

Rational Integration Tester



Reference Guide

Version 8.0.1



Note

Before using this information and the product it supports, read the information in “Notices” on page 590.

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

© Copyright IBM Corporation 2001, 2012.

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

Contents

About this Publication	viii
Intended Audience	ix
Scope	ix
Typographical Conventions	ix
Contacting IBM Support	ix
Integration and SOA Testing	1
Terminology	2
Getting to Know Rational Integration Tester	4
Service Oriented Architecture (SOA)	6
SOA Test Strategies	7
Who Uses Rational Integration Tester?	9
SOA and Rational Integration Tester	11
Rational Integration Tester Overview	27
Rational Integration Tester Projects	28
The Workbench	30
Test Artifact Documentation	40
Tags	42
Environments	43
Test Cycles	49
Email Project Files	57
Project Settings	61
Switching Users on a Project	74
Changing Preferences	75

Architecture School	87
Overview	88
The Logical View	90
The Physical View	117
Synchronisation	131
The Schema Library	136
The Rule Cache	138
Environment Tasks	143
 Requirements Library	 150
Overview	151
Creating Messages	154
 Recording Studio	 156
Overview	157
Using the Recording Studio	161
 Test Factory	 181
Overview	182
Tests	185
Performance Tests	222
Test Suites	223
Stubs	247
Test Templates	257
Test Data Sources	259
 Test Lab	 274
Overview	275
Working in the Test Lab	279
 Results Gallery	 293
Overview	294
Test Suite Results	295
Test Case Results	297

Test Cycle Results	299
Detailed Reports	301
Managing Reports, Console Output, and Notes	303
Notes.....	305
Deleting Test Suite Results.....	306
Populating Suite Details for Past Test Runs	308
Getting Started.....	309
Starting the Program	310
The Welcome Dialog	312
Opening an Existing Project.....	314
Creating a New Project	316
Cloning a Project	320
Fetching a Project from Source Control	325
Open a Resource Execution URL	330
Project Errors	332
Transports and Formatters	334
Transports	335
Formatters	339
Dynamic Formatters	342
Messages	343
Overview	344
Selecting a Transport and Formatter	345
Constructing Message Headers	346
Constructing Message Bodies	347
Repeating Elements.....	363
MIME, DIME, and Multipart Content.....	373
Using Attachments.....	374
Applying Specific Formats to a Message	376
Apply Integra Message Handlers	378
XML Message Properties.....	380
The Field Editor	382

Schemas	406
Creating Schema Resources	407
Using Schemas	411
Changing Schemas	414
Analysing Schemas	415
File Schemas	419
Record Layouts	423
 Databases	 442
Creating a Database Resource	443
Using the Database Connection	450
 Identity Stores and SSL	 462
Overview	463
Creating an Identity Store	464
Configuring an Identity Store	465
 Functions	 466
Availability and Usage	467
Assertion Functions	468
Mathematical Functions	470
String Functions	472
Other Functions	476
 Custom Functions	 484
Overview	485
The formatDate Example	486
Create a Plugin in Eclipse	487
Develop the Function Class	492
Create and Configure an Extension Point	495
Configure and Use the Function in Rational Integration Tester	500
Create a Function without Eclipse	501
Converting Existing Function Classes with Eclipse	503

Converting Existing Function Classes without Eclipse	504
External Tools.....	506
Overview	507
Command Line Execution	508
ANT	511
HP Quality Center.....	514
HP QuickTest Professional	516
Troubleshooting.....	517
Transport Diagnostics	518
Appendix A: Test Actions	519
Messaging Actions.....	520
BPM Actions	529
Flow Actions	530
General Actions	552
Performance Actions	582
Appendix B: Date & Time Patterns	583
Overview	584
Formatting and Parsing Date/Time Patterns.....	585
Glossary	588
Notices	590
Trademarks and service marks	593

About this Publication

Contents

Intended Audience

Scope

Typographical Conventions

Contacting IBM Support

This guide provides details of how to use IBM® Rational® Integration Tester. Deployment and configuration details can be found in *IBM Rational Integration Tester Installation Guide*. This version of Rational Integration Tester is compatible with a number of enterprise messaging solutions (for example, JMS, TIBCO EMS, and so on) by means of a collection of protocol- or format-specific plugins.

Intended Audience

This document intended to be read by those with a fair understanding and exposure to the concepts involved in both testing and development and in enterprise integration.

Scope

This document is about Rational Integration Tester, its environment, and its configuration – it does not discuss specific messaging technologies (for example, TIBCO EM, JMS implementations, and so on). If you wish to familiarize yourself with such technologies, please refer to the documentation that is provided by the relevant companies or individuals.

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/

Integration and SOA Testing

Contents

Terminology

Getting to Know Rational Integration Tester

Service Oriented Architecture (SOA)

SOA Test Strategies

Who Uses Rational Integration Tester?

SOA and Rational Integration Tester

This chapter provides a description of integration and Service Oriented Architecture (SOA) testing, the domain in which Rational Integration Tester is intended. Information is also included about IBM's view of the SOA and details about the different types of users who can call upon Rational Integration Tester to solve specific testing needs.

1.1 Terminology

This section provides more detail about the components and other terminology that are used when describing SOA and how it can be tested in Rational Integration Tester.

Component – an element of the system under test, including both the providers of services and infrastructure elements. All components can contribute to the collection of things that can be monitored.

Host – a logically named machine on which components run.

Infrastructure Component – the applications on which our SOA depends (for example, messaging transports, databases, and so on). These are logically named objects which have physical realisation in a given environment.

Interaction Pattern / Type – a template, devoid of application semantics, that describes a generic pattern for the exchange of information between agents.

These define the manner of interactions that take place between components (for example, one way, request response, and so on). These are not limited to messaging / web services and are the more generic description that will support testing of other exposed operations (for example, database stored procedures).

Logical Component – a provider of services through defined operations, these are either the items that are to be tested or the dependencies of other components and thus may need to be stubbed or monitored.

Logical View – a view of the components, their operations, dependencies and associated schemas.

MEP (Message Exchange Pattern) – the defined list of interaction patterns for a selected operation, including the interaction type (publish, subscribe, publish/subscribe, or request/reply), schema, and bindings.

Operation – a well-defined point of exposed functionality provided by a service. These may be both private and public (that is, whether or not they are visible outside of the component within which they are defined). Also known as Entry Point, End Point, API Call.

Performance Test – a distributed test that runs a configured load profile (for example, virtual users) to capture key performance indicators (for example, transactions per second, CPU load, and so on).

Physical View – a series of configured components for all of the items in the infrastructure grouped into environments.

Reference (Dependency) – the relationship from an operation to another operation or component indicating that when used, the operation requires an implementation of the reference to be available in order to provide its functionality.

Schema – an object source that provides information about services to be tested and can contribute to the infrastructure components and physical architecture values. Specific schema types supported in Rational Integration Tester include WSDL, XSD, DTD, Java DTO, File, and COBOL Copybook. Other specialized schemas (for example, TIBCO BW Private Process and Active Enterprise, SAP BAPIs, and webMethods IS schemas) are added to a project when synchronizing with the associated external source.

Stub – a stand in / replacement for an actual operation, its capability will depend upon the extent that the user has “programmed” it. These may also be known as “mock objects.”

SUT (System Under Test) – ultimately this would be the entire enterprise-based SOA (if the Rational Integration Tester project spanned that much content).

Synchronisation Source – an object source that provides information about processes to be tested (Operations) and connection details to infrastructure resources (infrastructure components and physical architecture values).

Test – a series of actions that exercise a particular operation and validate its correctness.

1.2 Getting to Know Rational Integration Tester

Rational Integration Tester provides a wide range of features and functionality that give users the ability to test their integration and SOA projects successfully. This section highlights a few of the most powerful features Rational Integration Tester – features that let users start testing complex systems as quickly as possible.

1.2.1 Synchronizing with External Resources

Few integration or SOA testing projects are simple, and they rely on complex systems and resources that are external to Rational Integration Tester (for example, WSDLs, SAP applications, TIBCO BusinessWorks projects, and so on). To enable reliable and accurate interactions with these external resources, Rational Integration Tester provides a powerful synchronisation feature. Synchronisation lets you reference an external resource and create all of the local artefacts needed to begin testing that resource – immediately. Synchronisation can also help to enforce standards across a project, encourage better change control, and ensure that proper environment bindings are utilized.

For more information about synchronisation in Rational Integration Tester, see [Synchronisation](#) in the [Architecture School](#) chapter.

1.2.2 Sharing Projects

Most testing projects require the interaction of multiple users who may or may not be working in the same location. To ensure that all users can access up-to-date resources in their current project, Rational Integration Tester projects can be shared using industry standard Source Control Management (SCM) systems. Utilizing SCM with Rational Integration Tester is a reliable way to share projects among multiple users without having to manually provide version control and rollback capabilities.

Support for different SCM systems is provided by the installation of Eclipse Team Provider plug-ins. For more information about SCM in Rational Integration Tester, refer to *IBM Rational Integration Tester SCM Application Guide*.

1.2.3 The Requirements Library

To help ensure that components and services are tested the way they are intended, software engineers (and other users) can create requirements – in the form of messages – that can be quickly dropped into tests. The use of requirements can save time and ensure that the right messages are used in required tests.

For more information about using requirements, see [Requirements Library](#).

1.2.4 Recording

Rational Integration Tester provides the ability to record events in the system under test, and recorded messages can be used to create tests and stubs more quickly and accurately. Recording is supported with HTTP, EMS, RV, and webMethods IS transports.

For more information about recording, see [Recording Studio](#).

1.3 Service Oriented Architecture (SOA)

An SOA consists of business processes that are implemented by linking individual services – programmatically or with tools based on Business Process Execution Language (BPEL) or Business Process Management (BPM). SOA testing can be challenging as there is no GUI (that is, the focus is on testing interfaces and services).

Rational Integration Tester is based on the Service Component Architecture (SCA), which provides a programming model for building applications and systems based on an SOA. For more information about SCA, see <http://www.osoa.org/display/Main/Service+Component+Architecture+Home>.

Based on this concept, the three main components used in Rational Integration Tester's SOA design are:

- **Operations** are transactions that represent single, logical units of work. The execution of an operation will typically cause one or more persistent data records to be read, written, or modified. An Operation has a structured interface for invocation and response.
- **Services** are granular, self-contained groupings of business operations. The focus on services is different from traditional GUI applications as a service is neither a complete application nor a single sub-system. Instead, a service is designed to maximize re-use and should be stateless. Additionally, a service will be able to accommodate multiple invocation patterns (for example, SOAP over HTTP for synchronous implementations or SOAP over JMS for asynchronous implementations).
- A **Business Process** is long-running set of actions or activities performed with specific business goals in mind. Business processes typically encompass multiple service invocations.

1.4 SOA Test Strategies

SOA is more than simple web services providing interfaces to existing applications. An SOA implementation is likely to involve multiple services and vendors and should be considered a complex computing problem.

Many companies have tried implementing automated testing for GUI solutions. Due to the changing nature of GUIs they have found it very difficult to achieve ROI. That is not to say that GUI testing can not be achieved, but it should be done during the project lifecycle by applying automated interface testing.

Focusing on programmatic interfaces and services in automated testing offers significant ROI, as the design contracts for these interfaces will change less often – making changes to business processes in a distributed environment is costly. When it comes to SOA, a different approach, skill-set, and mind-set are required for testing.

The main factors of a successful test methodology are as follows:

- Testing must start early in the project once business processes (for example, use cases) and interface definitions are agreed upon. If the SOA has already been deployed and does not have any documentation, then regression tests can be used to determine backwards compatibility.
- QA tests should be run at key stages throughout the project, and these tests should be shared with other groups as early as possible.
- A test plan should outline the high-level test suites that need to be executed along with the acceptance criteria.
- When designing tests, a data-driven approach to test generation can save time, particularly if an agile approach is being implemented.
- To aid test design, a focus on test coverage analysis is also needed along with requirements traceability.
- The following test phases are likely to be required when testing an SOA:
 - Policy testing to ensure, for example, continued compliance with an XML schema. This type of testing is required throughout the lifetime of an SOA ensuring there is always compliance to the interface definitions.
 - Service-component level testing (or unit testing) ensuring the individual components that make up a service work correctly. This level of testing is typically performed by a software engineer.

-
- Service-level testing focusing on functional, performance, and security testing of individual services within a controlled environment. This is an important level of testing that can help ensure that the quality of the system components is sufficient.
 - Integration-level testing focuses on testing the service interfaces. This level of testing aims to determine if the interface behavior and information sharing between linked services works as specified (interoperability, functional, regression and security). The interfaces must comply with the defined interface definitions (that is, standards, format, and data validation). Integration tests should also push the different layers of communications and network protocols. Stubs are likely to be required to simulate services that are not yet available.
 - Process-level testing ensures services are operating collectively as specified. This phase of testing would cover business logic, sequencing, exception handling and performance.
 - System-level testing (or acceptance testing) focuses on testing services that are linked together to deliver a business process. This level of testing focuses on the key business scenarios and requires test coverage from previous test runs to be compared so that quality can be assessed.
- Test execution is then achieved by executing the test plan with each test run associated with specific versions of the components under test and saved for future reference.
 - Simulation (stubs) enable project agility by crossing test phase boundaries (that is, development, unit testing, integration testing, performance testing). To achieve successful simulation requires proper control of test data so that stubs can be created and will use consistent data across the test environments. Stubs require maintenance across the different environments as both the data and endpoints change, meaning the deployment method for the stub may need to change. Basic stubs should be created when building a test project. These will form the basis for more focused stubs that will be needed as the project evolves.
 - Once an SOA is deployed then regression testing is needed ensuring continued compliance, backwards compatibility and performance.
 - A risk-based approach to testing should be taken where business impact, the likelihood of failure, and frequency of use are considered during test design and execution. This approach can help IT management and business users to make informed decisions quickly.
 - Performance testing can use the assets from the test projects to validate performance and capacity planning.
-

1.5 Who Uses Rational Integration Tester?

The following profiles describe different types of users who might utilize Rational Integration Tester in helping to test an SOA.

1.5.1 Systems Administrator / Architect

The Architect has overall technical knowledge about the components within the system and needs a way to share this information with the rest of the team. This knowledge needs to be shared early on rather than through “ad-hoc” questions (for example, where services are deployed, the database to be used, and so on).

1.5.2 Software Engineer

A software engineer (developer) has a technical background and needs to unit test the individual classes that he or she develops for services and processes. Developer testing may also mean integrating with other services that are supplied by another team that may or may not be external to the organisation. Typically, a software engineer will utilize automated tests that are produced by the QA team rather than writing a test harness and spending an extensive amount of time on testing and/or unit testing.

1.5.3 Software QA Engineer

The focus of a Software QA (SQA) Engineer is on integration and system testing within a controlled environment. The SQA engineer has a traditional testing background in GUI testing, but now needs to take on more SOA implementations that have progressed from simple web services integrating with existing applications to complex distributed service oriented architectures.

The SQA engineer needs to develop additional technical skills, particularly in the underlying SOA technologies and network security. The SQA engineer needs an automated test tool that lets him or her focus on test design and test coverage rather than on managing the tool or message payload syntax (which is handled by means of the numerous message formatters available in Rational Integration Tester).

Testing has moved forward for SOA projects and an agile approach is vital to help meet business expectations around delivery timeframes. Using this approach, software testing is carried out in parallel to other project activities (that is, development), which means that a time-saving solution is needed to ramp-up testing as quickly as possible.

1.5.4 Software Test Manager

The Software Test Manager is concerned with the effectiveness of the testing process, test design, test coverage, test execution, and the timely generation of comprehensive reports that provide guidance to the project team and business users. The top concern is the testing strategy for SOA projects and finding ways to reduce effort on manual tasks that distract from test design. A risk-based approach to testing is becoming increasingly more important, and reports need to reflect this approach so that business users can make informed decisions quickly.

1.5.5 IBM Consultant

IBM consultants have extensive knowledge of SOA testing and the Rational Integration Tester application, and they can be used to kick start a project in its first few days. A consultant would utilize the skills and knowledge of the customers (for example, architects and system administrators) to help guide key users in setting up a deployment so that testers can start constructing tests. An IBM consultant will leave the customer with a preconfigured project structure and staff that has been trained in the use and application of Rational Integration Tester.

1.6 SOA and Rational Integration Tester

Rational Integration Tester's primary intent is to simplify the process by which the user builds tests (that is, schema-driven message creation, test driven development, and so on). One of the main ways that Rational Integration Tester can simplify is through recording – capturing messages and events from an existing system to aid in the rapid creation of regression tests.

To do this, Rational Integration Tester separates the more complex task of learning about the system and configuring the application from the tasks of designing and executing tests.

Rational Integration Tester makes it easy for users to design tests by providing the ability to record a series of events and use a selection of these to provide the framework for the test steps from which they will work.

In a practical setting, different users will utilize Rational Integration Tester to design, build, execute, and evaluate tests. Rational Integration Tester utilizes different perspectives that are intended to be used by these different groups of users. The following sections describe the different phases of testing and include information about the perspective within Rational Integration Tester that will be used during each phase.

1.6.1 Design Phase – System Administrators and Architect

The design phase will be the main focus of System Administrators and Architects, and they will work in the [Architecture School](#) and [Requirements Library](#) perspectives in Rational Integration Tester. The main focus of the design phase is to create the following information in Rational Integration Tester so that users can start building tests:

- The components that make up the system along with the individual operations that are to be tested, including how they are to be invoked.
- A description of the system's physical infrastructure, including all of the transport and configuration details of the system's testable components.
- The dependencies between the system components (for example, when running Process A, it will invoke process B and C and write data to Database D).
- A collection of schemas that will be applied to the messages used in testing the various transports and components within the system.
- A comprehensive list of requirements, built in the form of messages containing all of the applicable test data. These messages can be used to quickly create tests and stubs that immediately contain realistic data.

-
- A list of environments that bind logical components to different physical components, letting users quickly test different parts of the development lifecycle using the same set of tests.

1.6.2 Building Tests – Software Engineers and SQA Engineers

Software and SQA Engineers build most of the test resources in Rational Integration Tester, and they will spend most of their time in the [Recording Studio](#) and [Test Factory](#) perspectives.

Once the system to be tested has been designed, users can start building their tests (unit, integration, acceptance, and so on). Rational Integration Tester provides a number of ways in which tests, test suites, and stubs can be created:

- Operations can be selected for monitoring, and Rational Integration Tester will “listen” on the transports and other components that these operations include. Live messages that are generated by or passed through the selected operation will be recorded in Rational Integration Tester, and these can be used to quickly create tests, suites, stubs, or triggers.
- Users can create tests manually, utilizing a wide array of “test actions” to create an almost unlimited number of scenarios.
- Stubs can be created to simulate missing or otherwise unavailable systems upon which other test resources depend.

1.6.3 Executing Tests – Software Engineers and SQA Engineers

When the time comes to execute tests, software engineers and testers have numerous options available – both within and outside of Rational Integration Tester.

Tests, stubs, and suites can be executed in the [Test Lab](#) perspective, using the console to monitor progress and results, and the Test Repair wizard to fix validation errors.

NOTE: While tests can be executed from other perspectives in Rational Integration Tester, the [Test Lab](#) provides the main functionality for doing so.

In addition to working directly in Rational Integration Tester, several methods of executing tests are available, including command line execution, ANT, and HP Quality Center. For more information, see [External Tools](#) and *IBM Rational Integration Tester Integration Guide for HP Quality Center*.

1.6.4 Evaluation – Software Test Manager

Throughout the testing lifecycle, the Software Test Manager needs to know how effectively the system is being tested, as well as the individual results of various tests and scenarios. The [Results Gallery](#) perspective provides a comprehensive collection of detailed reports, as well as coverage, error, and performance reports. Additionally, the Results Server provides a web-based view of project results, and users who have integrated Rational Integration Tester with HP Quality Center have access to an even wider range of results and reports.

Rational Integration Tester Overview

Contents

[Rational Integration Tester Projects](#)

[The Workbench](#)

[Test Artifact Documentation](#)

[Tags](#)

[Environments](#)

[Test Cycles](#)

[Email Project Files](#)

[Project Settings](#)

[Switching Users on a Project](#)

[Changing Preferences](#)

This chapter provides an overview of Rational Integration Tester, including information about projects and files, the Rational Integration Tester user interface, environments, and project settings and preferences.

2.1 Rational Integration Tester Projects

In Rational Integration Tester, all work is organized into individual projects, similar to a development IDE. This allows for simpler organisation of complex test scenarios.

Rational Integration Tester projects correspond to an underlying directory structure. For this reason, the projects themselves do not need to be explicitly saved, but the resources they contain do.

2.1.1 File Locations

While projects can be stored in any physical location, they would normally reside on a local drive of the machine that is running Rational Integration Tester. When sharing testing artifacts, it is best to employ configuration management software (for example, CVS, ClearCase, Visual Source Safe, and so on).

To ensure portability, internal Rational Integration Tester artifacts are referenced by a unique ID. When working with 3rd party elements (for example, XSDs, TIBCO AE repositories, and so on), a URL or the system variable `PROJECT_ROOT` should be utilized. When set up properly, projects can be accessed in the same way from different machines and even moved (in their entirety) from one machine to another.

2.1.2 Project Permissions

Rational Integration Tester supports project permissions based on users in a supported LDAP implementation (currently Active Directory). Permissions can be assigned to users by the project administrator, which can be configured when a project is created or added to an existing project. If an LDAP user creates a new project, that user is granted all available permissions by default. He or she, however, can not add new users to the project or modify permissions for existing users.

NOTE: If the current user attempts to select a perspective to which he or she does have access, including indirect access from a different perspective (for example, trying to run a test from the Test Factory when access has not been granted to the Test Lab), an error is displayed and access will be denied.

2.1.3 Configuration Management

Individual elements of a Rational Integration Tester project are persisted as single files. As such, configuration management software can be used to “version” your Rational Integration Tester projects, as desired.

NOTE: Details about using any specific configuration management software are beyond the scope of this document.

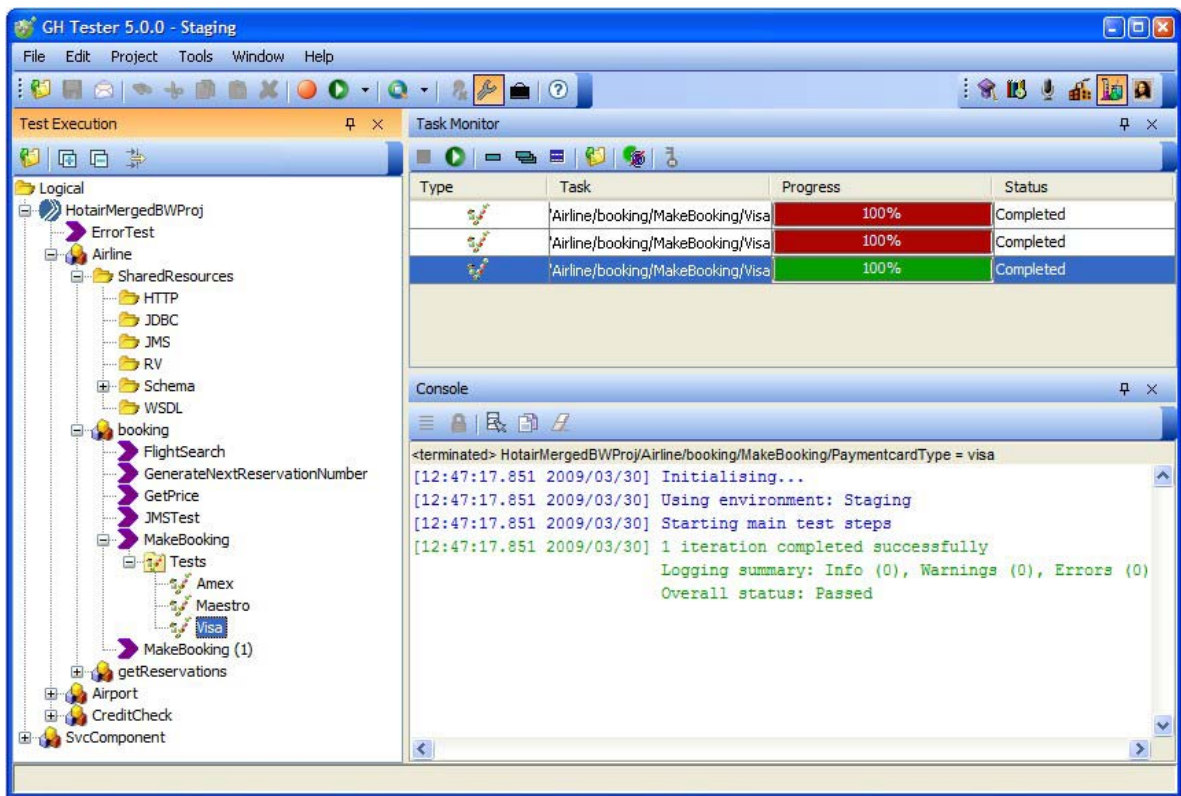
2.1.4 Naming Conventions

While there is no standard for naming the various components in a Rational Integration Tester project, the following guidelines may help you to better organize your projects.




- Tests, suites, and stubs should be named in a consistent and meaningful manner, and should include a description and location within the project hierarchy. You may also want to include an identifier in the name, which can be correlated with a test plan or a tracking tool.
- The names of system artifacts (for example, components and operations) should be brief yet descriptive. For example, they should include the transport type, sub-type, and detail: RV_CM_Pricing, JMS_TOPIC_Pricing, and so on
- Test components (for example, scenarios or data sources) should be named according to their contents or their intended use.

2.2 The Workbench

The main window of Rational Integration Tester is a workbench that contains several dockable windows. These windows are organized in a logical way, providing an intuitive, easy-to-use central workspace.



The initial layout of the workbench is pre-determined, and it can be restored at any time by selecting **Window > Reset Current Perspective** from the main menu. When the perspective is reset, the internal views are restored to their default positions and sizes, and the Rational Integration Tester application is maximized.

Many aspects of the workspace can be customized. Windows can be resized by clicking and dragging on their borders, they can be closed with the  button in the top right hand corner, and they can be set to hide automatically when not in use with  or to remain visible with .

2.2.1 Rational Integration Tester Perspectives

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



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



Requirements Library (F8) - create requirements that will help other users to create tests and test data more quickly and more accurately



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



Test Factory (F10) - create tests, test suites, and stubs
















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



Results Gallery (F12) - view historical test data and various reports for any stored test artefacts

2.2.2 Common Actions

Throughout most of Rational Integration Tester, the following general actions can be performed on test artifacts using the main toolbar, keyboard shortcuts, or context (right-click) menu.

Command	Icon	Keystroke	Action Taken
Expand			Expand the selected branch in the current tree.
Collapse			Collapse the selected branch in the current tree.
File > Open		Ctrl + O, Enter, double-click	Open the selected artifact for editing.
File > Save		Ctrl + S	Saves the test artifact that is selected in the current view.
File > Save All		Ctrl + Shift + S	Saves all test artifacts in the current view.
File > Send > Email			Opens a dialog that lets you select project resources, package them into a .zip archive, and send to a selected email address.
File > Close Project			Closes the current project and returns to the Rational Integration Tester Welcome dialog.
Edit > Cut		Ctrl + X	Cut the selected item to the clipboard.
Edit > Copy		Ctrl + C	Copy the selected item to the clipboard.
Edit > Paste		Ctrl + V	Paste the contents of the clipboard (previously cut or copied items) into the current location.
Edit > Delete		Delete	Delete the currently selected artifact.
Edit > Find		Ctrl + F	Opens a search field (as in most Internet browsers) to enter a string of text to find in the current view.
Edit > Find Files			
Rename		F2	Rename the currently selected artifact.
Window > Close Editor		Ctrl + W	Closes the active editor (in Requirements Library or Test Factory).
Window > Close All Editors		Ctrl + Shift + W	Closes all open editors (in Requirements Library or Test Factory).

NOTE: If a menu item or icon is unavailable, then the associated action is not permitted (for example, save unchanged item, rename multiple artifacts, and so on).

2.2.3 Additional Shortcuts


In addition to the common actions and shortcuts described in [Common Actions](#), the following shortcuts can help you work more quickly in Rational Integration Tester:

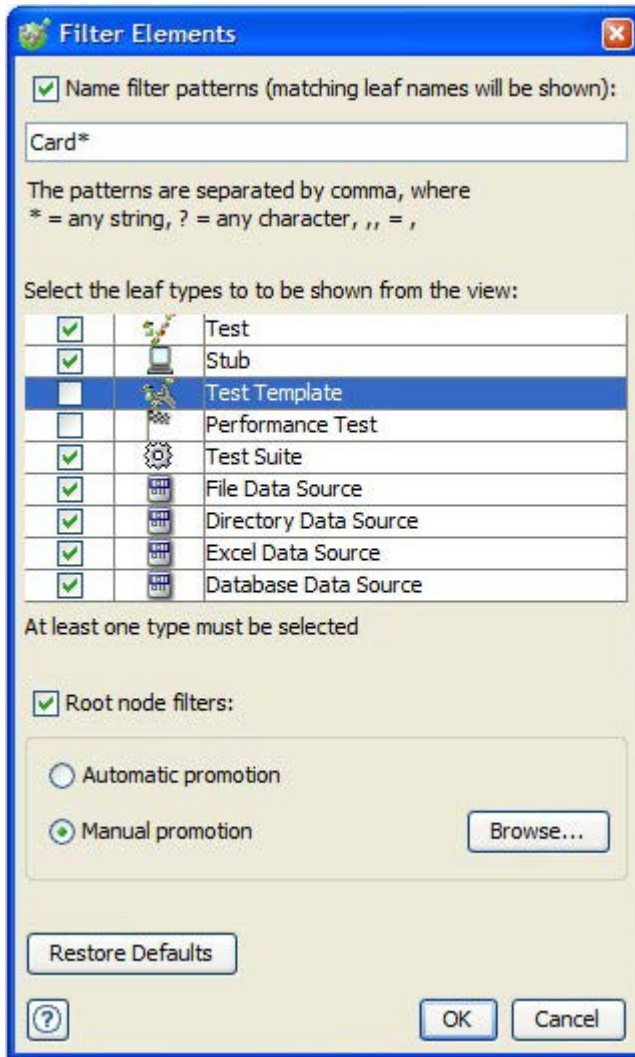
Key	Description	Notes
F1	Open Rational Integration Tester online help	
F5	Run the currently selected or last launched Test, Suite, or Stub	The current/last behavior can be modified in the application preferences.
Shift + F5	Open the custom execution dialog for running a Test, Suite, or Stub	
Ctrl + F5	Opens the custom execution dialog for running failed items in the selected Test Suite	

Key	Description	Notes
F7	Open the Architecture School perspective	
F8	Open the Requirements Library perspective	
F9	Open the Recording Studio perspective	
F10	Open the Test Factory perspective	
F11	Open the Test Lab perspective	
F12	Open the Results Gallery perspective	
Alt-F4	Close or quit	Closes the current window
Ctrl-E	Edit environments	Opens the Environment Editor
Alt-C	Cancel edit	Same as clicking the Cancel button.
Alt-O, Ctrl + Enter	Close and save	Same as clicking the OK button.
Alt-N	Moves to the next item when using the Find command	
Alt-P	Moves to the previous item when using the Find command	

NOTE: Throughout Rational Integration Tester, pressing the **Alt** key will reveal the shortcut key to activate buttons or menus (as in other Windows applications).

2.2.4 Filtering Component Trees

If desired, you can filter the contents of the component tree that is displayed in the Requirements Library, Test Factory, and Test Lab perspectives to show only specific test assets or test asset types. To configure the view filters, click the **Configure Filters** icon  in the toolbar.



You can enter a name filter to hide all test assets that do not match the specified pattern (for example, Card* will hide all assets whose name does not begin with “Card”). Name filter patterns are case sensitive.

Additionally, you can select specific test asset types to hide from the view by clearing the check box next to their type.

Finally, you can set manual root node filters by enabling the **Root node filters** option. Select the **Manual promotion** entry and click **Browse** to locate the node that you want to promote. The selected node will appear as the root of the project in the component view.



To remove any filters and restore the default view, click **Restore Defaults**.

2.2.5 Finding Project Components

The **Edit > Find** command (**Ctrl + F**) is available in most Rational Integration Tester perspectives to let users locate and highlight components whose name matches a specified search string.




As you enter the search string in the **Find** field, the first matching item within the view is selected and the total number of matches is displayed to the right of the field.

You can browse forward and backward through matching items with the **Next**  and **Previous**  icons. You can also press Alt + N to move to the next item, or Alt + P to move to the previous item.



NOTE: In the Logical View of the Architecture School perspective, you can enable the **Auto Zoom + Pan** option to have Rational Integration Tester automatically adjust the view when moving between found items.

If no items match the current search string, the background of the **Find** field is displayed in red and “0 matches” is displayed.




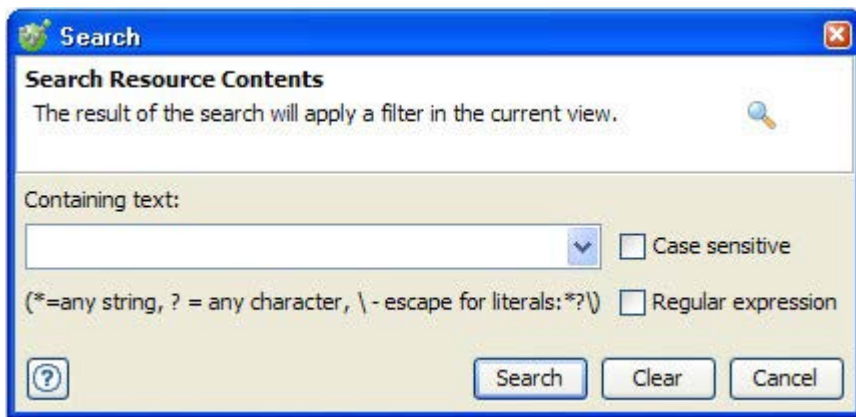
When finished, or to close the search area, click the  icon.

2.2.6 Expand and Collapse

Many items in the Rational Integration Tester GUI are displayed in a tree format. In such views, the **Expand Selection**  and **Collapse Selection**  icons are available to quickly expand or collapse all items within the selected branch.

2.2.7 Searching for Test Resources


You can search for test resources within the Test Factory and Test Lab perspectives by clicking the **Search Resource Contents**  icon. The search will examine the contents of every file within the project (for example, fields within messages of a test, the contents of a function within a test, and so on), and will apply a filter to the current tree view that will display only items whose contents match the search terms.



In the **Search Resource Contents** dialog, enter the search string and select or clear the **Case sensitive** check box, as desired. To select from previously used search strings, click the arrow on the search field box and select the desired string. If you want to use a regular expression for your search, tick the check box next to **Regular expression**.

NOTE: To filter on multiple strings you can use the pipe symbol ‘|’ as an OR operator (for example, “getData|sendData” will display both strings in the filtered view).

When finished, click **Search** to filter the component tree according to the selected search term.

To restore the tree to its unfiltered view, click the highlighted  icon and click **Clear** in the **Search** dialog.

2.2.8 Reload an Open Project

If you are adding resources to a project outside of the Rational Integration Tester Workbench (for example, copy and paste a folder containing test data from an existing project to the current project), you can refresh Rational Integration Tester so that newly added resources are displayed in the project.

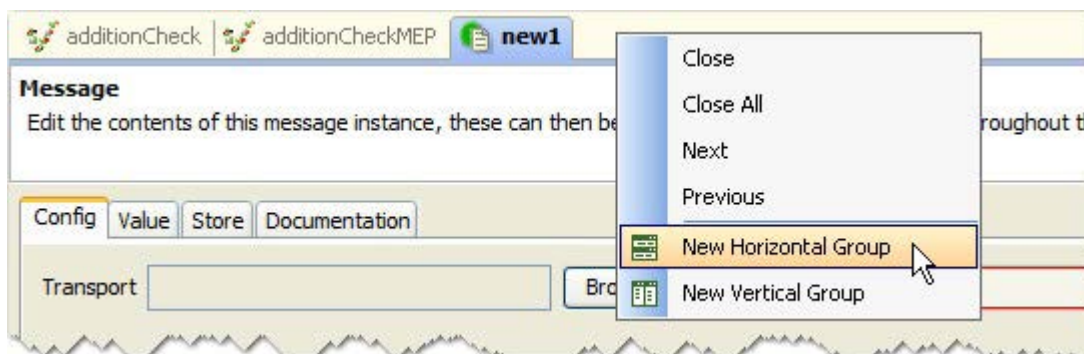
Select **Refresh** from Rational Integration Tester's **Project** menu to reload the contents of the project.

2.2.9 Editing Groups

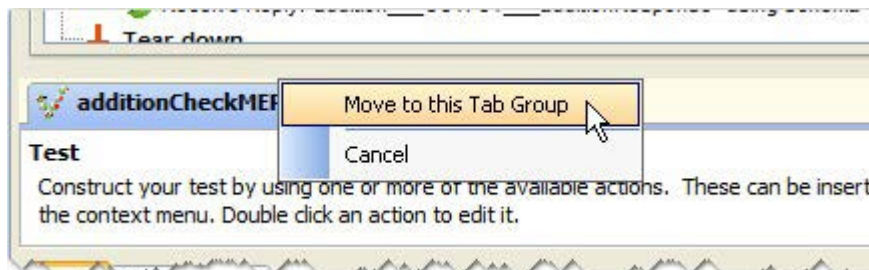
When multiple requirements and/or test assets are opened for editing (see [Requirements Library](#) or [Test Factory](#)), the opened items can be organized into groups by dragging their named tabs into an existing editing panel.

To create a new group, drag an open item into its own editor. When you drop the tab, you can create a new horizontal or vertical group from the context menu that appears.

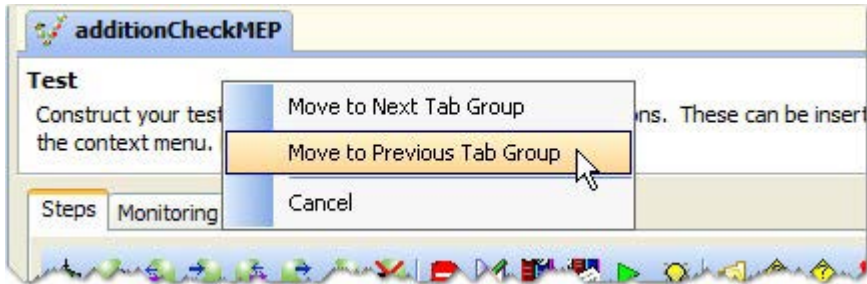
NOTE: You can not use vertical and horizontal groups at the same time.



To move an open item to an existing group, drag and drop it on the desired group and select **Move to this Tab Group** from the context menu.

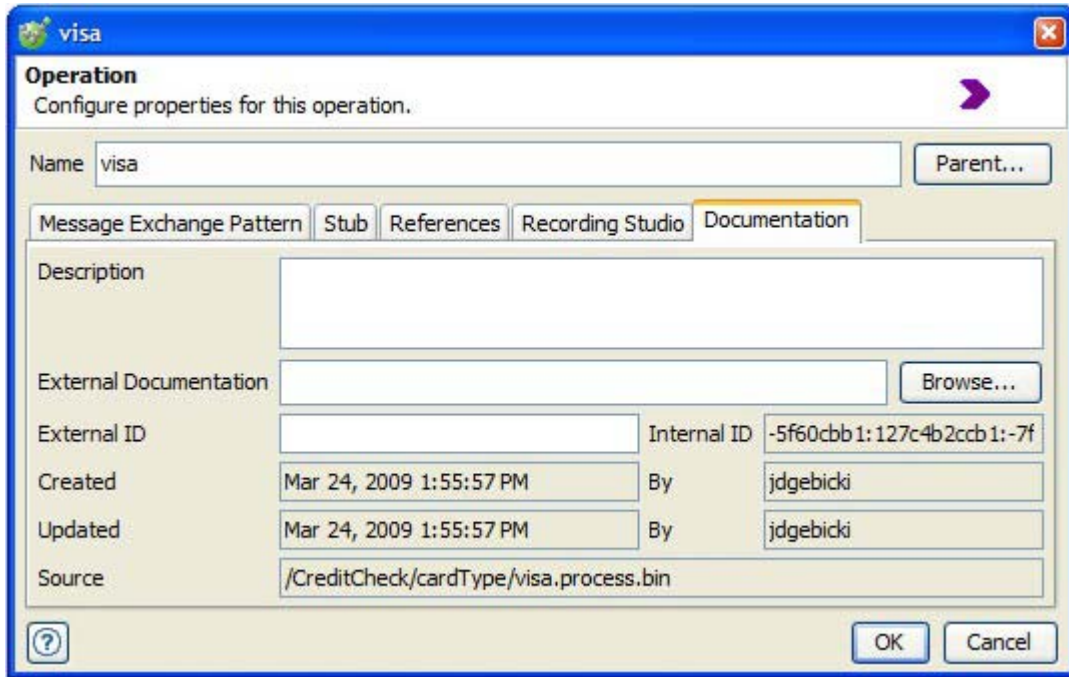


To move an open item to the next or previous group (when at least two groups exist), drag it into its own editor and select the desired option from the context menu.



2.3 Test Artifact Documentation

All Rational Integration Tester test artifacts (for example, tests, suites, messages, test templates, and so on) contain a **Documentation** tab in their respective editors, shown below.



The screenshot shows a window titled 'visa' with a blue header bar. Below the header, the text 'Operation' and 'Configure properties for this operation.' are displayed. A purple arrow icon is on the right. The 'Name' field contains 'visa' and a 'Parent...' button is to its right. Below this is a tabbed interface with five tabs: 'Message Exchange Pattern', 'Stub', 'References', 'Recording Studio', and 'Documentation'. The 'Documentation' tab is selected. It contains a 'Description' text area, an 'External Documentation' field with a 'Browse...' button, an 'External ID' field, an 'Internal ID' field containing '-5f60cbb1:127c4b2ccb1:-7f', a 'Created' field with 'Mar 24, 2009 1:55:57 PM' and a 'By' field with 'jdgebicki', an 'Updated' field with 'Mar 24, 2009 1:55:57 PM' and a 'By' field with 'jdgebicki', and a 'Source' field containing '/CreditCheck/cardType/visa.process.bin'. At the bottom are 'OK' and 'Cancel' buttons.

Within this tab users can enter a description of the artifact (in the **Description** field), an optional link to an external document related to the artifact (in the **External Documentation** field), and an optional External ID for the artefact. The default location for external documentation can be defined under [Project Settings](#), and external documents can be launched in an associated program or using a user-defined command, as described in [Applications](#).

The **Tools** menu and context menu contains a **Documentation** option for the selected artifact and any associated items (for example, parents, environments, and so on). If an external document has been defined for an artifact, the document name will be displayed next to the artifact. Selecting an item's documentation entry will open the artifact editor (with the **Documentation** tab highlighted) or the external document.

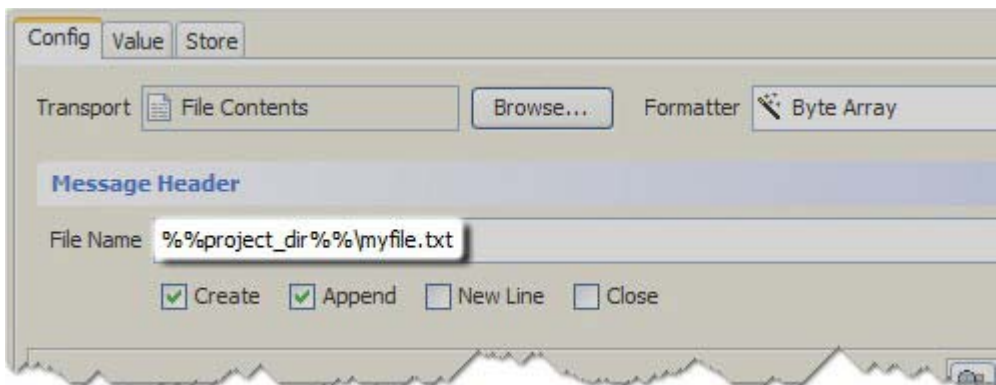
NOTE: If an external document's extension is not associated with an application or command, you will be prompted to provide one when trying to launch the document.

Additional details about the selected artifact (that is, date and time created and updated, internal ID, ID of the user who created and last updated the artifact, and the artifact's source within the project) are also provided within the **Documentation** tab.

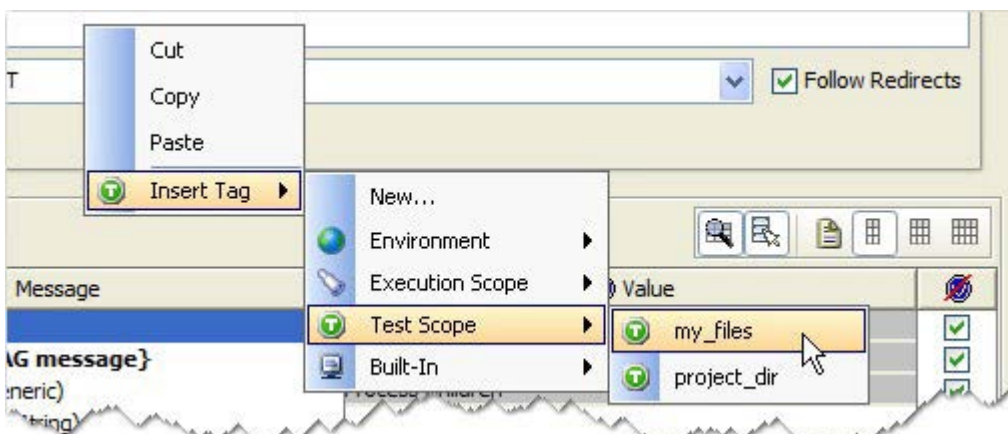
2.4 Tags

Tags are like data variables in Rational Integration Tester, consisting of a name and a default value. Tags can be used throughout the application to allow for dynamic data access. Tags can be used in most fields within Rational Integration Tester (for example, message headers, body elements, function definitions, and so on).

Tags can be entered manually by entering the tag name enclosed on both sides by two percent signs (for example, `%%tag_name%%`).



Tags can also be entered by right-clicking within a field and selecting a tag from the **Insert Tag** option of the context menu.



For more information about tags, see [The Tag Data Store](#).

2.5 Environments

Rational Integration Tester utilizes Environments to let users define groups of variables or tags that can be used in both tests and in transport definitions. Environments are a powerful feature that greatly improve the usability of Rational Integration Tester, and they are typically used to create configurations for different parts of a product workflow (for example, Development, QA, UAT, and so on). A single suite of tests can be reused in all cases when environments are used to modify run-time parameters that are appropriate to the current stage.

The default values for all tags (variables) are stored under Default Properties. When a tag is created here, that tag – if not explicitly defined (that is, value is shown as #undefined?) – will inherit its value from Default Properties.


NOTE: When Rational Integration Tester is launched, it will apply the environment that was in use when the application was last closed.

2.5.1 Create an Environment

A new environment can be created as follows:

- Click the small arrow next to the Environment icon in the main toolbar, then select the **Create New Environment** option.




- Click the **Create New Environment** icon  in the Environment Editor.
- Select the **Project > Create New Environment** menu option.

When prompted, enter a name for the new Environment. Once you do, the Environment is opened for viewing and/or editing (see [Edit an Environment](#)).

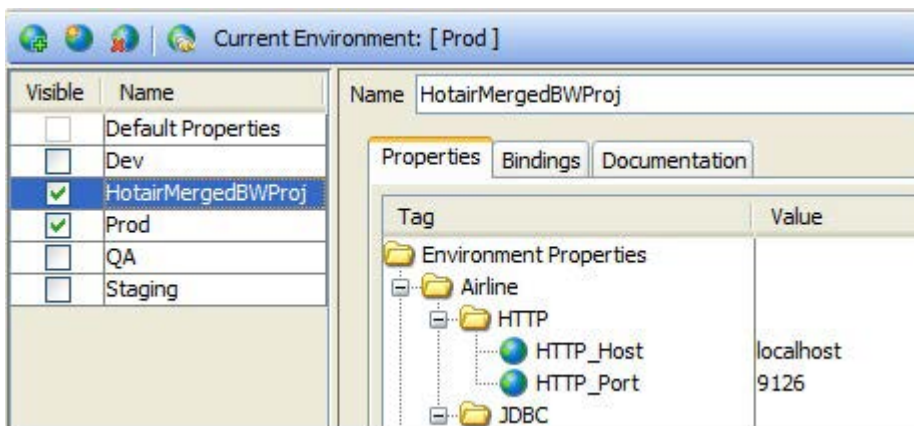
2.5.2 Edit an Environment

Immediately after creating a new environment, it is opened for editing. Otherwise, you can edit an existing environment as follows:

1. Open the Environment Editor in one of the following ways:
 - Click the Environment icon  in the main toolbar
 - Click the small arrow next to the Environment icon in the main toolbar and select the **Edit Environments** option.



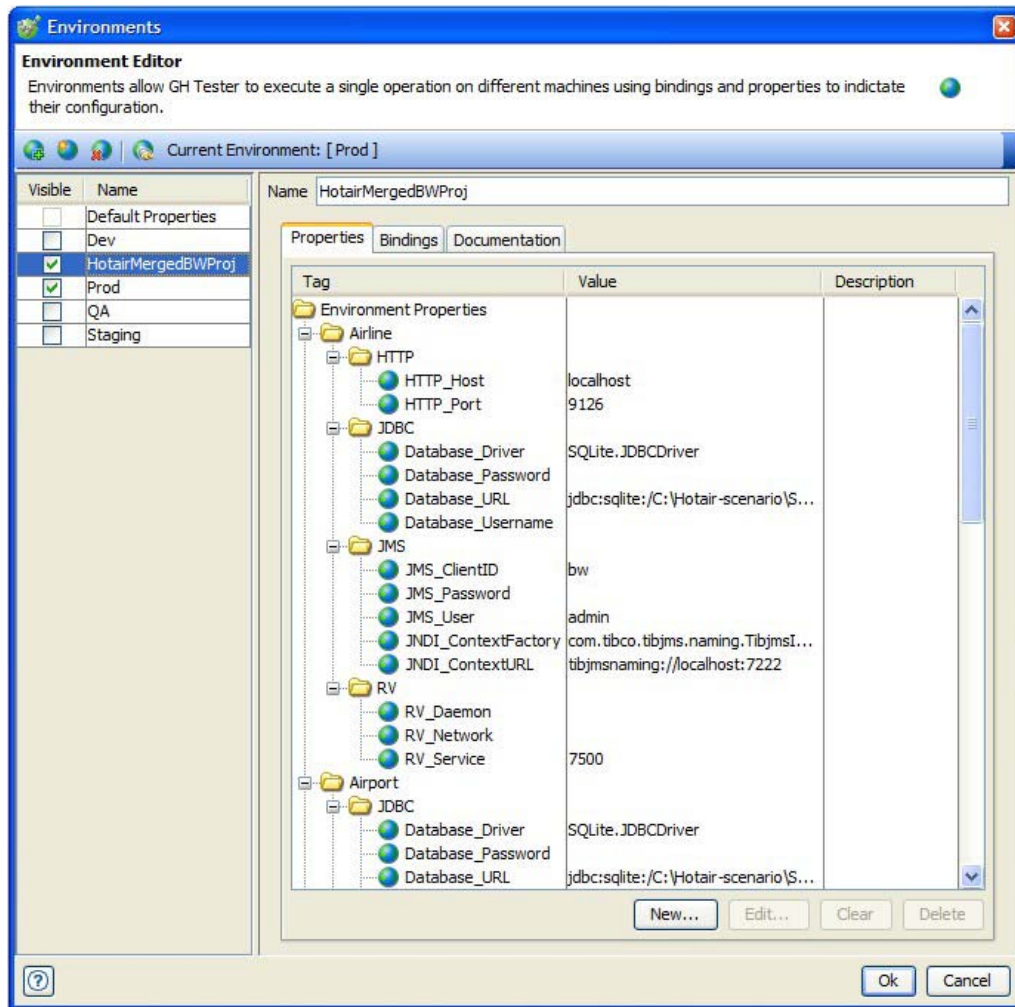
- Select the **Project > Edit Environments** menu option.
2. In the Environment Editor, select the desired environment from the list on the left side of the editor.



The contents of each environment are organized as **Properties**, **Bindings**, and **Documentation**.

Environment Properties

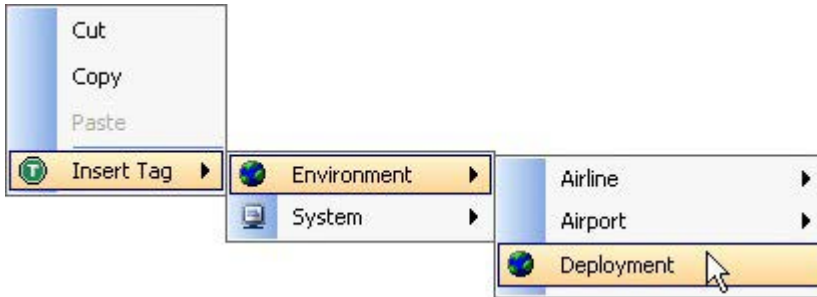
Environment Properties define all of the available tags in the project.



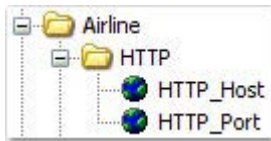
- To create a new tag click **New**.
- To edit an existing tag, select it and click **Edit**.
- To remove the current value of a tag, select it and click **Clear**.
- To remove a tag, select it and click **Delete**.

NOTE: To refer to tag values anywhere else in Rational Integration Tester, it is necessary to enter the tag name with two percent signs on either side of it. For example, to refer to the tag `WEBSERVER_HOST`, it must be typed as `%%WEBSERVER_HOST%%`.

NOTE: Tags appear in blue text when being used, (for example, `%%name%%`) and invalid tag names are underlined, (for example, `%%badname%%`). Any field that accepts tags allows input by right-clicking and choosing the Insert Tag option as shown.



Tag names can contain the '/' character to enable you to logically organize them into a folder-like structure. In the example shown below, the **HTTP_Host** and **HTTP_Port** tags would actually be named **Airline/HTTP/HTTP_Host** and **Airline/HTTP/HTTP_Port**.




Environment Bindings

Environment Bindings define the connection between the logical and physical components in the project.


Environment Documentation

Environment Documentation consists of an optional description, owner, and notes.

2.5.3 Clone an Environment

Rather than creating all new tags in a blank environment, you can clone an existing environment and add, edit, or modify tags as needed. In the Environment Editor, select the environment to clone and click the **Clone...** icon  in the toolbar.

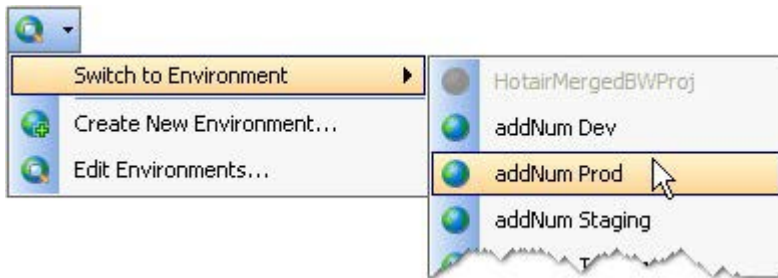
2.5.4 Delete an Environment


You can delete an existing environment in the Environment Editor. Simply select the environment to delete and click the **Remove...** icon  in the toolbar.

NOTE: The environment is not removed until you confirm your choice upon closing the editor.

2.5.5 Switching Environments

To switch from one environment to another, click the small arrow next to the Environment icon in the main toolbar and select the **Switch to Environment** option, then select the desired environment from those available.



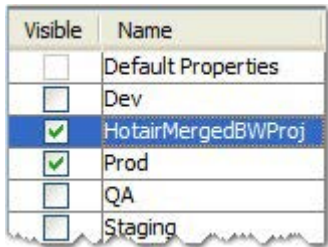
You can also change environments in the Environment Editor. Select the desired environment from the dropdown list and click the **Use...** icon  in the toolbar.

NOTE: When Rational Integration Tester is launched, it will apply the environment that was in use when the application was last closed.

2.5.6 Hiding and Showing Environments

If you use a large number of environments, you may want to select which ones are available when working in Rational Integration Tester. To change which environments

are hidden or displayed, simply toggle the check box under **Visible** next to the desired environment.



Visible	Name
<input type="checkbox"/>	Default Properties
<input type="checkbox"/>	Dev
<input checked="" type="checkbox"/>	HotairMergedBWProj
<input checked="" type="checkbox"/>	Prod
<input type="checkbox"/>	QA
<input type="checkbox"/>	Staging

You can select multiple environments using Shift or Ctrl, or select all with Ctrl + A, then toggle their current setting using the space bar.

NOTE: The Default Properties and the current environment can be hidden.



2.6 Test Cycles

A Test Cycle is a container for tests and test suites that spans multiple users and projects. The results of every test executed inside a Test Cycle are persisted to the database, regardless of whether the test is inside a suite or not.

Test cycles allow organizations to have a common parent for all test runs, such as those executed in HP Quality Center or by individual users who might be working across multiple projects. Users can then browse for the results of these test runs using the Test Cycle parent, navigating down to all the individual tests and test suites.

Users can create Test Cycles on-demand. Once a Test Cycle is created it can be immediately associated with projects, utilizing all project environments or a subset of them, and multiple projects can be associated with a single Test Cycle.

The state of the project with respect to Test Cycles is displayed in the status bar at the bottom of the Rational Integration Tester window. If the project is not associated with a Test Cycle, **<No Test Cycle>** is displayed. If the project is associated with a Test Cycle, the name of the Test Cycle will be displayed. A project can be added to or removed from (joined or unjoined) a Test Cycle at any time.

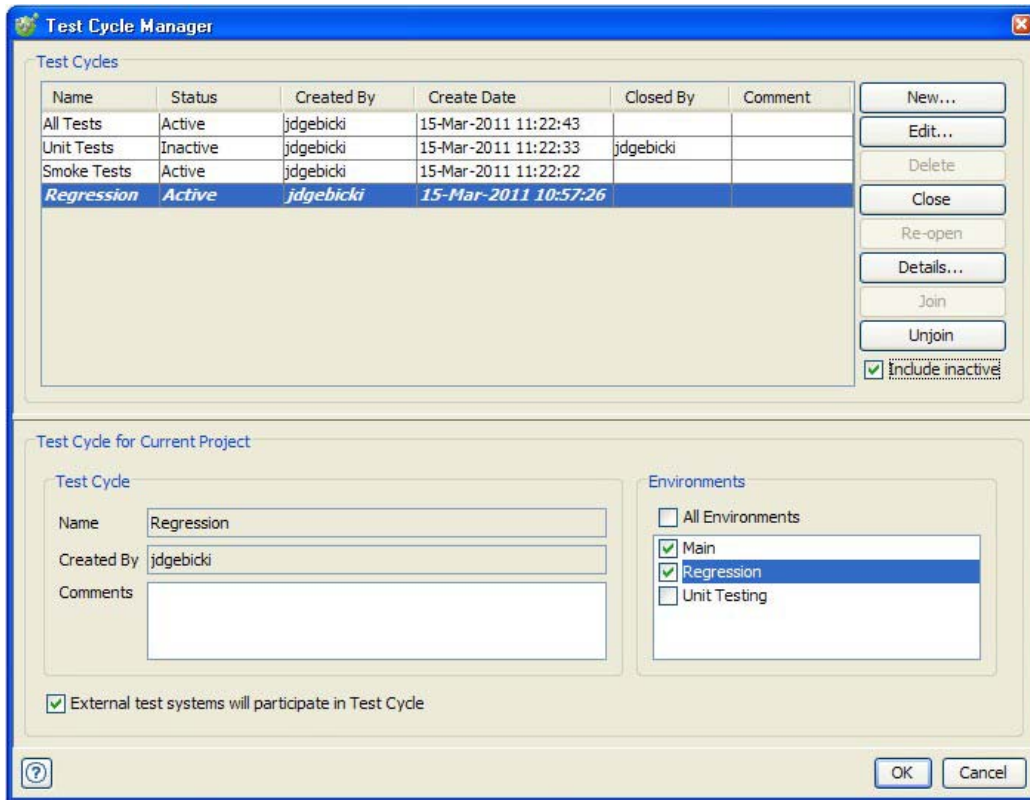
When the Test Cycle is active – indicated by the  icon – all executed tests will be stored against the Test Cycle. If the Test Cycle is paused – indicated by the  icon – Rational Integration Tester runs as before (that is, test results are not persisted, and test suite results are persisted for the suite but not for the Test Cycle).

Test Cycles can be marked to include tests that are executed externally (for example, from HP Quality Center or from the command line). In these cases, if tests are executed externally and their containing project is associated with a Test Cycle, all tests will be part of the Test Cycle and their results will be saved to the database.

NOTE: If a project has already been associated with a Test Cycle, Rational Integration Tester will prompt new users to join the Test Cycle the first time they open the project. If the user accepts, they will be joined immediately and remain so until the Test Cycle is closed or otherwise unjoined from the project. If the user chooses not to join when prompted, the project and Test Cycle can be associated later.

2.6.1 Manage Test Cycles

Test Cycles are managed in the Test Cycle Manager window, available by right-clicking the Test Cycle status indicator and selecting **Manage** from the context menu, or by selecting **Project > Test Cycle > Manage** from Rational Integration Tester's main toolbar.



The Test Cycle Manager displays a list of all Test Cycles and their details, as follows:

Name	The name of the Test Cycle, provided when the Test Cycle was created. The name can be modified when editing the Test Cycle.
Status	Active indicates that the Test Cycle is open and can be joined to the current project. Inactive indicates that the Test Cycle has been closed, and it can not be edited or joined to the project until opened again.
Created By	The ID of the user that created the Test Cycle.
Create Date	The date and time when the Test Cycle was created.
Closed By	If closed, shows the ID of the user that closed the Test Cycle.
Comment	Displays any comments that were added to the Test Cycle when created or edited.

If the current project is associated with one of the listed Test Cycles, its entry in the list will be displayed in bold and italicized type.

By default, only active Test Cycles are displayed in the Test Cycle Manager. To show all Test Cycles, including those that have been closed, enable the **Include inactive** option to the right of the Test Cycles table.

Regardless of which Test Cycle is selected in the list, the details of the current project's Test Cycle are displayed in the lower portion of the window. This is the Test Cycle to which the current project is associated (joined).

The **Test Cycle** panel displays the Test Cycle name, the user who created the Test Cycle, and any comments that have been added.



The screenshot shows a window titled "Test Cycle" with a light beige background. It contains three input fields: "Name" with the value "Regression", "Created By" with the value "jidgebicki", and a larger "Comments" field which is currently empty.

The **Environments** panel displays which environments in current project are active within the Test Cycle. Results will be stored only for tests that are executed within selected (active) environments.



The screenshot shows a window titled "Environments" with a light beige background. It contains a list of environments with checkboxes: "All Environments" (unchecked), "Main" (checked), "Regression" (checked and highlighted with a blue background), and "Unit Testing" (unchecked).

By default, the **All Environments** option is enabled when a project is joined to a Test Cycle. To use only a subset of the current project's environments, disable the **All Environments** option and check the box next to one or more desired environments.

NOTE: If an environment is not enabled and is selected in Rational Integration Tester, the Test Cycle icon will indicate that the Test Cycle is currently paused.

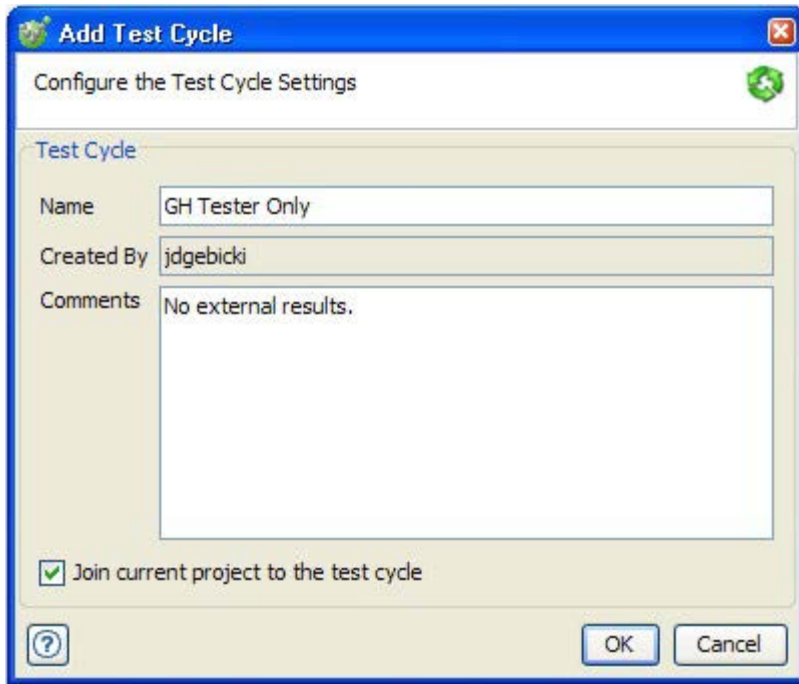
The **External test systems...** option enables or disables the storage of results for tests in the project that are executed externally (for example, in HP Quality Center, from the command line, and so on). This option is enabled by default.

Create a New Test Cycle

New Test Cycles can be created from the Test Cycle Manager.

1. If not already open, launch the Test Cycle Manager as described previously.
2. Click the **New...** button.

The **Add Test Cycle** dialog is displayed.



3. Provide a name for the new Test Cycle in the **Name** field.
4. If desired, enter any comments to be saved with the Test Cycle in the **Comments** field.

By default, the current project will be joined to the new Test Cycle upon its creation. To change this option and create the Test Cycle without joining the current project, disable the **Join current** option at the bottom of the dialog.

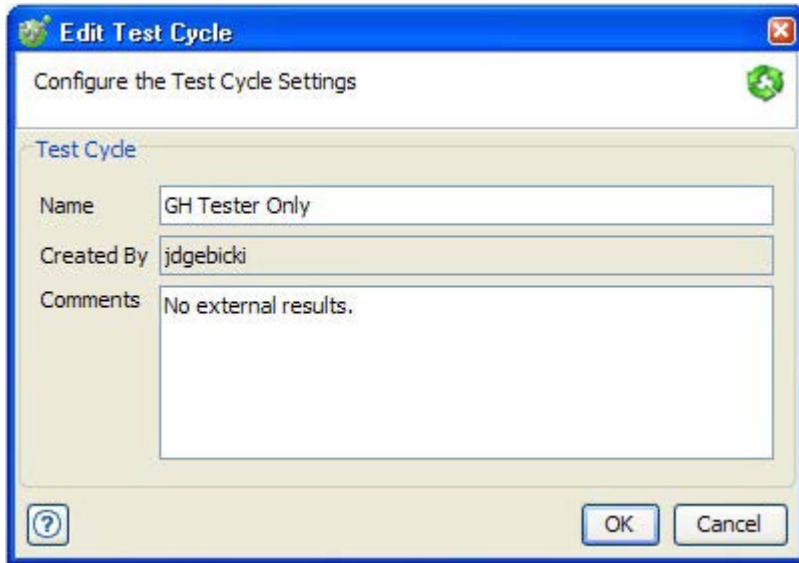
5. When finished, click **OK** to create the Test Cycle. Otherwise, click **Cancel** to return to the Test Cycle Manager without making any changes.

Edit an Existing Test Cycle

The details of an existing Test Cycle can be modified from the Test Cycle Manager.

1. If not already open, launch the Test Cycle Manager as described previously.
2. Select the Test Cycle to edit in the Test Cycles table, then click the **Edit...** button.

The **Edit Test Cycle** dialog is displayed.



3. If desired, modify the name of the Test Cycle in the **Name** field.
4. If desired, add or modify the Test Cycle comments in the **Comments** field.
5. When finished, click **OK** to save any changes and close the dialog. Otherwise, click **Cancel** to return to the Test Cycle Manager without making any changes.

Join a Test Cycle

The current project can be associated with a new or existing Test Cycle using the Test Cycle Manager.

- For information about joining a new Test Cycle, see [Create a New Test Cycle](#).
- To join an existing Test Cycle, select it in the Test Cycle Manager and click the **Join** button.

Once the current project is joined to a Test Cycle, you can specify which environments to use under the **Environments** panel in the Test Cycle Manager.

Unjoin a Test Cycle

The association between the current project and a Test Cycle can be removed using the Test Cycle Manager. To do so, simply select the project's current Test Cycle (displayed in bold and italicized type) and click the **Unjoin** button.

NOTE: The removal of the project from the Test Cycle occurs immediately for the current user. For any other users, the change will be realized the next time the project is loaded.

Close a Test Cycle

If desired, you can close an active test cycle and remove all associations to any Rational Integration Tester projects. Additionally, the closed Test Cycle will not be displayed in the Test Cycle Manager unless the **Include inactive** option is enabled.

To close a Test Cycle, select its entry in the Test Cycles table and click **Close**.

NOTE: Closing a Test Cycle will remove any associations immediately for the current user. For any other users, the change will be realized the next time the project is loaded.

Delete a Test Cycle

Inactive (closed) Test Cycles can be permanently deleted from the project database.

NOTE: If you delete a Test Cycle, all results that are stored with it will be deleted.

To delete an inactive Test Cycle, you must first enable the **Include inactive** option in the Test Cycle Manager to show closed Test Cycles. Once displayed, select the Test Cycle you want to delete and click the **Delete** button.

Reopen a Test Cycle

Inactive (closed) Test Cycles can be reopened at any time. Ensure that the **Include inactive** option is enabled in the Test Cycle Manager to show closed Test Cycles. Once displayed, select the Test Cycle you want to open and click the **Re-open** button.

Pause a Test Cycle

Users can pause the project's currently associated Test Cycle at any time, temporarily disabling Test Cycle functionality (that is, any tests executed while paused will not be stored with the Test Cycle).

To pause the current Test Cycle, right-click the Test Cycle icon in the status bar and select **Pause** from the context menu, or select **Project > Test Cycle > Pause** from Rational Integration Tester's main toolbar. A check mark will now be displayed in the context menu and in the **Project > Test Cycle** menu to indicate that the Test Cycle is paused.

If the Test Cycle is currently paused, you can resume normal functionality by selecting **Pause** again.

View Test Cycle Details

Additional Test Cycle details (that is, current and previously associated projects and joined users) can be displayed by selecting the desired Test Cycle and clicking the **Details...** button.

Test Cycle Details [GH Tester Only]

Details

No external results.

Created By: Closed By:

Creation Date: Closed Date:

Associated projects

Projects	Joined Users	
	User	Date Joined
<input type="text"/>	<input type="text"/>	<input type="text"/>

Projects no longer associated

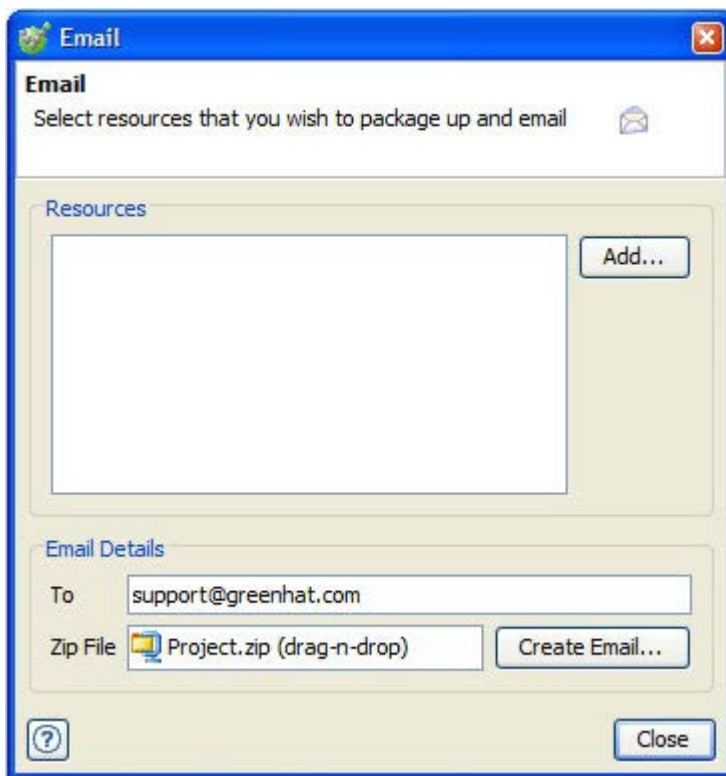
2.7 Email Project Files

Rational Integration Tester allows you to select project resources and email them to other users (for example, to have another team member execute a new test that you have created) instead of having to send the entire project or locate the project resources manually. The **Send > Email** option lets you select specific project resources and send them to one or more specified email addresses.

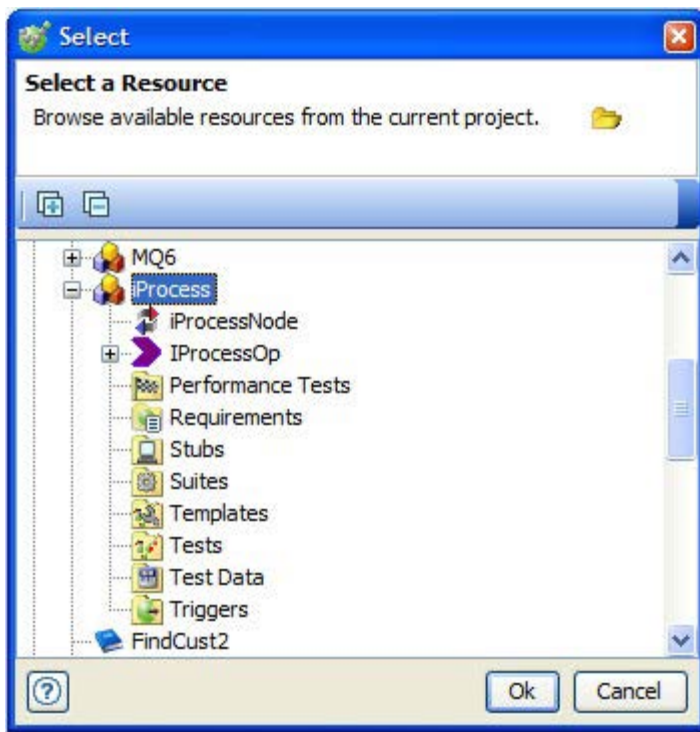
Follow the steps below to send project resources using Rational Integration Tester.

1. Select **File > Send > Email** from Rational Integration Tester's main toolbar.

The **Email** dialog is displayed.



-
2. Click **Add** to display the project resource dialog.

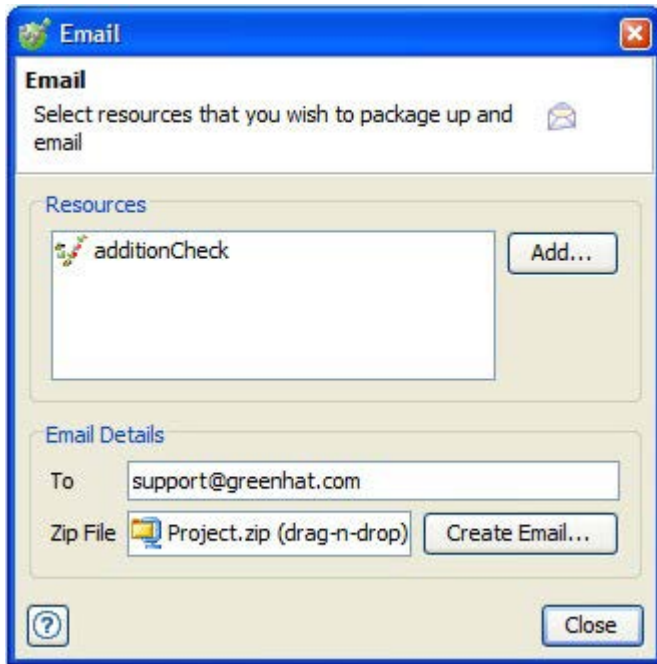


3. Select the resources you want to send (use **Shift** or **Ctrl** to add multiple resources).

NOTE: The children and other dependent resources will be added automatically for every component or operation selected. In the example shown above, everything under the **iProcess** service component will be included.

4. Click **OK** once all of the desired resources have been selected.

The **Email** dialog is displayed again and the selected resources will be listed under **Resources**.



NOTE: If desired, you can add more resources by repeating steps 2 through 4 above.

The generated .zip file will contain a Rational Integration Tester project file (*.ghp) and all of the parent and child resources that are required to use the attached resources.

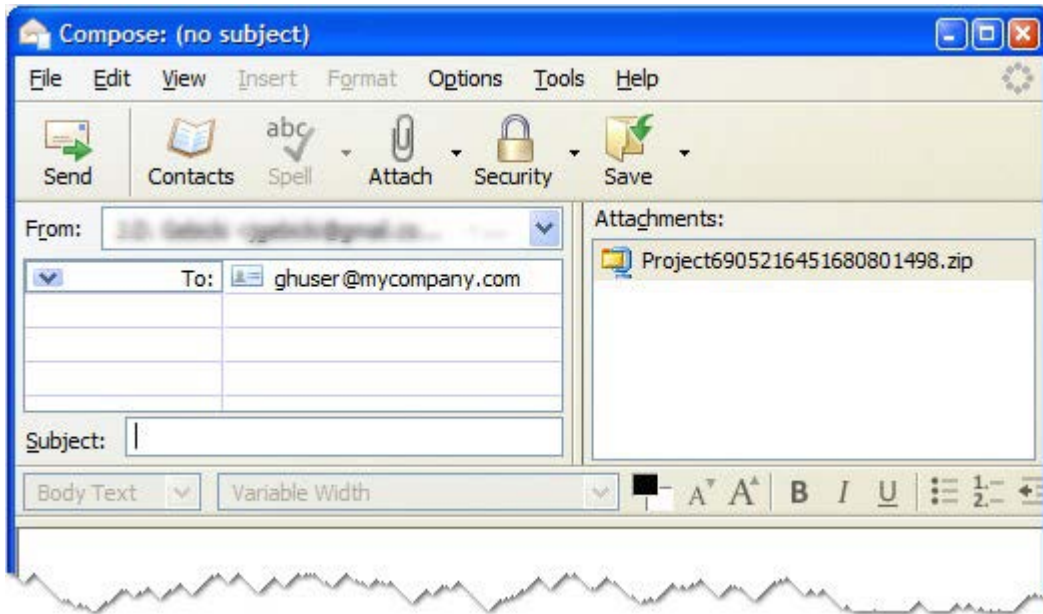
5. When all of the desired resources have been added, enter the target email addresses in the **To** field and click **Create Email**.

NOTE: If using multiple addresses, separate them with a comma and no spaces.

A new email message will be opened in your default email application (specified on your system), addressed to the specified recipients.

6. Enter a title, subject, and any desired message in the new email, then drag the attachment from the **ZipFile** field and drop it into the new email.

The generated file containing the Rational Integration Tester resources should be attached automatically to the email message.

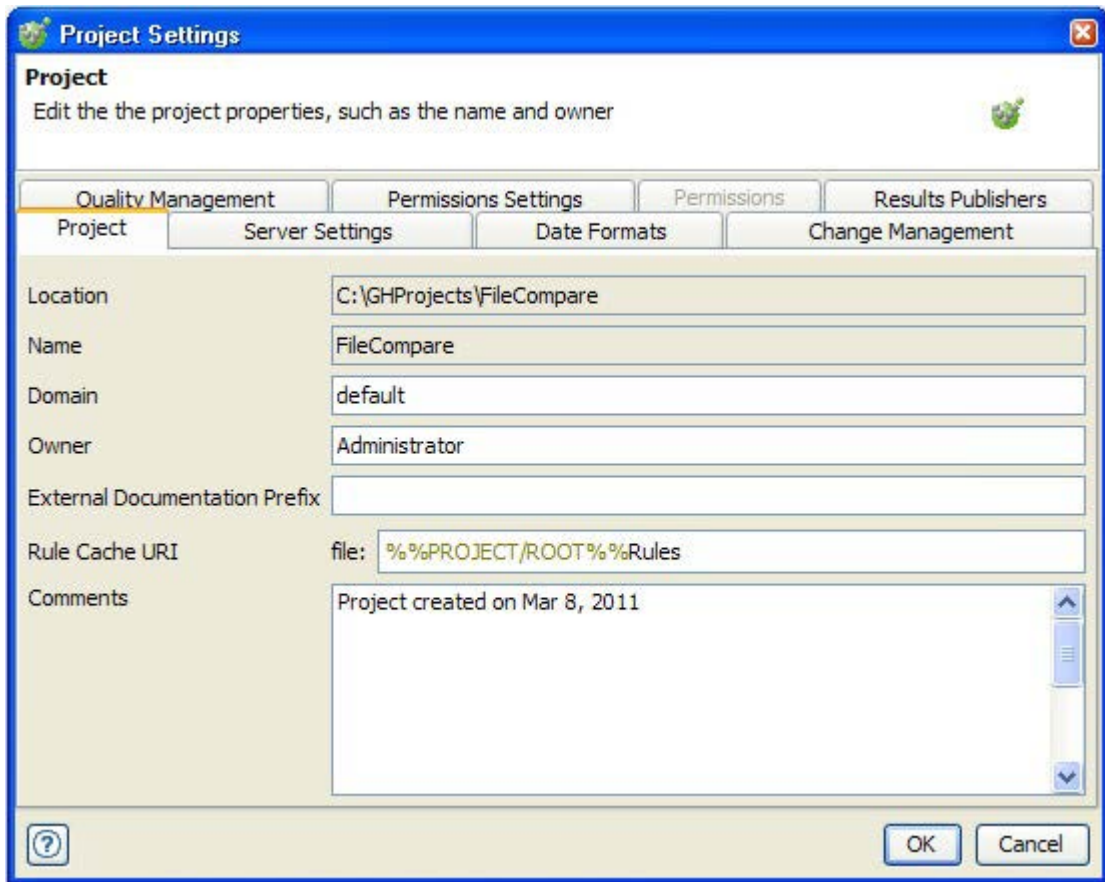


7. When ready, send the message to your intended recipients.

Users receiving the file can extract its contents into a new directory and open the project that it contains like any other Rational Integration Tester project. The project, however, will only contain the resources that were selected when sending.

2.8 Project Settings

The project settings can be viewed by selecting **Project > Project Settings** from the main Rational Integration Tester menu. The following dialog is displayed, letting you view or edit various properties and settings for the current project.



The screenshot shows the 'Project Settings' dialog box. The title bar is blue with a green globe icon and the text 'Project Settings'. Below the title bar, the word 'Project' is displayed in bold, followed by the instruction 'Edit the the project properties, such as the name and owner'. The dialog is divided into several tabs: 'Quality Management', 'Permissions Settings', 'Permissions', and 'Results Publishers'. The 'Quality Management' tab is selected and contains sub-tabs: 'Project', 'Server Settings', 'Date Formats', and 'Change Management'. The 'Project' sub-tab is active, showing a list of properties with corresponding text input fields. The properties and their values are: Location (C:\GHProjects\FileCompare), Name (FileCompare), Domain (default), Owner (Administrator), External Documentation Prefix (empty), Rule Cache URI (file: %%PROJECT/ROOT%%Rules), and Comments (Project created on Mar 8, 2011). At the bottom of the dialog, there is a help icon (question mark in a circle) on the left and 'OK' and 'Cancel' buttons on the right.

Project	
Location	C:\GHProjects\FileCompare
Name	FileCompare
Domain	default
Owner	Administrator
External Documentation Prefix	
Rule Cache URI	file: %%PROJECT/ROOT%%Rules
Comments	Project created on Mar 8, 2011

2.8.1 Project Properties

The **Project** tab contains summary information about the current project.

The screenshot shows a dialog box with four tabs: 'Project' (selected), 'Server Settings', 'Date Formats', and 'Change Management'. The 'Project' tab contains the following fields:

Field	Value
Location	C:\Documents and Settings\jedgebicki\Defects
Name	Defects
Domain	default
Owner	Administrator
External Documentation Prefix	
Rule Cache URI	file: %%%PROJECT/ROOT%%Rules
Comments	Project created on Oct 8, 2010

The fields in the **Project** tab are described in the following table:

Location	The full path where the project can be found.
Name	A meaningful name to help identify the project.
Domain	The default domain name to use when publishing stubs to Rational Test Control Panel (RTCP).
Owner	The project's creator/owner
External Documentation Prefix	The root directory where external documentation for Rational Integration Tester artifacts is stored.
Rule Cache URI	The location where validation rules are stored for the project (%%PROJECT/ROOT%%Rules, by default).
Comments	User-defined comments that can be used to enter information that might be relevant to the project.

2.8.2 Project Database

The **Server Settings** tab manages JDBC connection details for the project database, which can be specified when creating the project, as well as the Results Server URL and the Rational Test Control Panel (RTCP) URL.

The screenshot shows a dialog box with four tabs: **Project**, **Server Settings** (selected), **Date Formats**, and **Change Management**. The **Server Settings** tab is divided into three sections:

- Database Server**:
 - Database Provider**: A dropdown menu with "Oracle" selected.
 - Connection Details**: A group box containing:
 - Database URL**: `jdbc:oracle:thin:@localhost:1521:XE`
 - User Name**: `ghtester51924`
 - Password**: A field with 10 dots.
 - Schema version required**: `1.9.24.c`
 - Test Connection**: A button.
- Other Servers**:
 - Results Server**: `http://ghregression:8080/ResultsServer`
 - GH Server**: `http://localhost:8080/GHServer`

Below the **Database Server** section is a table with two columns: **Property** and **Value**. The table is currently empty.

Use the fields under **Database Server** to create or modify your database connection, select the provider and enter the required connection details (specific to where and how the database has been set up for Rational Integration Tester).

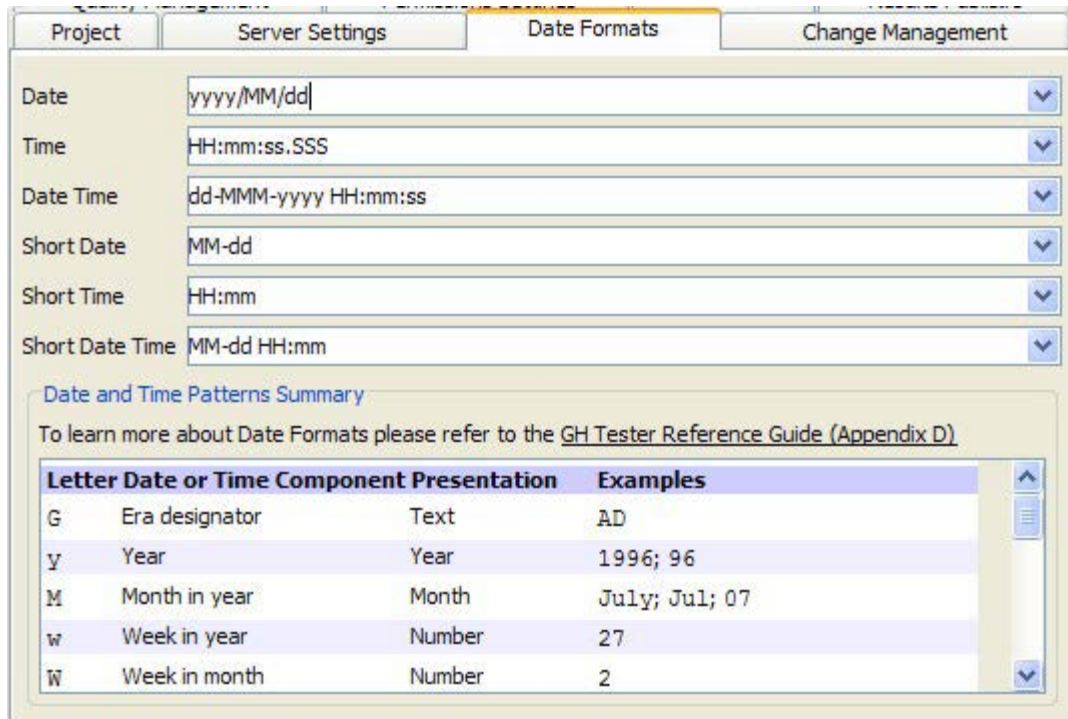
To test the provider and connection settings, click **Test Connection**. If the test is successful, no further configuration is required. If the test is unsuccessful, you will need to verify/modify the connection settings and try again.

Under **Other Servers**, specify the URLs for the Results Server and Rational Test Control Panel. The Results Server URL (`http://localhost:7172/ResultsServer`, by default) is used to create links to detailed execution results for test suites that are run from test sets in HP Quality Center (for more information, refer to *IBM Rational Integration Tester Integration Guide for HP Quality Center*). The Rational Test Control Panel URL (`http://localhost:7819/RTCP`, by default) should point to the base URL of the Rational Test Control Panel instance that will

run published stubs and manage scheduled execution (for more information, refer to *IBM Rational Test Virtualization Server Reference Guide*).

2.8.3 Date Formats

The **Date Formats** tab lets you define the output format of certain date and datetime tags in Rational Integration Tester (for example, TEST / START_DATE).



The screenshot shows the 'Date Formats' tab in the Rational Integration Tester settings. It contains several input fields for different date and time formats, each with a dropdown arrow. Below these fields is a 'Date and Time Patterns Summary' section, which includes a link to the 'GH Tester Reference Guide (Appendix D)' and a table of date and time components.

Letter	Date or Time Component	Presentation	Examples
G	Era designator	Text	AD
y	Year	Year	1996; 96
M	Month in year	Month	July; Jul; 07
w	Week in year	Number	27
W	Week in month	Number	2

For more information, see [Appendix B: Date & Time Patterns](#).

2.8.4 Change Management

The **Change Management** tab is used to manage change management systems that can be used with Rational Integration Tester. Currently, Rational Integration Tester supports integration with OSLC compliant change management providers and Atlassian's JIRA for issue and project tracking.

The integrations that are defined here can be accessed directly from Rational Integration Tester to raise defects in the context of an executed test or test suite.

The screenshot shows the 'Change Management' tab in the Rational Integration Tester interface. It contains two main sections: 'Change Management Integrations' and 'Change Management Integration Details'.

Change Management Integrations: This section displays a list of integrations. One integration is listed: 'GH RTC [OSLC CM Integration Provider] (default)'. To the right of the list are three buttons: 'Add', 'Remove', and 'Set Default'.

Change Management Integration Details: This section contains four input fields for configuring the selected integration:

- Name:** GH RTC
- Root Services URI:** https://localhost:9443/jazz/rootservices
- Username:** ghuser
- Password:** (masked with dots)

To add an integration, click **Add**. In the **Select Provider** dialog, select one of the supported providers and click **OK**.

The screenshot shows the 'Select Provider' dialog box. It has a title bar with a close button. Inside, there is a label 'Available Providers:' followed by a dropdown menu. The dropdown menu is currently showing 'OSLC CM Integration Provider'. At the bottom of the dialog are two buttons: 'OK' and 'Cancel'.

OSLC Providers

For OSLC providers, enter the following configuration details:

- **Name** – a descriptive name for the selected OSLC instance, used to identify the integration in a meaningful way when creating defects.
- **Root Services URI** – a URI that points to the root services document on the selected integration from which service discovery will take place.
- **Username and Password** – the user name and password to be used when accessing protected resources on the selected provider and when creating defects.

When you are finished, click **OK** to save the changes and close the **Project Settings** dialog.

Atlassian JIRA Providers

For Atlassian JIRA providers, enter the following configuration details:

- **Name** – a descriptive name for the selected JIRA instance, used to identify the integration in a meaningful way when creating defects.
- **Base URL** – specifies a base URL for all relative links on the JIRA server (this should match the Base URL that is set in the server configuration).
- **Username and Password** – the user name and password to be used when accessing protected resources on the selected provider and when creating defects.

To verify the configuration details, click **Connect**. If you are unable to connect to the selected instance, verify the provided configuration details and try again.

NOTE: The selected JIRA server must permit remote access by Rational Integration Tester. This is achieved by enabling the “Accept remote API calls” setting under **General Configuration, Options** in the server configuration.

Once you have successfully connected to the specified JIRA instance, you can set the following default options to use when creating defects.

- **Default Project ID** – select one of the available projects that have been configured on the selected instance.
- **Default Issue Type** – select one of the available issue types on the selected instance.
- **Default Priority** – select one of the available priorities on the selected instance.

When you are finished, click **OK** to save the changes and close the **Project Settings** dialog.

2.8.5 Quality Management Settings

The **Quality Management** tab is used to manage quality management systems that can be used with Rational Integration Tester. Currently, Rational Integration Tester supports integration with IBM Rational Quality Manager.

The screenshot shows a dialog box titled "Quality Management Settings" with four tabs: "Quality Management", "Permissions Settings", "Permissions", and "Results Publishers". The "Quality Management" tab is active. It contains two main sections: "Quality Management Integrations" and "Test Management Integration Details".

The "Quality Management Integrations" section features a list box with one entry, "Local [IBM Rational Quality Manager]", which is highlighted. To the right of the list box are three buttons: "Add", "Remove", and "Set Default".

The "Test Management Integration Details" section contains five labeled text input fields:

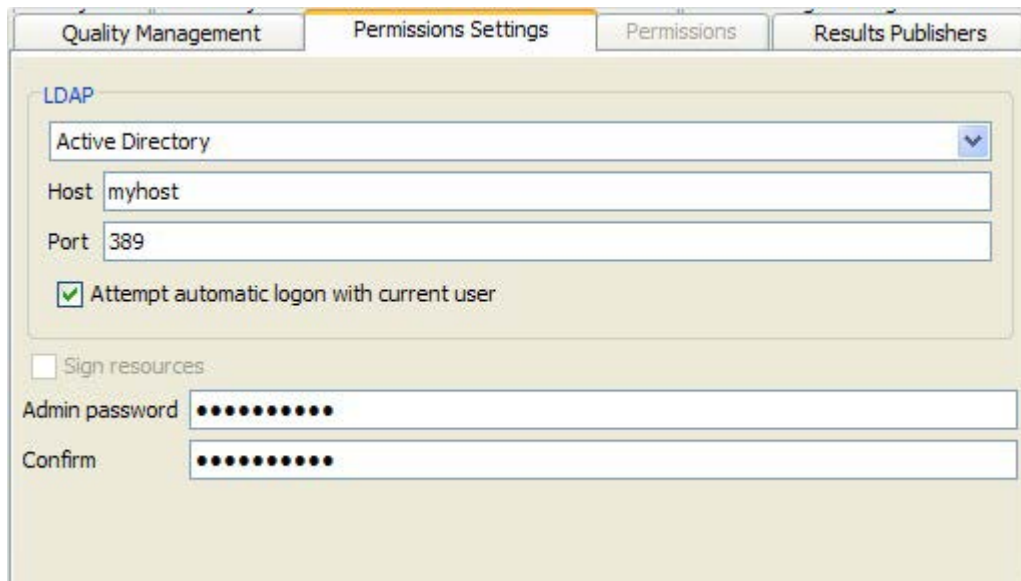
- Name: Local
- URL: http://gh-local:9080/jazz
- Project: Regression
- Username: ghuser
- Password: (masked with seven dots)

For specific details about how to configure the integration between Rational Quality Manager and Rational Integration Tester, refer to *IBM Rational Integration Tester Integration Guide for IBM Rational Quality Manager*.

2.8.6 Permissions Settings

The **Permissions Settings** tab is used to manage whether or not LDAP permissions are enabled for the project, host information for LDAP, and the password for the project administrator.

NOTE: Once permissions have been enabled on a project, only the project administrator has access to the **Permissions Settings** tab.



The screenshot shows a dialog box titled "Permissions Settings" with four tabs: "Quality Management", "Permissions Settings" (selected), "Permissions", and "Results Publishers". The "LDAP" section is expanded, showing a dropdown menu set to "Active Directory". Below this are text fields for "Host" (containing "myhost") and "Port" (containing "389"). A checkbox labeled "Attempt automatic logon with current user" is checked. Below this is an unchecked checkbox labeled "Sign resources". At the bottom are two password fields: "Admin password" and "Confirm", both containing masked characters (dots).

To enable permissions on the project, select the LDAP provider (currently Active Directory is the only option), then specify the host name or IP address and port of the LDAP server to use.

If desired, you can have Rational Integration Tester attempt to log in to each LDAP-enabled project using the ID of the current user.

Finally, you can add or modify the password of the project administrator by entering it in the **Admin password** and **Confirm** fields. The password entered in each field must match.

NOTE: A blank password is permitted, but it is not recommended.

2.8.7 Permissions

The **Permissions** tab is used to manage which LDAP users have access to the project and the areas of the project to which they should have access.

NOTE: Only the project administrator has access to the **Permissions** tab.

Quality Management | Permissions Settings | **Permissions** | Results Publishers

Group or user names:

Type	Name
	John D. Gebicki
	Peter DiStefano

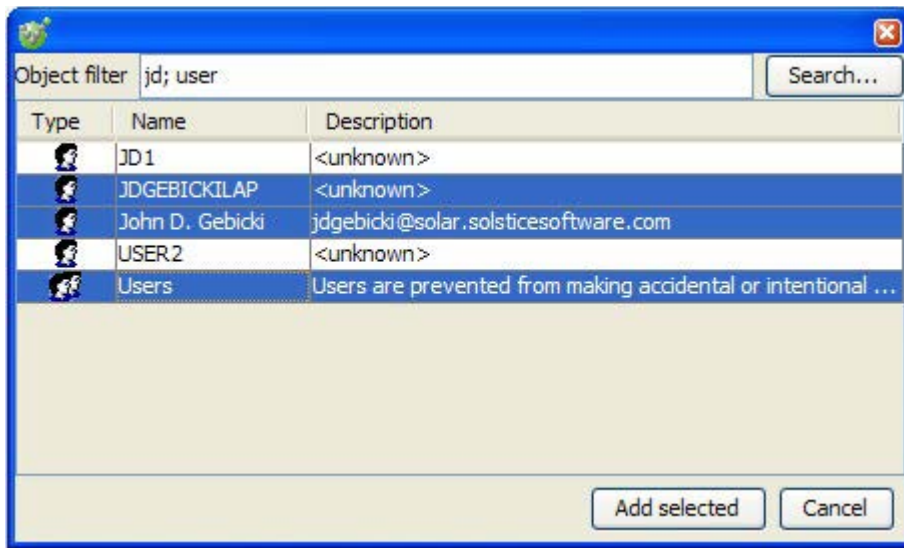
Actions	Allow	Deny
<input type="checkbox"/> Create		
Environment	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Delete		
Resource	<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input type="checkbox"/> Modify		
Environment	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> View		
<input type="checkbox"/> Perspective		
Architecture School	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Requirements Library	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Recording Studio	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Test Factory	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Test Lab	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Results Gallery	<input checked="" type="checkbox"/>	<input type="checkbox"/>

NOTE: A valid domain login must be supplied before the admin user will have access to the list of available users and groups. You will be prompted to login when attempting to add users, or you can log into the project as a different user by selecting **Switch user...** from the **Project** menu in Rational Integration Tester.

Users can be added to the project by clicking **Add...** and locating the users in the search dialog that is displayed.

Enter a search string (multiple strings can be used, separated by semicolons) and click **Search**. To add any of the located users or groups to the project, select them and click **Add Selected**.

NOTE: Users or groups that have already been added to the project will not be displayed in the filtered list.



For users or groups that have been added to the project, permission to perform certain actions must be explicitly allowed or denied using the appropriate check box. Permissions can be set for one or multiple users at the same time.

NOTE: To quickly allow or deny all available actions for the selected users or groups, right-click anywhere within the Actions table and select **Allow All** or **Deny All** from the context menu.

To remove a user or group from the project, select its entry in the current list and click **Remove**.

2.8.8 Results Publishers

The **Results Publishers** tab is used to manage results publishers that can email test suite results or publish results to a configured CentraSite server. Details about configuration within a test suite can be found in [Test Suites](#). For more information about results publishers for CentraSite, refer to *IBM Rational Integration Tester Integration Guide for Software AG CentraSite*.



For each publisher in the list, the following details are displayed:

- **Name** - the name given to the publisher when it was created (the name can not be modified once the publisher has been created) and the publisher type
- **Default** - indicates which publisher (if any) has been designated as the default for all test suites (the default publisher can not be set here, but it can be removed)
- **Enabled** - indicates whether or not the selected publisher is enabled for use in test suites

Click **Add** to create a new publisher, click **Edit** to modify the configuration of the selected publisher (see [Creating a New Email Publisher](#) for more information on publisher configurations), or click **Remove** to delete the selected publisher.

To modify the publication settings of the default publisher, select it and click **Edit Defaults** (see [Add or Modify Results Publisher Settings](#) for more information).

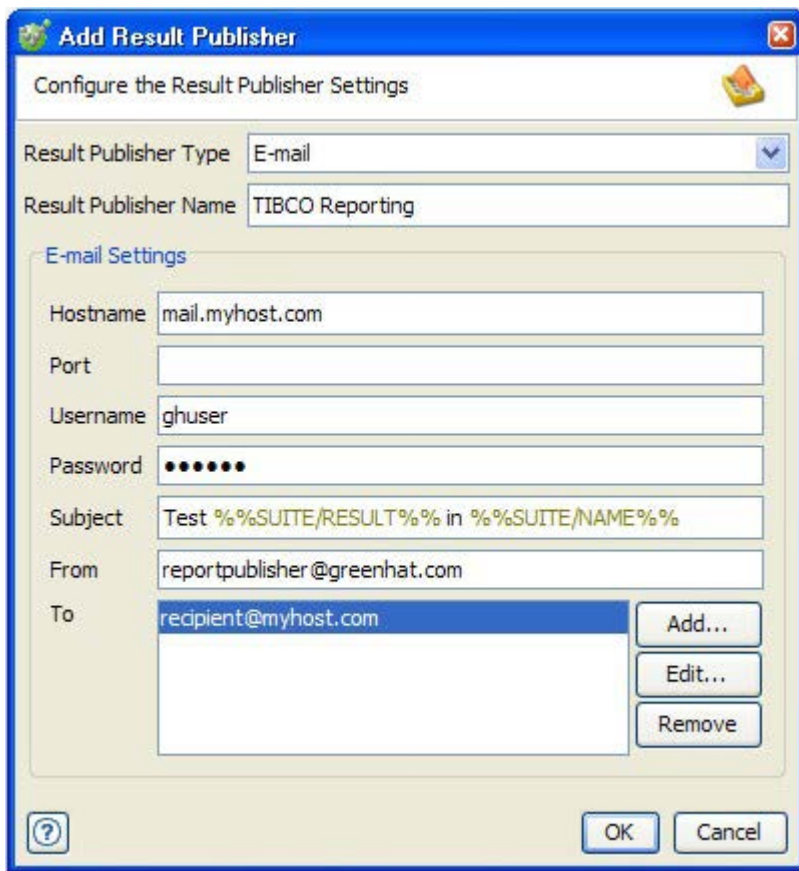
To remove the “default” designation of the default publisher, select it and click **Remove Defaults** – this will not delete the publisher. If you want to designate a new default publisher, it must be done in the test suite editor.

Publishing can be enabled or disabled on a project-wide basis by toggling the results publishing icon Rational Integration Tester's main toolbar.



Creating a New Email Publisher

The **Add Result Publisher** dialog is used to create a new publisher.



Select **E-mail** as the publisher type from the **Result Publisher Type** combo box and provide a name for the publisher in the **Result Publisher Name** field (only alpha-numeric characters, hyphens, and underscores are permitted).

In the **Hostname** field, enter the host name or IP address of the mail server and enter the SMTP port in the **Port** field (port 25 is used by default). If the server requires authentication for sending, enter the user name and password of a valid email user.

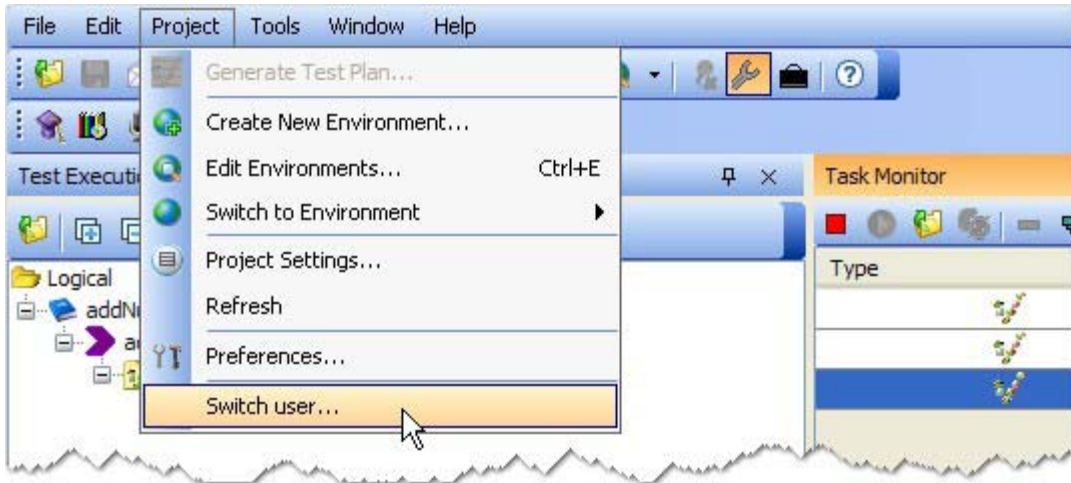
A default subject is supplied in the **Subject** field, which can be changed if desired (tags are supported). In the From field, enter the email address to use as the sender when results are published by Rational Integration Tester.

Recipients for the new publisher are managed in the **To** field. To add one or more recipients, click **Add** and enter one or more email addresses in the **Add Recipient** dialog (multiple addresses separated by a semi-colon can be entered or pasted). To modify an existing recipient, select it and click **Edit**. To remove a recipient from the list, select it and click **Remove**.

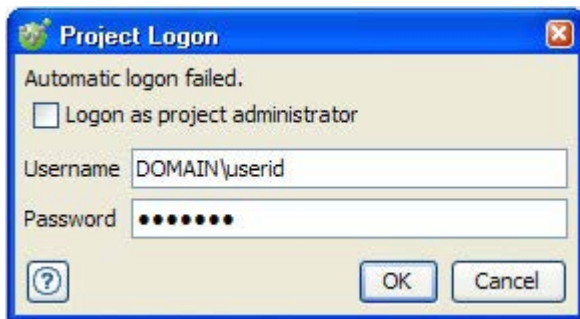
2.9 Switching Users on a Project

If the current project was created or configured to use permissions (see [Creating a New Project](#) or [Project Settings](#)), you can log into the project as a different user.

1. Select **Switch user...** from Rational Integration Tester's **Project** menu.



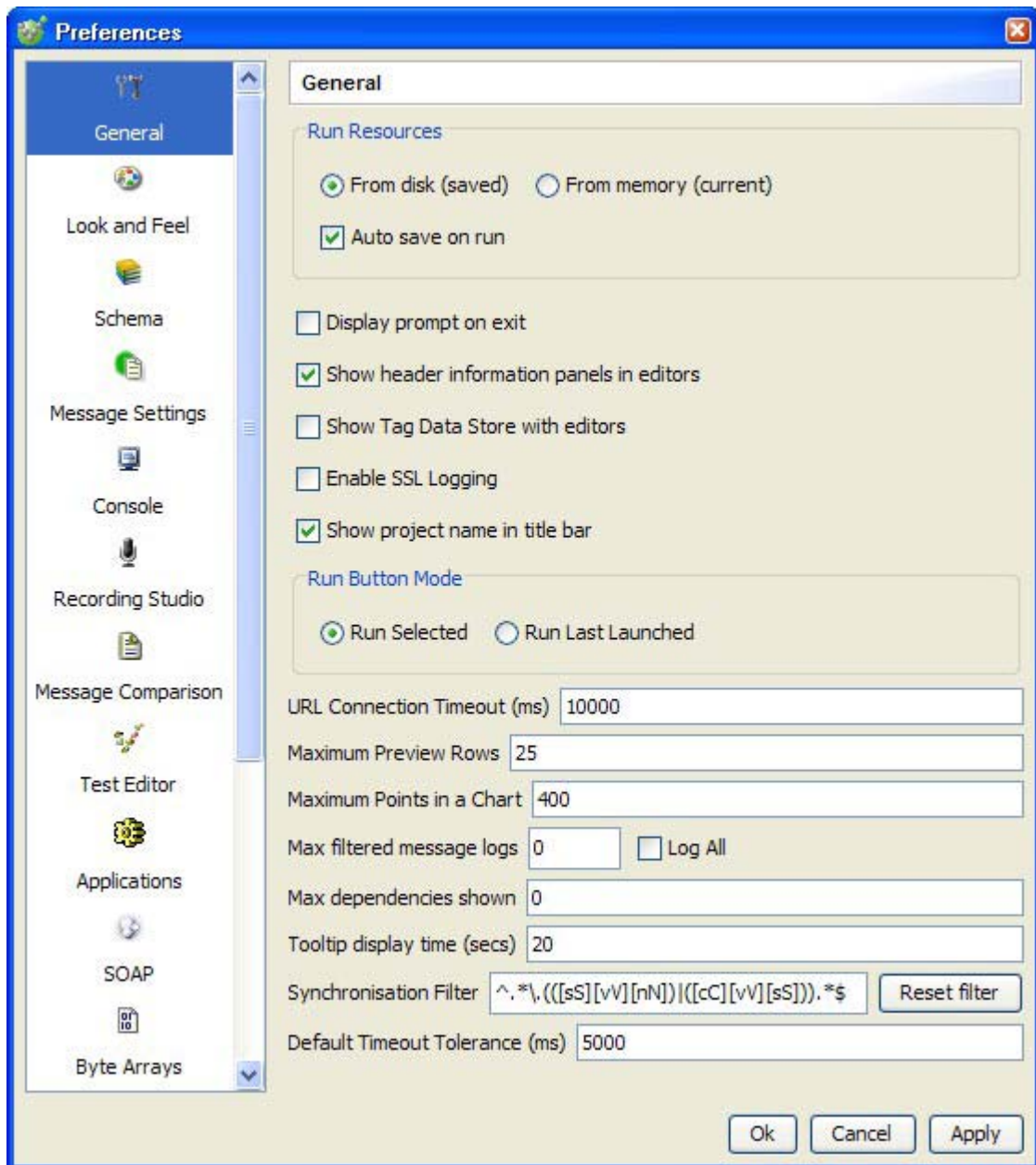
The **Project Logon** dialog is displayed.



2. Enter a valid domain user/password combination, or enable the “Logon as project administrator” option and enter the admin password, then click **OK**.

2.10 Changing Preferences

Much of the behavior exhibited by Rational Integration Tester can be customized in the application preferences. To view or modify these preferences, select **Window > Preferences** in the main Rational Integration Tester menu.



2.10.1 General

General preferences control the behavior of Rational Integration Tester on the whole, as follows:

Run Resources	Controls how test resources are executed. From disk (saved) will execute the last saved version of the resource. From memory (current) will execute the resource in its current state, including any unsaved changes. If Auto save on run is enabled (when From disk is in use), test resources will be saved automatically when executed.
Display prompt on exit	Asks the user to confirm whether or not to exit Rational Integration Tester.
Show header information panels in editors	In message editors, displays a description panel in the headers area (once you are more familiar with Rational Integration Tester, these can be disabled to save screen space).

Show Tag Data Store with editors	Displays the Tag Data Store automatically whenever a message editor is opened.
Enable SSL Logging	Enables or disables SSL logging for use in Transport Diagnostics . NOTE: When enabled, any tests that use SSL will run more slowly due to the overhead of additional logging. NOTE: If changing this option, you must restart Rational Integration Tester before the change takes effect (a dialog is displayed asking you to save changes and restart or to restart later).
Show project name in title bar	Toggles the display of the current project name in the Rational Integration Tester title bar.
Run Button Mode	Controls the behaviour of the Run button and F5: Run Selected always runs the selected test resource; Run Last Launched always runs the test resource that was executed last.
URL Connection Timeout	Specifies a default timeout to use for connections in test actions.
Maximum Preview Rows	The maximum number of rows to display in previews (for example, when importing a WSDL).
Maximum Points in a Chart	The maximum number of points to plot in charts (for use with performance testing).
Max filtered message logs	Specifies the maximum number of messages to log when using a filter on a message. To log all messages, enable the Log All option.
Max dependencies shown	Specifies the maximum number of dependencies to display at one time in Architecture School's logical view.
Tooltip display time (secs)	Specifies the amount of time (in seconds) that tooltips should be displayed when the cursor rests on an applicable icon. If left empty or if an invalid entry is set, the default value (that is, four seconds) is used. If set to zero, tooltips will not be displayed at all.
Synchronisation Filter	Specifies a filter that will ignore certain items found within a synchronisation source (currently only applies to TIBCO BusinessWorks projects). The filter string can be modified as needed. Click Reset filter to restore the default value, which ignores <code>.svn</code> and <code>.cvs</code> folders (that is, for projects in SVN or CVS repositories).
Default Timeout Tolerance	The default timeout tolerance (in milliseconds) to use for Receive Reply and Subscribe actions.

2.10.2 Look and Feel

Look and Feel preferences let users specify which fonts (**Fixed width** and **Proportional**) to use throughout Rational Integration Tester. Additionally, the colour used for highlighting null fields in data sources can be set.

Fonts	Provides combo boxes for selecting an available fixed width or proportional system font to use in Rational Integration Tester. Below each selection is a Sample field that displays a sample of text in the selected font.
Colours	Controls the Color for null values option, which specifies the color to use for highlighting fields treated as null when previewing the contents of a file or Excel data source (see Test Data Sources). To change the current color, click on it and select a new color using the Select color dialog that is displayed.

NOTE: If the proportional font is modified, Rational Integration Tester must be restarted for the change to take effect. After closing the Preferences dialog, you will be prompted to restart now or restart later. If you restart later, the proportional font changes that you made will not be immediately apparent in Rational Integration Tester.

2.10.3 Schema

Schema preferences control how Rational Integration Tester works with schemas, as follows:

Analysis Settings	Controls the default selections to use for schema analysis (see Analysing Schemas), which can be overridden during analysis.
Cache	Displays the number of files currently stored in Rational Integration Tester's schema cache, as well as the cache's current size. To clear the cache (for example, to save disk space), click the Clear button.
Rebuild when saving resource	Controls how Rational Integration Tester will rebuild referenced schemas when saving resources (for example, a SOAP message).
Schema Library	Specifies whether to save schemas automatically (that is, when clicking away from the schema after changes have been made).
Record Layouts	Controls the appearance of grouping rows in record layouts (see Record Layouts), allowing you to change the background color and the color of the text used in such rows.

2.10.4 Message Settings

Message Settings preferences let you enable or disable default options to be applied when working with messages, as follows:

- **Message Expansion Level** controls how many levels of a message to expand in editors (used only when the editor is opened for the first time). A value of “0” will expand all nodes in the message.
- The **Content** options control the default settings to use when creating messages:
 - **Include Text Nodes** will add text nodes to message elements automatically when creating a message.
 - **Include Optional Fields** will add all optional fields to a message according to the schema in use.
- The **Assert** options set default validation options when creating a new message:
 - **Accept fields in any order** controls whether or not the order of message fields should be considered during validation.
 - **Ignore missing fields in received message** controls whether or not missing fields in the received message will cause a validation error.
 - **Ignore additional fields in received message** controls whether or not additional fields in the received message will cause a validation error.
 - **Enable validation of time based fields** controls whether or not time based fields (which are likely to be different) should be considered during validation.

2.10.5 Console

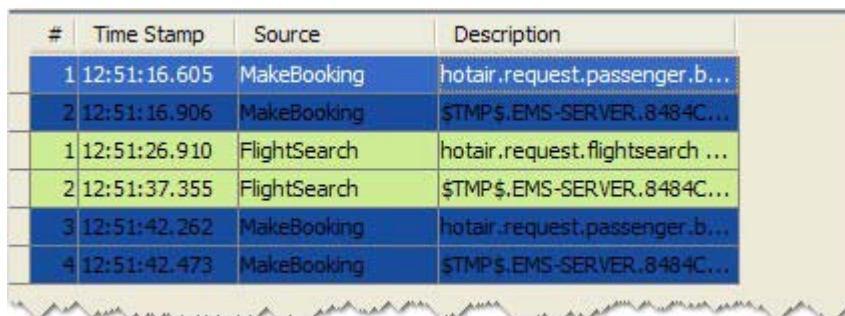
Console preferences control the appearance and behavior of the output console in the Test Lab.

Success, Error, Information color	Controls the colours used to indicate message status in the console. By default, green is the color of success messages, red is used for errors, and blue indicates information. To modify the default colors, click on an existing colour and select a new one in the Select color dialog.
Retain Message Context Data	Enables or disables the selection of console output (where validation errors have occurred) to invoke the message comparison tool.
Enable Debugging Output	Enables or disables debug messages in the console when running a test. When enabled, additional details of the various actions processed in the test are provided.
Action Prefix	Determines if the Technical or Business view descriptions are displayed in the console for test step action names.
Console Caching	When enabled, users can specify that console output for no more than the selected number of tests be cached by Rational Integration Tester. NOTE: Caching will be disabled if no project database provider has been specified.

2.10.6 Recording Studio

Recording Studio preferences let you control how the sequence of recorded events are numbered in the Recording Studio.

- **Unified counter** will number events in the order that they are received, regardless of the monitor that recorded them (for example, six events from two different monitors will be numbered 1, 2, 3, 4, 5, 6).
- **Counter per monitor** will number events in the order that they are received for each monitored operation (that is, six total events from two different monitors could be numbered 1, 2, 1, 2, 3, 4, as shown below).



#	Time Stamp	Source	Description
1	12:51:16.605	MakeBooking	hotair.request.passenger.b...
2	12:51:16.906	MakeBooking	\$TMP\$.EMS-SERVER.8484C...
1	12:51:26.910	FlightSearch	hotair.request.flightsearch ...
2	12:51:37.355	FlightSearch	\$TMP\$.EMS-SERVER.8484C...
3	12:51:42.262	MakeBooking	hotair.request.passenger.b...
4	12:51:42.473	MakeBooking	\$TMP\$.EMS-SERVER.8484C...

2.10.7 Message Comparison

Message Comparison preferences control the appearance and behavior of the Message Differences window.

Colours	Controls the colours that are used to indicate difference types or ignored nodes. To modify a selected colour, click on it and select a new one in the Select color dialog.
Reset to Colour Scheme	Click one of the buttons to set all colours to the Pastel or Vibrant scheme.
Navigation	Controls whether or not looping is enabled in the Message Differences window. If Do not loop when navigating differences is enabled, the Next difference and Previous difference buttons will be disabled when the last or first difference is selected (that is, you can not loop back to the beginning or end of the differences). If this option is disabled, the buttons are always enabled and they will loop to the beginning or end of the message to view the next or previous difference. Both buttons will be disabled if there are no differences between the messages.
Always retain the rules	Sets the default behavior when overwriting an expected message (that is, to repair a validation failure) that uses validation rules (see Overwriting Messages that Use Validation Rules for more information).
Always apply changes without prompting	If enabled, any changes made to the expected message in the Message Differences window will be applied without prompting when the window is closed. If disabled, the user will be prompted to save/apply changes, discard any changes, or cancel the action and return to the window.

2.10.8 Test Editor

Test Editor preferences control the display color for missing business view descriptions, the rendering of comments within a test, and the behavior of the test/stub action editor (inline or popup).

Missing business descriptions color	Controls the color to use for a test name or test step that has not been customized in the Business View. To modify the default color, click on the existing color and select a new one in the Select color dialog.
-------------------------------------	--

Render comments as HTML 2.0	Controls whether or not the text within new comment actions should be rendered as HTML, by default. This can be overridden on a per-comment basis when such actions are created.
-----------------------------	--

Action Editor Settings

Action Editor	Enables and sets the position of the inline Action Editor (for editing the selected action in a test or stub below or to the right of the editing panel). If set to Popup , test and stub actions can be edited by double-clicking them.
---------------	---

Apply changes automatically	When enabled, changes made in the Action Editor are applied automatically. If disabled, changes must be applied or discarded when clicking away from the item that is currently open in the Action Editor.
-----------------------------	--

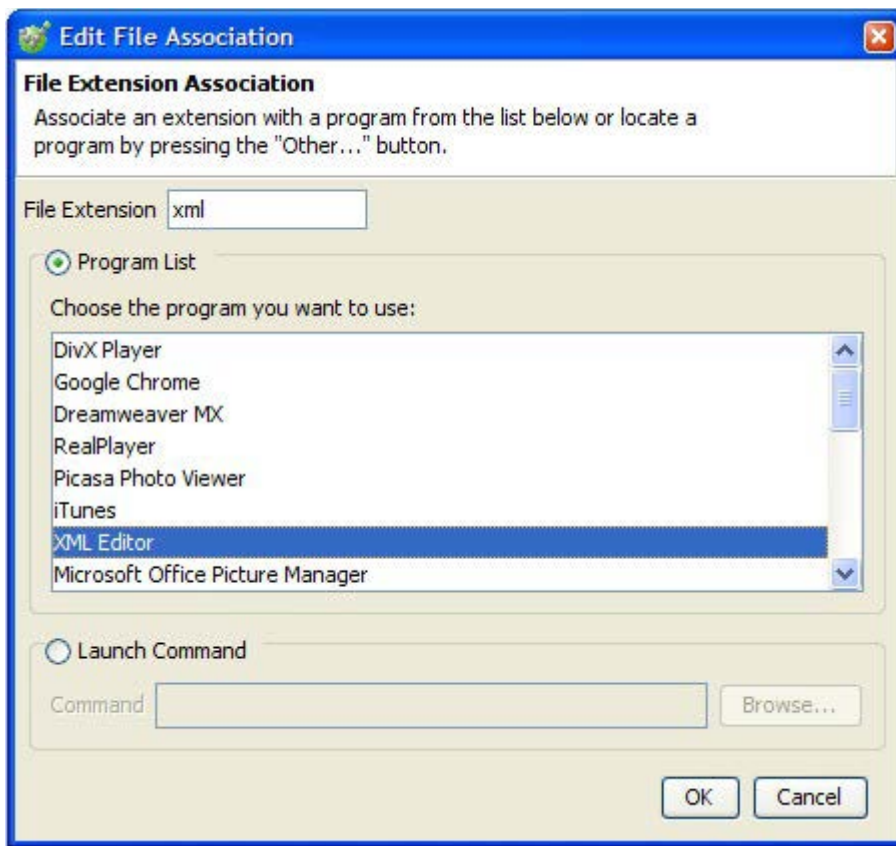
2.10.9 Applications

Applications preferences let you control how files can be opened and edited by external applications from within Rational Integration Tester.

Add a File Association

1. Click **Add** to add a new file association.

The **Edit File Association** dialog is displayed.



2. Enter the file extension to associate in the **File Extension** field.
3. Select the program to associate under **Program List**, or select **Launch Command** and enter the full path to the associated program (click **Browse** to locate the program).

Remove a File Association

1. Select the desired file association in the list of those available.
2. Click **Remove**.

Edit a File Association

1. Select the desired file association in the list of those available.
2. Click **Edit**.
3. Follow the steps listed under [Add a File Association](#).
4. Click **Add** to add a new file association.

2.10.10 SOAP

SOAP preferences control the default options in the Field Properties dialog for SOAP messages.

SOAP Version	Select the default SOAP version (1.1 or 1.2) to apply in the Field Properties dialog.
SOAP Headers	Controls whether or not SOAP headers should be enabled by default in the Field Properties dialog.

2.10.11 Byte Arrays

Byte Arrays preferences let you set the default encoding to apply to byte arrays, either UTF-8 or UTF-16.

2.10.12 Copybook

Copybook preferences control the default behavior for handling Copybook schema files, including Schema Processing, File Serialisation, and Processing Properties.

2.10.13 Files

File preferences control the default retry interval to use in when a subscriber is configured to use the File transport. For more information, refer to *IBM Rational Integration Tester Reference Guide for Files*.

2.10.14 WS -* Extensions

WS-* Extensions preferences control whether or not WS-Addressing and/or WS-Security extensions are enabled (by default) in the Field Properties editor for SOAP messages. Both options can be enabled or disabled on a per-message basis.

WS-Addressing	Enables or disables WS-Addressing extensions in the Field Properties editor, by default.
WS-Security	Enables or disables WS-Security extensions in the Field Properties editor, by default, as well as whether or not millisecond precision for timestamp tokens (within WS-Security) should be enabled by default.

2.10.15 XML

XML preferences let you control the default options used by Rational Integration Tester when processing XML.

File Serialisation

Encoding The default encoding type for serialising XML, either UTF-8 or UTF-16.

Processing Properties

Formatting The default formatting for XML, either Single-line or Multi-line indented.

Encoding The default encoding to use when processing XML, either UTF-8, UTF-16, or ISO-8859-1.

Normalise the document Enables or disables the normalization of empty text nodes to a single space character, by default.

Send NULL values Enables or disables the sending of XML elements whose text contains only a Tag, the value of which is NULL (not empty string ""). If **Send NULL values** is enabled, this element would be sent. If **Send NULL values** is not enabled, this element would not be sent.

Include XML Declaration Enables or disables the inclusion of the XML declaration, by default.

Treat CDATA as Text Enables or disables the treatment of CDATA fields as text, by default.

Default xml:space to preserve Enables or disables the preservation of white space in XML.

XML Name Matching

Qualified Name Enables the use of namespace prefixes during message comparison.

Local Name Enables local name matching (that is, ignores namespace prefixes) during message comparison.

Message Editing Preview

Depth Sets the default number of levels to be expanded in XML previews.

Length Sets the default message field length to display in XML previews.

Architecture School

Contents

Overview

The Logical View

The Physical View

Synchronisation

The Schema Library

The Rule Cache

Environment Tasks

This chapter provides an overview of Rational Integration Tester's Architecture School perspective, including details about how to build a system under test using the logical, physical, and synchronisation views. Details are also provided about using the Schema Library to manage or view the schema's available in your project.

3.1 Overview

Rational Integration Tester takes a service oriented approach to testing. Using this approach, business logic is modeled as one or more services, each one containing a number of operations with which the system can interact.

Architecture School is Rational Integration Tester's initial view (or perspective). In this perspective, users (that is, Architects) build the system to be tested in a simple, graphical fashion. The Architect defines the logical and physical system components and their relationships to one another.

NOTE: Logical components are bound to physical components using one or more environment.

When a new project is opened in Rational Integration Tester, an empty palette is presented in Architecture School. How the system is built may vary, but it will probably follow the same basic steps – create a service component, add operations to the service component, add infrastructure components, create physical components, create environments and bindings, and so on

Rational Integration Tester considers the following items when building a system under test:

- **Service Components** are named components that expose one or more operations to be tested. A service component can be simple (that is, has no child components) or composite (that is, has one or more child components).
- **Infrastructure Components** are named components (for example, a database, a JMS domain, an HTTP connection, and so on) that can be bound to physical resources by means of an environment. Infrastructure components do not expose any testable operations.
- **Operations** are interfaces that define a service component's functionality. An operation is defined by a Message Exchange Pattern (MEP) that describes the message pattern (Publish or Request/Reply) required by the operation's transport to establish communications.
- **Physical Resources** are testable resources within the enterprise (for example, a database or a web server) that represent a single configuration of an infrastructure component. The binding between the infrastructure component and the physical resource is configured in one or more environments.
- **Dependencies** represent a reference from one operation to another, or from an operation to an infrastructure component. For example, if an operation's output is stored in a database, the operation would have a reference to the database (that is, it would depend upon it). Outgoing dependencies are displayed in **lavender**, and

incoming dependencies are displayed in **green**. Dependencies are only displayed for items in the diagram that are selected.

By itself, the logical system does not contain enough information to be testable. To enable the creation and execution of tests, the configuration details of the infrastructure components that make up the logical system must be provided. These details are stored in physical resources which can be bound to infrastructure components (by means of an environment).

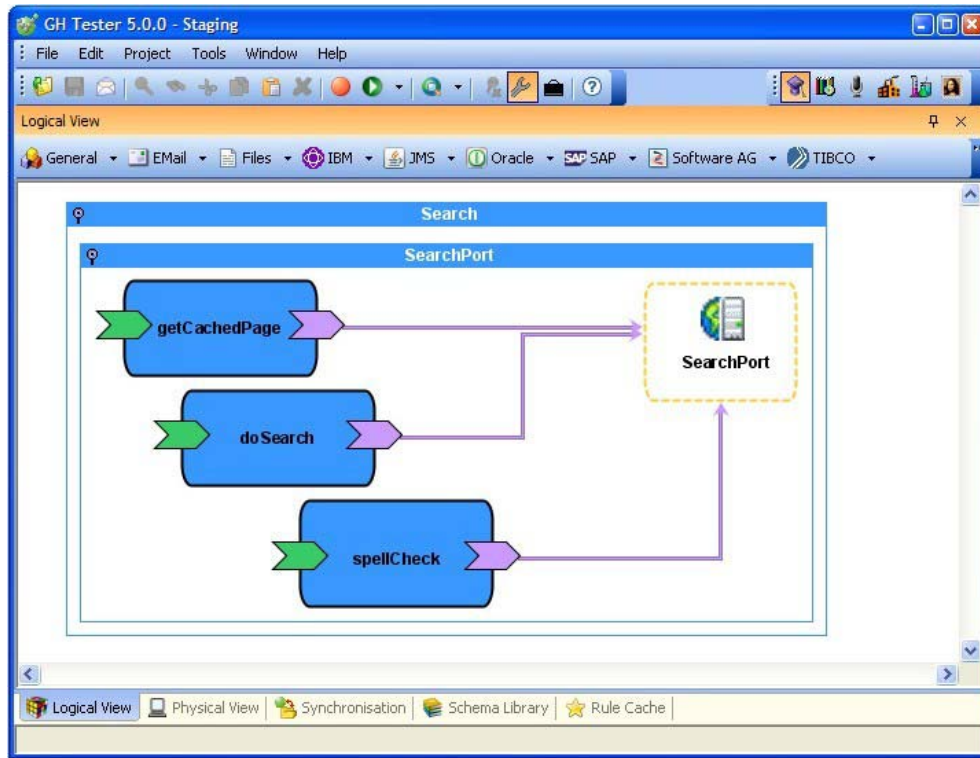
NOTE: An infrastructure component may have any number of physical resource implementations, but only one binding may exist when running a test.

Users can do the following in Architecture School:

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











3.2 The Logical View

The **Logical View** displays the logical layer of the system under test (that is, Service and Infrastructure Components and the dependencies between them). Use this view to lay out the logical service and infrastructure components, as well as their dependencies.



3.2.1 Component Shortcuts

The left side of the toolbar above the drawing palette provides shortcuts for creating components and operations. The following table lists the shortcuts available under each menu.

 General ▾	BPM Domain, Copybook, Database Server, DTD, Java Object, Operation, SCA Domain, Service Component, TCP/UDP Connection, UDDI Server, WSDL, XSD
 EMail ▾	Email
 Files ▾	File Contents, File Schema
 IBM ▾	IBM WebSphere® MQ Broker
 JMS ▾	JMS Domains (Destinations, Queues, and Topics), Sonic Domains (Destinations, Queues, and Topics), TIBCO EMS Domain
 Oracle ▾	HTTP Connection, SCA Domain, UDDI Server
 SAP ▾	SAP System
 Software AG ▾	CentraSite Server, webMethods Broker Domain, webMethods Integration Server Domain
 TIBCO ▾	TIBCO BusinessWorks Project, TIBCO Design Time Library, TIBCO EMS Domain, TIBCO Rendezvous Bus
 Web ▾	DTD, HTTP Connection, WSDL, XSD

3.2.2 Diagramming Tools

The right side of the toolbar above the drawing palette provides all of the tools you will need to build and modify your system under test. The following table lists the tools and their usage.



Adds a new external resource to the diagram, launching a wizard that lets you select the resource type and helps import it



Creates a dependency (from an operation) to a component or operation



Provides options to toggle the display of dependencies, clear all dependency filters, or edit dependency filters



Enables the “select” cursor, which is the default cursor, for selecting, moving, and editing diagram items



Increases the zoom of the current view (zoom in)



Decreases the zoom of the current view (zoom out)



Lets you draw a box around a specific area in the palette and zooms the view to that area



Resets the current view to the default zoom



Zooms the current view so that all diagram items fit on the screen



Enables the “pan” cursor that lets you grab a spot in the diagram and drag it around to change the view



Organizes the contents of the palette



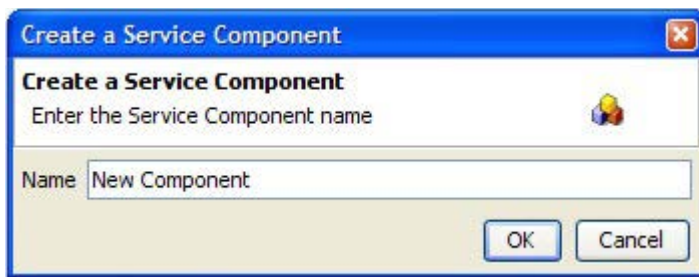
Toggles the drawing grid (when enabled, items will snap to the grid when created or moved)

3.2.3 Create a Service Component

A new service component can be created in one of two ways:

- **From the toolbar:** Click the desired location of the new service component (that is, an existing component or the drawing palette to create a top-level component) and select **General > Service Component**.
- **From the context menu:** Right-click the desired location of the new service component (that is, an existing component or the drawing palette to create a top-level component) and select **New > General > Service Component**.

When the **Create a Service Component** dialog is displayed, provide a name for the new component and click **OK**.

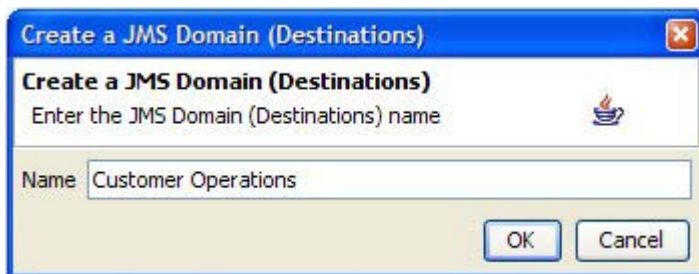


3.2.4 Create an Infrastructure Component

A new infrastructure component can be created in one of two ways:

- **From the toolbar:** Click the desired location of the new infrastructure component (that is, an existing component or the drawing palette) and select the desired component type from the menus across the top of the drawing palette.
- **From the context menu:** Right-click the desired location of the new infrastructure component (that is, an existing component or the drawing palette) and select the desired component type from the **New** menu.

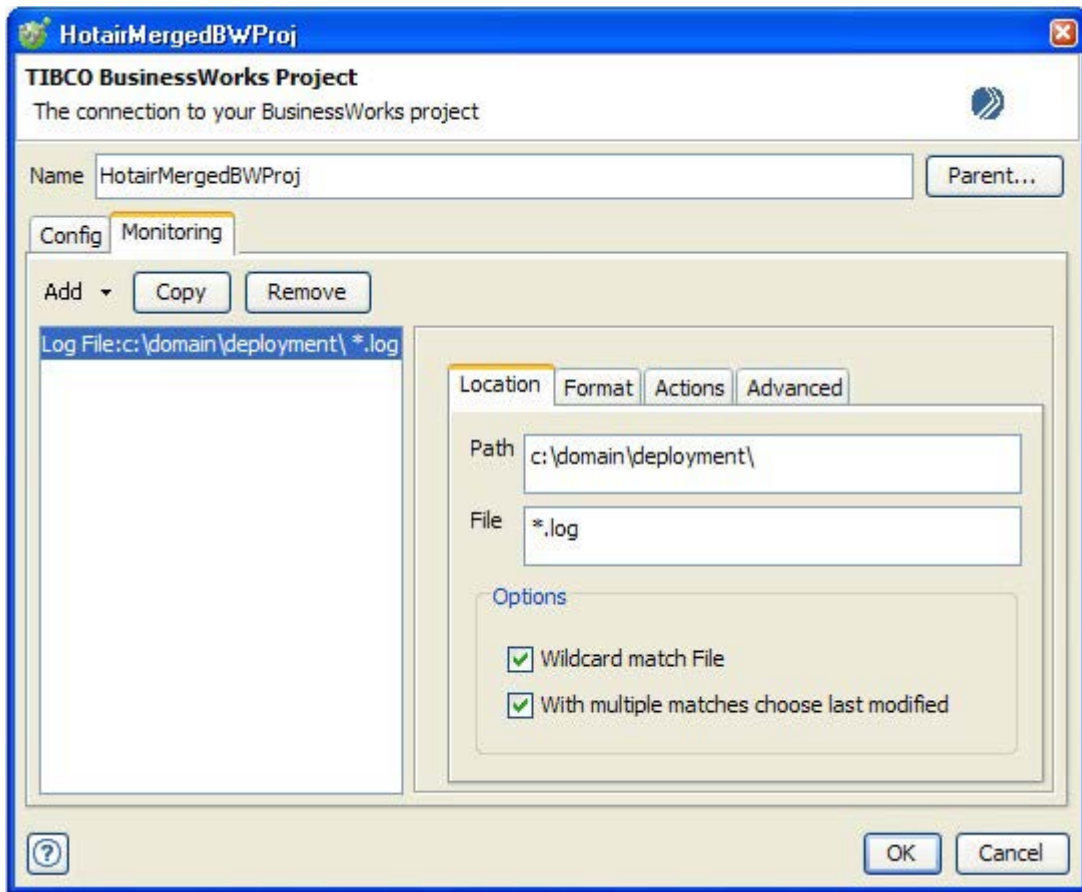
In the **Create...** dialog, provide a name for the new component and click **OK**.



3.2.5 Add a Log File to Monitor on an Infrastructure Component

Log files that are associated with infrastructure components (that is, logical components that can be bound to physical resources) can be configured for monitoring when the component is used in a test.

To add or modify a log file configuration for any of the supported components, double-click the component in the Logical View and select the **Monitoring** tab.

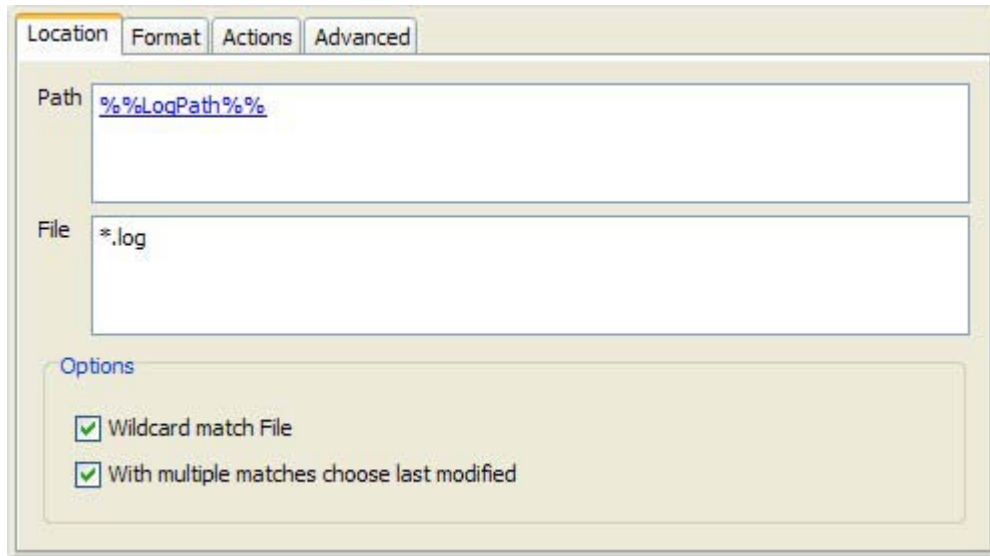


To add a log file configuration to the selected component, click **Add** and select **Log File** from the menu. If you want to copy an existing log file configuration, select its entry and click **Copy**. To remove an existing log file configuration, select its entry and click **Remove**.

The details of any log file configuration can be modified using the **Location**, **Format**, **Actions**, and **Advanced** tabs. See the following sections for configuration details: [Log File Location and Options](#), [Log File Format Options](#), [Log File Actions](#), and [Advanced Log File Options](#).

Log File Location and Options

Under the **Location** tab you can configure the path to the log file, the log file name, and options for controlling whether or not to use a wildcard match for the log file (for example, *.log) and how to treat multiple files that match a specified wildcard.



The screenshot shows a dialog box with four tabs: Location, Format, Actions, and Advanced. The Location tab is selected. It contains two text input fields: 'Path' with the value '%%LogPath%%' and 'File' with the value '*.log'. Below these fields is an 'Options' section with two checked checkboxes: 'Wildcard match File' and 'With multiple matches choose last modified'.

In the **Path** field, enter the full path to the location where log files attached to the selected component are stored. By default, the path is c:\domain\deployment\.

In the **File** field, enter the name of the log file to be monitored. Wildcard entries can be used for the filename (for example, *.log, *.txt, and so on). If using a wildcard, ensure that the appropriate matching options are selected under **Options**.

Under **Options**, enable one or both of the available matching options, as follows:

- If a wildcard has been specified in the **File** field, ensure that the **Wildcard match File** option is enabled.
- If wildcard matching is enabled and more than one log file may be matched, enable the **With multiple matches choose last modified** option to use only the most recently modified file. If multiple files are matched and this option is not enabled, errors may occur.

Log File Format Options

Under the **Format** tab you can configure the format of individual log file entries so that Rational Integration Tester can distinguish one entry from the next, timestamp options to help recognize the timestamp format used in the selected log file, and technical correlation matching options to identify different transactions that have been logged.

The screenshot shows the 'Format' tab of a configuration window. It is divided into three main sections: 'Entry format', 'Timestamp', and 'Technical Correlation'. Each section contains input fields and radio button options.

Entry format

- Entity:
- Treat gaps between entries as:
 - ☒ Attached to last
 - ☐ Attached to next
 - ☐ An error
 - ☐ Ignore

Timestamp

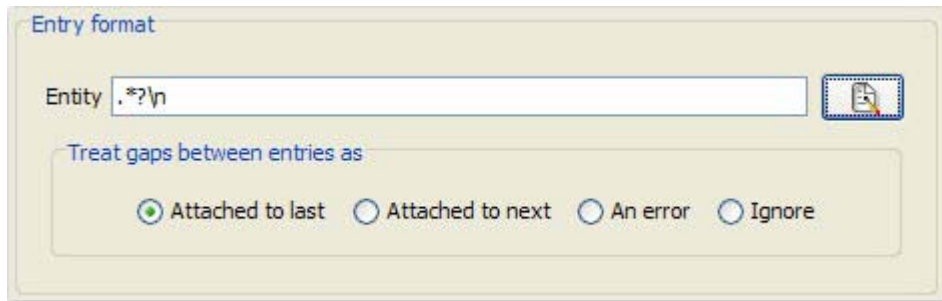
- Regex:
- Parse Format:
- Time Offset:
- Treat entries without timestamps as:
 - ☒ Same as last
 - ☐ Same as next
 - ☐ An error

Technical Correlation

- Regex:
- Treat unresolved entries as:
 - ☒ Excluded
 - ☐ Included
 - ☐ Last resolved
 - ☐ Next resolved
 - ☐ An error

Entry Format

In the **Entity** field, enter a regex that defines a single event (entity) in the log. The specified expression should match the start of one entity and any text that comes before the start of the next.

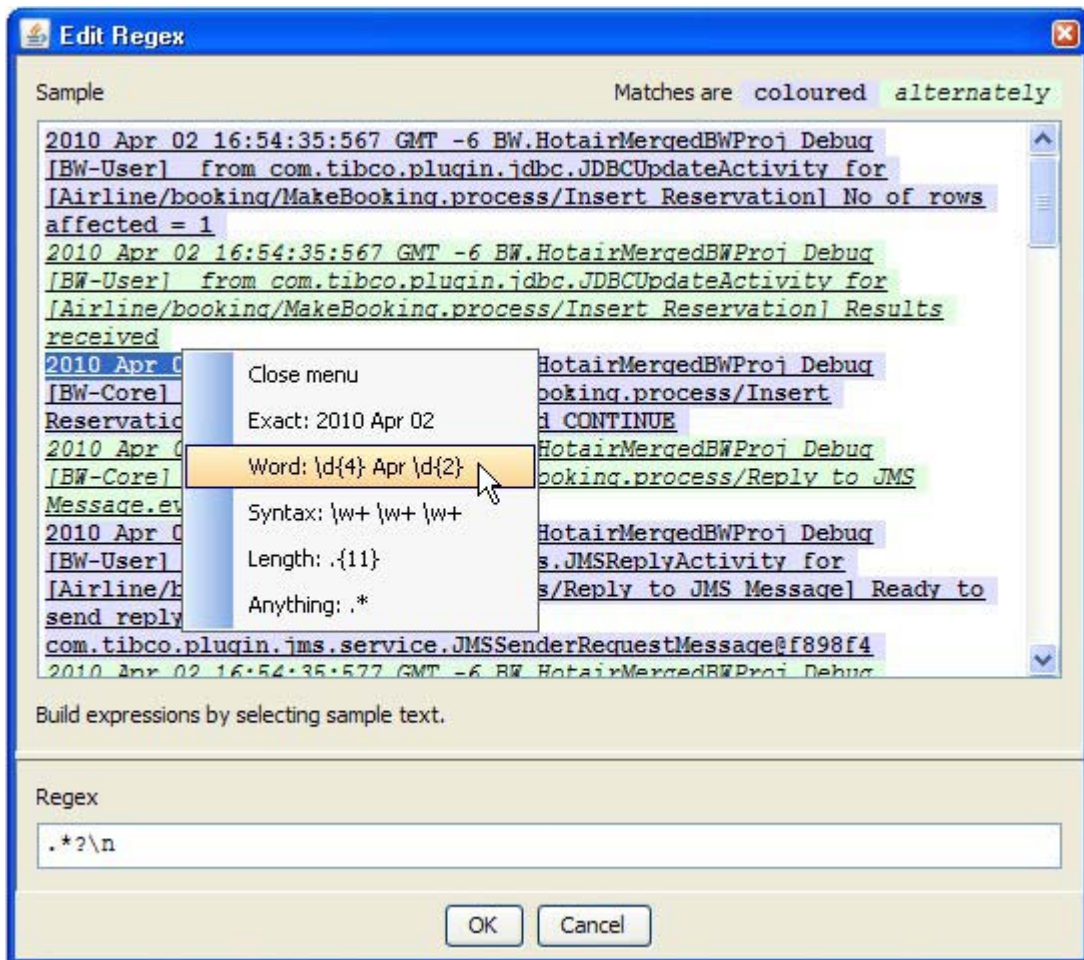


The screenshot shows a dialog box titled "Entry format". It contains an "Entity" text field with the value ".*?\n". Below this is a section titled "Treat gaps between entries as" which contains four radio button options: "Attached to last" (which is selected), "Attached to next", "An error", and "Ignore".

In the **Treat gaps between entries as** panel, select one of the four options to define how Rational Integration Tester should treat any gaps that it finds in the log, as follows:

- **Attached to last** – treats the gap as being part of the previous log entity
- **Attached to next** – treats the gap as being part of the next log entity
- **An error** – produces an error if any gaps are found
- **Ignore** – ignores any gaps that are found in the log

If you click the **Edit** button to the right of the **Entity** field, the regex editor is displayed, showing a sample of the log file that is defined under the **Format** tab.



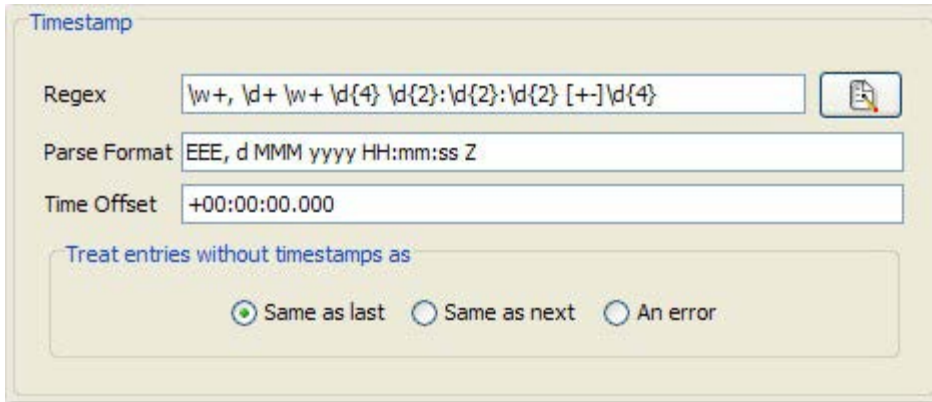
You can edit the expression in the **Regex** field and view a live preview of the entities that it defines. Matching entities are coloured (alternatively) blue and green to illustrate how the expression will behave at runtime.

You can build expressions by selecting a portion of the preview text. Once you release the mouse button, the expressions are displayed in a context menu. To use one of the suggestions, simply click on its entry in the list. To dismiss the menu without using one of the suggested expressions, select the **Close menu** option at the top of the list.

To save any changes you have made to the expression, click **OK**. To return to the log file configuration without applying any changes, click **Cancel**.

Timestamp

The **Timestamp** settings are optional, but they can be used to help Rational Integration Tester determine how to present timestamps within the log file, and calculate when a specific entry was written since logs are generated in an asynchronous manner.



The screenshot shows a dialog box titled "Timestamp" with the following fields and options:

- Regex:** A text field containing the regular expression `\w+, \d+ \w+ \d{4} \d{2}:\d{2}:\d{2} [+-]\d{4}`. To the right of the field is a small icon of a document with a magnifying glass.
- Parse Format:** A text field containing the date format `EEE, d MMM yyyy HH:mm:ss Z`.
- Time Offset:** A text field containing the time offset `+00:00:00.000`.
- Treat entries without timestamps as:** A section containing three radio button options:
 - ☒ Same as last
 - ☐ Same as next
 - ☐ An error

The options under **Timestamp** can be used as follows:

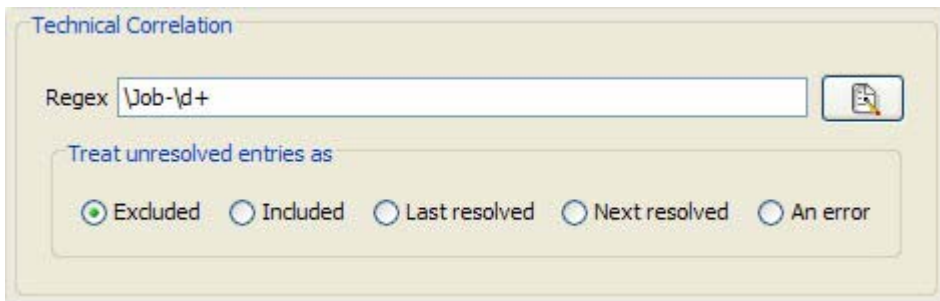
- **Regex** – the regular expression used to locate the timestamp (see the previous section, Entry Format, for more information about using the expression editor)
- **Parse Format** – a date format/pattern that can be used to create a SimpleDateFormat object
- **Time Offset** – a time value that will be added to the time stamp value once it has been parsed

In the **Treat entries without timestamps as** panel, select one of the three options to define how Rational Integration Tester should treat entities that are missing timestamps, as follows

- **Same as last** – apply the same timestamp as the previous entity
- **Same as next** – apply the same timestamp as the next entity
- **An error** – produce an error and fail the test if no timestamp is resolved

Technical Correlation

The options under **Technical Correlation** can be used to identify separate transactions that may have been logged concurrently.

The image shows a dialog box titled "Technical Correlation". It has a "Regex" text field containing the text "\Job-Id+" and a small icon to its right. Below this is a section titled "Treat unresolved entries as" which contains five radio button options: "Excluded" (which is selected), "Included", "Last resolved", "Next resolved", and "An error".

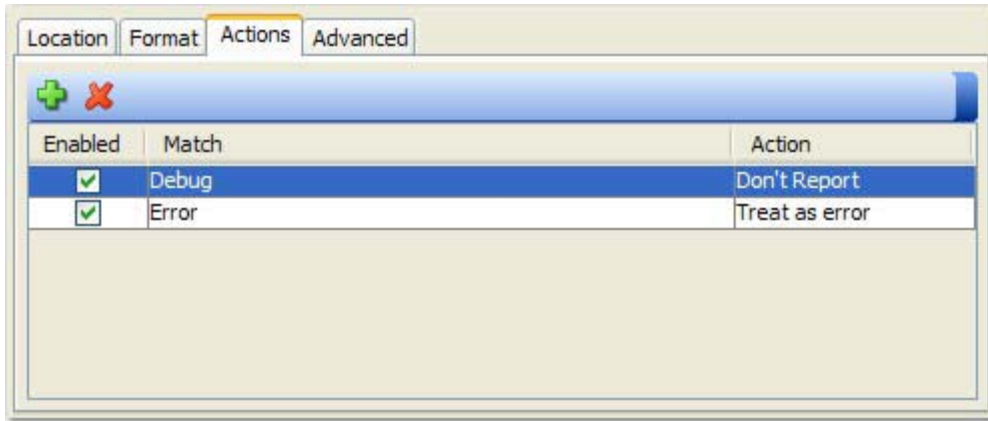
In the **Regex** field, enter a regular expression that will identify a unique transaction ID within log entities (see the previous section, **Entry Format**, for more information about using the expression editor).

In the **Treat unresolved entries as** panel, select one of the five options to define how Rational Integration Tester should treat entities that do not contain the transaction ID, as follows

- **Excluded** – exclude entries from all transactions
- **Included** – include entries in all transactions
- **Last resolved** – treat entries in the same transaction as the previous log entry
- **Next resolved** – treat entries in the same transaction as the next log entry
- **An error** – produce an error and fail the test if no transaction ID is resolved

Log File Actions

Under the **Actions** tab you can define specific test actions that should be taken when a log entity matches the specified expression.



To add an action to the list, click the icon.

For each action in the list, check the box under **Enabled** to have the action applied to tests at runtime. To leave the action in the list but not use it at runtime, leave the **Enabled** check box empty.

Double-click the field under **Match** to preview the log file and create an expression to match within entities. You can enter the expression manually, or use the expression editor as described previously (under [Log File Format Options](#)).

From the **Action** field, select one of the following three actions to take when the specified expression is matched in a log entity:

- **Don't Report** – ignore the entry and do not generate any console output
- **Treat as error** – flags the entry as an error in the console and fails the test
- **Treat as warning** – flags the entry as a warning in the console but does not fail the test

NOTE: Actions are matched in the order that they are defined in the editor. If no actions are defined or none of the defined actions are matched, the entry will be sent to the test console as a normal message.

To remove an action from the list, click the icon.

Advanced Log File Options

Use the options under the **Advanced** tab to configure log file encoding and the method to use for determining if/when the log file rolls over.



The screenshot shows a dialog box with four tabs: Location, Format, Actions, and Advanced. The Advanced tab is selected. It contains two sections: Character Set and Rollover. Under Character Set, the File Encoding is set to windows-1252. Under Rollover, the Method is set to When file shrinks.

Under **Character Set**, from the **File Encoding** menu, select the one of the available file encoding methods to use when processing the log file.

The **Rollover** method helps to determine when the log file has rolled over and a new file has been started. This option is not currently used, however, and the default value (**When file shrinks**) should be left unchanged.

3.2.6 Add an External Synchronisation Resource

The following Infrastructure Components can be added as external resources that can be synchronized with your Rational Integration Tester project: WSDL, webMethods Integration Server, TIBCO BusinessWorks Project or Design Time Library, SAP System, and SCA Domain (Oracle Fusion). For more information about external resources and synchronisation, see [Synchronisation](#).

3.2.7 Create Dependencies

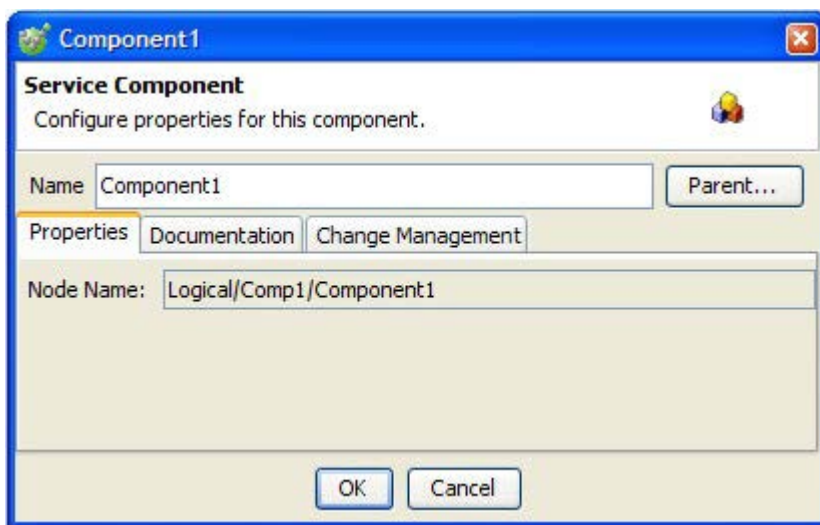
Dependencies are references between one operation and another, or between an operation and an infrastructure component. Dependencies can be created within the same service component, or they can extend beyond it. Follow the steps below to create dependencies between items in the logical view.

1. Click the Dependency icon in the main Rational Integration Tester toolbar.
The selection cursor turns into a cross hair.
2. Click the operation from which you are creating the dependency.
A line forms from the selected operation and follows the corsair cursor.
3. Click the operation or infrastructure component upon which the operation depends.

3.2.8 View Component Details

Follow the steps below to view the detailed properties of an existing component:

1. Double-click the component or right-click it and select **Open** from the context menu.
The component editor is displayed.



-
2. The component name is displayed, with additional information in various tabs, as follows:

Field	Description
Node Name	The name of the component (see Change a Component's Name).
Type	For infrastructure components, the type of the component.
Parent	Click this button to change the component's parent (see Change a Component's Parent for more information).

Properties

Resource Path	The fully qualified name of the component, including its location in the project hierarchy.
---------------	---

Bindings (Infrastructure Components only)


Environment	The name of the environment (one row is shown for each environment so that a binding can be configured for each one).
Binding	The physical resource to which the infrastructure component should be bound in the corresponding environment.

Documentation

Description	The description of the component (see Change a Component's Description).
Created	The date and time when the component was created, as well as the ID of the user who created it (in the By field).
Updated	The date and time when the component was last updated, as well as the ID of the user who made the update (in the By field).
Source	

Change Management

Override	Lets users specify a different change management integration than one inherited from another resource (higher up the project resource tree) or the default integration specified in the Project Settings. See Change Management for more information.
----------	---

3. When finished, click **OK** to save any changes and close the editor. Otherwise, click  or **Cancel** to close the editor without making changes.

3.2.9 Change a Component's Name

Follow the steps below to change the name of an existing component:

1. Open the component editor (see [View Component Details](#)) and enter a new name for the component in the **Name** field.
- or -
2. Right-click the component, select **Rename** from the context menu, and enter a new name for the component in the **Rename to** dialog.
3. Click **OK** to save your changes and close the editor/dialog.

3.2.10 Change a Component's Parent

Follow the steps below to move a child component from one parent to another:

1. Open the component editor (see [View Component Details](#)).
2. Click the **Parent** button.

The **Select a Resource** dialog is displayed.

3. Locate and select the new parent component.
4. Click **OK** to save your changes and close the dialog.

3.2.11 Change a Component's Bindings

For Infrastructure Components, you can configure bindings in one or more environments in the component editor, by means of the context menu, or in the environment editor (see [Edit an Environment](#)).

Using the Component Editor

1. Open the component editor (see [View Component Details](#)) and select the **Bindings** tab.
2. Click in the **Binding** field next to the desired environment and select the physical resource to bind to in that environment.
3. Click **OK** to save your changes and close the dialog.

Using the Context Menu

1. Right-click the desired component and place the cursor over the **Set Binding In** option.

NOTE: A submenu is displayed for each environment that has been created. The environment that is currently selected is displayed in bold.

2. Select the name of the environment where you want to set the binding.
3. If any have been created, select the physical resource to which the selected component should be bound.

NOTE: If desired, or if no physical resources of the selected type exist, you can create a new physical resource by selecting the **Create new <resource type>** option.

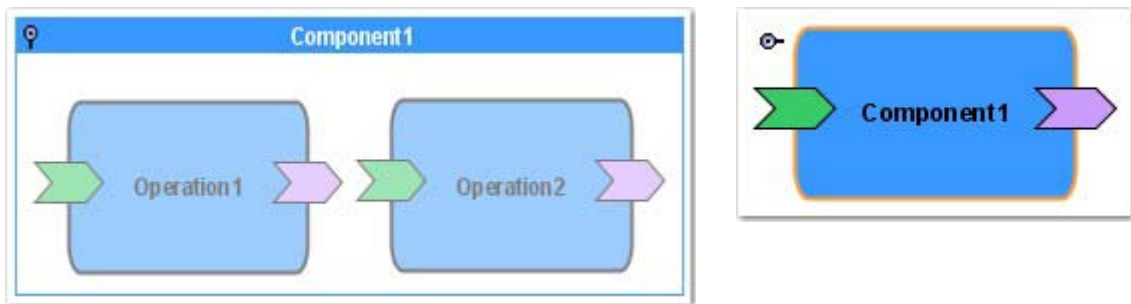
3.2.12 Change a Component's Description

Follow the steps below to change the description of an existing component:

1. Open the component editor (see [View Component Details](#)).
2. Click the **Documentation** tab.
3. Enter a new description for the component in the **Description** field.
4. Click **OK** to save your changes and close the editor.

3.2.13 Expand and Collapse Components

Service components can be viewed in an expanded (default) or collapsed state. To change the current state of a component, simply click the expand/collapse icon (in the title bar when expanded, to the left of the component when collapsed). The image below illustrates the same component in its expanded and collapsed states.

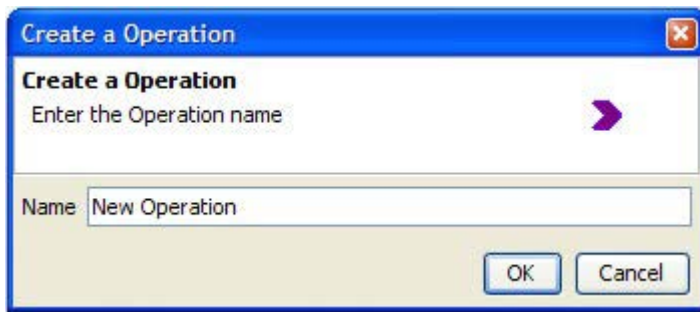


3.2.14 Create an Operation

A new operation can be created in one of two ways:

- **From the toolbar:** Click the service component that will contain the operation and select **General > Operation**.
- **From the context menu:** Right-click the service component that will contain the operation and select **New > General > Operation**.

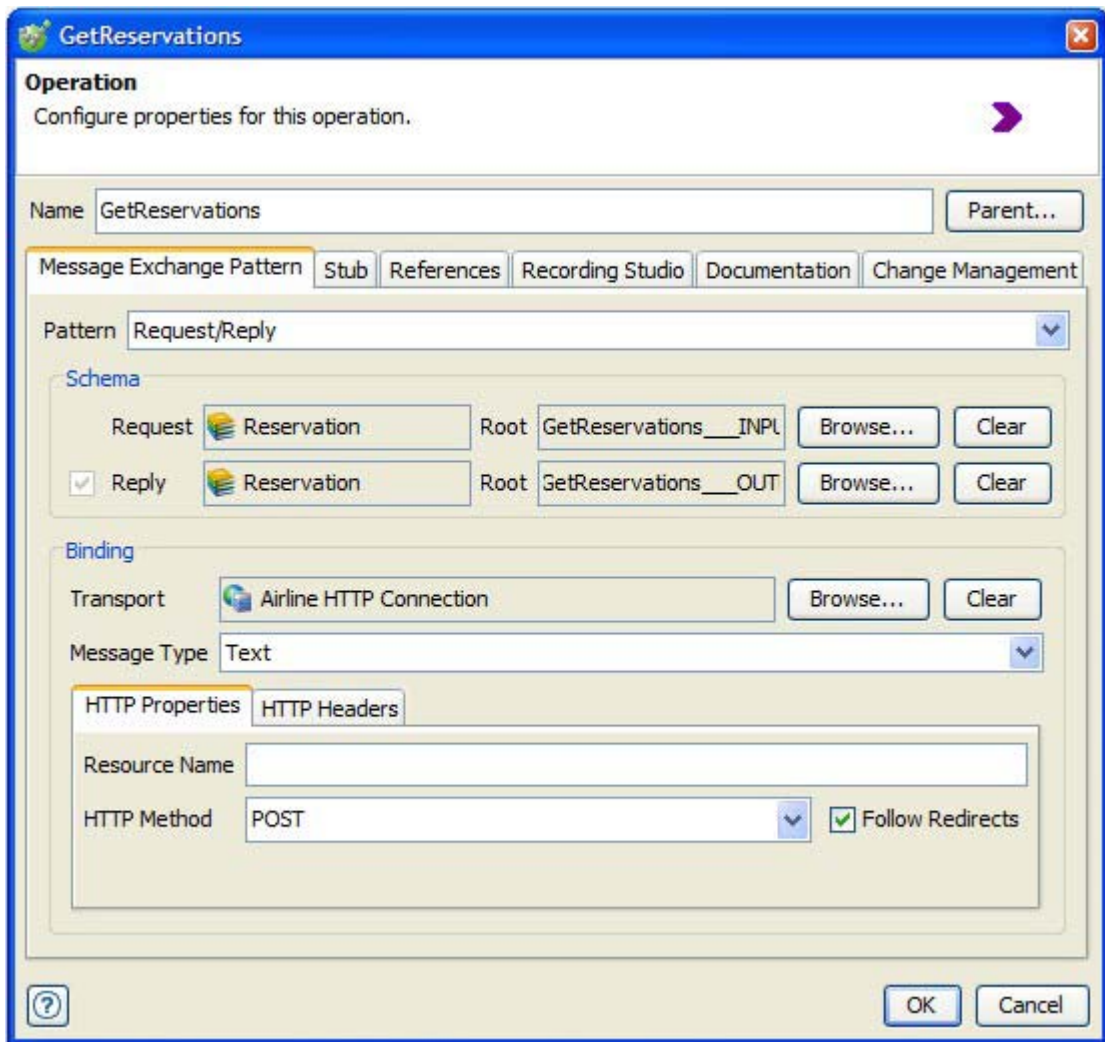
When the **Create an Operation** dialog is displayed, provide a name for the new operation and click **OK**.



3.2.15 View Operation Details

Follow the steps below to view the detailed properties of an existing operation:

1. Double-click the operation or right-click it and select **Open** from the context menu. The operation editor is displayed.



2. The component details are displayed, with additional information available in the dialog's multiple tabs.
3. Details about viewing or editing specific properties can be found in the sections that follow.

Change an Operation's Name

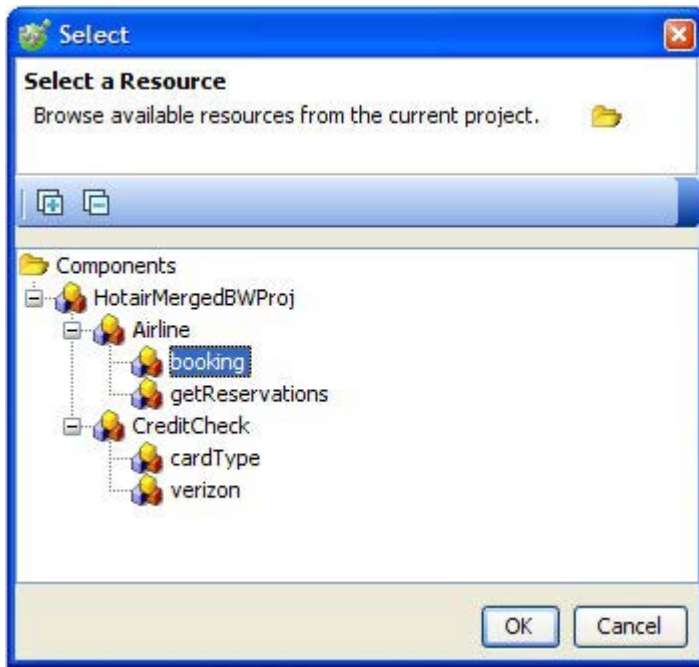
Follow the steps below to change the name of an existing operation:

1. Open the operation editor (see [View Operation Details](#)) and enter a new name for the operation in the **Name** field.
- or -
2. Right-click the operation, select **Rename** from the context menu, and enter a new name for the operation in the **Rename to** dialog.
3. Click **OK** to save your changes and close the editor/dialog.

Change an Operation's Parent

Follow the steps below to move an operation from one parent to another:

1. Open the operation editor (see [View Operation Details](#)).
2. Click the **Parent** button (the **Select a Resource** dialog is displayed).

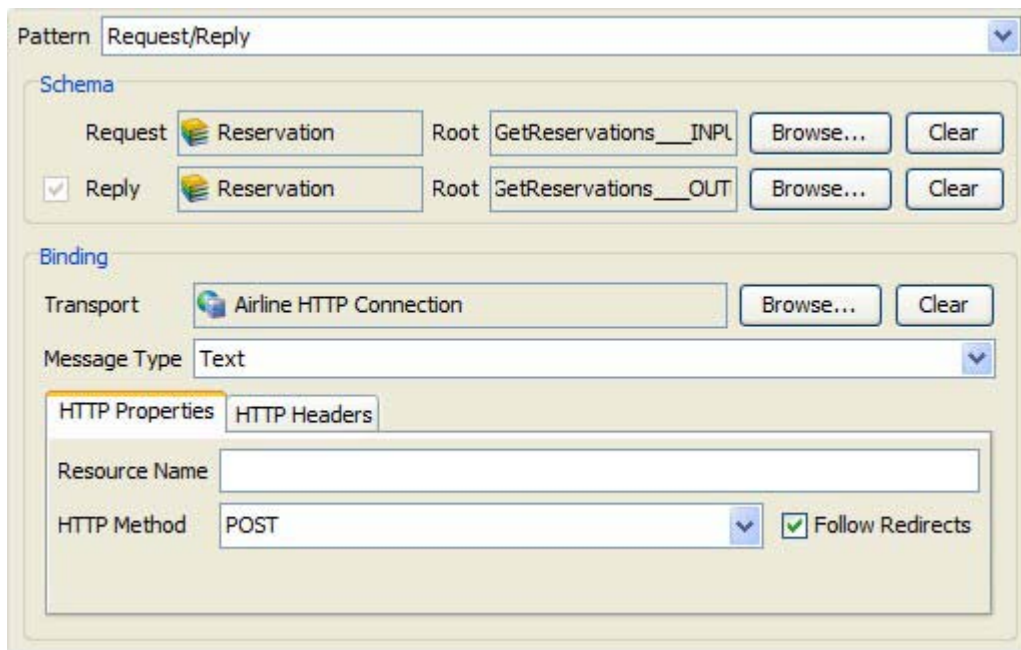


3. Locate and select the new parent component.
4. Click **OK** to save your changes and close the dialog.

Change an Operation's Message Exchange Pattern


An operation's Message Exchange Pattern (MEP) defines the pattern, schema, and binding for messages that are exchanged by the operation. Follow the steps below to view or modify an operation's MEP:

1. Open the operation editor (see [View Operation Details](#)). The **Message Exchange Pattern** tab is highlighted, by default.



2. Modify the various properties of the MEP as described in the table below:

Property	Description
Pattern	Select the desired message pattern (Publish or Request/Reply).
Schema	Select the schema or format that should be applied to each applicable message type. Click Browse to locate and select a schema (see Schemas for more information), or click Clear to clear the current selection.
Binding	Click Browse to select an existing transport to apply to the operation's messages. Depending on the transport, configure additional message options using the tabs below the transport selection. See Transports and Formatters for more information.

3. When finished, click **OK** to save any changes and close the editor. Otherwise, click  or **Cancel** to close the editor without making changes.

Change an Operation's Stub Properties

An operation's Stub properties define the binding and timeout details that should be used when creating a stub from the operation. Follow the steps below to view or modify an operation's Stub properties:

1. Open the operation editor (see [View Operation Details](#)) and select the **Stub** tab.

The screenshot shows a dialog box titled "Stub Properties". It has two main sections: "Binding" and "Default Timeout".

Binding Section:


- Transport:** A text box containing "Airline EMS Connection" with "Browse..." and "Clear" buttons to its right.
- Message Type:** A dropdown menu showing "Text Message".
- Destination:** A text box, currently empty, with a "Durable" checkbox to its right.
- Filter:** A text box, currently empty.
- Mode:** Two radio buttons: "Watch" (unselected) and "Participate" (selected).

Default Timeout Section:

- Minimum Response Time (ms):** A text box containing "0".
- Maximum Response Time (ms):** A text box containing "0".
- Average Response Time (ms):** A text box containing "0".

2. Modify the various properties in the dialog as described in the table below:

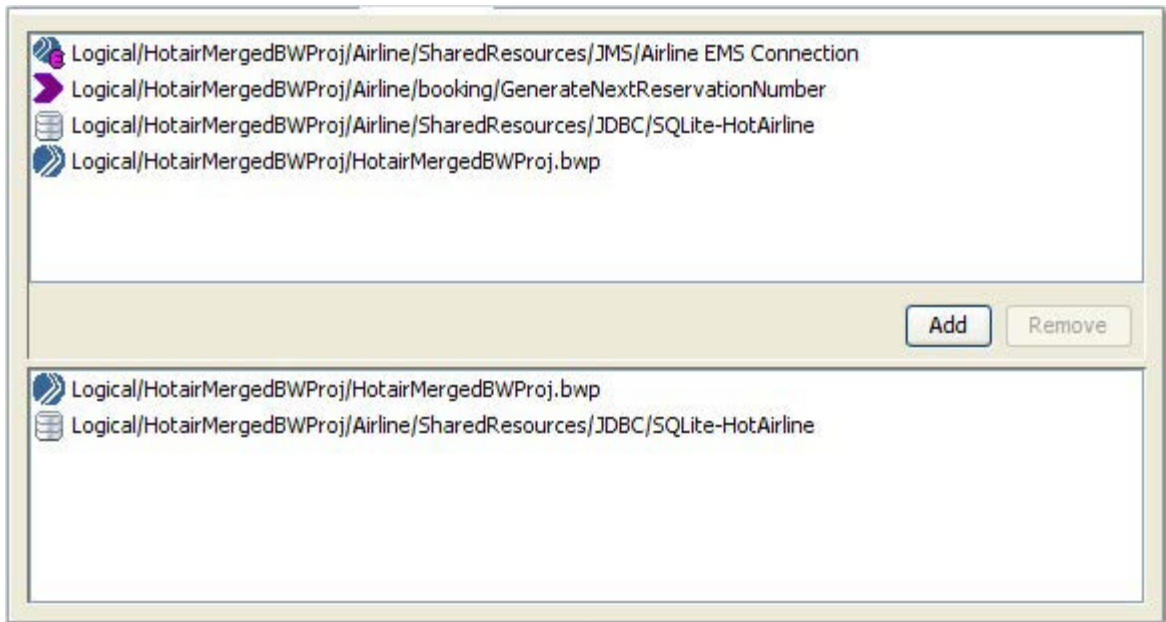
Property	Description
Binding	Click Browse to select an existing transport that should be applied to the operation's Stubs. Depending on the transport, you can configure additional message options using the fields below the transport selection. See Transports and Formatters for more information.
Default Timeout	Enter the minimum, maximum, and average response times (in milliseconds) that the stub should use when replying to requests.

3. When finished, click **OK** to save any changes and close the editor. Otherwise, click  or **Cancel** to close the editor without making changes.


Change an Operation's References

References show the dependencies that the selected operation has on other items within the system under test. The references that are displayed will correspond directly to the operation's dependencies that are displayed in the diagram. Follow the steps below to view or modify an operation's references:

1. Open the operation editor (see [View Operation Details](#)) and select the **References** tab.



Direct references (no components between the operation and the dependency) are displayed in the top portion of the dialog, while indirect references (one or more components link the operation with the dependency) are displayed in the bottom.

2. To add a reference, click **Add** and select the resource to which you want to create a dependency.
3. To remove a reference, select it and click **Remove**.
4. When finished, click **OK** to save any changes and close the editor. Otherwise, click  or **Cancel** to close the editor without making changes.

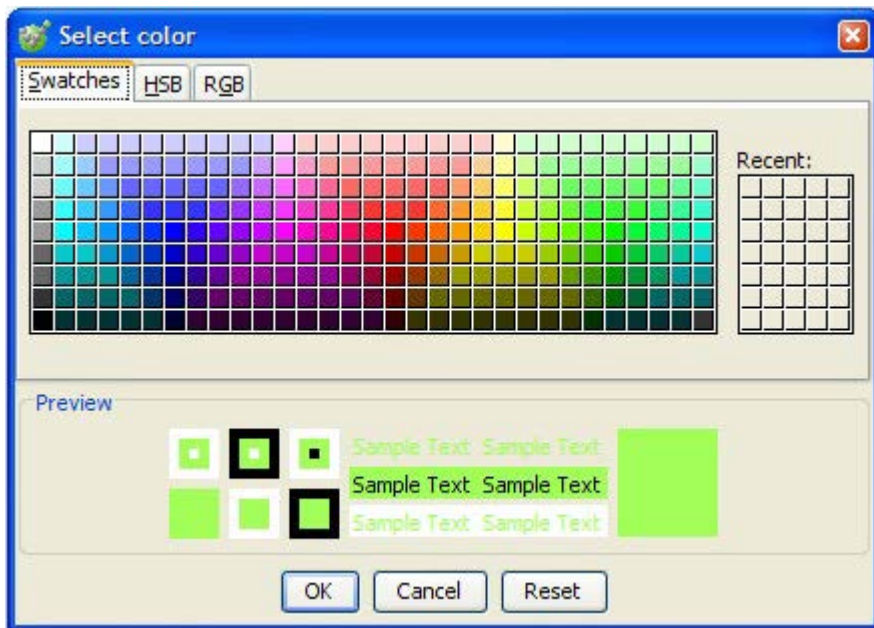
Change an Operation's Recording Studio Color



When monitoring events in the Recording Studio perspective, events for each operation are displayed in their own unique color. This is especially useful if you are viewing events from multiple operations. Follow the steps below to view or modify an operation's event color:

1. Open the operation editor (see [View Operation Details](#)) and select the **Recording Studio** tab.



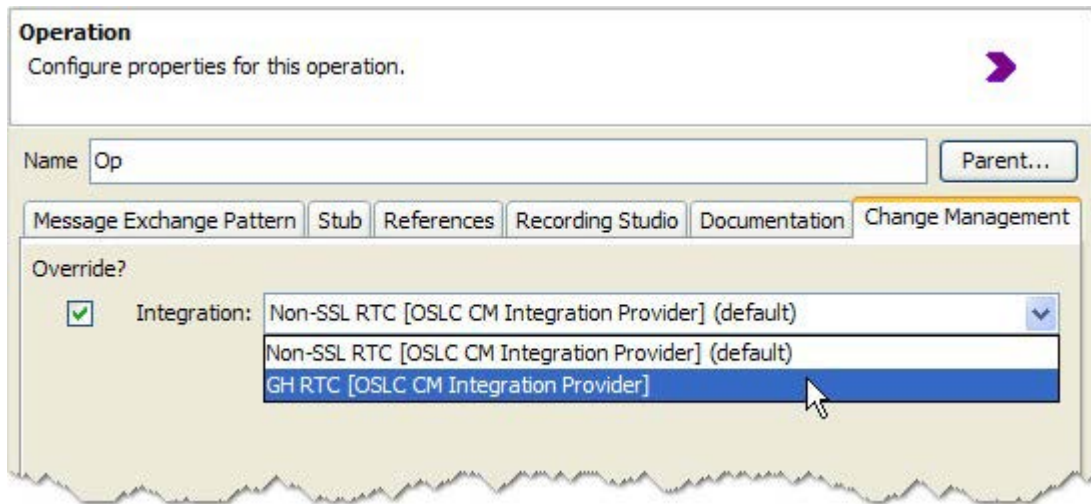
2. Click the color box next to **Display Color** to display the **Select color** dialog.



3. Select the tab of the desired color model (Swatches, HSB, or RGB) and select the desired color.
4. Click **OK** to save the new selection, click Reset to revert to the original color, or click  or **Cancel** to close the dialog without making changes.
5. When finished, click **OK** to save any changes and close the editor. Otherwise, click  or **Cancel** to close the editor without making changes.

Change an Operation's Preferred Change Management Integration

Rational Integration Tester can be integrated with OSLC compliant change management systems for raising defects within the context of a Rational Integration Tester test asset. If multiple change management integrations have been configured in the current project, you can override an inherited integration (from an operation or service component higher up the project resource tree) or the default integration under the **Change Management** tab, letting you specify a different integration to be used for items below the selected operation.



To specify a different or non-default integration for the current operation and its children, enable the **Override** option and select the existing integration from the list of those available.

3.2.16 View a Component's Physical Resource

If an infrastructure component is bound to a physical resource in the current environment, you can open the physical resource for viewing or editing by right-clicking the component and selecting **Physical Resource** from the context menu.

NOTE: The **Physical Resource** option is not available for non-bindable components (for example, service components or operations) or for components that are unbound in the current environment.

3.2.17 Set the Focus on a Specific Component

For some projects, even for a simple BusinessWorks or web services project, you can quickly accumulate a large number of operations, components, and dependencies in Architecture School's logical view. To simplify the view and temporarily hide items that are beyond a single service component, use the **Focus ...** option.

1. Right-click a component and select **Focus on** *<component>* from the context menu. Everything outside of the selected component will be hidden from the logical view.
2. To restore the view of the entire system, right-click anywhere within the logical view and select **Remove focus from** *<component>* from the context menu.

3.2.18 Hide Dependencies

Dependencies can be hidden, or filtered out of view, in a number of different ways.

- To hide all dependencies in the system, click the arrow next to the filter icon and select **Hide Dependencies**.

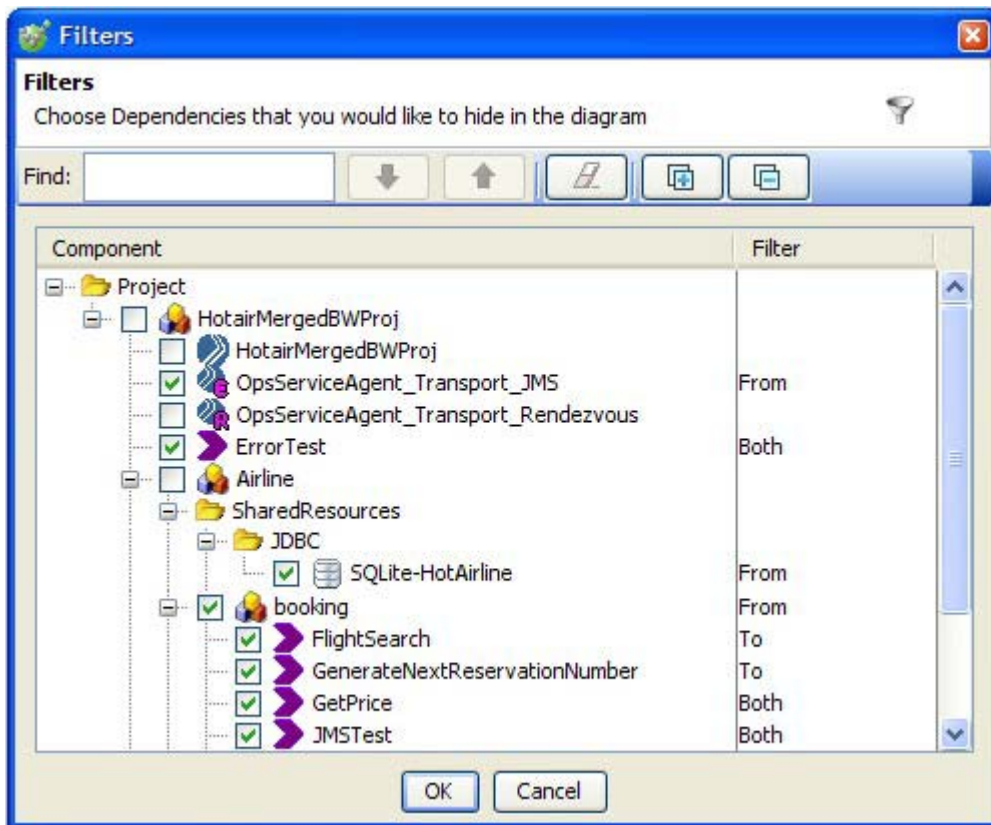
NOTE: To reveal dependencies again, select **Show Dependencies** from the same menu.

- To hide dependencies from an operation, right-click the operation and select **Hide Dependencies > From** *<operation name>*.
- To hide dependencies on an operation or component, right-click the item and select **Hide Dependencies > On** *<item name>*.
- To hide dependencies on all operations or all of a specific component type, right-click the item and select **Hide Dependencies, On all** *<item type>*. For example, to hide all dependencies on database servers, select any database server and select **Hide Dependencies, On all Database Servers**.

3.2.19 Edit Dependency Filters

Dependency filters control how dependencies are displayed for the entire diagram.

1. Click the Filter icon in the main Rational Integration Tester toolbar, or click the arrow next to the icon and select **Edit Filters**. The filters dialog is displayed.



2. Select the operations or components whose dependencies you want to hide, and deselect any of the operations or components whose dependencies should be shown.
3. For each selected item, select the type of filtering to apply (**From**, **To**, or **Both**) from the dropdown menu under the “Filter” column.

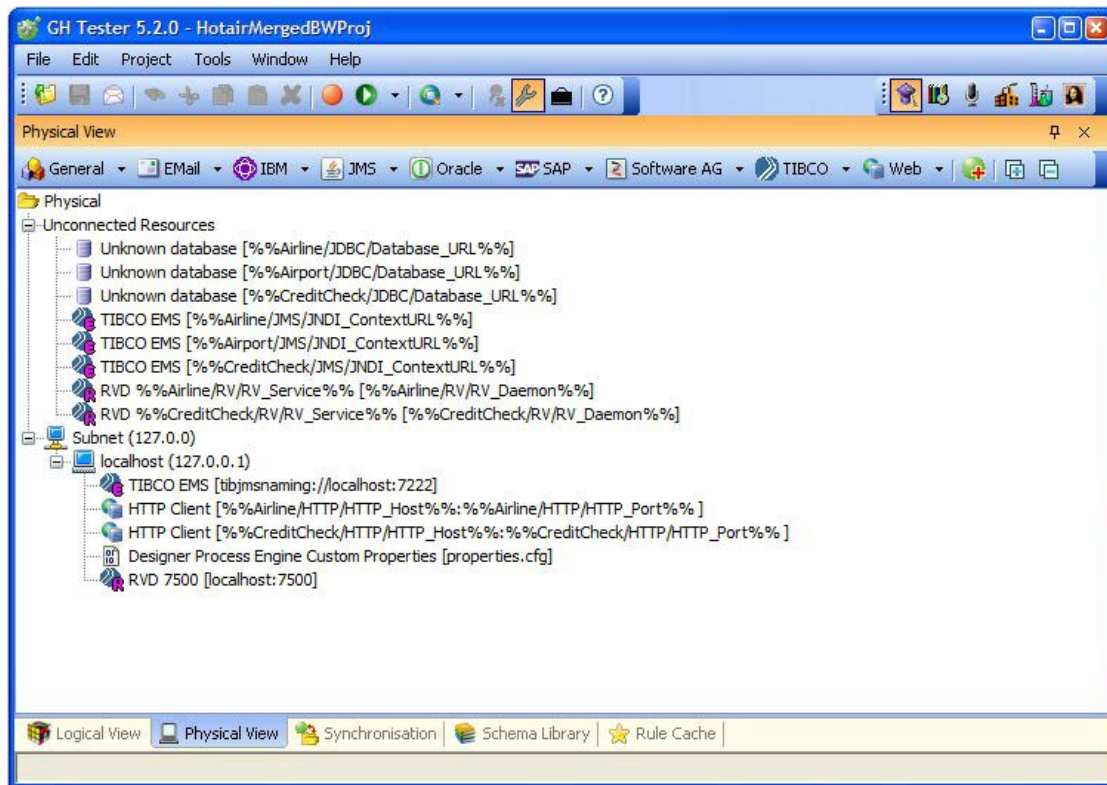
NOTE: Choosing the empty option from the dropdown menu will remove the filter from the selected item.

4. When finished, click **OK** to apply the changes and close the dialog.

NOTE: To clear all dependency filters, click the arrow next to the Filter icon and select **Clear Filters**.

3.3 The Physical View

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



NOTE: Physical resources represent testable items in your system, but they are only accessible when bound to a logical resource by means of the environment that is currently in use by a test.

NOTE: Since a logical resource can be bound to only one physical resource at a time, you will normally create multiple versions of a physical resource for use in different environments.

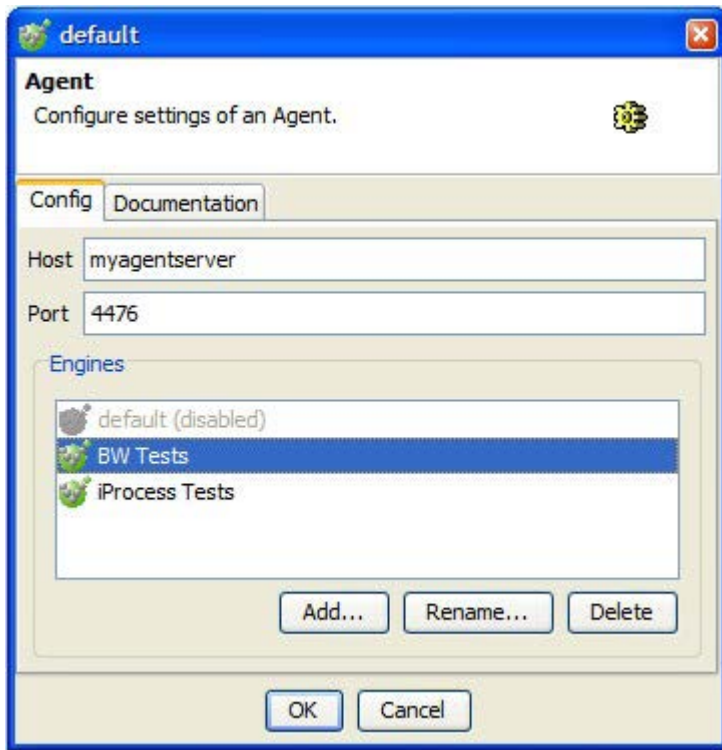
3.3.1 General Resources

Numerous types of general physical resources can be added to the Physical View, all of which are available under the **General** resource menu. The following sections describe each resource type and the menu option for adding it. In all cases, double-click a physical resource to open it for editing.

- [Agent](#)
- [Cluster](#)
- [Database](#)
- [Host](#)
- [Identity](#)
- [Identity Store](#)
- [Subnet](#)
- [TCP/UDP Server](#)
- [UDDI Server](#)

Agent

An agent is a Java process used by the Performance Test Controller to start test engines and probes. An Agent may be on the local machine or on a remote server. Select **General > Agent** to add a Rational Performance Tester Agent.



Host The host name or IP address of the machine where the agent is installed.

Port The port number on which the agent is running.

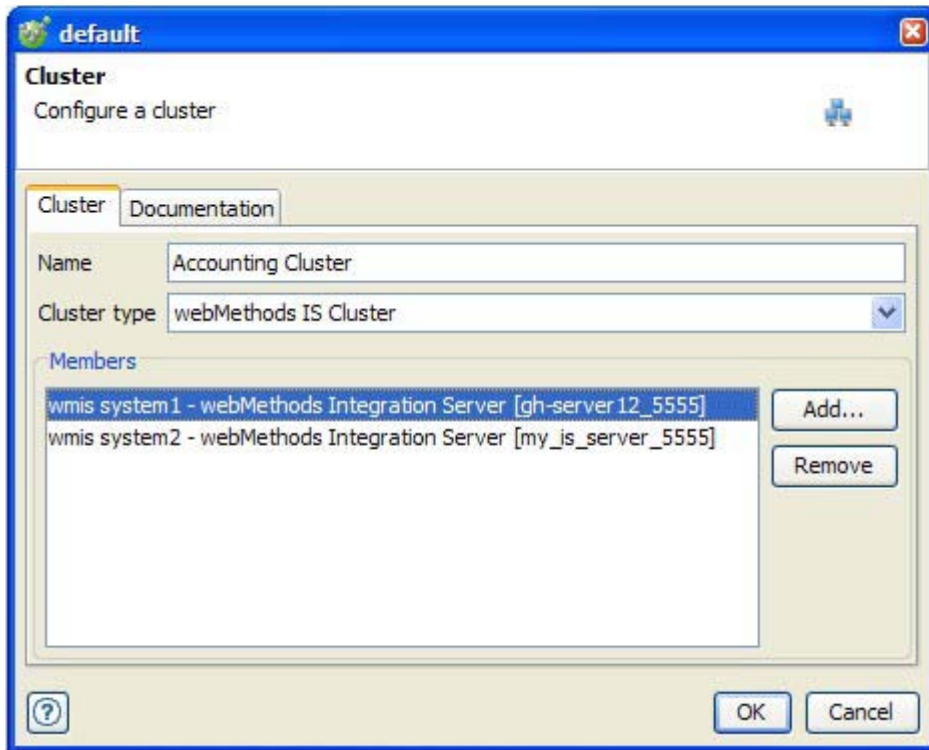
Engines Lists the test engines that the agent is running. To add a test engine to the agent, click **Add** and provide a name for the new engine. To rename an existing test engine, select it and click **Rename**. To delete a test engine, select it and click **Delete**.

NOTE: For more information about Agents and Test Engines, refer to *IBM Rational Performance Test Server Reference Guide*.

Cluster

A cluster is one or more of the same type of physical resource that have been configured as a cluster outside of Rational Integration Tester.

Select **General > Cluster** to add a new cluster to the Physical View.



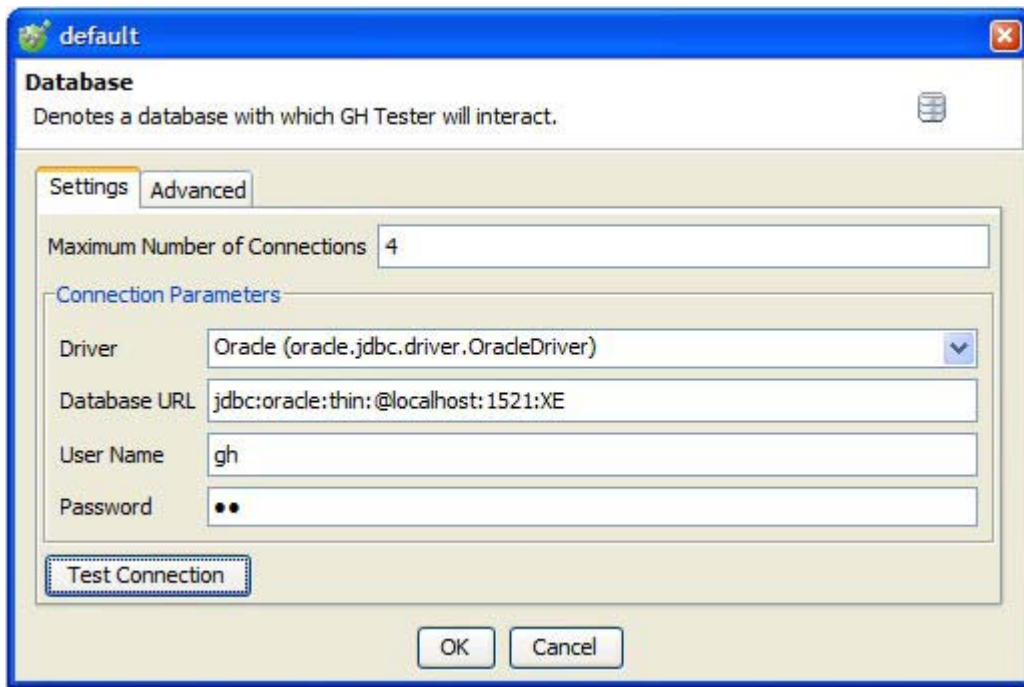
-
- | | |
|--------------|---|
| Name | Provide a descriptive name for the cluster to help identify it. |
| Cluster type | Select the type of cluster to create (that is, the type of physical resources it will contain). |
| Members | Click Add to locate and select one or more of the selected resource types that should be added to the cluster. To remove one or more resources from the cluster, select them and click Remove.
NOTE: Only physical resources matching the Cluster type selection can be added to the cluster. |
-

In environments, a logical resource can be bound to a cluster in the same way that any standalone physical resource can. When invoking services on a cluster, the node that manages the clustered resources will determine which resource to use. When recording on a resource in a cluster, all traffic in the cluster will be seen by Rational Integration Tester. If Rational Integration Tester runs a stub, the service will be executed through the stub if it is executed on any of the machines in the cluster.

Database

A database can be used to supply Rational Integration Tester with live test data during the execution of different test resources (for example, tests, test suites, and so on).

Select **General > Database** to add a DB instance with which Rational Integration Tester can interact.



NOTE: Before a database can be accessed by Rational Integration Tester (for example, in a test), the binding between a logical database component and a physical database must be established (see [Change a Component's Bindings](#)).

See [Databases](#) for more information about creating and configuring database resources.

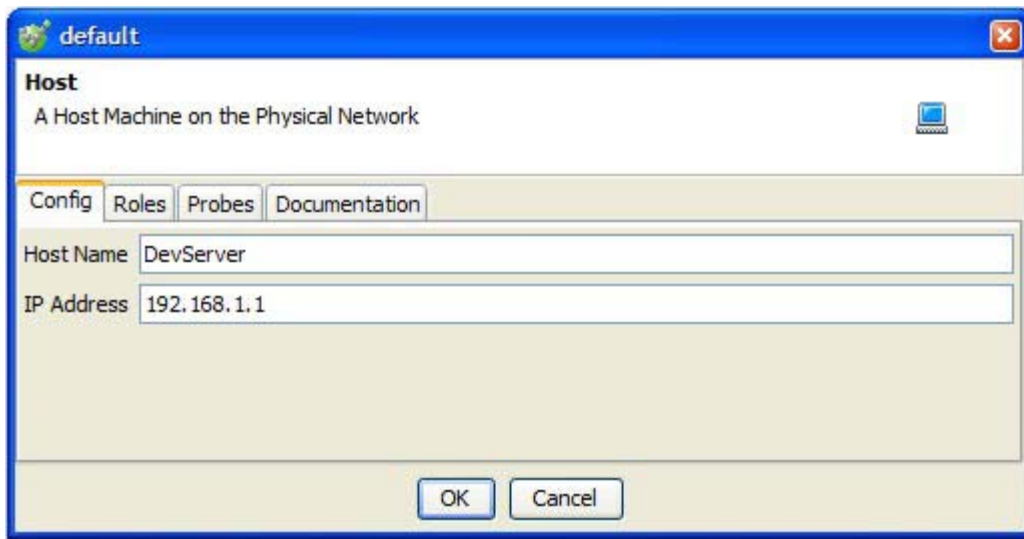
Host

A host can be used to group other physical resources or for adding Rational Performance Test Server Probes to the system.

Select **General > Host** to add a new host machine to the physical network.

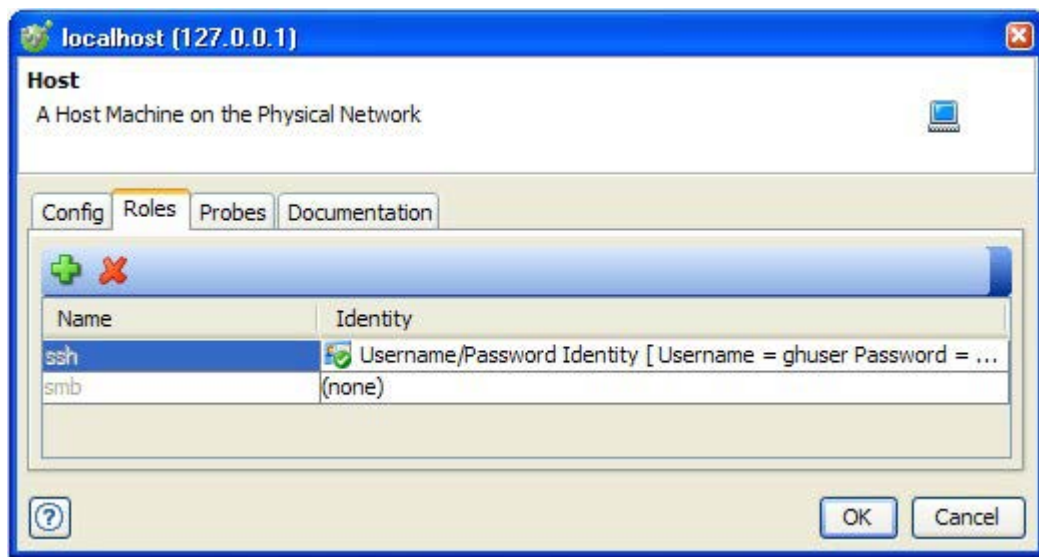
NOTE: For more information about probes, refer to *IBM Rational Performance Test Server Reference Guide*.

Enter the name of the new host and Rational Integration Tester will automatically try to resolve the machine's IP address. If resolution fails, you must add the IP address manually.





NOTE: If the new host is part of a new subnet, the subnet will be added automatically to the Physical View.

Under the **Roles** tab you can manage new or predefined roles that can be utilized to connect to a host remotely.

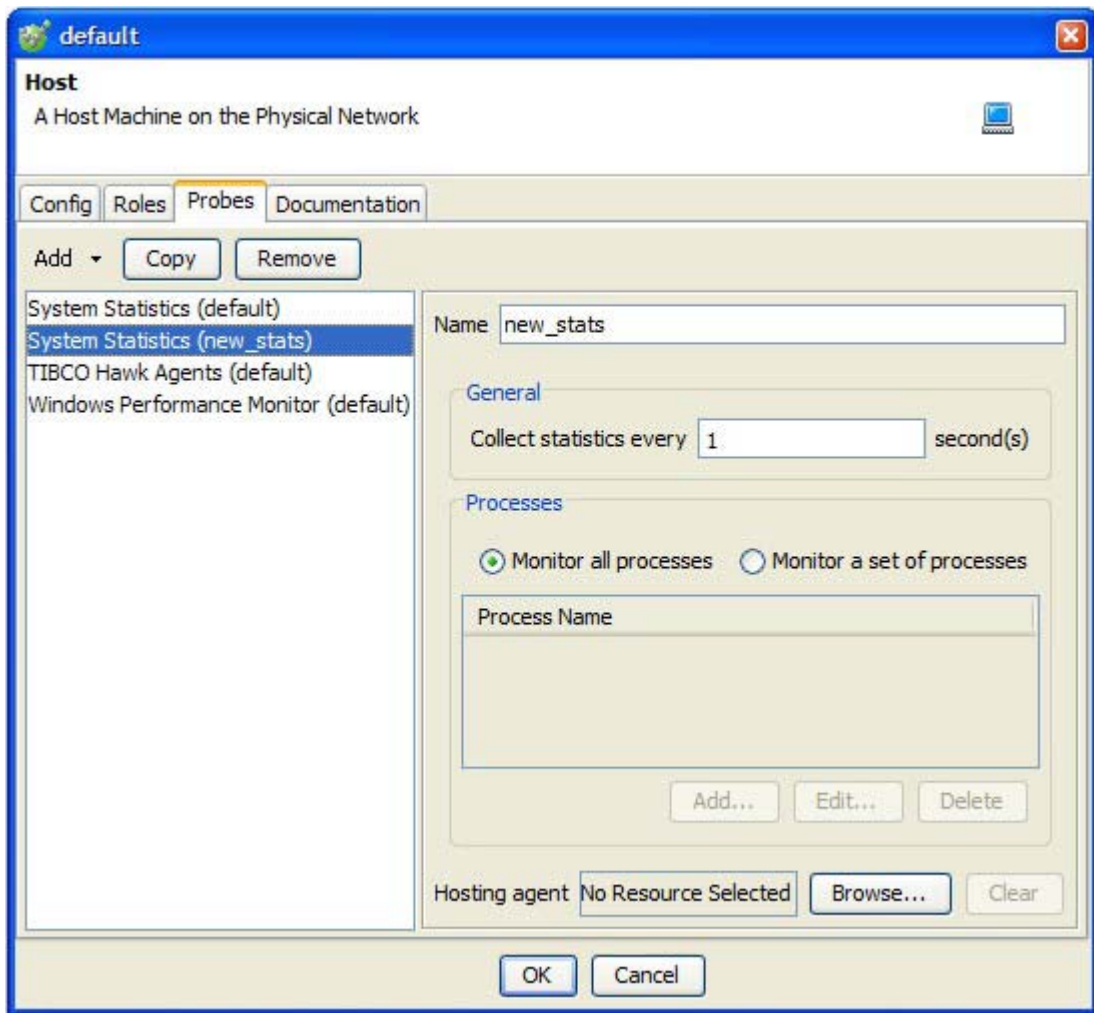


The connection method used by a role is defined by an Identity that has already been configured in the current project (see [Identity](#) for more information). To add an Identity to a role, double-click the **Identity** field for the role and select an Identity in the project resource dialog.

By default, SSH and SMB roles are included for all hosts.

- To add a new role, click the  icon then double-click the **Name** field to provide a name for the role.
- To remove a role other than **ssh** or **smb**, select the role and click the  icon.

Under the **Probes** tab you can manage (that is, add, modify, or delete) the Probes that are configured on the selected host.



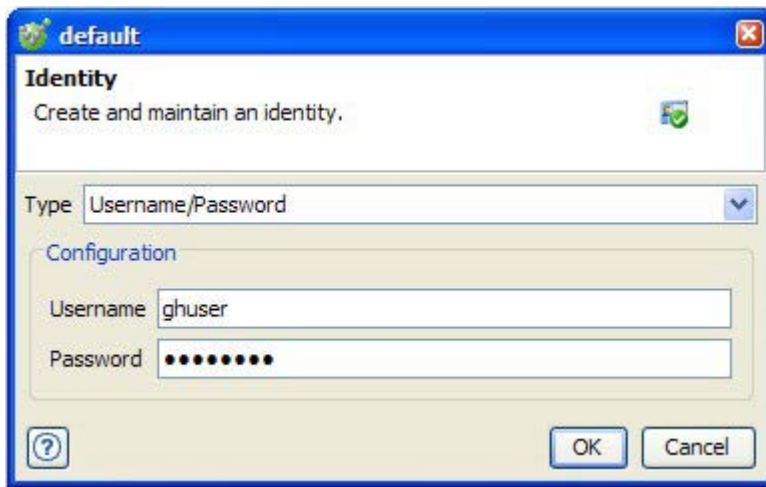
Select a new probe to add from the **Add** menu. To copy an existing probe, select it and click **Copy**. To delete an existing probe, select it and click **Remove**.

NOTE: For more information about adding probes to a host, refer to *IBM Rational Performance Test Server Getting Started Guide* or *IBM Rational Performance Test Server Reference Guide*.

Identity

An identity can be used when it is necessary to authenticate to a third party (for example, when importing or synchronizing with a WSDL using a secure URL, when monitoring a remote log file over SSH, and so on).

Select **General > Identity** to add a new identity to your project.

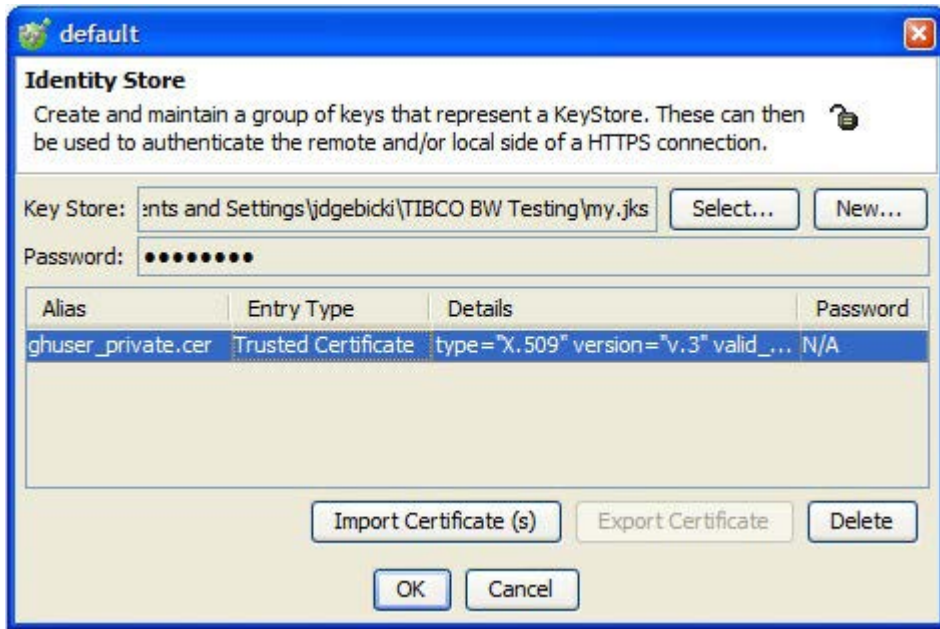


Type	Select the type of authentication to use, either a Username/Password or Certificate/Private Key combination.
Configuration	<p>Enter or select the authentication details according to the selected authentication type, as follows:</p> <ul style="list-style-type: none">• For Username/Password, enter the user ID and password that should be sent to authenticate.• For Certificate/Private Key, select an existing identity store and a private key that it contains that should be used for authentication. See Identity Stores and SSL for more information.• For SSH Private Key, enter a user name and browse to select the appropriate private key. If a passphrase is required for the selected key, enter it in the Passphrase field.

NOTE: The name of the identity resource in the Physical View will reflect the type of authentication and the authentication details.

Identity Store

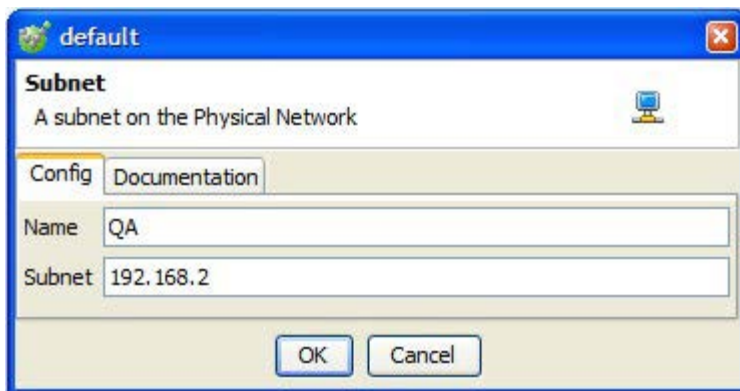
An identity store can be used to enable SSL communications in various messaging transports in Rational Integration Tester. Select **General > Identity Store** to add an identity store (keystore). You can then add keys for use in authenticating connections.



Additional configuration details can be found in [Identity Stores and SSL](#).

Subnet

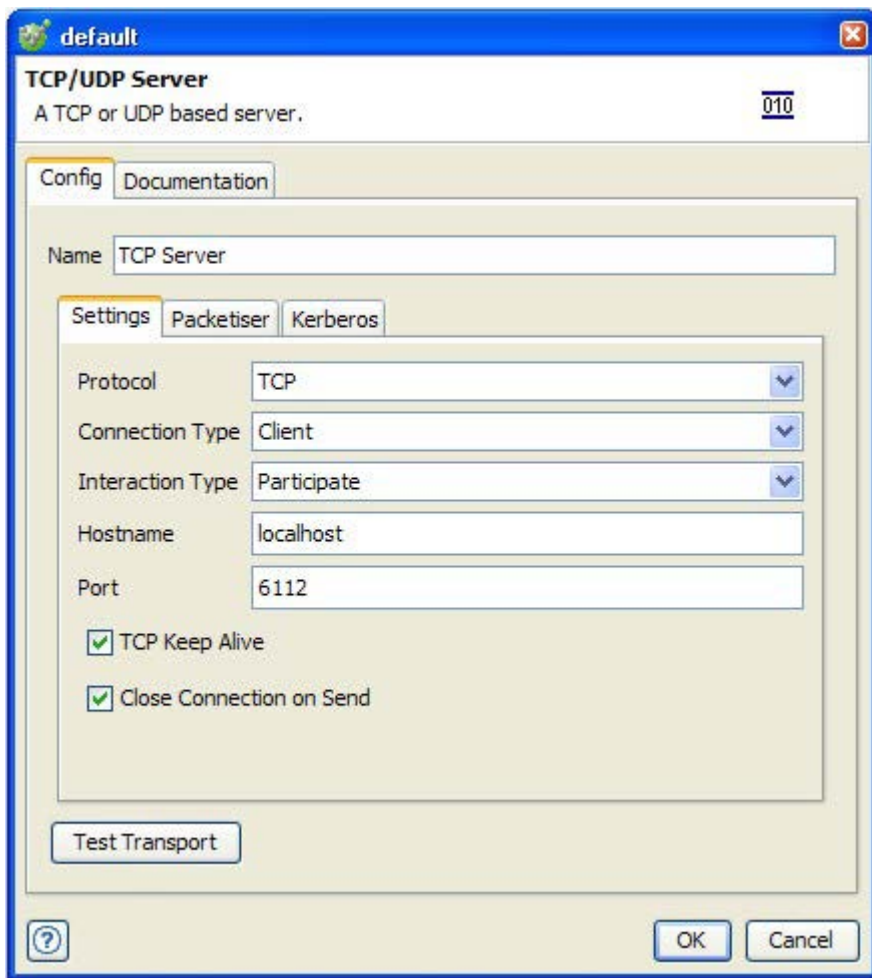
A subnet can be used to group hosts and other physical resources in Rational Integration Tester. Select **General > Subnet** to add a new subnet.



The subnet must be configured with a name and address, which is the first three octets to which all hosts or resources should belong, (for example, 127.0.0 for localhost).

TCP/UDP Server

A TCP/UDP server provides Rational Integration Tester with the ability to communicate between clients and servers using both TCP and UDP based sockets. Select **General > TCP/UDP Server** to add a TCP/UDP server.

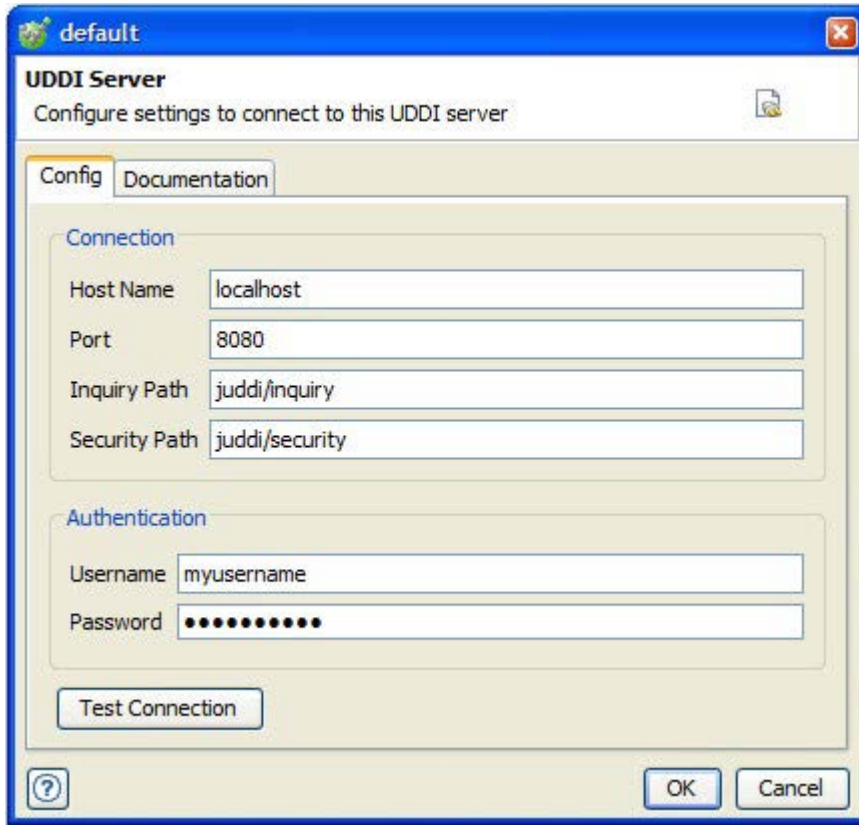


NOTE: Before a TCP/UDP server can be accessed by Rational Integration Tester (for example, in a test), the binding between a logical TCP/UDP connection and the physical TCP/UDP server must be established (see [Change a Component's Bindings](#)).

For more information about configuring TCP/UDP servers, refer to *IBM Rational Integration Tester Reference Guide for TCP/UDP Sockets*.

UDDI Server

A Universal Description, Discovery, and Integration (UDDI) server lets users import WSDL documents from a specified UDDI registry. Select **General > TCP/UDP Server** to add a TCP/UDP server.



The screenshot shows a Windows-style dialog box titled "default" with a sub-title "UDDI Server". Below the title bar, it says "Configure settings to connect to this UDDI server". There are two tabs: "Config" (selected) and "Documentation". The "Config" tab contains two sections: "Connection" and "Authentication". Under "Connection", there are four text input fields: "Host Name" (containing "localhost"), "Port" (containing "8080"), "Inquiry Path" (containing "juddi/inquiry"), and "Security Path" (containing "juddi/security"). Under "Authentication", there are two text input fields: "Username" (containing "myusername") and "Password" (containing a series of dots). At the bottom of the "Config" section is a "Test Connection" button. At the bottom of the dialog box are three buttons: a help button (question mark icon), an "OK" button, and a "Cancel" button.

Enter the server configuration details, which you can obtain from your network or system administrator, as follows:

Host Name	The host name or IP address of the UDDI server to use.
Port	The port on which the server is listening for requests.
Inquiry Path	The server URL to which inquiry requests should be directed.
Security Path	The server URL to which authentication requests should be directed.
Username/ Password	The user name and password to send when connecting to the server.

Click **Test Connection** to check your connection and authentication settings. If the test is unsuccessful, verify the settings and try again.

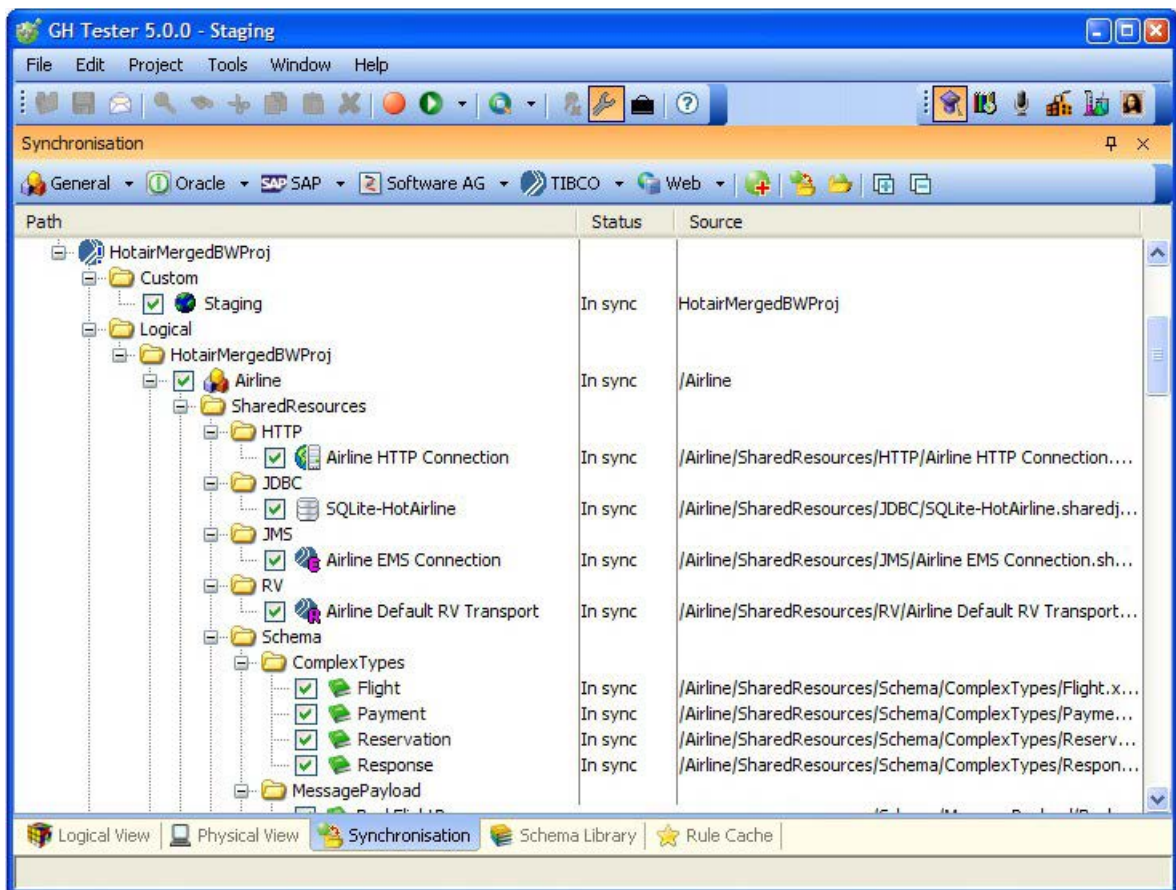
3.3.2 Messaging Transports

The remainder of the available resource types in the Physical View are messaging plugins. Each of these resources represent a transport that can be applied to messaging actions in Rational Integration Tester. The details of these transports are discussed in a collection of Rational Integration Tester reference guides, as follows:

Transport/Resource	Document
EMail Server	<i>IBM Rational Integration Tester Reference Guide for Email</i>
IBM WebSphere Broker	<i>IBM Rational Integration Tester Reference Guide for IBM WebSphere MQ</i>
JMS Brokers	<i>IBM Rational Integration Tester Reference Guide for JMS Messaging</i>
Sonic Brokers	<i>IBM Rational Integration Tester Reference Guide for SonicMQ</i>
SOA Server	<i>IBM Rational Integration Tester Reference Guide for Oracle Fusion</i>
SAP Application Server	<i>IBM Rational Integration Tester Reference Guide for SAP</i>
CentraSite Server	<i>IBM Rational Integration Tester Integration Guide for Software AG CentraSite</i>
My webMethods Server	<i>IBM Rational Integration Tester Reference Guide for Business Process Management Systems</i>
webMethods Broker and Integration Server	<i>IBM Rational Integration Tester Reference Guide for Software AG webMethods</i>
TIBCO BW TRA File, EMS Broker, Rendezvous Daemon	<i>IBM Rational Integration Tester Reference Guide for TIBCO</i>
TIBCO iProcess Server	<i>IBM Rational Integration Tester Reference Guide for Business Process Management Systems</i>
Web Server	<i>IBM Rational Integration Tester Reference Guide for HTTP & Web Services</i>


3.4 Synchronisation

Synchronisation lets you add external resources to your project, view resources that have been added, and synchronize those resources with your local system. Currently, you can synchronize with a WSDL, a webMethods Integration Server, a TIBCO BusinessWorks Project or Design Time Library, a SAP System, and an SCA Domain (Oracle Fusion).



The biggest benefit of synchronisation is that it can create many test artifacts in a very short period of time, and it does it without errors. If you're testing one of the supported resources, synchronisation does all the work of building the system under test automatically.

3.4.1 Add a New Resource

You can add an external resource to the project by selecting it from a specific collection in the toolbar (General, TIBCO, Web, or webMethods) or by clicking the **Add New Item** button .

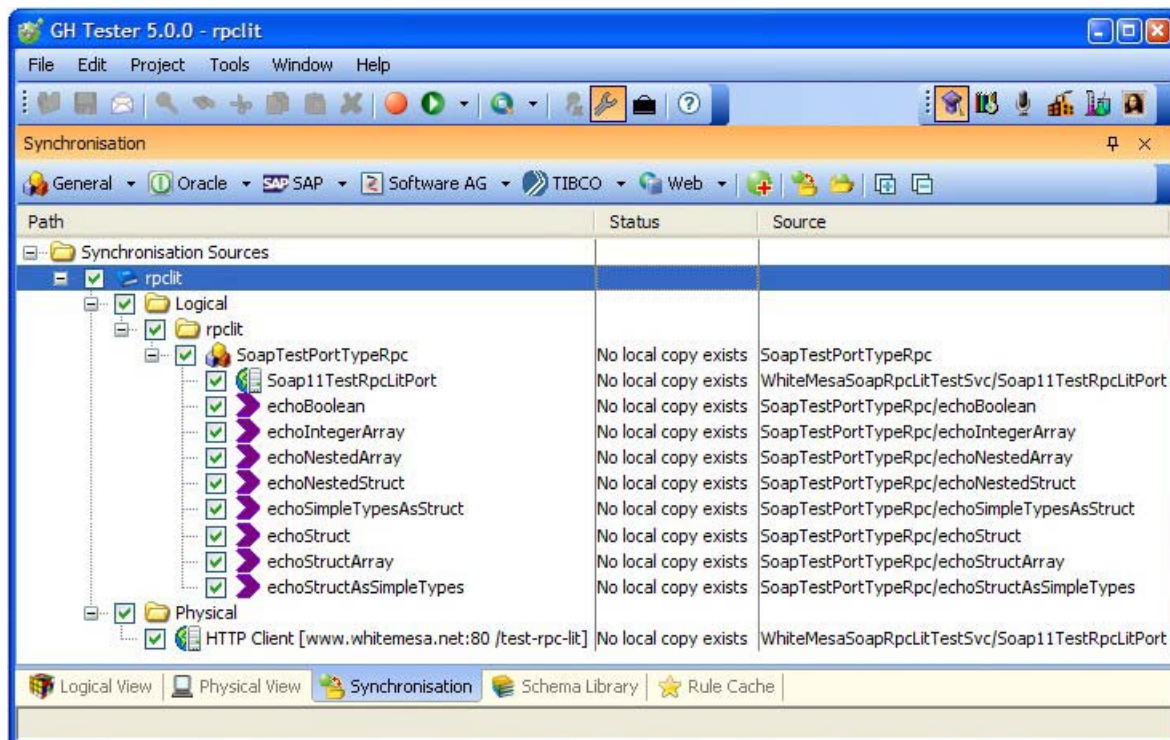
If you select a specific resource, a appropriate wizard is launched to guide you through the import. If you select the **Add New Item** option, you must select the resource type to add before the appropriate wizard is launched.

Currently, the following components can be added as an external (synchronized) resource to a Rational Integration Tester project:

- A webMethods Integration Server Domain (refer to *IBM Rational Integration Tester Reference Guide for Software AG webMethods*)
- A WSDL (refer to *IBM Rational Integration Tester Reference Guide for HTTP & Web Services*)
- A TIBCO BusinessWorks Project or Design Time Library (refer to *IBM Rational Integration Tester Reference Guide for TIBCO*)
- A SAP System (refer to *IBM Rational Integration Tester Reference Guide for SAP*)
- An SCA Domain (refer to *IBM Rational Integration Tester Reference Guide for Oracle Fusion*)

3.4.2 Viewing Resources


After one or more resources have been added to the project, they are displayed in the Synchronisation view, as shown below:



The synchronisation view displays the following information about added resources:

Path	Displays the test artifacts that have been generated for the resource in the same structure that exists in the original resource.
Status	Indicates the current status of the selected artifact, as follows: No local copy exists: No local artifacts have been created. In sync: A local artifact has been created and it is in sync with the external resource. Source copy updated: The external source has been updated since the last synchronisation, so the unedited local copy could be out of date. Local copy updated: The local copy of the artifact has been modified since the resource was added. Source and local copies updated: Both the local copy of the resource and its source have been modified since the resource was added. Source copy no longer exists: The selected artifact has been deleted from the synchronisation source (for example, from the BusinessWorks project) but still exists in the Rational Integration Tester project. Unknown state: The current status of the artifact can not be determined.
Source	The full path in the external resource from which the artifact originated.

3.4.3 Checking Synchronisation

Once you have added an external resource or if you have made changes to the local project, you can check synchronisation to update the status of the resource's local artifacts. To check synchronisation, click the **Check Synchronisation** button  in the toolbar.

3.4.4 Selecting Artifacts for Synchronisation

After adding a resource to your project, all of the generated artifacts are enabled (selected) for synchronisation by default. If desired, though, you can disable any number of artifacts by deselecting their node in the synchronisation tree.



- To enable/disable an artifact for synchronisation, tick/untick the check box next to the artifact name.

NOTE: Selecting or deselecting the root of the resource or any sub-tree will select or deselect all artifacts within the same branch of the tree.

- To enable/disable all artifacts for synchronisation, right-click any node in the tree and select **Select All Paths/Deselect All Paths** from the context menu.

3.4.5 Expanding and Collapsing Artifacts

The artifacts generated by adding an external resource to your project are organized in a tree structure. The entire tree or any one or more branches within it can be expanded or collapsed.


- To collapse the selected branch and all branches below it, click the **Collapse All** button .
- To expand the selected branch and all branches below it, click the **Expand All** button .

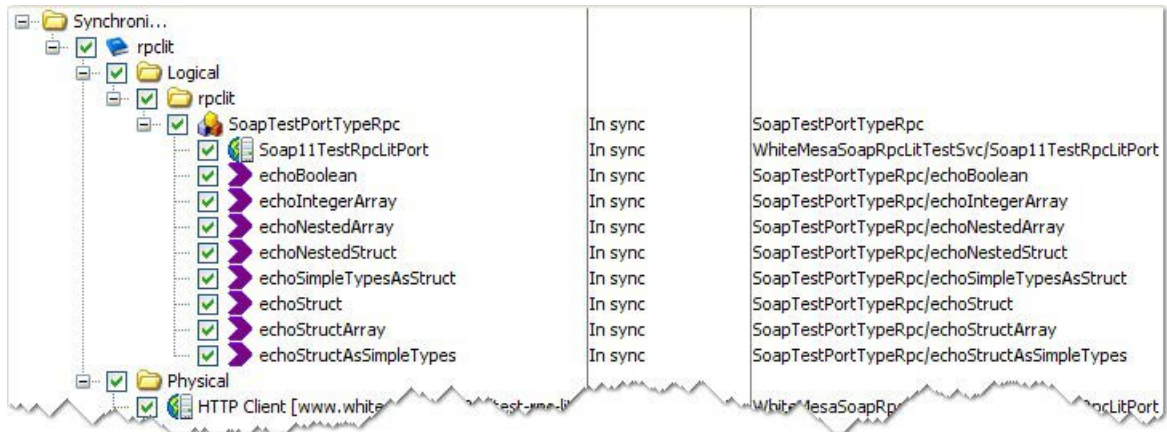
3.4.6 Resolving Artifacts

If changes are made to both the local copy and source of a test artifact, the changes can be resolved to indicate that you are satisfied with the.

For example, if you edit a BW transport in Rational Integration Tester and edit the same transport in the BW project, the local copy will be merged following synchronisation. To “approve” the change, right-click the artifact and select **Mark as resolved** from the context menu.

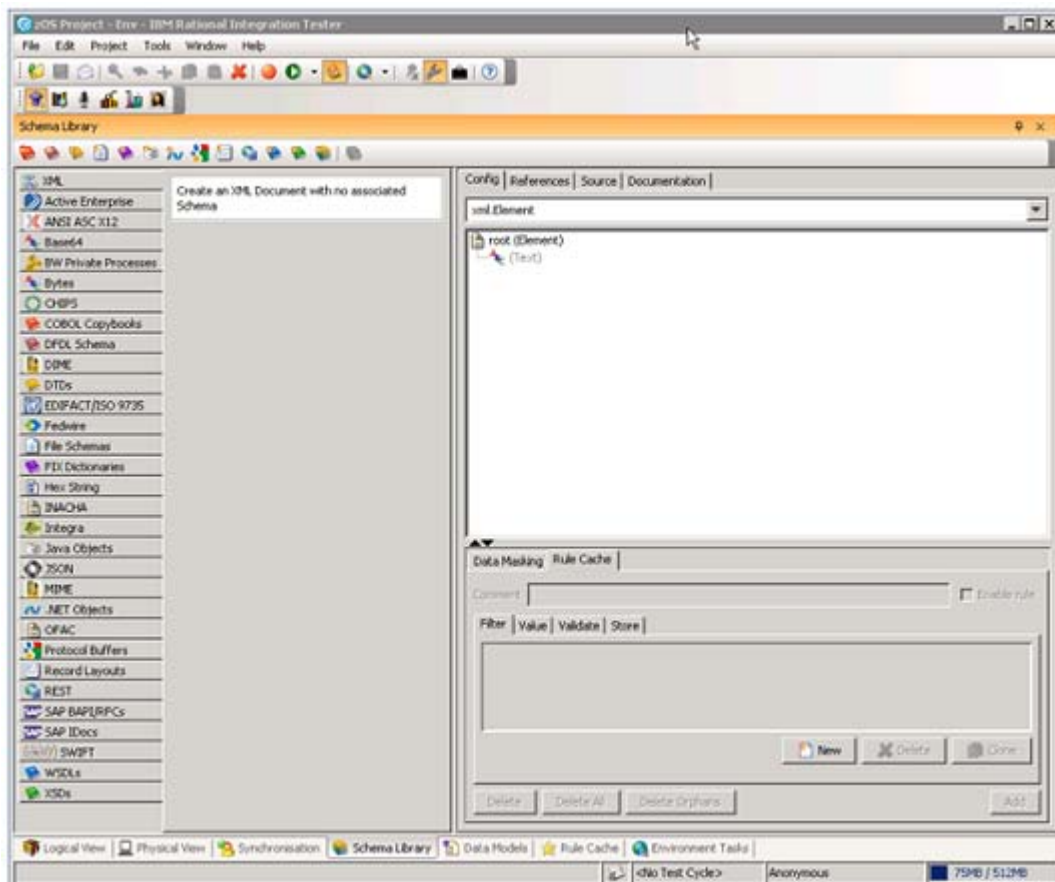
3.4.7 Synchronizing Resources

When you are ready to synchronize your project with external resources, click the **Synchronize** button  in the toolbar. Rational Integration Tester will scan the external resources and create all of the test artifacts that you will need to successfully test the resource.



3.5 The Schema Library

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

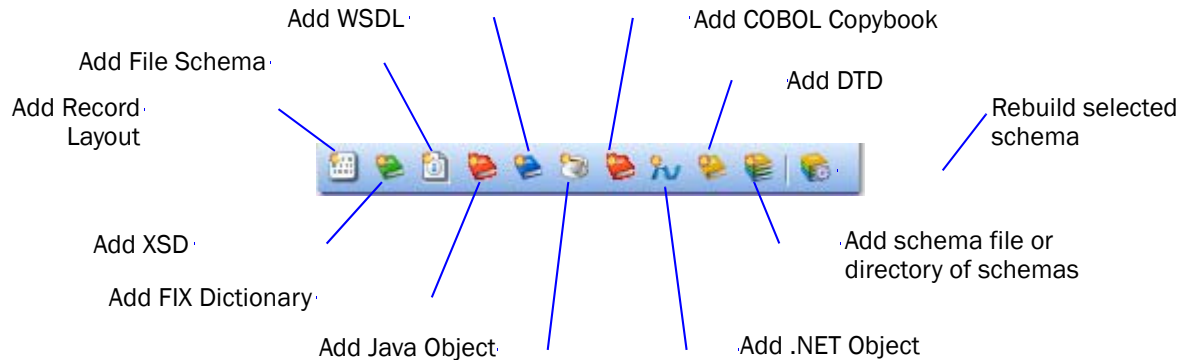



Each schema category is listed on the left side of the view. When a category is selected, information about and options for the category are displayed in the center view.

When you select an option (where provided) in the center view, you can view and update the schema information in the right view. When a schema is selected in the library, the contents of the schema are refreshed with its source.

3.5.1 Adding Schemas

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



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

3.5.2 Schema Details

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

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

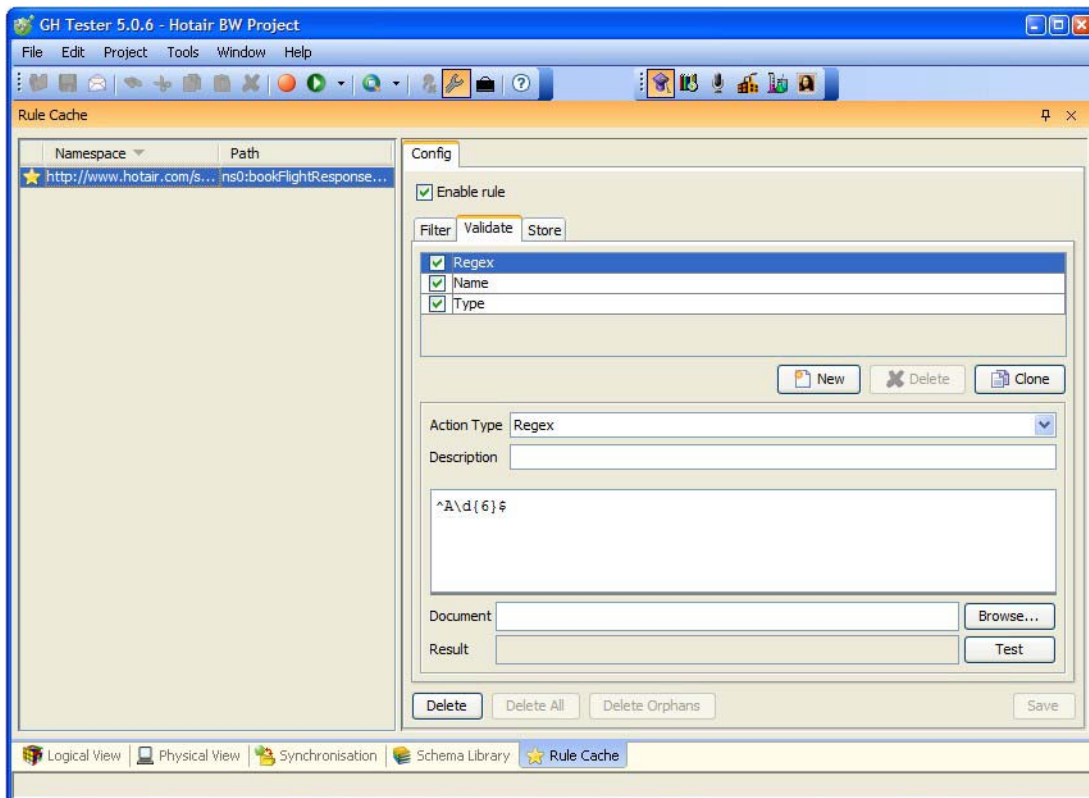
3.5.3 Format Details

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

3.6 The Rule Cache

When validation errors are repaired and cached (see [Repair Validation Failures](#)), the repair method is saved as a validation rule. These rules will be applied to the same field from which they are created when the same schema is in use.

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



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

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

Under the **Config** tab on the right side of the view, the selected rule can be enabled or disabled with the **Enable rule** option. Validation actions are listed under the **Validate** tab. Actions can be created, modified, cloned, and deleted, the same way as in a message editor (see [Messages](#)). When the selected action includes a rule, it is displayed

below. The rule can be edited, and a description can be added if desired.

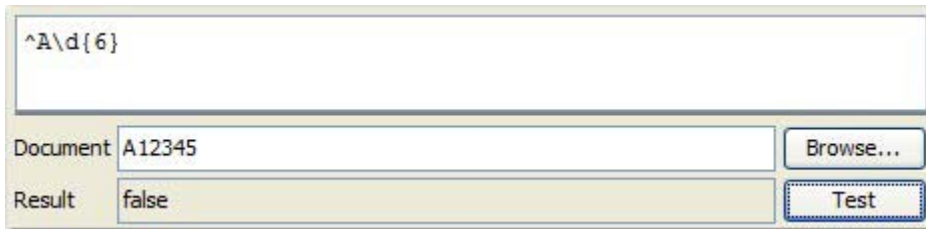
The screenshot shows a 'Config' window with the following elements:

- Enable rule:** A checked checkbox.
- Tabs:** 'Filter', 'Validate', and 'Store'. 'Filter' is the active tab.
- Rule List:** A table with columns 'Regex', 'Name', and 'Type'. The 'Regex' column is highlighted in blue. There are three rows, each with a checked checkbox in the 'Regex' column.
- Buttons:** 'New' (with a document icon), 'Delete' (with an 'X' icon), and 'Clone' (with a document icon).
- Action Type:** A dropdown menu currently showing 'Regex'.
- Description:** An empty text field.
- Regex Field:** A text field containing the pattern `^\\d+?$`.

At the bottom of the **Config** tab, you can click **Delete** to delete the selected rule or click **Delete All** to remove all rules from the selected schema. Click **Delete Orphans** to remove all rules from the selected schema that are associated with fields that no longer exist (due to changes to the schema). This button is only available when “orphaned” rules actually exist for the schema.

3.6.1 Testing Rules

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

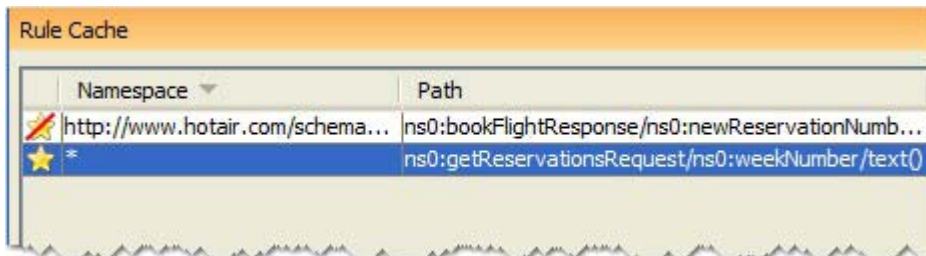


The screenshot shows a dialog box for testing a rule. At the top is a large text field containing the regular expression `^A\d{6}`. Below this are two rows of controls. The first row is labeled 'Document' and contains a text field with the value 'A12345' and a 'Browse...' button. The second row is labeled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

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

3.6.2 Generic Rules

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

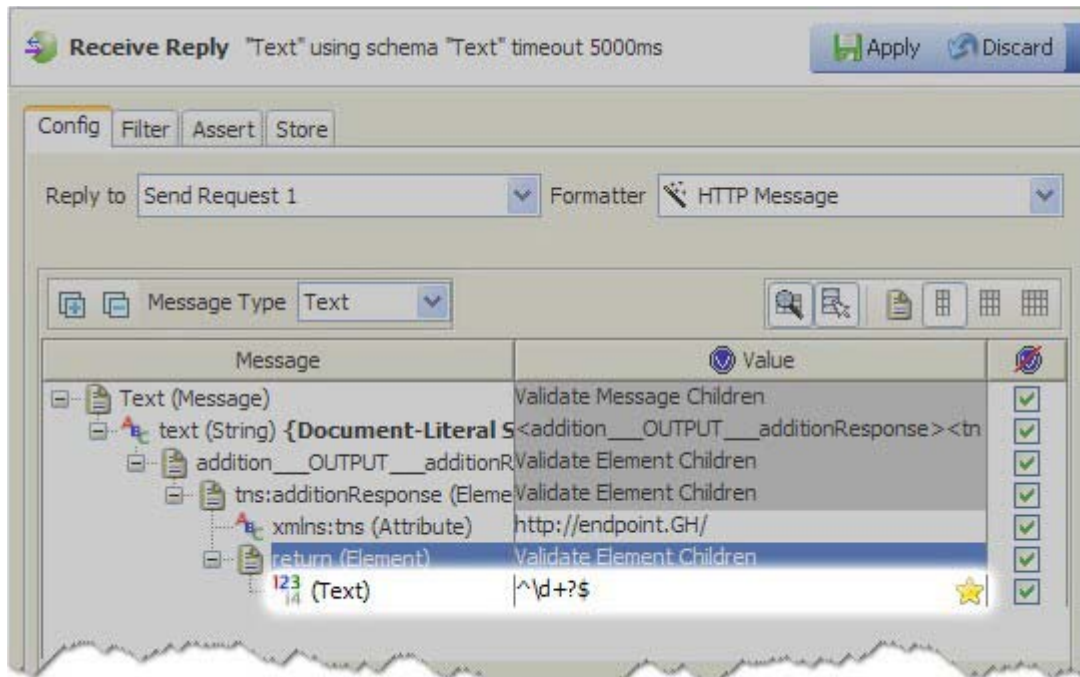


The screenshot shows the 'Rule Cache' view, which is a table with two columns: 'Namespace' and 'Path'. The first row has a red star icon in the 'Namespace' column, the value 'http://www.hotair.com/schema...' in the 'Namespace' column, and 'ns0:bookFlightResponse/ns0:newReservationNumb...' in the 'Path'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

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

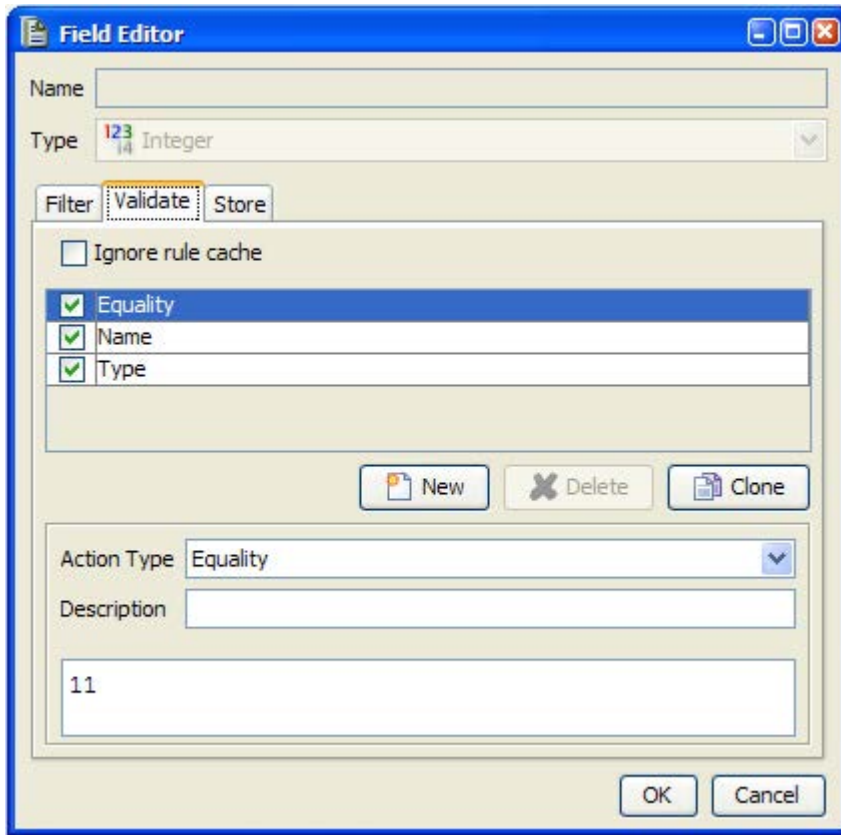
3.6.3 Rules in Message Fields

When viewing messages in an action editor, the status of rules on message fields is indicated in the same way as in the Rule Cache view.

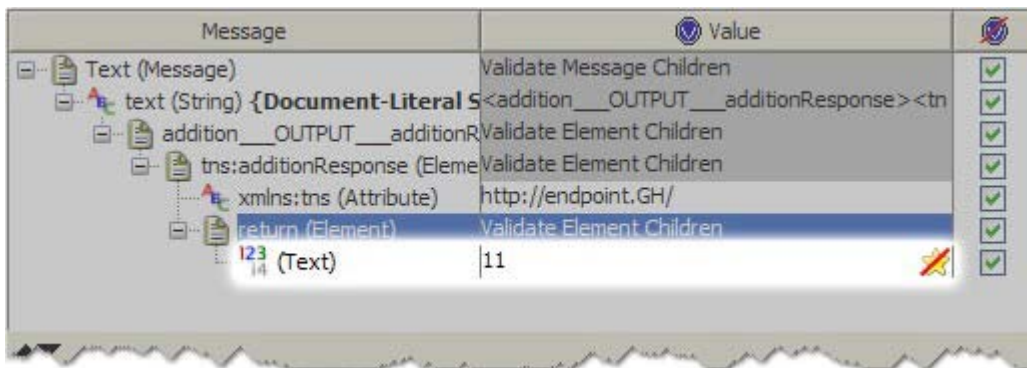


In the example shown above, the highlighted message field is using a validation rule.

If you double-click the field to open it in the field editor, you will see the local value that is used in the field.

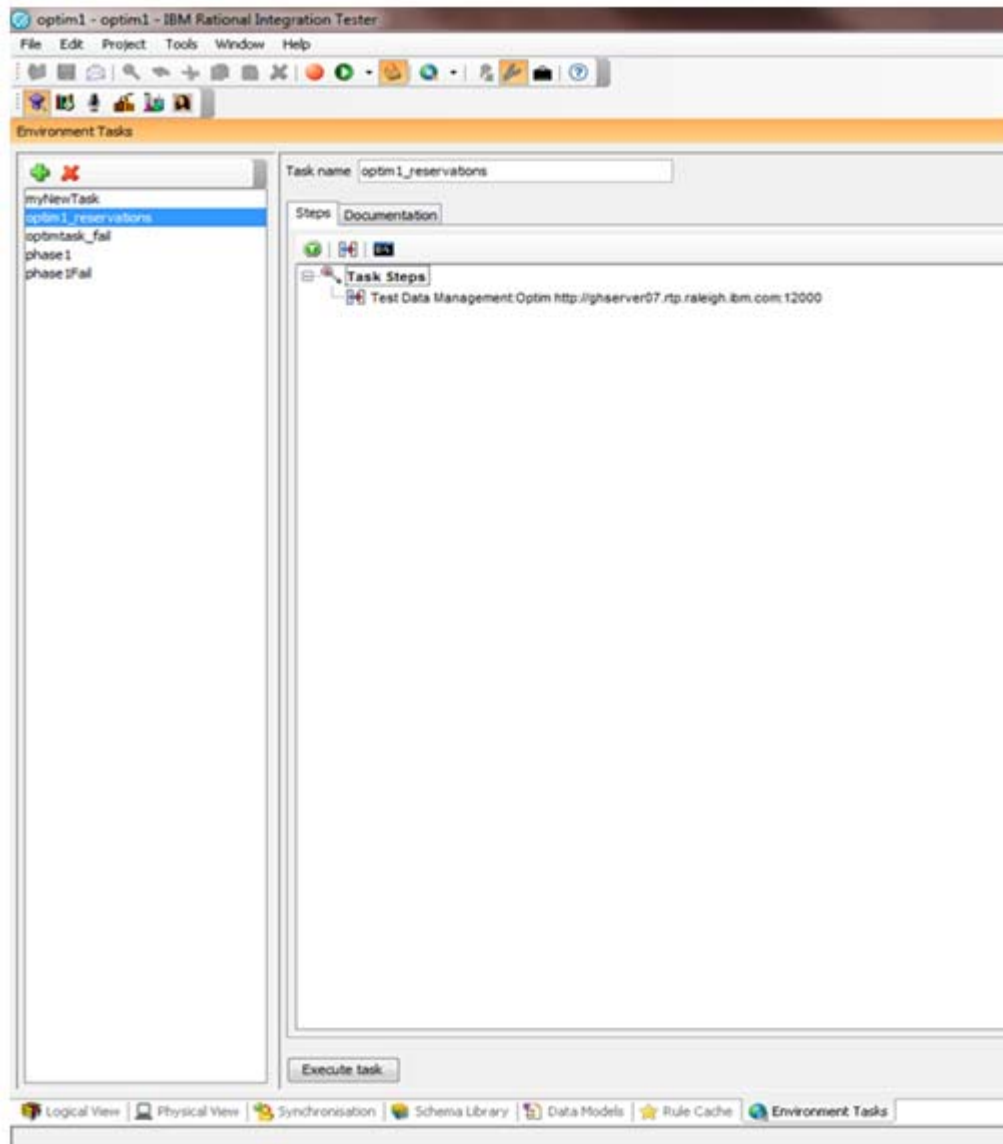


If you disable the rule (that is, tick the box next to **Ignore rule cache**), the icon in the action editor changes accordingly, and the local value for the field is displayed again.



3.7 Environment Tasks

Use the Environment Tasks view to configure test environments. You can define environment configuration tasks by creating test-data-management or run-command actions. You can then associate these environment tasks with stubs to ensure that test environments are set up correctly before the stubs are run.



For example, you could create an environment task that calls out to a test data management system to refresh test data used by stubs. This version of IBM Rational Integration Tester supports integration with IBM InfoSphere Optim (a test data management system). You could use InfoSphere Optim to create test data from a

customer database that contains private information. Using the masking capabilities in InfoSphere Optim, you could mask the sensitive data fields, and then use that data for realistic testing. In Rational integration Tester, you could create an environment task that would invoke InfoSphere Optim to populate a database as part of setting up the test environment, so that stubs could be run using data in a known state.

Alternatively, a run command action could be used to invoke a script or program, for example, to clear queues in a messaging system.

The high-level tasks to define and use environment tasks are as follows:

1. If you decide to use the test data management action, define the InfoSphere Optim provider and the `runservice-tool.jar` file in Library Manager.
2. Create environment tasks in the Architecture School perspective.
3. Associate the environment tasks with stubs in the Test Factory perspective so that the environment tasks run first.
4. Run the stub in Rational Integration Tester or publish it to Rational Test Control Panel, and then deploy it to a computer running a Rational Test Virtualization Server agent. The associated environment tasks will execute before the stub is started.

The first three tasks are documented in the following sections. The remaining tasks are standard Rational Integration Tester procedures, and are explained in *IBM Rational Test Virtualization Server Reference Guide*.

3.7.1 Defining the Provider and JAR file in Library Manager

To define the InfoSphere Optim provider and the `runservice-tool.jar` file:

1. Click **Start > (All) Programs > IBM Rational Integration Tester > Library Manager**


The Library Manager application window is displayed.

2. Click **Test Data Management**.
3. In **Provider**, add InfoSphere Optim.
4. In **Settings**, add the resource name and path for the `runservice-tool.jar` file. (This file is included in the InfoSphere Optim Designer.)
5. Click **OK**.

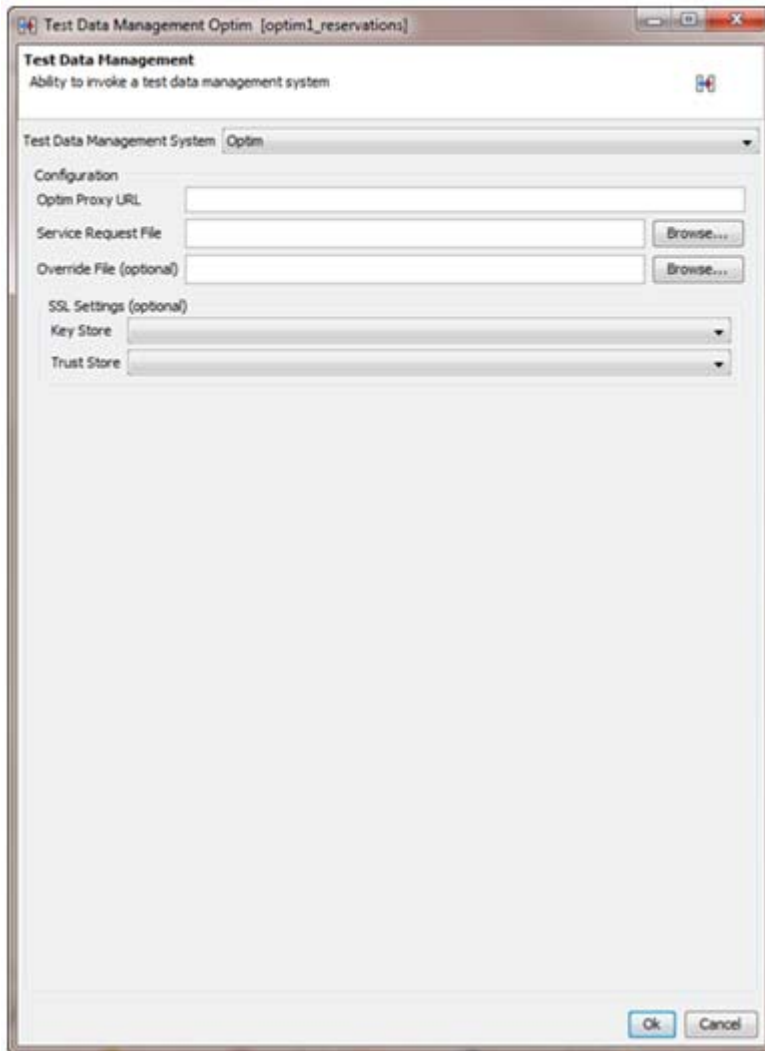
3.7.2 Creating Environment Tasks

In Environment Tasks view, you can create test data management actions or run command actions.

To create environment tasks for configuring test environments:

1. In Architecture School perspective, click the **Environment Tasks** tab.
2. In the left panel, click the plus sign (+) and provide a name for the new environment task.
3. Click the **Test Data Management action** icon () and double-click the new task step.

The Test Data Management screen is displayed.

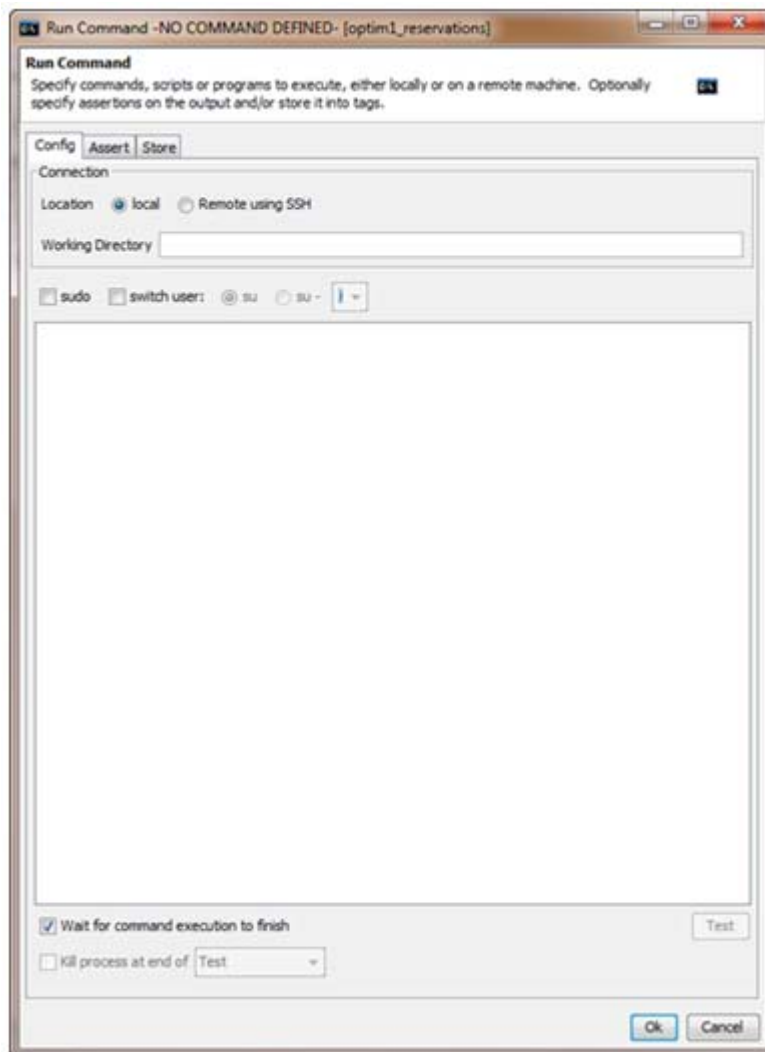


4. In the **Optim Proxy URL** field, enter the URL of the InfoSphere Optim proxy server.
5. In the **Service Request File** field, enter the location of a service request JAR file that has been exported from InfoSphere Optim Designer.
6. **Conditional:** If the Optim Proxy URL uses HTTPS, using the lists provided, select the SSL settings for key store and trust store.
7. Click **OK**.

To create a run command action:

1. In Architecture School, click the **Environment Tasks** tab.
2. In the left panel, click the plus sign (+) and provide a name for the new environment task.
3. Click the command prompt icon (⌨).
4. Right-click the new task.

The Run Command screen is displayed.



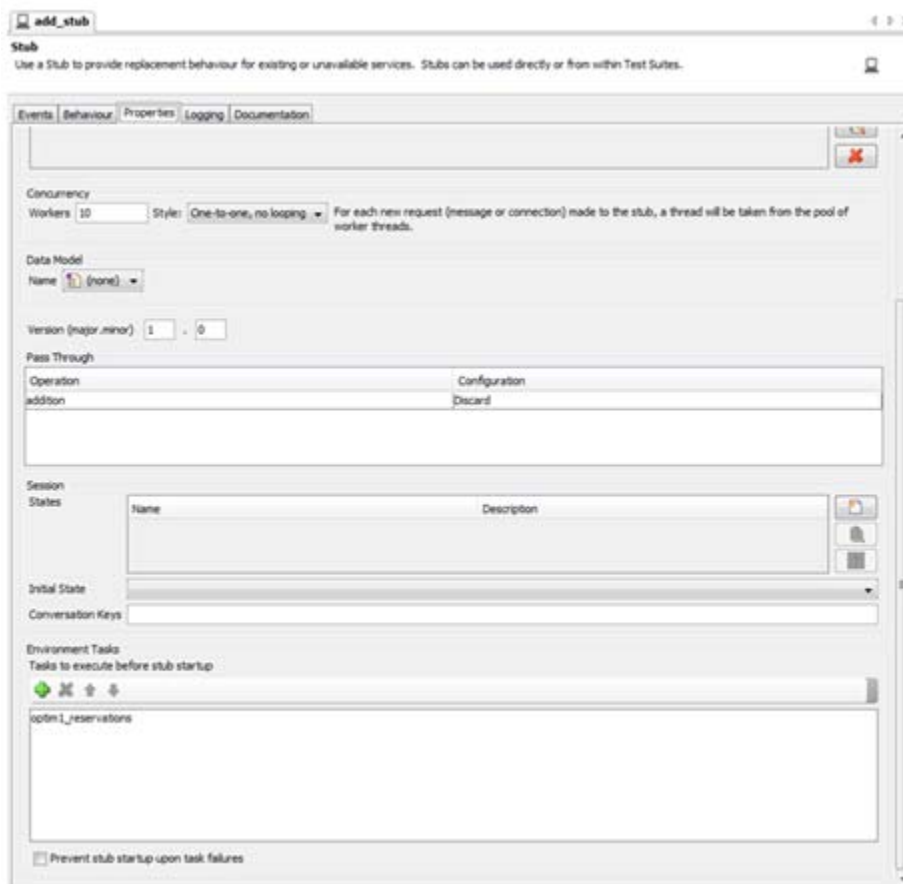
5. Specify commands, scripts or programs to execute, either locally or on a remote computer.

6. **Optional:** Specify assertions on the output or store them in tags.
7. Click **OK**.

3.7.3 Associating Environment Tasks with Stubs

In the Test Factory view, either create a new test or stub, or open a stub for editing.

1. Click the **Properties** tab.



2. In the **Environment Tasks** section, click the plus sign icon (+).

The Environment Tasks window opens and displays all the environment tasks that were created in Architecture School.



3. Select one or more environment tasks.

NOTE: You can select multiple tasks by holding the CTRL key and clicking the tasks.

4. Click **OK**.

The tasks display in Environment Tasks.

5. **Optional:** Select the **Prevent stub startup upon task failures** check box.

Requirements Library

Contents

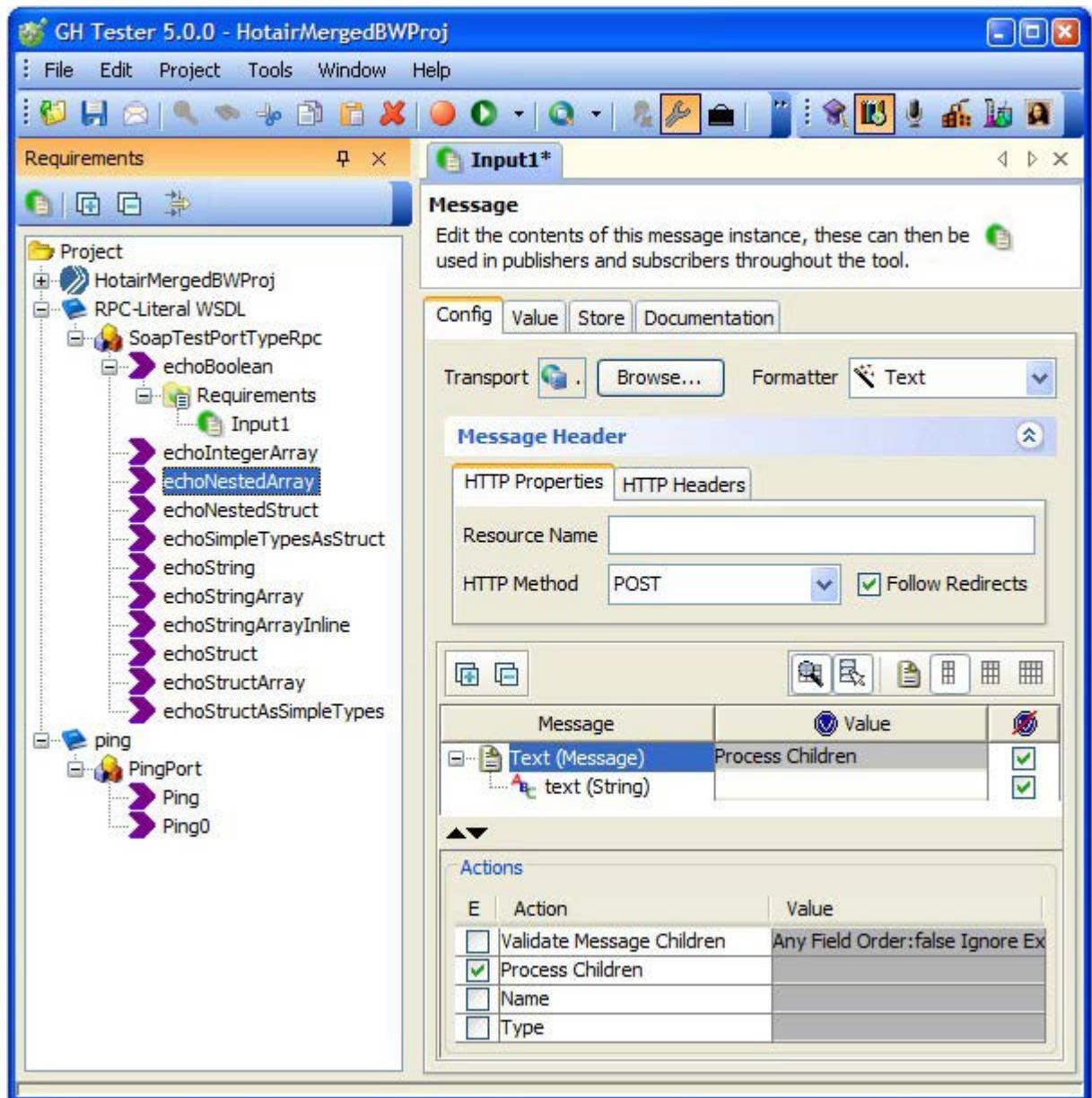
Overview

Creating Messages

This chapter provides an overview of Rational Integration Tester's Requirements Library perspective, including details about how to create messages that can be used to build tests in other parts of Rational Integration Tester.

4.1 Overview

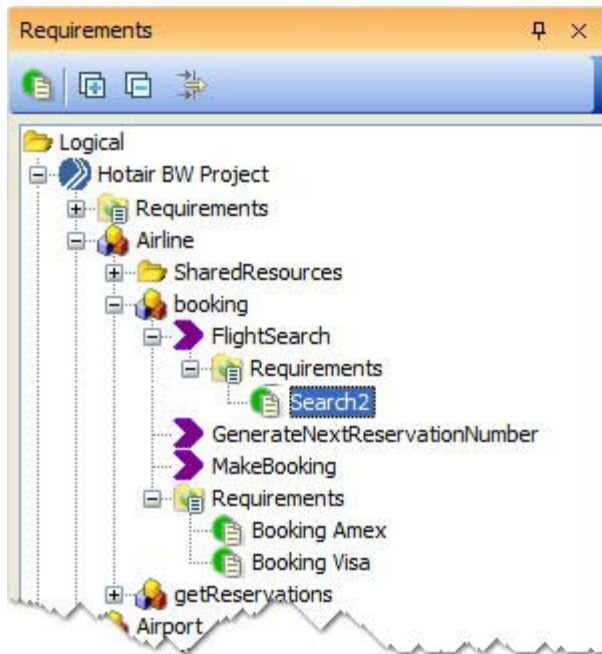
Testing requirements can be defined by the messages that are used to test an operation. The Requirements Library allows you to create any number of messages that can be dragged into messaging actions in the Test Factory, saving time and mistakes when it comes to building tests.



NOTE: Messages can be created under service components, operations, or virtual folders.

4.1.1 The Requirements Tree

The requirements tree, docked on the left side of the Requirements Library perspective, displays all of your project resources that may contain requirements. As requirements are added, they are automatically organized in a “Requirements” folder under the folder, component, or operation in which they are created.



To focus the view on a specific item, right-click it and select **Promote to Root** from the context menu. To restore the normal view, click the root of the tree and select **Demote from Root** from the context menu.

4.1.2 The Requirements Toolbar

The toolbar in the Requirements Library contains shortcuts for creating messages and for manipulating what is displayed in the tree.

Create a new message.

Filter the component view



Expand the current branch.

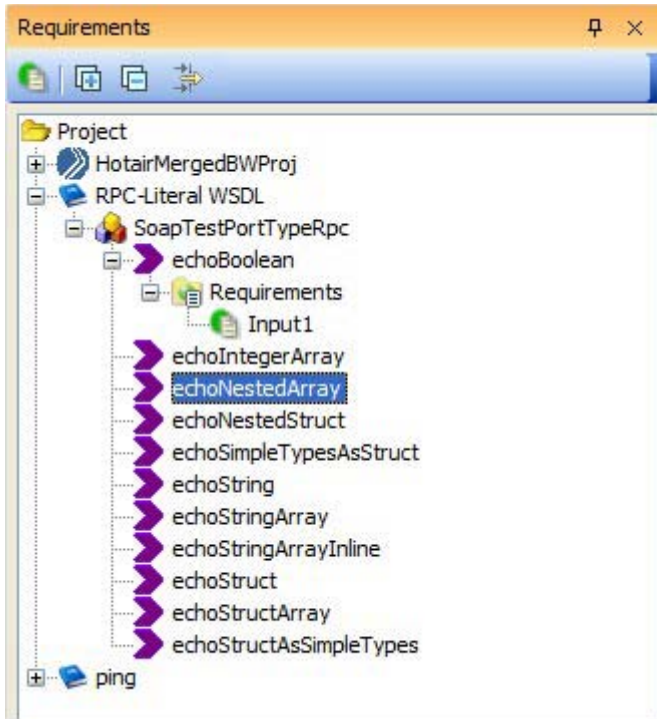
Collapse the current branch.


The “Create a New Message” icon is not available when the root of the tree is selected.

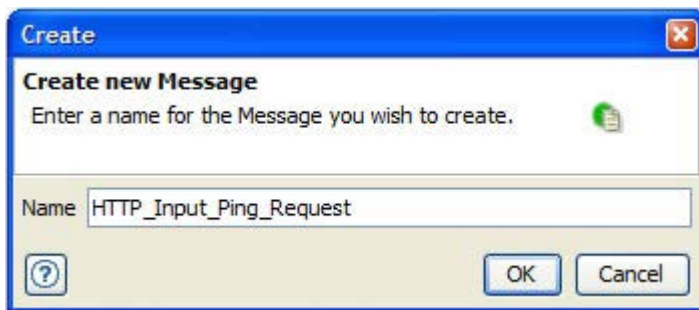
4.2 Creating Messages

Follow the steps below to create a message in the Requirements Library:

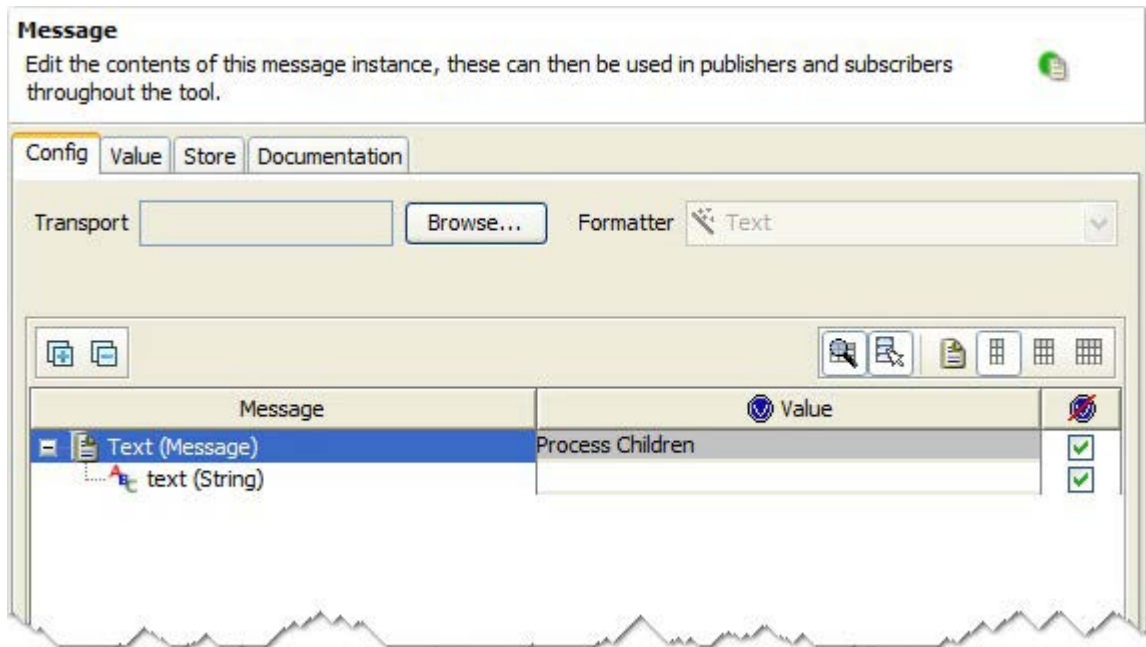
1. In the requirements tree, locate and select the folder, component, or operation in which you want to create a message.



2. Click the **Create a new Message** icon  at the top of the tree.
3. When prompted, enter a name for the new message and click **OK** to proceed.



-
4. The new (empty) message will be created and opened for editing in the work area to the right of the project tree.



5. You can now build the message that can be used later to test your desired requirements.

NOTE: See [Messages](#) for more information about building the different message components (that is, transport and formatter, header, body, validation options, and so on).

NOTE: Messages that are created in the Requirements Library are displayed under their respective operations, in a *Requirements* folder, in the Test Factory perspective.

NOTE: Messages can be dragged and dropped on the root of a Message Switch action, replacing existing data or adding new message cases. If new cases are added, you can opt to copy only the message structure, use the message as a filter, use the message as an assertion, or all three. If the “Structure” option is selected, the message structure and all values are copied, but filtering and assertions are disabled for all fields.

Recording Studio

Contents

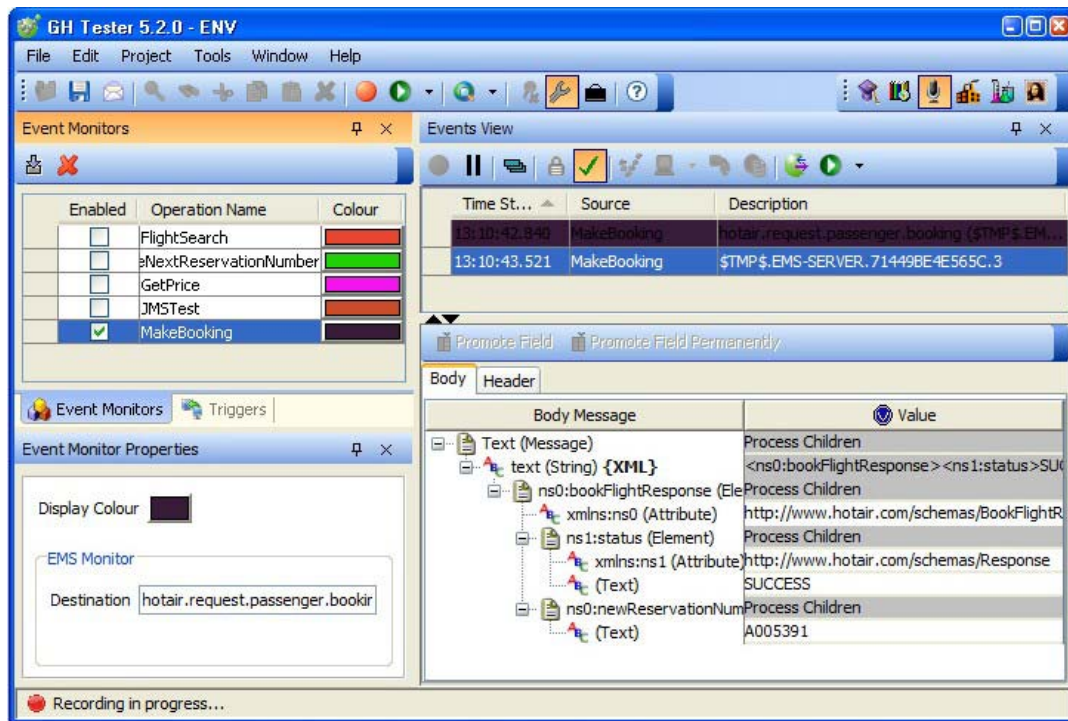
Overview

Using the Recording Studio

This chapter provides an overview of Rational Integration Tester's Recording Studio perspective, including details about how to observe and capture events generated by system components and use these events to generate tests, stubs, triggers, and requirements.

5.1 Overview

The Recording Studio lets you monitor and record system events. You can then use the recorded events to create tests, stubs, triggers, and requirements (messages).



NOTE: The following transports can be monitored and recorded in the Recording Studio: HTTP, IBM Websphere MQ, TIBCO EMS, and webMethods. For information about recording MQ events, refer to [Recording IBM Websphere MQ Events](#). For information about recording TIBCO EMS events, refer to [Recording TIBCO EMS Server Events](#).

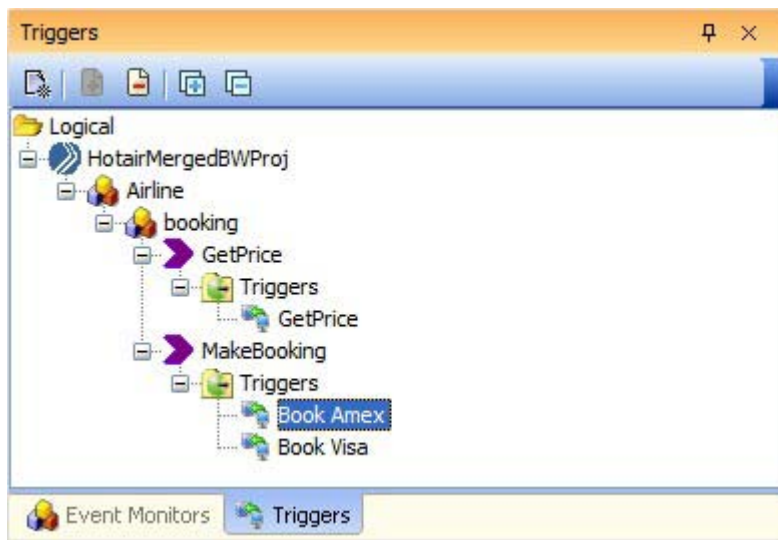
NOTE: Generic JMS topics can be recorded using the topic that is configured within the operation.

NOTE: For TIBCO EMS, if security is enabled on the EMS server, a valid JNDI user name and password are required (in the configuration of the physical TIBCO EMS Broker) to monitor events.

The Recording Studio contains four views: Triggers, Event Monitors, Event Monitor Properties, and the Events View. Each view is described in the following sections.

5.1.1 Triggers

A trigger is essentially a publisher that is used to send an event to the system under test. Existing triggers are displayed in the Triggers view, which is located in the upper-left portion of the Recording Studio perspective. It shares the same space with the Event Monitors view. To show the Triggers view, click the **Triggers** tab at the bottom of the view.

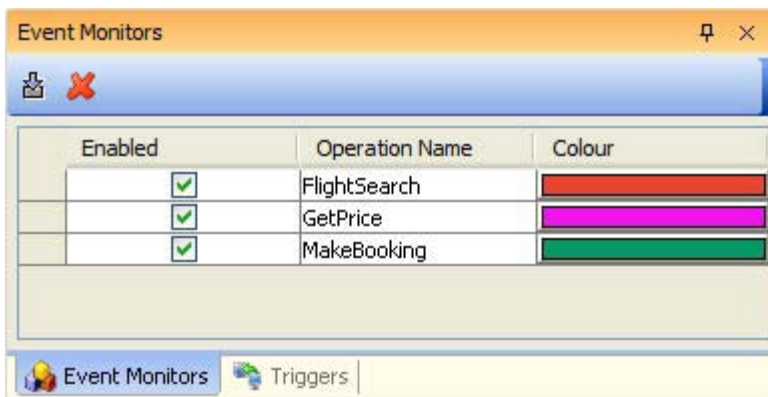


Triggers are displayed within the project tree, in a Triggers folder under the containing operation.

Like other resource trees within Rational Integration Tester, the contents of the Triggers view can be expanded or collapsed using the appropriate toolbar icons.

5.1.2 Event Monitors

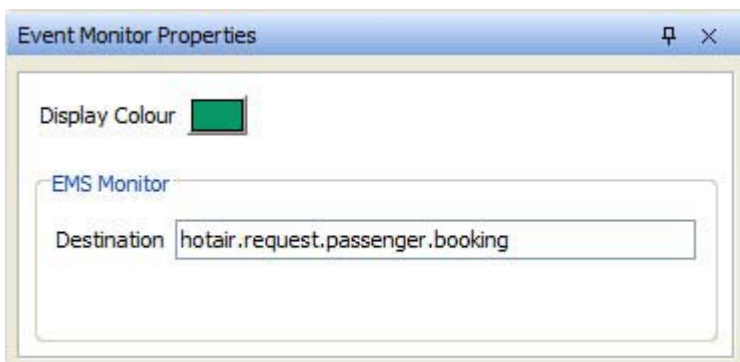
Operations that have been selected for monitoring are displayed in the Event Monitors view, which is located in the upper-left portion of the Recording Studio perspective. It shares the same space with the Triggers view. To show the Event Monitors view, click the **Event Monitors** tab at the bottom of the view.



The view displays operations in a grid. The monitoring status (enabled or disabled) is displayed for each operation, along with the background color that is used (in the Events View) to distinguish events captured for the corresponding operation. The highlight color can be changed in the operation (see [Change an Operation's Recording Studio Color](#)).

5.1.3 Event Monitor Properties

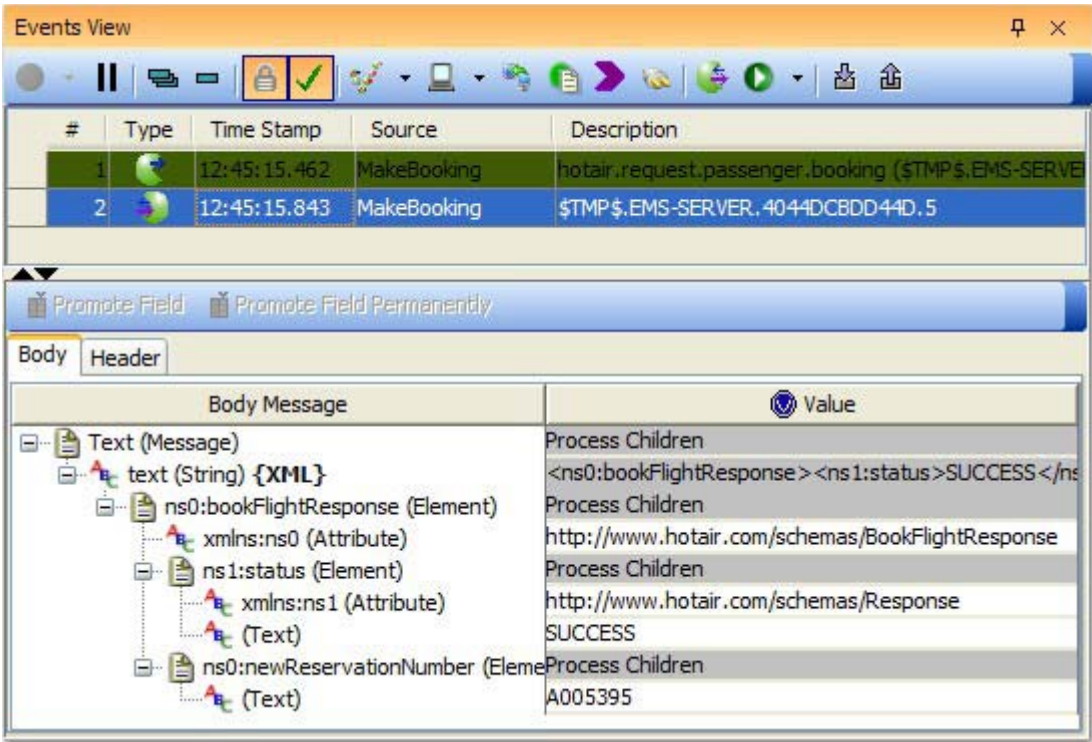
The Event Monitor Properties view, located directly below the Event Monitors and Triggers view, displays information about the operation that is currently selected in the Event Monitors view.



The display color (used to highlight the operation's events in the Events View) is displayed, as well as the destination that is being monitored for the operation.

5.1.4 Events View

The Events View displays events that have been captured by enabled event monitors. Located on the right side of the Recording Studio perspective, the view is divided into a list of captured events on top and message details (body and header) for the selected event on the bottom.



The events table displays the captured events and includes the event sequence ID (indicating the order in which the selected event was captured during this Rational Integration Tester session, relative to other events), the event type (indicating whether the event is a request or a reply) , the time when the event was captured, the source of the event, and the event description.

Captured events can be saved as tests, stubs, triggers, or requirements. Within the view you can send a response to selected events or run and manage triggers.

5.2 Using the Recording Studio

This section describes how to use the features and functions in the Recording Studio to monitor and capture events, as well as how to use captured events to tests and other artifacts for use within your Rational Integration Tester project.

- [Recording IBM Websphere MQ Events](#)
- [Recording TIBCO EMS Server Events](#)
- [Recording HTTP\(S\) Traffic](#)
- [Recording TCP Traffic](#)
- [Recording SQL](#)
- [Adding Operations to Event Monitor View](#)
- [Removing Monitored Operations](#)
- [Creating New Standalone Triggers](#)
- [Running Triggers](#)
- [Adding a Trigger to the Triggers Menu](#)
- [Removing a Trigger from the Triggers Menu](#)
- [Capturing Events](#)
- [Managing the Events View](#)
- [Creating Test Resources from Recorded Events](#)

5.2.1 Recording IBM Websphere MQ Events

Recording for IBM Websphere MQ transports is carried out in a different way than other transports. The Websphere MQ transport does not provide a native method for watching a queue in a non-destructive manner (that is, events can not be captured without taking them off the queue).

For this reason, Rational Integration Tester utilizes Websphere MQ's browse facility to capture messages. To record Websphere MQ events, Rational Integration Tester polls the queue at a predefined interval and copies event information to the Events View.

NOTE: Due to the way that Rational Integration Tester records Websphere MQ events, it is possible that an event could be placed on the queue and removed from the queue before it is seen by Rational Integration Tester. Such events would not be recorded, and they would not be listed on the Events View window. Additionally, if you stop recording an operation and start again, all of the events on the queue will be placed in the Events View, even if they had been recorded previously.

5.2.2 Recording TIBCO EMS Server Events

Users can record all traffic from a TIBCO EMS server thus removing the requirement to specify in advance which server operations are to be recorded.

5.2.3 Recording HTTP(S) Traffic

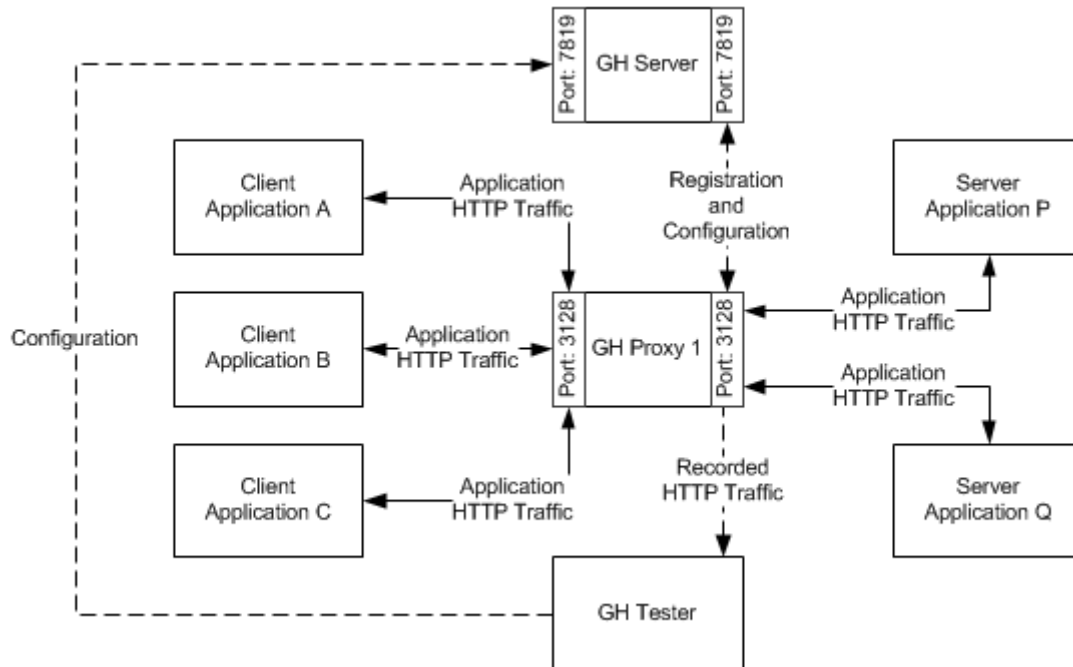
The HTTP proxy in the Rational Integration Tester Platform Pack enables Rational Integration Tester to record all HTTP(S) traffic routed through the proxy.

NOTE: The HTTP proxy in the Rational Integration Tester Platform Pack can proxy/route both HTTP(S) and TCP traffic. (For information about deploying the Rational Integration Tester HTTP proxy, refer to *IBM Rational Integration Tester Platform Pack Installation Guide*.)

NOTE: If using Rational Test Virtualization Server, the HTTP proxy can also be used to route HTTP(S) traffic to Rational Test Virtualization Server stubs automatically when they start up instead of routing traffic through the live system. (For information about using Rational Test Virtualization Server, refer to *IBM Rational Test Virtualization Server Reference Guide*.)

The following diagram shows an example network configuration where a Rational Integration Tester HTTP proxy in recording mode is acting as an intermediary

enabling HTTP messages to appear in Rational Integration Tester's Recording Studio while also passing messages back and forth between the original client and server applications.



NOTE: The port numbers specified in the diagram are default port numbers.

5.2.3.1 Before Recording

Before recording HTTP(S) traffic:

1. Install the Rational Integration Tester HTTP proxy and configure it to record HTTP(S) traffic. (For information about this, refer to *IBM Rational Integration Tester Platform Pack Installation Guide*.)
2. In Rational Test Control Panel, ensure that the proxy is registered with Rational Test Control Panel. (For information about this, refer to *IBM Rational Test Control Panel System Administration Guide*.)

5.2.3.2 Recording

To use Rational Integration Tester and the HTTP proxy to record HTTP(S) traffic:

1. In Architecture School's Physical View, click **Web > Web Server** on the toolbar.

Alternatively, double-click an existing HTTP transport on the Physical View window to modify it.

For information about creating and modifying HTTP transports in Rational Integration Tester, refer to *IBM Rational Integration Tester Reference Guide for HTTP & Web Services*.

2. On the **Config** tab of the Web Server dialog box, click the **Recording** tab.
3. In the **Recording Mode** list, click **External Proxy Server**.
4. Click **OK**.
5. In Recording Studio, create an event monitor for the HTTP transport. (For information about this, refer to [Adding Operations to Event Monitor View](#).)
6. On the Event Monitor window, select the monitor for the HTTP transport and click



5.2.4 Recording TCP Traffic

The Rational Integration Tester HTTP proxy enables Rational Integration Tester to record general TCP traffic routed through the proxy.

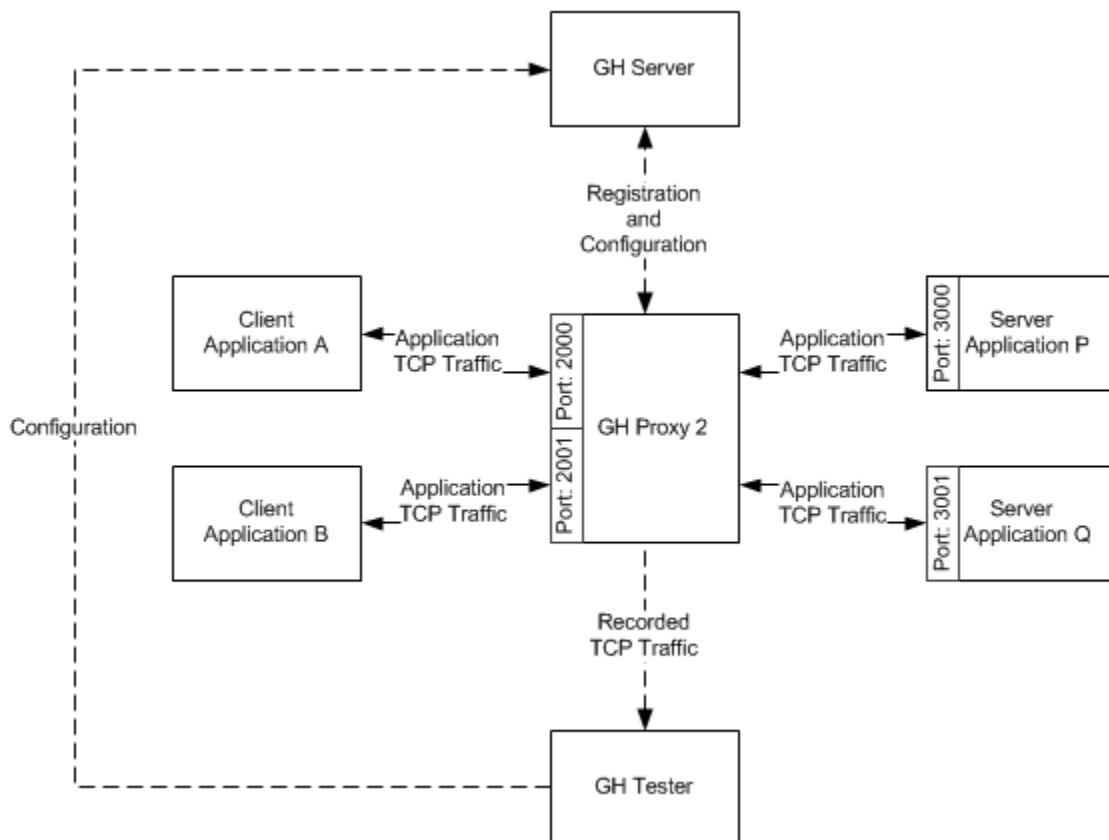
NOTE: The Rational Integration Tester HTTP proxy can proxy/route both HTTP(S) and TCP traffic. (For information about deploying the Rational Integration Tester HTTP proxy, refer to *IBM Rational Integration Tester Platform Pack Installation Guide*.)

NOTE: If using Rational Test Virtualization Server, the Rational Integration Tester HTTP proxy can also be used to route TCP traffic to Rational Test Virtualization Server stubs automatically when they start up instead of routing traffic through the live system. (For information about using Rational Test Virtualization Server, refer to *IBM Rational Test Virtualization Server Reference Guide*.)

The Rational Integration Tester HTTP proxy supports any data sent over TCP/IP but it also supports richer functionality if the messages being sent are Financial Information eXchange (FIX) protocol messages.

This additional support enables content-based routing, that is, routing messages by deriving a destination from the actual content of each message rather than by using a specified end-point reference. (For information about Rational Integration Tester's support for the FIX protocol, refer to *IBM Rational Integration Tester Reference Guide for Financial Information eXchange (FIX)*.)

The following diagram shows an example network configuration where a Rational Integration Tester HTTP proxy in recording mode is acting as an intermediary enabling TCP messages to appear in Rational Integration Tester's Recording Studio while also passing messages back and forth between the original client and server application.



NOTE: The port numbers specified in the diagram are default port numbers.

5.2.4.1 Before Recording

Before recording TCP traffic:

1. Install the Rational Integration Tester HTTP proxy and configure it to record TCP traffic. (For information about this, refer to *IBM Rational Integration Tester Platform Pack Installation Guide*.)
2. In Rational Test Control Panel, ensure that the proxy is registered with Rational Test Control Panel. (For information about this, refer to *IBM Rational Test Control Panel System Administration Guide*.)


5.2.4.2 Recording

To use Rational Integration Tester and a HTTP proxy to record TCP traffic:

1. In Architecture School's Physical View, click **General > TCP/UDP Server** on the toolbar.

Alternatively, double-click an existing TCP transport on the Physical View window to modify it.

For information about creating and modifying HTTP transports in Rational Integration Tester, refer to *IBM Rational Integration Tester Reference Guide for TCP/UDP Sockets*.

2. On the **Config** tab of the TCP/UDP Server dialog box, click the **Recording** tab.
3. In the **Recording Mode** list, click **External Proxy Server**.
4. Click **OK**.
5. In Recording Studio, create an event monitor for the TCP transport. (For information about this, refer to [Adding Operations to Event Monitor View](#).)
6. On the Event Monitor window, select the monitor for the TCP transport and click .

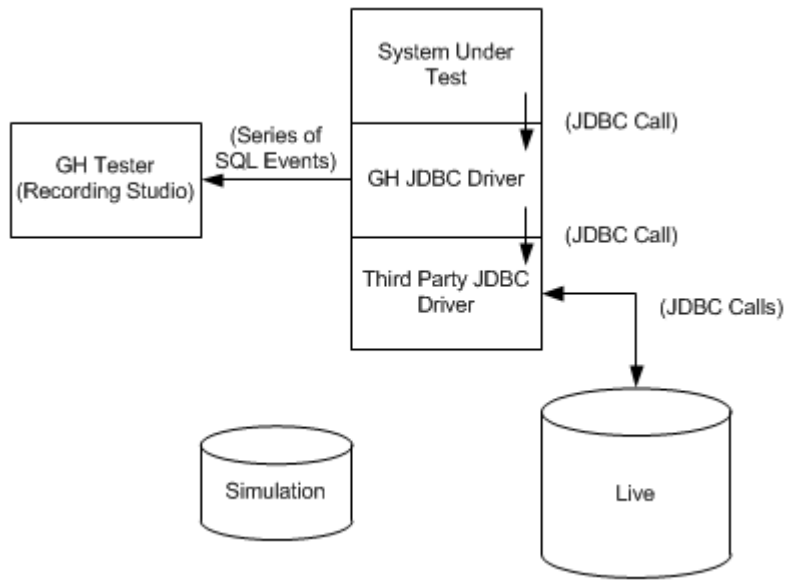
5.2.5 Recording SQL

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. (Rational Test Virtualization Server users can also create, edit, and start/stop database stubs. For information about using Rational Test Virtualization Server, refer to *IBM Rational Test Virtualization Server Reference Guide*.)

The ability to record SQL (but not create any database stubs) is useful if you are building test cases in Rational Integration Tester and want to extract SQL from queries on a database for analysis or for use in test cases.

NOTE: In Rational Integration Tester 8.0.1 (or later), you can also record stored procedures.

The following diagram illustrates how Rational Integration Tester records SQL from databases that use JDBC.



5.2.5.1 Before Recording

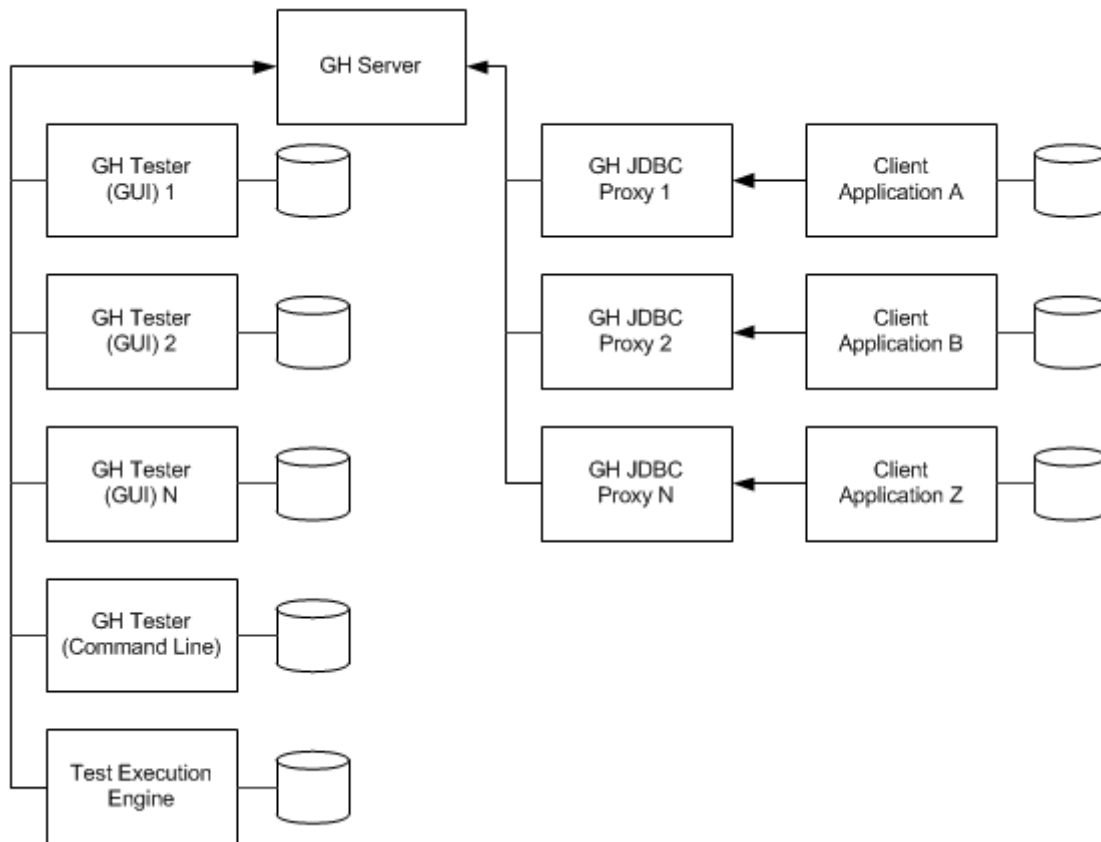
Before recording SQL events:

1. Install the Rational Integration Tester JDBC proxy. (For information about this, 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 upgrade your Rational Integration Tester Platform Pack installation and upgrade the JDBC proxy wherever it is deployed.

2. In Rational Test Control Panel, ensure that the proxy is registered with Rational Test Control Panel. (For information about this, refer to *IBM Rational Test Control Panel System Administration Guide*.)

The following diagram shows an example network configuration.



5.2.5.2 Recording

Recording SQL requires choosing a database. In Rational Integration Tester, there are two methods of selecting a database for recording:

- Architecture School method
- Recording Studio method

Both methods achieve the same result but the Architecture School method is a more direct method.

Recording Studio Method

To use Recording Studio to record a database:

1. Click **+** on the Event Monitors toolbar to create a new event monitor.

The Select a Resource dialog box is displayed.

-
2. On the Select a Resource dialog box, click the **Show/Hide Recordable Resources** button.

NOTE: For more information about the Select a Resource dialog box, refer to [Adding Operations to Event Monitor View](#).

3. Select the relevant logical database.
4. Click **OK**.
5. Click **Record** on the Events View toolbar.
The Create Stubbed Database Whilst Recording? dialog box is displayed.
6. Click the **No thanks, just record the SQL** option button.
7. Open the Recording Studio wizard for creating database stubs.
The first screen of the wizard is displayed.
8. Click the **Start recording** option button.
9. Click **Finish**.

When the application under test makes JDBC SQL calls, they will be visible in Recording Studio's Events View window and you can click each SQL event to view its details or to copy and paste the SQL to a test case in Rational Integration Tester or to another application.

Architecture School Method



There are two ways to use Architecture School's Logical View to record a database:

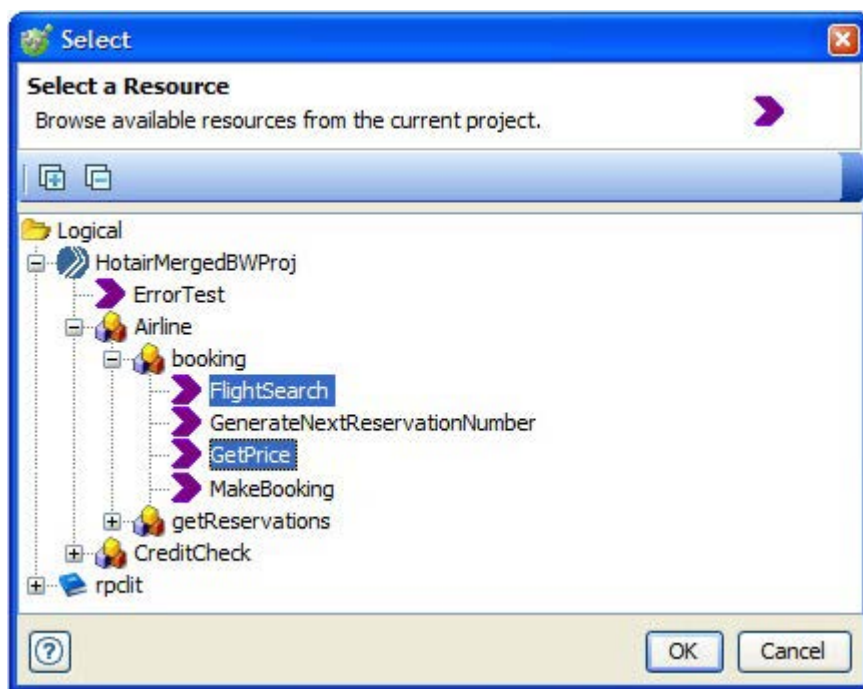
- Right-click the logical database that is to be recorded, and click **Record** on the shortcut menu.
- Alternatively, select the logical database that is to be recorded, and click **Record** on the toolbar. This will open Recording Studio with the resource listed in the Event Monitor window. To start recording, click **Record** on the Events View toolbar.

When the application under test makes SQL calls, they should be visible in Recording Studio.

5.2.6 Adding Operations to Event Monitor View

Once the Recording Studio is opened, you can add new operations to monitor from the Event Monitors view.

1. Open the [Event Monitors](#) view.
2. Click the **Select Operations to Monitor** icon  in the Event Monitors toolbar.
3. Select one or more operations to monitor (use **Ctrl** or **Shift** to select multiple operations) in the project resource tree, then click **OK**. Alternatively, you can select the operations to monitor in the Logical View of Architecture School, then click the **Record selected operations** icon  in Rational Integration Tester's main toolbar.




NOTE: Tags can be used to define the destination for recorded events (that is, in the message header properties of selected operations).

4. The selected operations will be displayed (and enabled for recording, by default) in the Event Monitors view.

NOTE: To enable or disable recording for a specific operation, select or clear the **Enabled** check box for the operation. If an event monitor cannot be created for an operation, an error message containing more details will be displayed.

5.2.7 Removing Monitored Operations



Follow the steps below to remove event monitors for one or more operations:

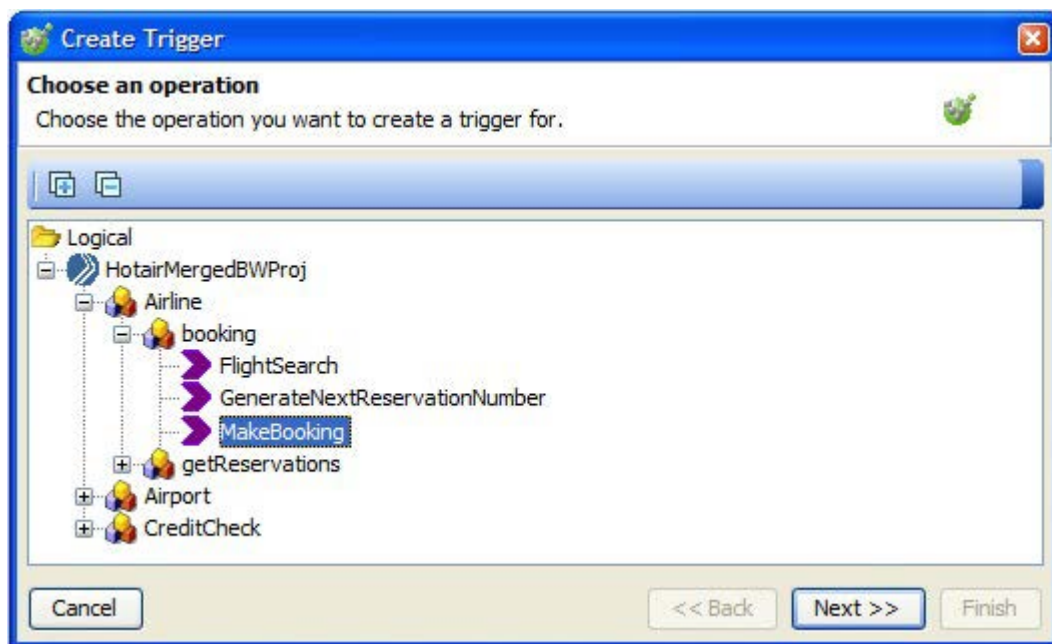
1. Open the [Event Monitors](#) view.
2. Select one or more event monitors to be removed (press CTRL or SHIFT to select multiple event monitors).
3. Click the **Delete the Selected Monitors** icon  in the Event Monitors toolbar.

The selected event monitors will be removed from the view.

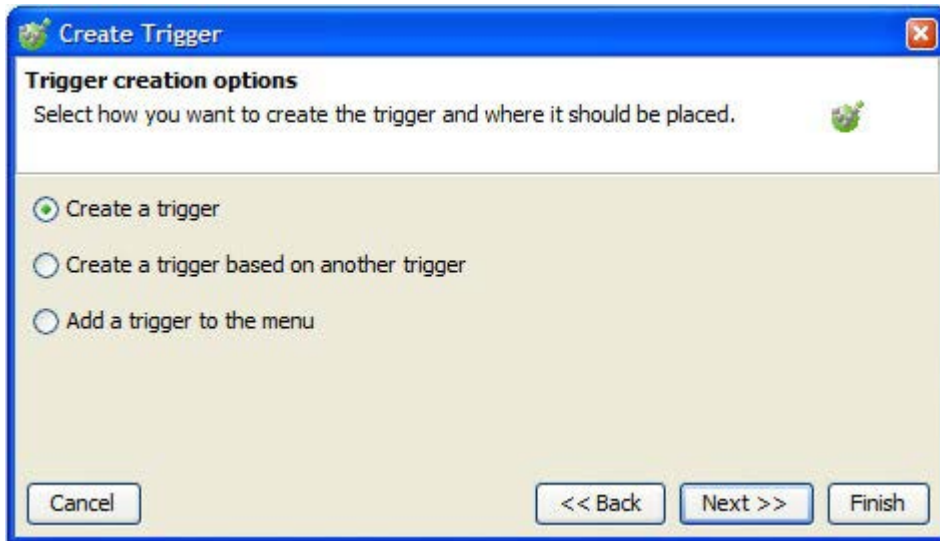
5.2.8 Creating New Standalone Triggers

Using the transport settings configured in an operation's MEP or using the settings in an existing trigger, new triggers can be created from the Triggers view or the Events View. Follow the steps below to create a new trigger:

1. In the Triggers view, select the **New Trigger** icon  or select **New > Trigger** from the context menu. In the Events View, click the small arrow next to the Triggers menu and select the  **Trigger** option.
2. Select the operation in which you want to create the new trigger, then click **Next**.

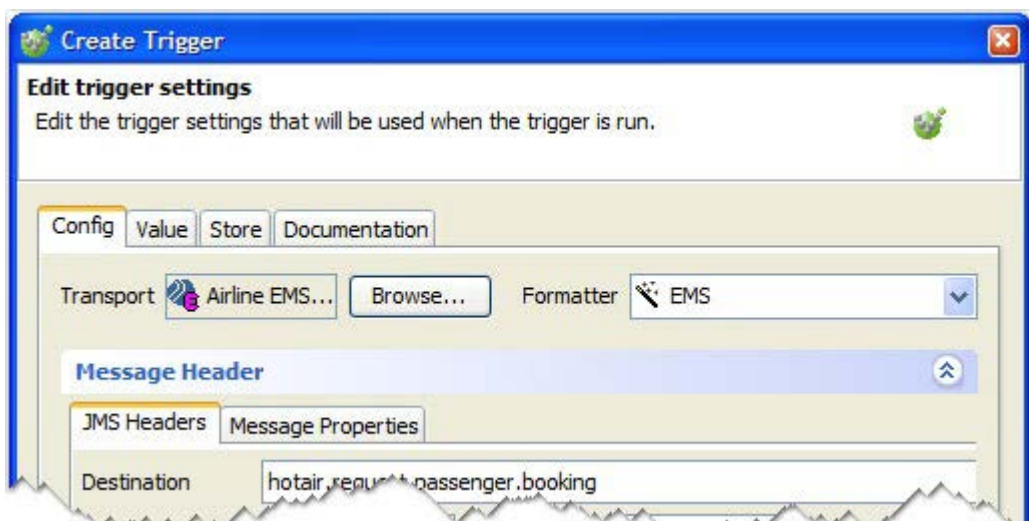


3. Select whether you want to create a brand new trigger or create one based on an existing trigger, then click **Next**. If you base the new trigger on an existing one, you must select the existing trigger before proceeding.

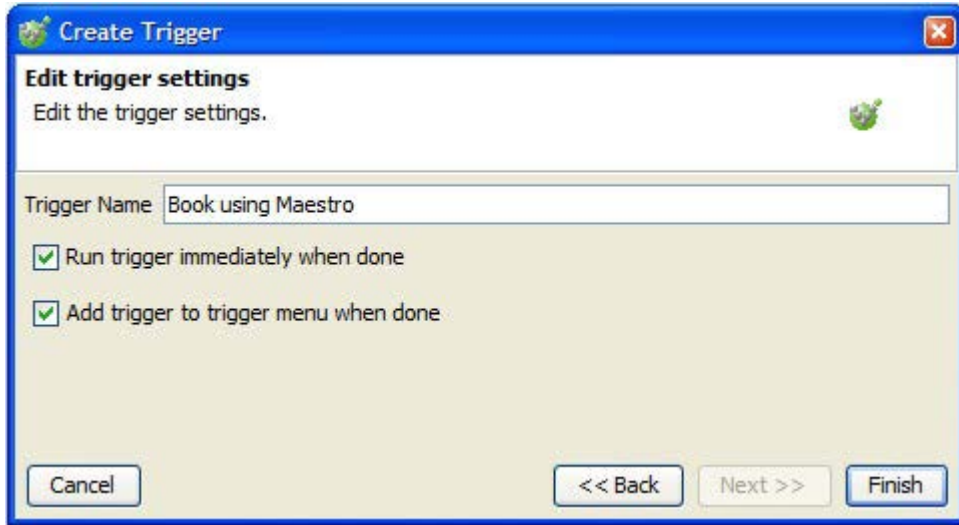


NOTE: If you select the **Create a trigger** option, the selected operation must have a transport configured in its MEP settings.

4. Configure the message to be sent by the trigger in the trigger editor. The trigger is configured like a publisher (see [Publish](#) for more information).



-
5. Provide a name for the trigger and select whether you want to run the trigger after it has been created and whether or not to add the new trigger to the Triggers menu in the Events View.



6. Click **Finish** to create the trigger and close the wizard.



5.2.9 Running Triggers

Existing triggers can be run in several ways:

- In the Triggers view, right-click the desired trigger and select **Run Trigger** from the context menu.
- In the Triggers view, select the desired trigger and click the **Run** icon in Rational Integration Tester's main toolbar (or press **F5**). The Test Lab perspective will be displayed and the trigger will be executed there.
- In the Events View, click the Triggers menu icon to run the trigger that was last run.
- In the Events View, if the trigger has been added to the Triggers menu (see [Adding a Trigger to the Triggers Menu](#)), click the small arrow next to the Triggers menu and select the desired trigger.
- In any view, if the trigger is one of the last ten test items to have been executed, click the small arrow next to the **Run** icon in Rational Integration Tester's main toolbar and select desired trigger. The Test Lab perspective will be displayed and the trigger will be executed there.

5.2.10 Adding a Trigger to the Triggers Menu



In the Triggers view, a trigger can be added to the Triggers menu as follows:

- Select the trigger to add and click the **Add...**  icon in the toolbar.
- Right-click the trigger to add and select the  **Add...** option from the context menu.


NOTE: If the selected trigger has already been added to the Triggers menu, the option to add it will not be available.

5.2.11 Removing a Trigger from the Triggers Menu

A trigger can be removed from the Triggers menu from the Triggers view or from the Triggers menu itself, as follows:

- In the Triggers view, select the trigger to remove and click the **Remove...**  icon in the toolbar.
- In the Triggers view, right-click the trigger to remove and select the  **Remove...** option from the context menu.

NOTE: If the selected trigger has not been added to the Triggers menu, the option to remove it will not be available.

- In the Events View, click the small arrow next to the Triggers menu and select the  **Remove...** option. Select one or more triggers to remove (use **Ctrl** or **Shift** to select multiple triggers) and click **OK**.

5.2.12 Capturing Events

The Events View is used in conjunction with the Event Monitors view to select events and start capturing them.

NOTE: Tags can be used to define the destination for recorded events (that is, in the message header properties of selected operations).

1. In the Event Monitors view, enable an event monitor for the desired operations (tick the **Enabled** option for each). See [Adding Operations to Event Monitor View](#) for more information.

NOTE: If an event monitor can not be created for an operation, an error dialog containing more details will be displayed.

2. In the Events View, click the **Start Recording** icon  in the toolbar.
3. Generate events in the operations that are being recorded (for example, send a request to a web server).

NOTE: New events will be displayed in the top portion of the Events View as they are captured.

4. When you have captured the desired events, click the **Pause Recording** icon  in the Events View toolbar.



5.2.13 Managing the Events View



The Events View displays events that have been captured in monitored operations, including the details of message bodies and headers for selected events. This section describes the options that are available for managing the way information is displayed in the view.

- [Clearing Events](#)
- [Toggling Event Scrolling](#)
- [Showing/Hiding Used Events](#)
- [Adding/Removing Columns in the Event Table](#)
- [Sorting Data in the Events Table](#)
- [Filtering Data in the Events Table](#)

Clearing Events


You can clear some or all of the events in the table if it contains too many entries or if you have utilized all of the events that you want before capturing new events.

To clear specific events, select them (use **Ctrl** or **Shift** to select multiple events, or click one event and drag over the others) and click the **Clear selected events**  icon in the toolbar, or right-click any of the events and select the  **Clear Selected Events** option.

To clear all of the events, click the **Clear all events**  icon in the toolbar or right-click any of the events and select the  **Clear All Events** option.

NOTE: After confirming your selection, the applicable events – whether they have been used or not – will be cleared from the view.

Toggling Event Scrolling


As new events are captured, they are added to the bottom of the events table. By default, the contents of the table will not scroll when new events are added (that is, scrolling is locked). You can toggle this behavior by clicking the **Scroll Lock** icon  in the toolbar.

- When the icon is highlighted, scrolling is locked.
- When the icon is not highlighted, the contents of the table will scroll, showing new events at the bottom of the table as they are added.

Showing/Hiding Used Events

Once you create a test, stub, trigger, or requirement from an event in the event table, that event is considered to have been “used.”

NOTE: Once an event has been used, a green checkmark is displayed to the left of it in the event table.

By default, all events are displayed in the table, regardless of whether or not they have been used. You can toggle the display of used events by clicking the **Show...** icon  in the toolbar.

- When the icon is highlighted, all events are displayed in the table.
- When the icon is not highlighted, events that have been used will be hidden from the table view.

Adding/Removing Columns in the Event Table

By default, the event table displays the time stamp, source, and description for each captured event. If desired, you can add message fields from a selected event as columns in the event table. You can also remove any custom columns that you add.

1. Select a captured event that contains the field you want to add to the table.
2. Locate the desired element or field in the message body or header using the appropriate tabs in the message viewer (below the event table).
3. Select the element or field and click **Promote Field** above the message viewer.

A new column is added to the event table, named according to the element or field that was promoted.

NOTE: If you want to promote a field permanently (that is, the field will stay promoted after closing and reopening the project – and only for the current project), click **Promote Field Permanently**.

4. To remove an added column, right-click the column header or any field in the column and select **Demote** from the context menu (to remove a permanently promoted field, select the **Demote Permanently** option).

Sorting Data in the Events Table

By default, the events in the event table are displayed top to bottom in the order in which they were captured. If desired, you can sort the events by the values contained in any of the displayed columns.

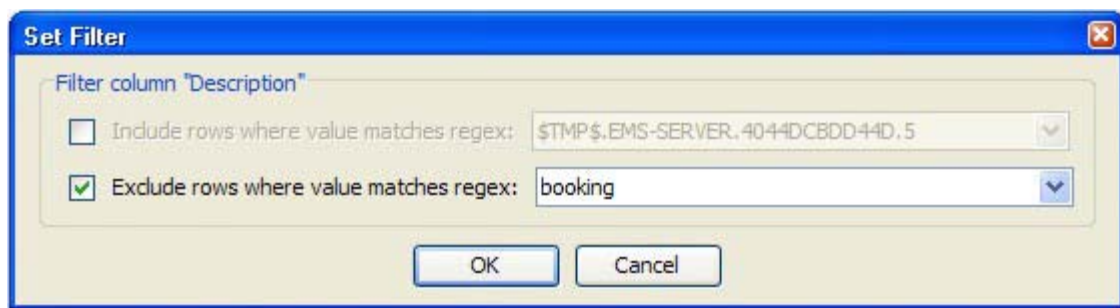
- Click any column heading to sort the table in ascending order ▲ by that column.
- Click any column heading a second time to sort the table in descending order ▼ by that column.
- Click any column heading a third time to stop sorting by that column.

NOTE: To sort with multiple columns, hold the **Ctrl** key and click a column header while the table is already being sorted by another column.

Filtering Data in the Events Table

You can filter the contents of the event table based on existing values in the columns that are displayed in the table, and multiple columns can be filtered at one time.

1. Right-click the column heading or any field in the desired column and select **Filter** from the context menu. The **Set Filter** dialog is displayed.



2. Select one or both of the filter type options (“Include” or “Exclude”) – events must match an “include” filter to be displayed, and events matching an “exclude” filter will be hidden.

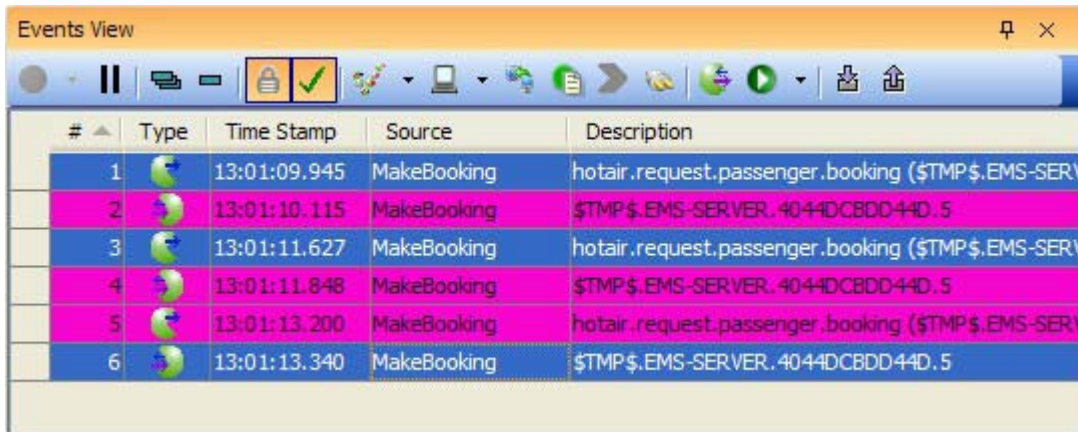
NOTE: All enabled “include” filters must pass for a row to be displayed, but only one “exclude” filter must match for a row to be hidden.





3. In the text field/combo box of the active option, enter a regular expression to use for filtering the events or click the arrow to select a recent value, then click **OK**.
4. To remove a filter, right-click the filtered column heading or any field in the column and select **Remove filter** option from the context menu.

5.2.14 Creating Test Resources from Recorded Events

Once events have been recorded in a monitored operation, you can create a test, stub, trigger, or requirement from selected events in the Events View.

1. Select one or more recorded events in the Events View (use **Ctrl** or **Shift** to select multiple events, or click one event and drag over the others).



2. Click one of the test resource icons (test , stub , trigger , or requirement ) in the toolbar or right-click the selected event(s) and choose the desired option from the context menu.

When creating tests, you can create an Integration Test or a Unit Test (see [Tests](#) and [Creating Tests from Recorded Events](#)). When creating stubs, you can create a stub or a parameterized stub (see [Stubs](#), [Create a Parameterized Stub](#), and [Configuring the Contents of a Stub](#)).

3. Select the operation where the resource(s) should be saved and provide a name (or prefix) for the new resource(s).

NOTE: When creating tests or stubs you will provide a name since only one test or stub is created from one or more recorded events. When creating triggers or requirements, you will provide a naming prefix since one trigger or requirement will be created for each selected event.

4. If desired, enable the **Edit...** option to edit the new resource (or the first resource if creating more than one) when finished, then click **OK**.

The new resource(s) will be created under the selected operation and (optionally) opened for editing.

Test Factory

Contents

Overview

Tests

Performance Tests

Test Suites

Stubs

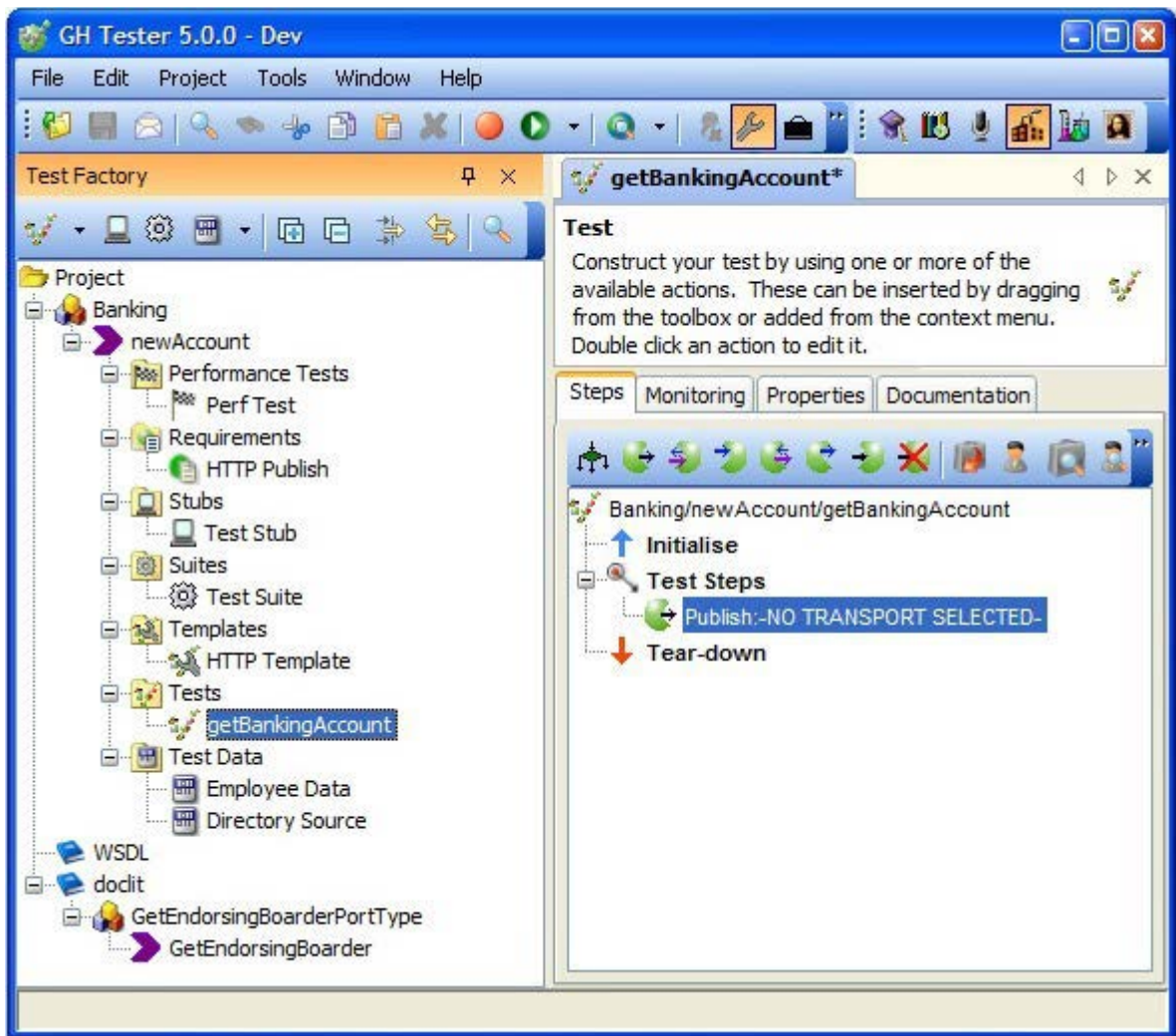
Test Templates

Test Data Sources

This chapter provides an overview of Rational Integration Tester's Test Factory perspective, including details about how to build tests (including performance tests), test suites, stubs, test templates, and test data.

6.1 Overview

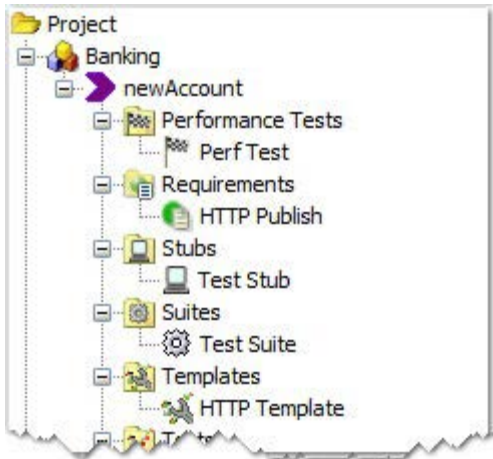
The Test Factory perspective is one of Rational Integration Tester's main work areas – the place where you will create all of your tests and the artifacts/data to support them.



All test items are created in the Test Factory tree, on the left side of the perspective. Any test or test-related item that is created in the Test Factory can be edited in the editing panel, on the right side of the perspective (for example, a test can be edited here, but a message must be opened in the Requirements Library perspective).

6.1.1 The Test Factory Tree

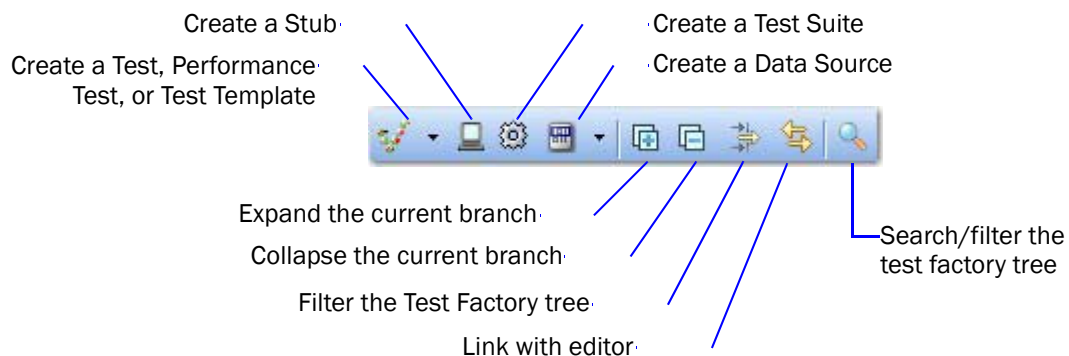
The Test Factory tree, docked on the left side of the Test Factory perspective, contains all of your project resources (that is, components, operations, and test items). As tests and test items are added to the tree, they are automatically organized, by folder, under the operation in which they are created.



To focus the view on a specific item, right-click it and select **Promote to Root** from the context menu. To restore the normal view, click the root of the tree and select **Demote from Root** from the context menu.


6.1.2 The Test Factory Toolbar

The Test Factory toolbar contains shortcuts for creating test items and supporting artifacts, and for manipulating what is displayed in the tree.



Different shortcuts will be available/unavailable, depending on the item that is selected in the Test Factory tree (for example, the Stub and Test Suite icon are unavailable when a message/requirement is selected).

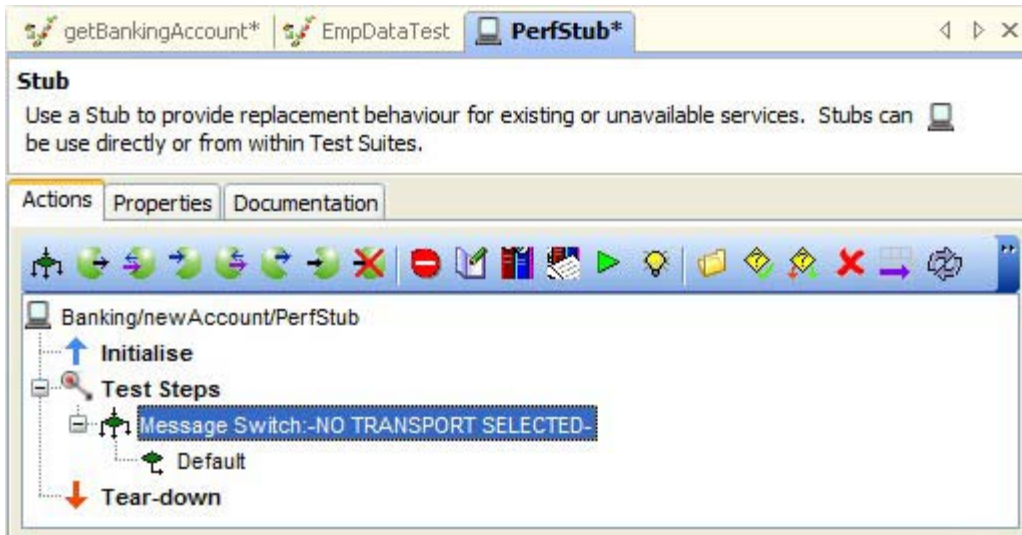
6.1.3 Linking Items with the Editor

You can enable/disable the link between an open item in the [The Editing Panel](#) and the Test Factory tree by toggling the **Link with Editor** icon . When enabled, the test asset that is currently selected in the editing panel will be highlighted and brought into view in the tree. Similarly, if an open item is selected in the tree, it will be brought into focus in the editing panel.

6.1.4 The Editing Panel

The editing panel is used to modify test items that are created in the Test Factory perspective.

NOTE: Items that have been created in a different perspective (for example, messages, triggers, and so on) must be edited in their own perspective.



Double-click on any Test Factory item to open it for editing. Details about editing each item can be found in the following sections:

- [Tests](#)
- [Performance Tests](#)
- [Test Suites](#)
- [Stubs](#)
- [Test Templates](#)
- [Test Data Sources](#)

6.2 Tests

Tests are the fundamental building blocks of integration and SOA testing. A test sends an input message to an external service or system in a protocol-specific way and validates the reply against an expected result.

The following sections contain information about managing tests in Rational Integration Tester:

- [Test Phases](#)
- [Test Properties](#)
- [Change Management Properties](#)
- [Test Views](#)
- [The Action Editor](#)
- [Constructing Tests](#)
- [Manipulating the Contents of a Test](#)
- [Monitor a Log File During a Test](#)
- [The Tag Data Store](#)
- [Add a Message Case](#)
- [Setting Pass Paths](#)
- [Setting Failure Paths](#)
- [Executing Tests](#)

6.2.1 Test Phases

A new, empty test will always consist of three basic phases: *Initialise*, *Test Steps* and *Tear-down*. The essential usage of the three steps is as follows:

- The *Initialise* phase should be restricted to executing commands that start or configure plugins, servers, or target resources. This phase may be used for test preparation (for example, populating environment variables), but if these variables are overwritten in the test steps, they will not be restored to their original values.

NOTE: Test Data Set values are not accessible from the *Initialise* phase.

- The *Test Steps* phase should contain the actions used in the test. If the test is part of a repeated or concurrent execution (by means of a stub or a performance test), then the *Test Steps* phase is the only one that will be executed for each iteration.
- Similar to initialization, the *Tear-down* phase should be used only for deallocation of resources, termination of processes, and so on. The *Tear-down* phase is only executed by the last test to be processed when executed repeatedly or concurrently.

NOTE: The *Initialise* and *Tear-down* phases are always run around the test, and the *Tear-down* phase runs even if the test fails. These phases are useful in stubs and performance tests where the body of the test is called multiple times, but actions within the other phases should only run once each.

6.2.2 Test Properties

The **Properties** tab defines which tags should be available on the input and output interface for the selected test. When a tag is created, the interface definition can be set by enabling the “Expose as input” and/or “Expose as output” option. Additionally, new tags can be created under either of the input or output interfaces within the test.

Test

Construct your test by using one or more of the available actions. These can be inserted by dragging from the toolbox or added from the context menu. Double click an action to edit it.




Steps Monitoring **Properties** Documentation Change Management

Interface

Input

☐ None ☐ All Tags ☒ Select Tags




Name	Description
Response	
Input2	
Input3	
Input1	




Output

☐ None ☐ All Tags ☒ Select Tags

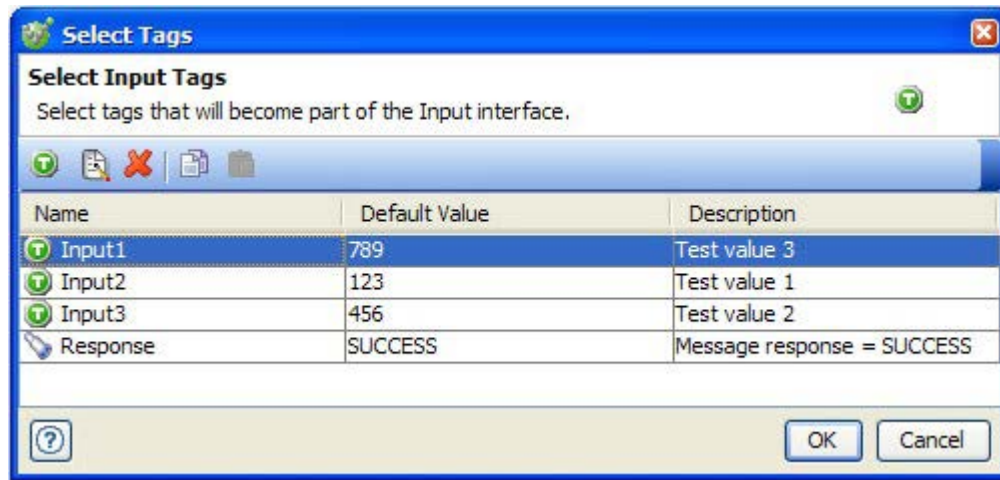
Name	Description
Response	
Input2	
Input3	
Input1	



  

Managing Input/Output Tags

You can create, edit, or delete the tags that are available under the input and output interface using the , , and  icons.

- To create a new tag, click the  icon next to the table under Input or Output. The **Select Tags** dialog is displayed. Tags can be created, modified, or deleted in the same way as in [The Tag Data Store](#).



- To edit an existing tag in the Input or Output interface, select it and click the  icon. The Edit Tag dialog is displayed (see [Creating Tags](#) for more information).
- To delete a tag from the Input or Output interface, select it and click the  icon next to the table.

Designating Tags for Input/Output

You can specify which of the tags under the Input or Output interface should be made available to scenarios with the radio buttons above the tag tables.

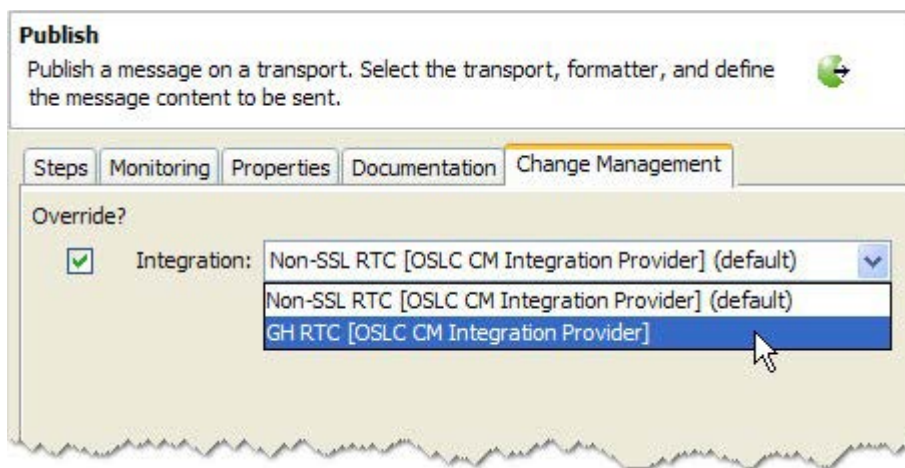


- If you choose **None**, none of the tags will be available and the list of tags can not be edited.
- If you choose **All Tags**, all tags will be available and the list of tags can not be edited.
- If you choose **Select Tags**, only the tags shown in the table will be available in scenarios (that is, add and delete tags to the table as described in [Managing Input/](#)

Output Tags).



6.2.3 Change Management Properties

Rational Integration Tester can be integrated with OSLC compliant change management systems for raising defects within the context of a Rational Integration Tester test asset. If multiple change management integrations have been configured in the current project, you can override an inherited integration (from an operation or service component higher up the project resource tree) or the default integration under the **Change Management** tab, letting you specify a different integration to be used.

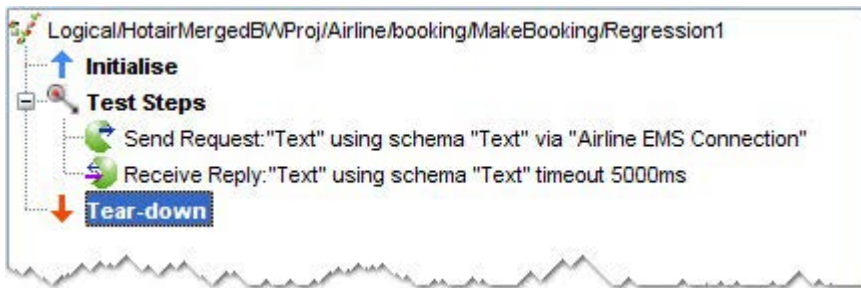


To specify a different or non-default integration for the current test, enable the **Override** option and select the existing integration from the list of those available. Ensure that you save the test after making any changes.

6.2.4 Test Views

Two views are available for a test, the Technical View  and the Business View . The current view can be toggled by selecting the appropriate icon in the Rational Integration Tester toolbar.

In the Technical View, a test and each of its actions is represented by its icon and an automatically generated technical description. The test description defaults to the full path of the test, and the action description includes the action type and basic operations. As you change the details of the action, the description will also change.



In the Business View you can enter a custom description for a test or any action by selecting **Rename** from the context menu or by selecting the test/action and pressing **F2**.



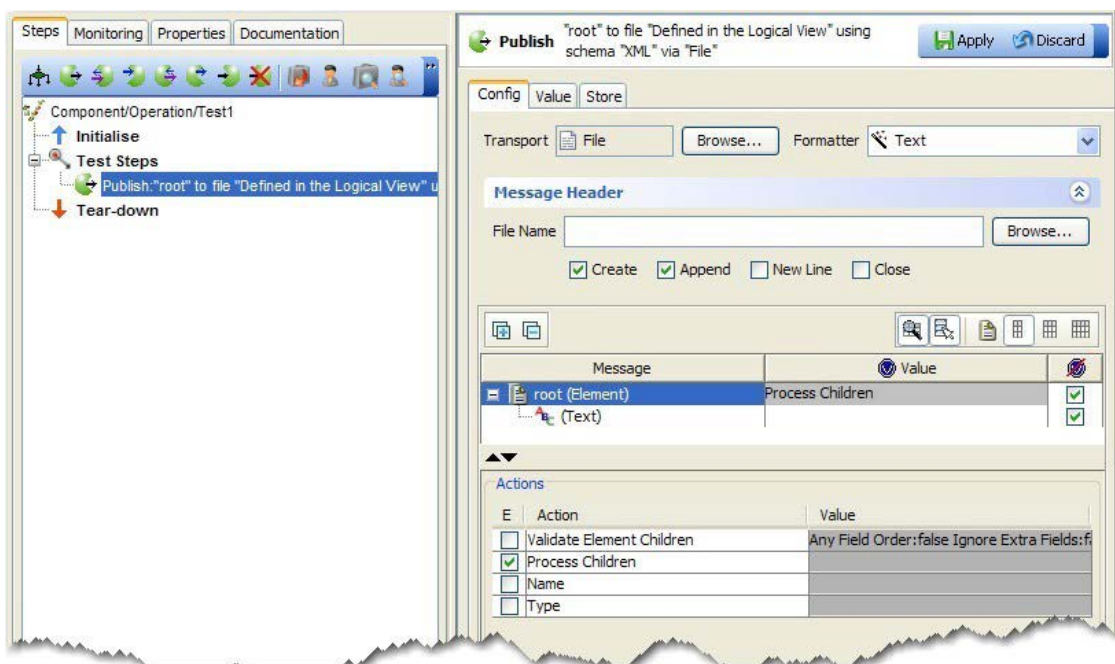
If no custom description is written for a test or any of its actions, the technical description is used. In this case, the test and action descriptions are displayed in a different color. The default color is grey, but it can be changed in the project preferences (**Test Editor > Missing business descriptions color**).

NOTE: Once the test label or action has been renamed in the Business View, the text color will change. From now on, or until the custom name is cleared, the step's functions will not be copied across to the business description.

NOTE: For projects created in an earlier release of Rational Integration Tester, the business text will come from the previous “Description” field.


6.2.5 The Action Editor

The Action Editor is used to modify the actions contained within a test or stub. Set in Rational Integration Tester’s preferences (see [Test Editor](#)), the editor can be opened as a popup dialog by double-clicking an action, or it can be displayed inline, below or to the right of the editing panel. The details of available test actions are provided in [Appendix A: Test Actions](#).



If the Action Editor is displayed inline, when a new test or stub is created or if nothing in the current test or stub is selected (use **Ctrl + click** to deselect items), the Action Editor displays information about using the test editor. Once an editable action is selected, the editor associated with that action is displayed. The displayed editor is the same as the popup editor, except that the **OK** and **Cancel** buttons are not used, and the **Import** and **Export** buttons are moved into the **File** menu.

You can set the Action Editor preferences to apply changes automatically (that is, like clicking **OK** in a standard editor). If changes are not applied automatically and you click away from an edited action (that is, click on another action or on a test phase), a dialog will prompt you to **Apply** or **Discard** the changes. If you click **Cancel**, the selection will be reset and the previously selected item in the test will be selected.

If changes are not being applied automatically, you can use the **Apply** button to apply the current changes. To undo any changes that have not been applied and restore the action to its last saved state, click **Discard**. Once you are finished making changes to an action, you can save the test by clicking  or pressing **Ctrl + S**.

6.2.6 Constructing Tests

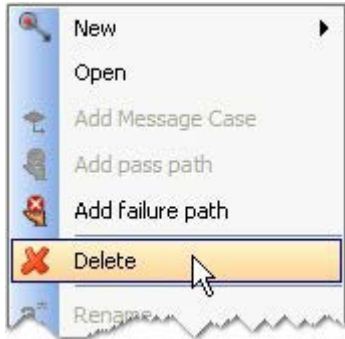
A test can be added to an existing operation by means of the context menu or the Test Factory toolbar. Four different methods can be used for creating tests, and each method is detailed in the following sections: [Creating a Single Empty Test](#), [Creating a Single Test using MEP](#), [Creating Multiple Tests using MEP](#), [Creating a Test from a Template](#), [Creating Tests by Pasting](#), and [Creating Tests from Recorded Events](#).

NOTE: If the operation has not yet been configured to point to a specific transport, the **Test from Template**, **Test from MEP**, and **Tests from MEP** are unavailable from the **New, Tests** context menu.

After a test is created and named, it is opened in [The Editing Panel](#) with a palette of available actions displayed above the test tree (see [Manipulating the Contents of a Test](#)).



Once added to a test, a test action is fully editable and configurable. To open an action in a test, simply double-click it. Additional options are available from the action's context menu, which is displayed when you right-click on the action.



NOTE: When executed, a test's steps are processed from top to bottom.

Creating a Single Empty Test

Follow the steps below to create a new empty test.

1. Select the operation or folder in which you want to create the test.
2. Right-click the operation or folder and select **New > Tests > Test** from the context menu, or click the arrow next to the Test icon in the Test Factory toolbar and select **Test**.

NOTE: If at least one test has already been created, you can right-click the **Tests** folder or one of the existing tests and select **New > Test**.

3. Provide a name for the new test when prompted, then click **OK**.

The new test is opened for editing. See [Manipulating the Contents of a Test](#) for more information about working with the new test.

Creating a Single Test using MEP

An operation's Message Exchange Pattern (MEP) defines the pattern, schema, and binding for messages that are exchanged by the operation. You can use this information to quickly create a test that adheres to the rules defined in the MEP (see [Change an Operation's Message Exchange Pattern](#) for more information).

Follow the steps below to create a single test using the selected operation's MEP.

1. Select the operation or folder in which you want to create the test.

-
2. Right-click the operation or folder and select **New > Tests > Test using MEP** from the context menu, or click the arrow next to the Test icon in the Test Factory toolbar and select **Test using MEP**.

NOTE: If at least one test has already been created, you can right-click the Tests folder or one of the existing tests and select **New > Test using MEP**.

3. Provide a name for the new test when prompted, then click **OK**.

The new test is opened for editing and includes the appropriate messaging actions as defined by the operation's MEP. See [Manipulating the Contents of a Test](#) for more information about working with the new test.

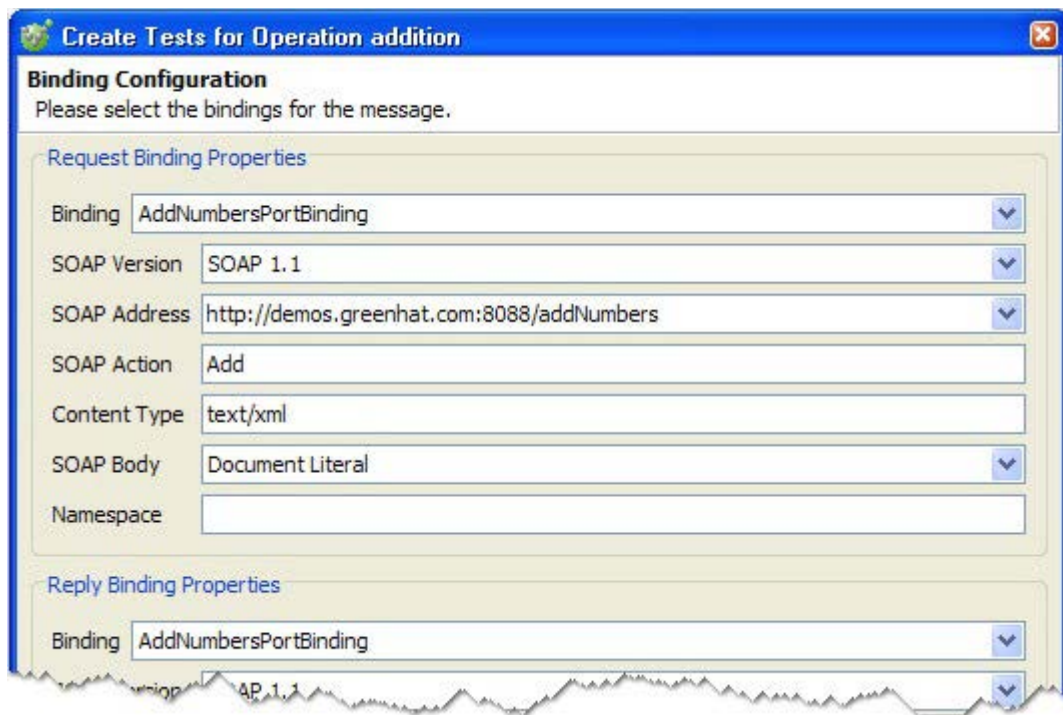
Creating Multiple Tests using MEP

Similar to [Creating a Single Test using MEP](#), you can create multiple tests using the MEP with the help of the MEP wizard, as described below.

1. Select the operation or folder in which you want to create the test.
2. Right-click the operation or folder and select **New > Tests > Test using MEP** from the context menu, or click the arrow next to the Test icon in the Test Factory toolbar and select **Test using MEP**.

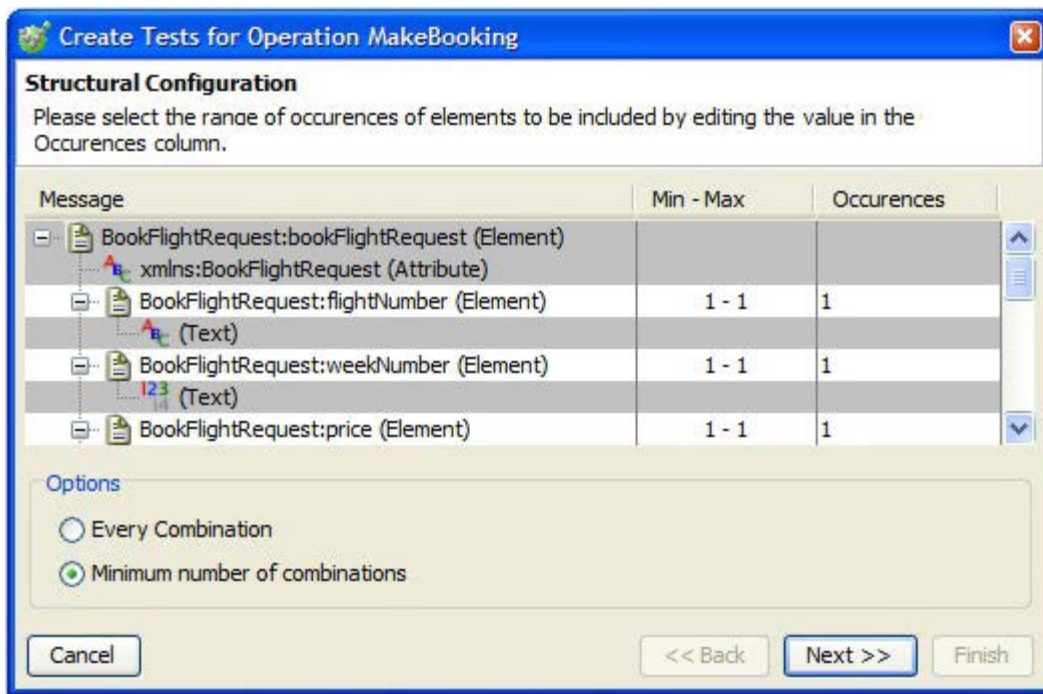
NOTE: If any tests already exist, you can right-click the Tests folder or one of the existing tests and select **New > Tests using MEP**.

If the operation containing the tests has a WSDL as its parent, the **Binding Configuration** dialog is displayed.



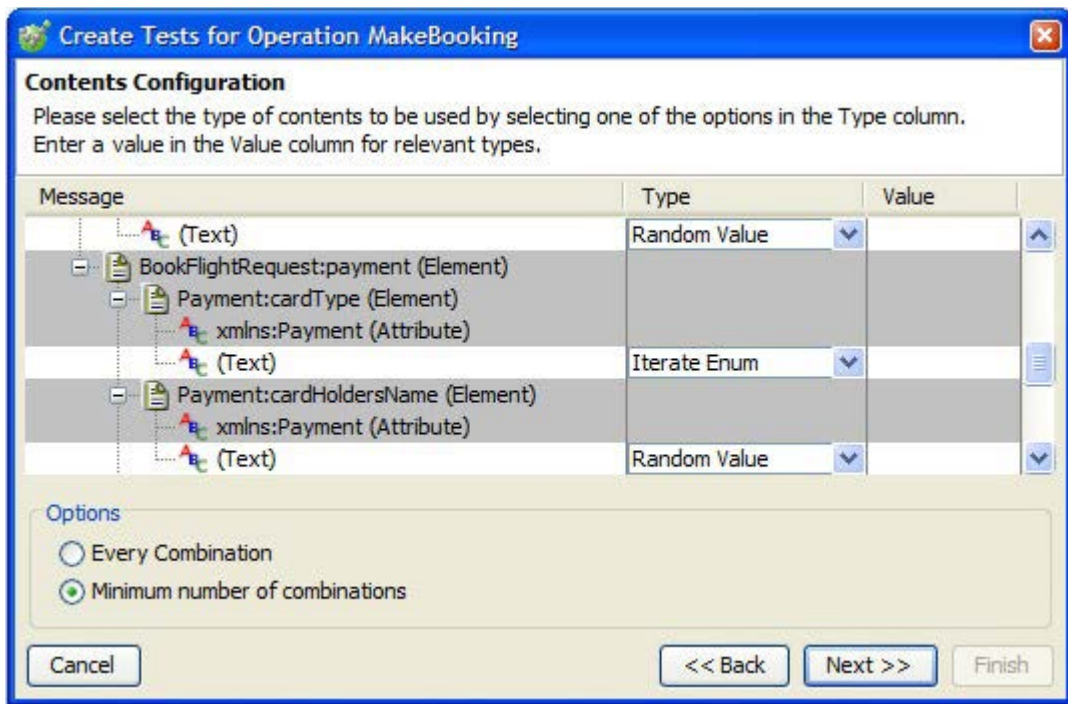
3. Select the binding properties for the request and reply messages, then click **Next**.

The **Structural Configuration** dialog is displayed.



4. Set the number of occurrences for each message element (0 through n) and select the desired combination option for generating tests.
 - If **Minimum number of combinations** is selected, at least one unique element will be generated in a message for each occurrence, but only enough tests will be generated to satisfy the highest single occurrence value (that is, if the most occurrences for any element is three, then three tests would be generated).
 - If **Every Combination** is selected, a test will be created so that every unique combination of occurrences is satisfied (that is, if one element specifies two occurrences and another specifies three, then six tests would be generated).
5. When finished, click **Next**.

The **Contents Configuration** dialog is displayed.

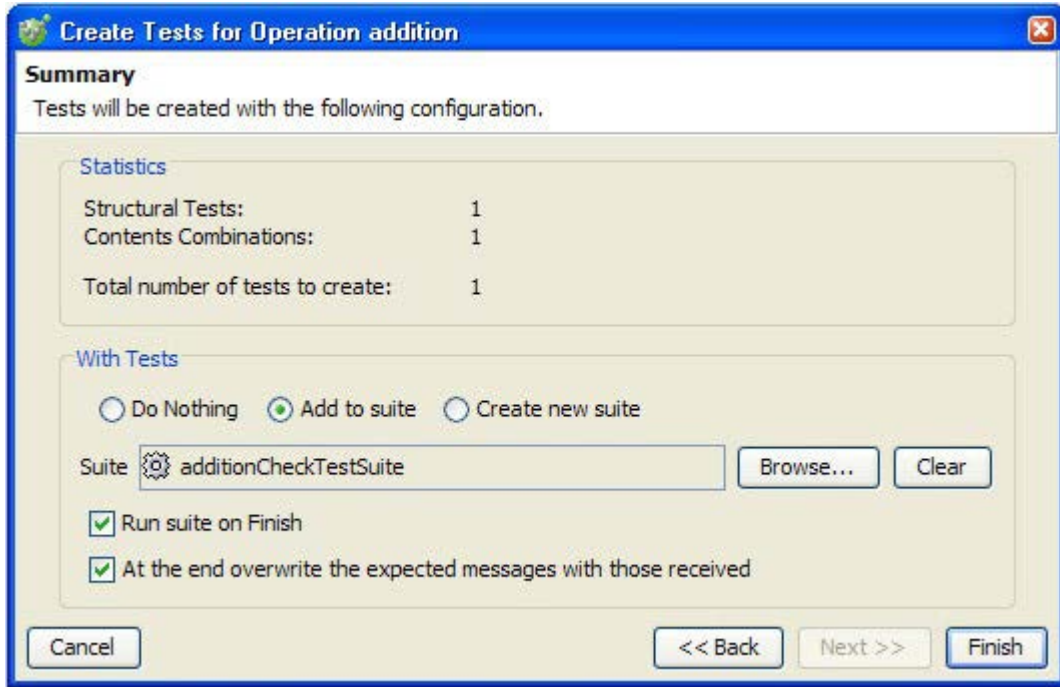


6. For each element, set the value to generate according to the element type, as follows:

Random Value	(Default for all elements, available for all field types) Generates a random value according to the element type.
Constant	(Available for all field types) Populate the field with the name of the element's parent in reverse. If desired, this value can be modified.
Random Regex	(Available for all field types) Lets you enter a regular expression pattern for the value, and Rational Integration Tester will generate a random value that matches the pattern for every instance of the message field.
Boundary	(Available for numeric fields) Generates a value that is on the defined boundary of the field.
Beyond Boundary	(Available for numeric fields) Generates a value that is outside the defined boundary of the field.
Iterate Enum	(Available for fields with enumerated values) Generate a unique test for each enumeration.

7. Set the combination options as described in the previous dialog, then click **Next**.

-
8. The **Summary** dialog is displayed, showing the total number of tests to be generated (that is, the product of the number of structural tests and the number of content combinations).



9. Under the **With Tests** panel you can perform additional actions with the tests to be created, as follows:
- **Do Nothing** (default) simply creates the applicable tests.
 - **Add to suite** will add the new tests to an existing test suite, selected by clicking Browse and selecting the desired suite.
 - **Create new suite** will add the tests to a new test suite, the name of which can be specified in the Suite field.
 - Enable the **Run suite on Finish** option to execute the new or existing test suite when the MEP wizard closes.
 - Enable the **At the end...** option to overwrite expected messages in the executed tests with the messages that are received.
10. Click **Finish** to create the tests and return to the Test Factory's main view.

The new tests are created, named according to the content options that were selected in the wizard. See [Manipulating the Contents of a Test](#) for more information about working with the new tests.

Creating a Test from a Template

Once you have created a test template (see [Test Templates](#)), you can quickly create new tests from it. Follow the steps below to create a test from a test template.

1. Select the operation in which you want to create the test.
2. Right-click the operation or folder and select **New > Tests > Test from Template** from the context menu, or click the arrow next to the Test icon in the Test Factory toolbar and select **Test from Template**.

NOTE: If at least one test has already been created, you can right-click the Tests folder or one of the existing tests and select **New > Test from Template**.

3. Select the template to use from the project resource dialog that is displayed.
4. Provide a name for the new test when prompted, then click **OK**.

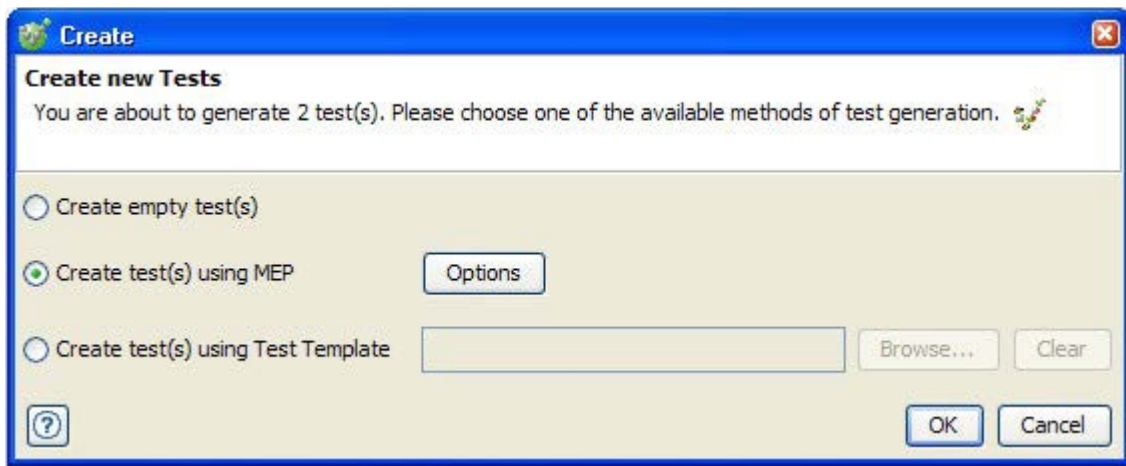
The new test is opened for editing. See [Manipulating the Contents of a Test](#) for more information about working with the new test.

Creating Tests by Pasting

You can create one or more tests by pasting one or more lines of text that are currently on the clipboard. New tests will be named according to the pasted text, each line of text on the clipboard will create a unique test, and the tests will be created within the component or folder that is selected when pasting.

1. Place the test names to be created on the clipboard by copying them in the desired application (for example, Notepad).
2. Select the component or folder in which the new tests should be created.
3. Select **Edit, Paste** or press **Ctrl + V** to paste the contents of the clipboard.

The **Create new Tests** dialog is displayed.



4. Select the desired option to use for creating the new tests – create empty tests, create tests using MEP, or create tests using a selected template. See [Creating a Single Empty Test](#), [Creating a Single Test using MEP](#), [Creating Multiple Tests using MEP](#), or [Creating a Test from a Template](#) for more information.

If creating tests using MEP, click **Options** to configure the message options to use when creating the tests. Additionally, you can enable the **Save...** option to save the selected settings for the next time pasted tests are created using MEP.

If creating tests using a template, click **Browse** to locate and select the desired template within the current project.

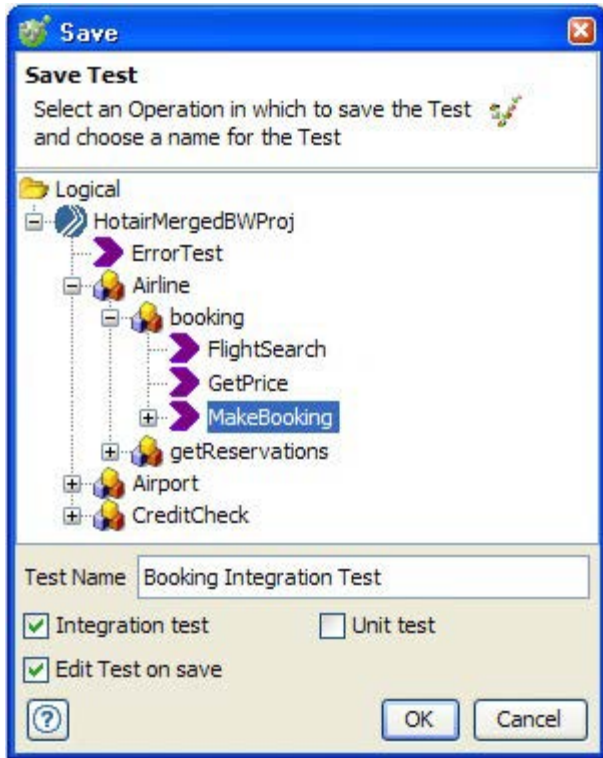
5. When finished, click **OK** to create the tests as configured.

NOTE: Case is not checked when creating tests (that is, “a” is the same as “A”). Therefore, if any created tests would have the same name, a unique number will be appended to the test name (for example, “test” and “Test(2)” instead of “test” and “Test”).

Creating Tests from Recorded Events

Tests can be created using events captured in the Recording Studio (see [Recording Studio](#), [Capturing Events](#), and [Creating Test Resources from Recorded Events](#)).

1. In Recording Studio, select one or more recorded events in the Events View.
2. Click the **Save Test from selected events** icon in the Events View toolbar, or right-click the messages and select **Save Test** from the context menu.



NOTE: If a promoted column's tag reference is used as part of the test name (for example, col = %%Col A%%), grouping will be enabled, as follows:

Any promoted column's cell which does not contain a value will inherit its value from the first non-null value above it.

All rows which share the same value will belong to the same group.

If more than one tag name is used in the name, uniqueness will be applied to the applicable columns.

Each group will produce a set of tests (unit and or integration).

Generated tests will be driven from only the subset of events defined in each group.

3. Provide a name for the test in the **Test Name** field.
4. Select the type of test you want to create – **Integration test** will create a test based on the actions in the selected events, and **Unit test** will create a test based on the MEP types of the selected operations.

NOTE: If you select both **Integration test** and **Unit test**, two tests will be created using “_integ” or “_unit” in their name.

5. Enable the **Edit Test on save** option to open the new test (or the first of multiple tests) for editing once it has been created.
6. When finished, click **OK**.

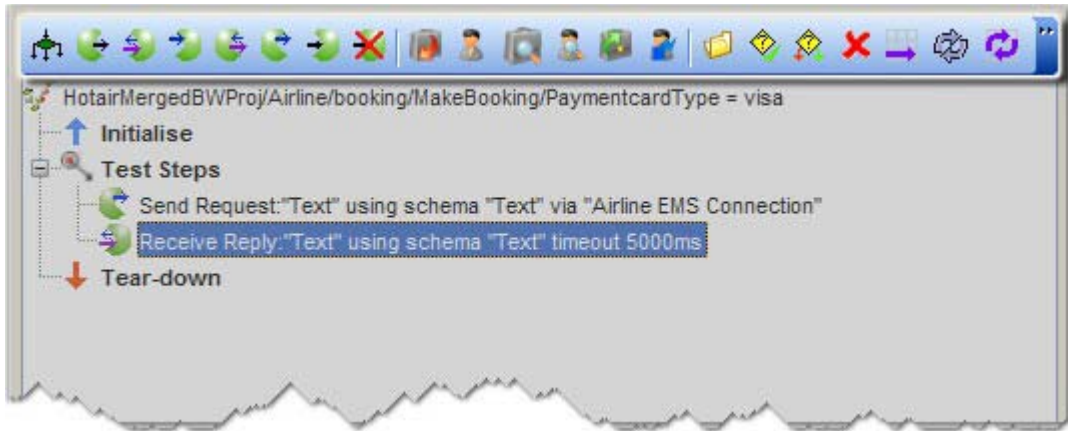
6.2.7 Manipulating the Contents of a Test

After creating a new test or opening an existing test, you can manipulate its contents in the test editor, which displays the test and its phases to the right of the Test Factory tree.

- [Add Test Actions](#)
- [Edit Test Actions](#)
- [Delete Test Actions](#)
- [Rename Test Actions](#)
- [Change a Test Action's Type](#)
- [Cut/Copy/Paste Test Actions](#)
- [Move Test Actions](#)
- [Enable or Disable Test Actions](#)
- [Save Messages as Requirements](#)

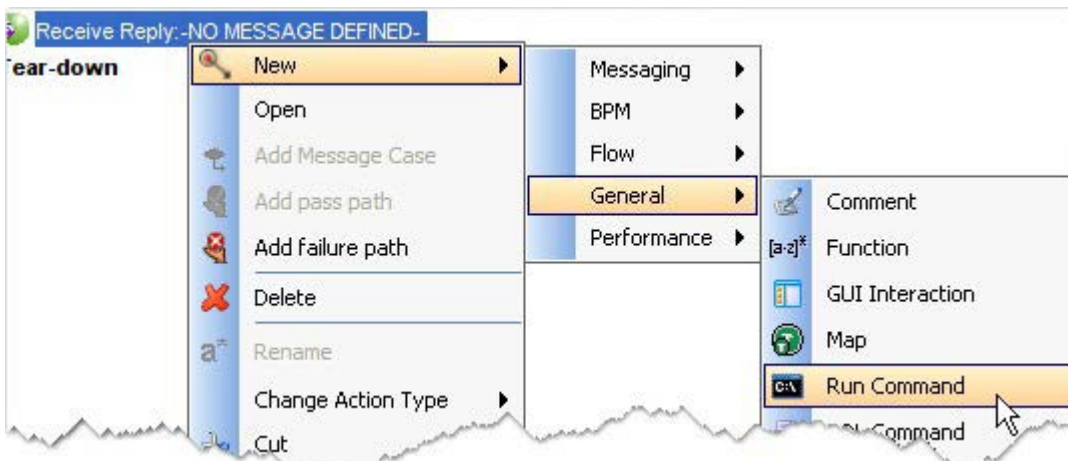
Add Test Actions

Actions can be added to a test by clicking on them in the Actions palette, which is the horizontal toolbar located above the test editor.



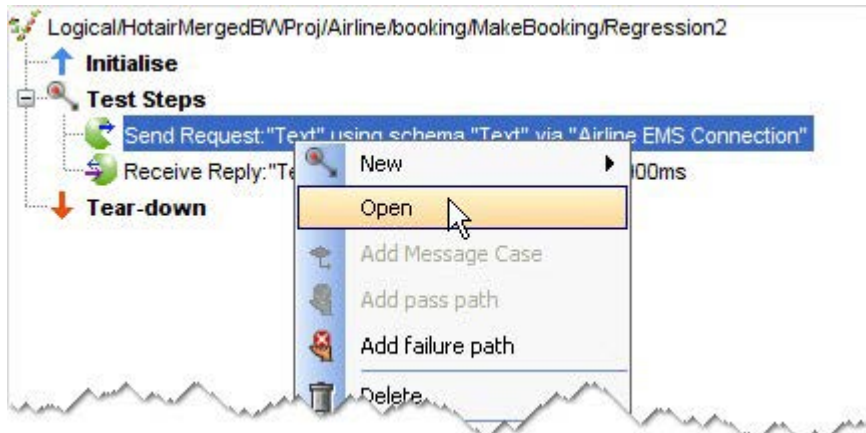
NOTE: The selected action will be inserted below the currently selected phase or action.

Additionally, actions can be added from the test editor's context menu by right-clicking on the phase or step after which you want to insert an action. Actions are available by group under the **New** context menu item.



Edit Test Actions

Test actions can be edited by double-clicking them in the test editor, or by selecting **Open** from the action's context menu. If the Action Editor is enabled, test actions can be edited inline, below or to the right of the test editor (see [The Action Editor](#)).



NOTE: The details of individual test actions are described in [Appendix A: Test Actions](#).

Delete Test Actions

Test actions can be deleted by selecting **Delete** from the action's context menu, or by selecting the desired action and pressing the **Delete** key.

Rename Test Actions

When working in the Business View, you can rename a test's label or any of the actions in its various phases. This lets you name the test and any desired test actions in a way that can be easily understood by other users.

To rename an action, select **Rename** from its context menu or select the action and press **F2**.

Change a Test Action's Type

When working in a test, any existing messaging actions, except Unsubscribe, can be quickly changed to another type of messaging action.

To change an existing action, select the desired action type under **Change Action Type** in the action's context menu.

NOTE: When changing an action's type, the existing transport and message content will be retained.

Cut/Copy/Paste Test Actions

Test actions can be cut, copied, and pasted using the action's context menu. When pasting, the action being pasted is inserted below the phase or action that is currently selected.

Move Test Actions

Test actions can be reordered by moving them up or down the list of existing actions. You can move a test action by selecting **Move Up** or **Move Down** from the action's context menu, or you can simply drag an action and drop it into a new position within the test.

When using drag and drop, you can select multiple test actions (sequential or non-sequential) and move them to a new position within the test.

NOTE: When moving an action by means of the context menu, you can not move it out of its current test phase.

Enable or Disable Test Actions

By default, every test action is enabled in a test when it is first added. If desired, however, a test action can be disabled, letting you skip its execution within the test without having to remove it.

To change the state of test actions, select them in a test (use **Shift** or **Ctrl** to select multiple actions), right-click any of the actions, then select the desired state (**Enable** or **Disable**) from the context menu. If the selected actions are currently in both states, then both the **Enable** and **Disable** options will be available. If all actions are in one state, then only the opposite state will be available in the context menu (for example, if all actions are enabled, then only **Disable** will appear in the menu).

If using the docked action editor, you can toggle the state of the selected test action using the **Enabled** option in the editor.

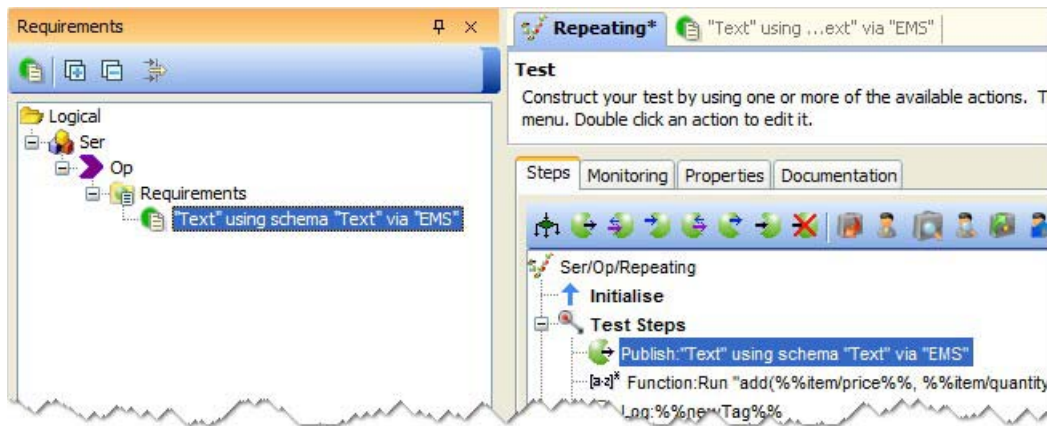


NOTE: When a test action is disabled in a test, its result in the console is reported as "skipped."

Save Messages as Requirements

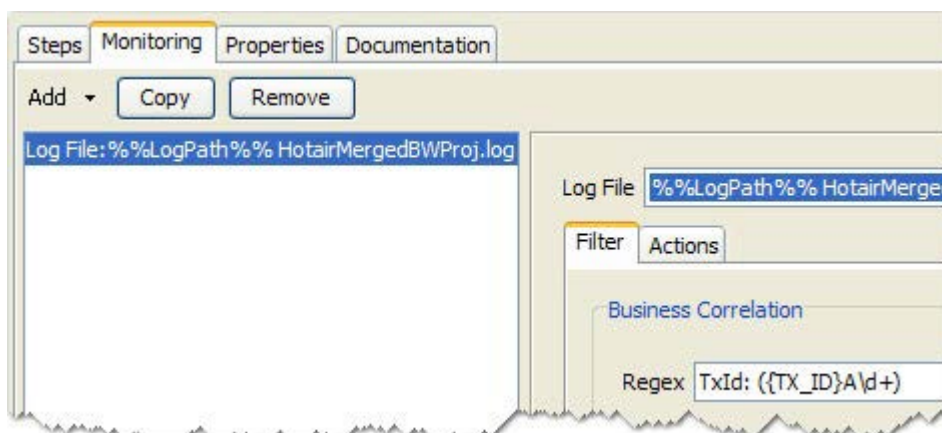
Messaging test actions (for example, Publish, Subscribe, and so on) can be saved as requirements by dragging them from an open test and dropping them into a valid branch within the tree in the Requirements or Test Factory perspective. The contents of the message (that is, header options, transport, and so on) in the selected action will be reproduced in the new message. The new message will be named according to the description of the action that created it.

In the example below, the Publish action was dragged into the “Op” operation. The resulting message was then named according to the Publish action’s description.



6.2.8 Monitor a Log File During a Test

If a log file has been configured on any of the bound infrastructure components of the project (see [Add a Log File to Monitor on an Infrastructure Component](#)), that file can be monitored during the execution of a test. To configure how the file should be monitored, click the **Monitoring** tab of an open test in the Test Factory (**F10**).



To add an existing log file to monitor, click **Add** and select **Log File** from the menu. If you want to copy an existing log file, select its entry and click **Copy**. To remove an existing log file, select its entry and click **Remove**.

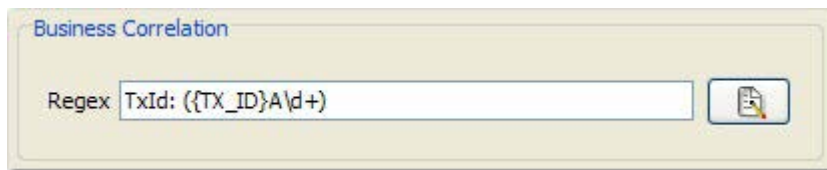
On the right side of the window, select the specific log file (that is, one that has already been configured for monitoring in the Logical View of Architecture School) to monitor from the **Log File** menu.

Monitoring Remote Log Files

If the log file to be monitored is on a remote host (defined in the current environment), you must configure a Role on the remote host that defines how Rational Integration Tester should connect to it. See [Host](#) and [Identity](#) for more information about configuring roles on a host.

Filter Options

Under the **Filter** tab, you can filter the log entities that you want to capture using the Business Correlation ID and a starting and ending transaction filter.



In the **Regex** field under **Business Correlation**, you can specify an expression that is unique to each business transaction (for example, a reservation number). You can also click Edit to use the expression builder to create the regular expression (see [Log File Format Options](#) for more information).

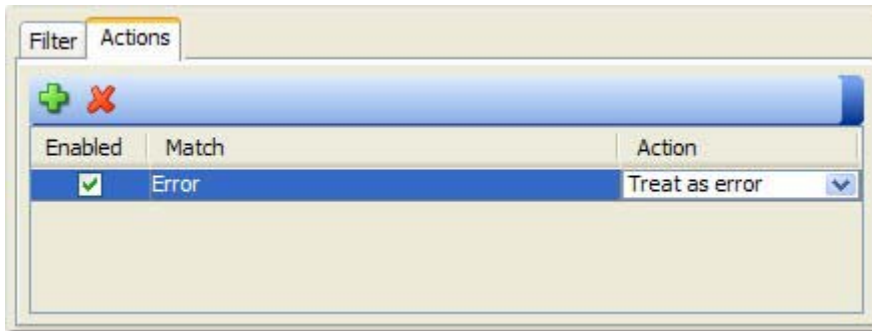
The Business Correlation is used in conjunction with the Technical Correlation, which is configured on the logical component's log file setup. If you can match the Technical Correlation in entities where the Business Correlation has been found, you can then filter and identify only the relevant log entries in which you are interested.

In the example shown above, TX_ID is used in a capture group defined in the regular expression (TX_ID is a tag into which the returned reservation number is being stored). At runtime, TX_ID will be extracted from the log and must match the value in the tag. If it does, the Business Correlation is matched and Rational Integration Tester continues to try and match the Technical Correlation as well.

In the **Transaction** panel, you can use the **Starts with** and **Finishes with** filters to reduce the number of entries that are being matched. In the Starts with field, enter an expression that will filter anything coming before the expression. In the Finishes with field, enter an expression that will filter anything coming after the expression.

Log Monitor Actions

Under the **Actions** tab you can define specific test actions that should be taken when a log entity matches the specified expression. When monitoring a log file during a test, you would typically create actions that will handle unexpected conditions (for example, errors or exceptions) in the log.








NOTE: Actions are matched in the order that they are defined in the editor. If no actions are defined or none of the defined actions are matched, the entry will be sent to the test console as a normal message.

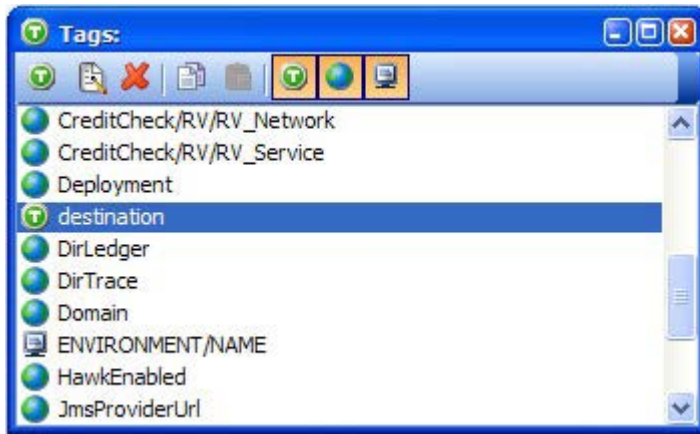
For more information about configuring log monitoring actions, see [Log File Actions](#).

6.2.9 The Tag Data Store

The Tag Data Store, accessed from the context menu anywhere within a test, is the repository and access point for all of the data tags that are available to a given test. The data store contains the following types of tags:

-  Environment tags
-  System tags
-  Test tags
-  Global test tags
-  Overridden Environment tags

The image below illustrates a typical Tag Data Store. The tags are listed in alphabetical order.



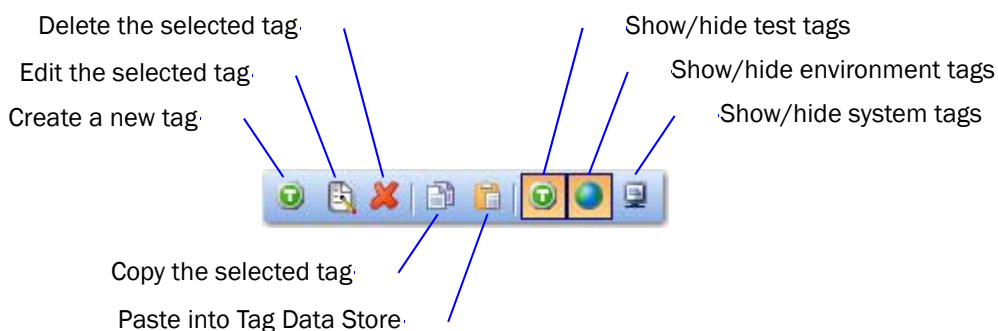
NOTE: The tool tip for the tag under the mouse-pointer displays the optional description field for the tag.

You can create a logical hierarchy of tags by separating multiple name sections with the '/' character (for example, 'JMS/URL' and 'JMS/User' will be displayed together under the 'JMS' section).

When the Tag Data Store window is open, you can drag tags directly into appropriate fields of the test step that is currently being edited.

The Tag Data Store Toolbar

The toolbar at the top of the Tag Data Store provides all of the actions needed for working with tags. The icons available in the toolbar are described below.



The same actions for creating and modifying tags from the toolbar can be accessed from the context menu.

Filtering Displayed Tags


You can show or hide the available tag types in the Tag Data Store by toggling the tag type icons in the toolbar.

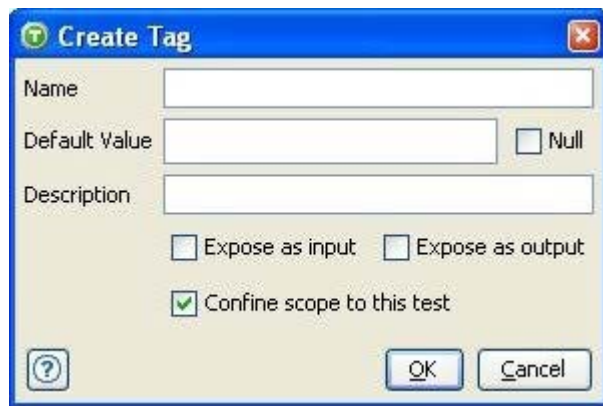
When the icon for a specific tag type is highlighted in the toolbar, that type of tag will be displayed. If the icon is not highlighted, that type of tag will be hidden.

By default, all tag types are displayed in the Tag Data Store.

Creating Tags

New test tags can be created in one of two ways:

- Create one manually by clicking on the  icon




In the **Create Tag** dialog, enter the name, optional default value, and description for the new tag. To set the default value of a new tag to Null, select the “Null” check box.

Use the “Expose as input” and “Expose as output” options to specify whether or not the tag should be available as input, output, or both at runtime. These options determine which tags are available when configuring the Input and Output interface for tests and stubs (see [Test Properties](#) for more information).

Enable or disable the “Confine scope to this test” option. If disabled, a global test tag will be created. See [Test Tags](#) and [Global Test Tags](#) for more information.

- Create multiple tags by pasting a list of tag names directly into the tag data store window. First create a list of tag names in a text-based file – the names may contain '/' characters to create a hierarchy as mentioned above (for example,

'MYTEST/Name'). Next, copy the list of tag names and paste it into the data store window (press **Ctrl + V** or click the paste icon .

NOTE: Blank lines are ignored when pasting tags.

Tag Data Store Preferences

If desired, you can configure the Tag Data Store to pop up automatically when you are editing a test step. This behaviour can be set with the “Show Tag Data Store with editors” option in Rational Integration Tester’s General Preferences.

Test Tags

Test tags are used within tests to store data values. They are normally created within the Tag Data Store itself at the beginning of the test design, but it is also possible to create them when editing some test steps (for example, when storing the output of an action).

NOTE: The scope test tag value is limited to the currently running test only.
Multiple tests can not use the same test tag to store common data.

Global Test Tags

Global test tags can be used for the duration of the execution session. This means that tests run from within a suite can share the same tag value.

NOTE: Global test tags must be defined in all tests that use this shared value.

Environment Tags

Environment tags are created in the environment and can not be edited in the Tag Data Store window (see [Environments](#) for more information).

Overridden Environment Tags

It is possible to create a test tag with the same name as an environment tag. In this case, the environment tag value is no longer used – it is overridden. The modified icon is displayed and the value may be overwritten in the test.

NOTE: If the tag is not overridden it will have an empty value.

System Tags

System tags, listed below, are read-only and are populated by the system at execution time.

Name	Content
AGENT/NAME	When executing as part of a distributed performance test, this contains the name of the agent.
ENVIRONMENT/NAME	The name of the environment that is currently in use.
PROJECT/ROOT	The path to the root of the current project.
PROJECT/ROOT_DIRECTORY	The OS specific path to the current project directory.
SUITE/NAME	The name of the suite directly containing the executing test.
SUITE/PASSED	When running a suite, contains a boolean that will be true if the suite has passed.
SUITE/PATH	The path to the suite directly containing the executing test.
SUITE/RESULT	When running a suite, this contains the current result.
SUITE/SCENARIO/NAME	The name of the scenario currently being executed by the suite.
SYSTEM/CURRENT_DATE	The current system date (yyyy/MM/dd).
SYSTEM/CURRENT_DATE_TIME	The current system date and time (MMM d, yyyy h:mm:ss a).
SYSTEM/CURRENT_TIME	The current system time (HH:mm:ss.SSS).
SYSTEM/CURRENT_UTC	The current UTC system time.
SYSTEM/FAILURE	Contains information about the last exception thrown during test execution.
SYSTEM/HOST_NAME	This host name of the local machine.

Name	Content
SYSTEM/INSTANCE_ID	The instance number of the test.
SYSTEM/USER_NAME	The currently logged in user.
TEST/ITERATION/NUMBER	The iteration number of the task with which the tag is associated.
TEST/ITERATION/START_DATE	The start date of the task with which the tag is associated.
TEST/ITERATION/START_TIME	The start time of the task with which the tag is associated.
TEST/NAME	The short form of the tests name.
TEST/PASSED	When running a test, this contains a boolean that will be true if the test has passed.
TEST/PATH	The full form of the tests name.
TEST/PHASE/NUMBER	When executing as part of a distributed performance test, this contains the current phase number.
TEST/RESULT	When running a test, this contains the current result.
TEST/START_DATE	The start date of the test with which the tag is associated.
TEST/START_TIME	The start time of the test with which the tag is associated.

6.2.10 Add a Message Case

Message cases are used in Message Switch actions. To add a message case to an existing message switch, right-click the message switch and select **Add Message Case** from the context menu.

Each message case is configured to match a specific message that has been captured (more generically) by the message switch. When combined, the message switch and any one of its message case actions work like a complete Subscribe action. The message switch specifies the general transport and destination information, while the message case performs detailed filtering and contains the assert and store options.

If the message switch is part of a stub, the **Config** tab provides the option to check and/or set the state of the stub. Under **Start State** and **Finish State**, you can select one of the available session states (configured under the **Properties** tab of the stub). If a start state is selected, the actions within the message case will only be executed if the stub session equals the selected state. If a finish state is selected, the session will be set to that state once the actions under the message case finish executing.

Message Case "Text" using schema "Text" [addNumbers/addition/Ad...

Message Case
Configure the filtering and validation for this Case action

Config | Filter | Assert | Store

Session

Start State: FLIP

Finish State: FLOP

Filter Expressions

lt(%%accountBalance%%, -1250)

eq(%%accountType%%, "CURR")

'OR' Expressions ☐

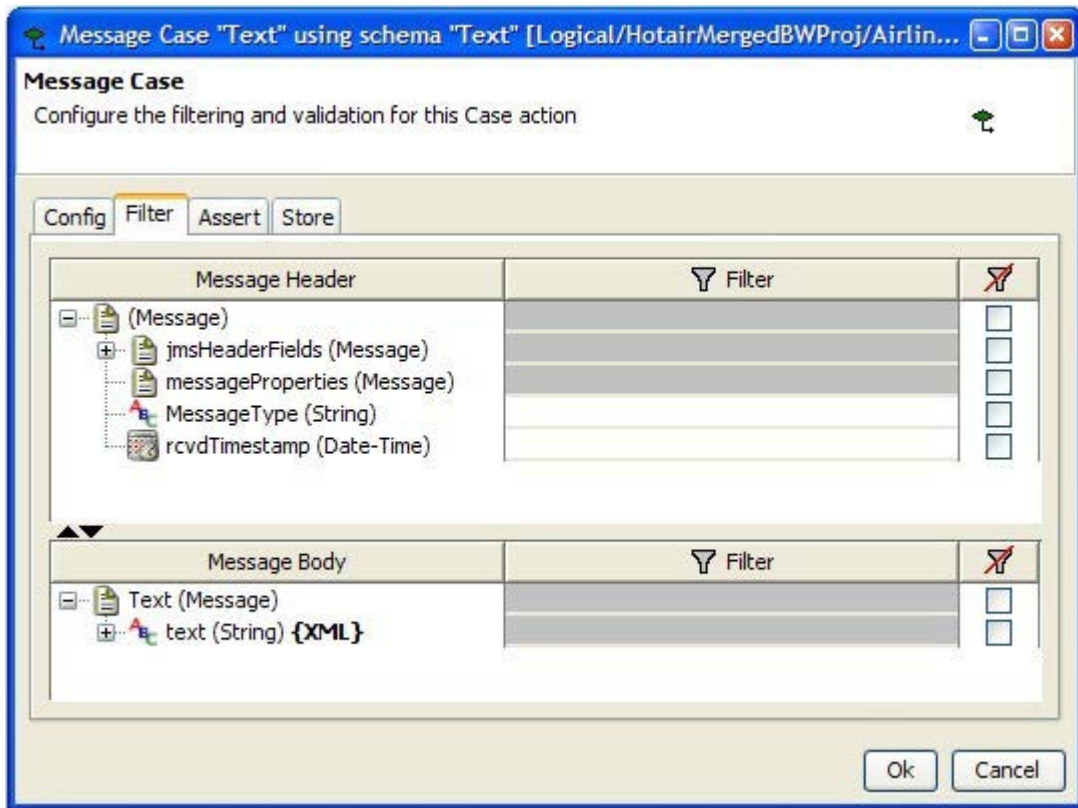
Add Delete Test Expressions

Ok Cancel

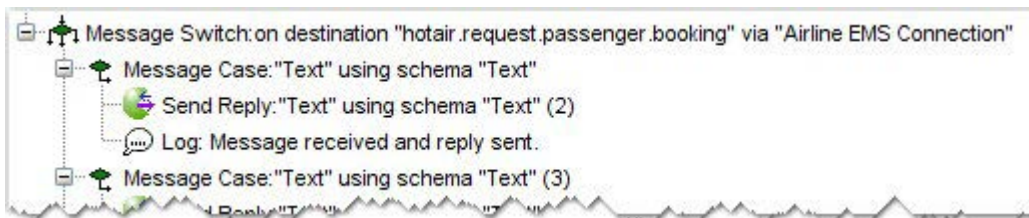
Additionally, before any of the actions within the message case can be executed, any filter expressions have been added under the **Config** tab must evaluate to true (or at

least one if the 'OR' expressions option is enabled). Filter expressions are available for message cases in both tests and stubs.

The standard filter, assert, and store actions are available under the remaining tabs within the message case.



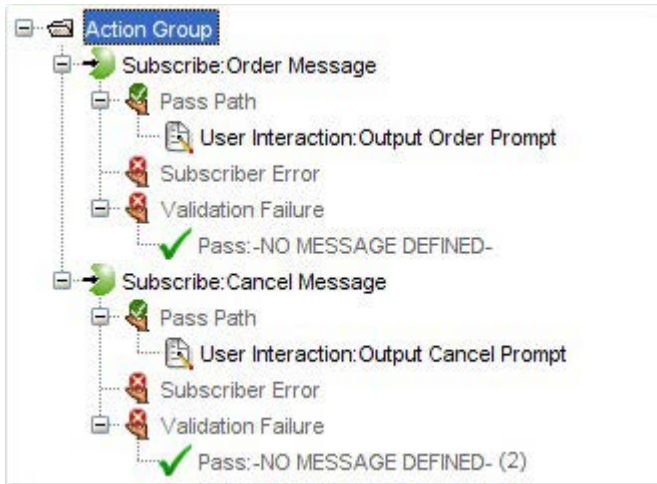
The message case contains its own actions that can be carried out when a matching message is received.



6.2.11 Setting Pass Paths

Rational Integration Tester lets you define action sequences to be carried out when a subscription-based action passes (that is, Subscribe, Receive Request, and Receive Reply). The **Add pass path** option can be selected from a test action's context menu.

The image below illustrates how you can configure a test to handle two different messages that are sent on the same subject or topic:



Both subscribers are using the same topic name and they are placed within an Action Group so that they both receive the same messages. The validation is set up so that it will pass in the first one with an *Order Message* and in the second one with a *Cancel Message*. If an order message comes in, the first subscriber will pass validation and execute the pass path steps – in this case, a user interaction is run, but any number of steps is possible. The same order message is processed in parallel by the second subscriber – in this case it fails validation but the error is trapped and the pass step is executed.

For a *Cancel Message*, the opposite happens (that is, the first subscriber ignores the message while the second subscriber processes it).

NOTE: If you change the action type for a test step (for example, Subscribe to Publish), any existing pass paths will be removed.

6.2.12 Setting Failure Paths

If you want your test to take different steps when a specific action fails, you can create a failure path and define sequences of actions for it (for example, a Publish action can fail to publish a message, or a subscriber can fail to subscriber properly or an incoming message might not pass validation). The **Add failure path** option can be selected from a test action's context menu:

When this option is selected, nested failure path actions are automatically added to the sequence:



A series of actions can then be nested under the failure path actions, as shown below. In this case, the corresponding failure notification messages are displayed and the test is set to fail.



NOTE: If you change the action type for a test step (for example, Subscribe to Publish), any existing failure paths will be removed.

6.2.13 Executing Tests

Test execution can be initiated in the Test Factory, but the test will actually run in the Test Lab perspective. After starting a test (as described below), the Test Lab perspective will be displayed to let you monitor the test's execution. See [Test Lab](#) for more information.

Running Tests in the Current Environment

Tests can be executed in the currently selected environment in any of the following ways:

- Select one or more tests and press **F5**.
- Right-click one or more selected tests and select **Run** from the context menu.
- Select one or more tests and click the **Run** icon in the main Rational Integration Tester toolbar.
- Click the small arrow next to the **Run** icon in the main Rational Integration Tester toolbar and select the test you want to run (if the test had been run recently).

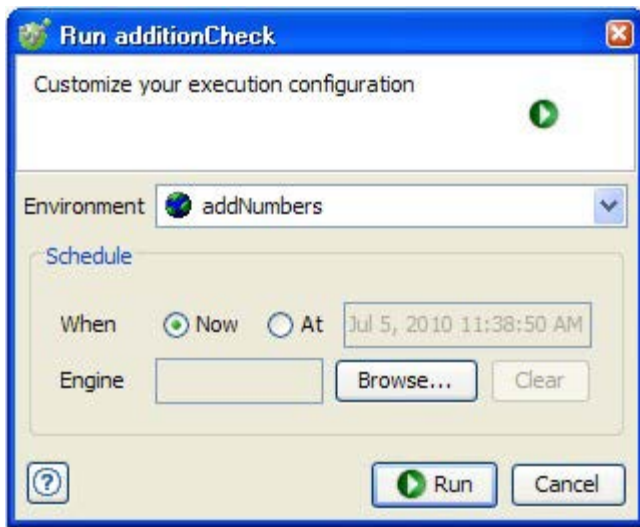
NOTE: Depending on the current “Run Resources” preferences, the test will be executed from its current state (including any unsaved changes) or from its last saved state. See [Changing Preferences](#) for more information.

Customizing the Execution of a Test

A selected test can be executed in an environment other than the current environment by using the custom execution dialog, as follows:

1. Open the custom execution dialog in one of the following ways:
 - Select a test and press **Shift + F5**.
 - Right-click a test and select **Run...** from the context menu.
 - Select a test, click the small arrow next to the **Run** icon in the main Rational Integration Tester toolbar, then select the **Run...** option.

The custom execution dialog is displayed.




2. Select the environment in which the test should be executed from the **Environment** dropdown.
3. To execute the test remotely, click **Browse** next to **Engine** and select the agent that should execute the test – only one test per engine.
4. Click **Run** to execute the test.

NOTE: No other custom execution options are currently supported.

6.3 Performance Tests

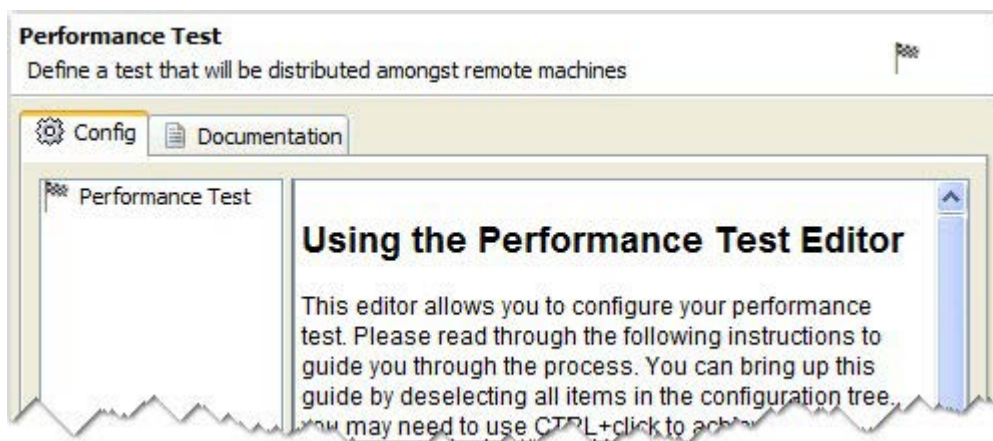
A performance test includes one or more instances of a Rational Integration Tester test plus a number of probes that measure the load on external systems (for example, messaging servers). This section describes how to create a performance test.

1. Select the operation in which you want to create the performance test.
2. Click the small arrow next to the **New** icon  in the Test Factory toolbar and select **Performance Test**, or right-click the operation and select **New > Tests > Performance Test** from the context menu.

NOTE: If at least one performance test has already been created, you can right-click the Performance Tests folder or one of the existing tests and select **New > Performance Test**.

3. Provide a name for the new performance test when prompted, then click **OK**.

The new test is opened to the right, in the editing panel. The left side of the panel is a configuration tree, the right side displays documentation or a configuration panel when an item is selected in the tree.



NOTE: Once you select the performance test or any of its components in the editing panel, the test properties are displayed to the right. To display the documentation again, press **Ctrl** and click the root of the test.

Please see the *IBM Rational Performance Test Server Reference Guide* for additional details about configuring and running performance tests.

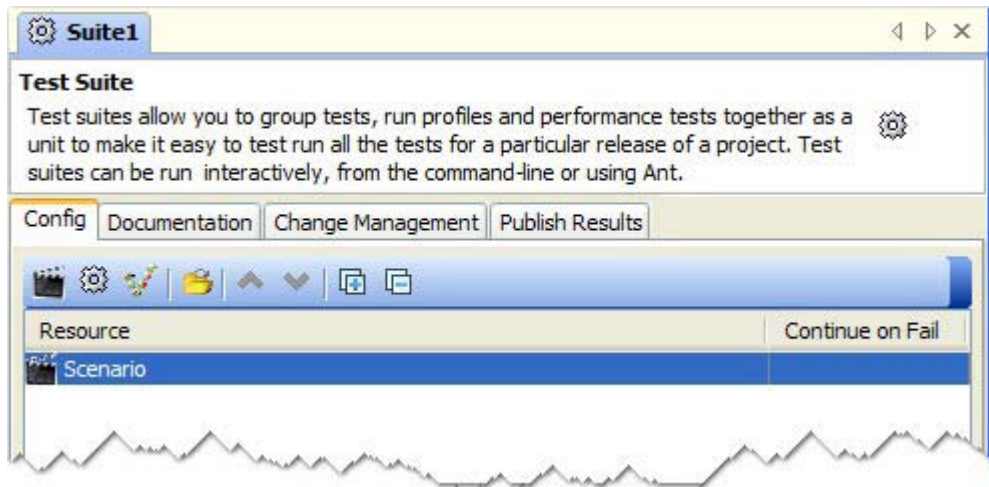
6.4 Test Suites

A Test Suite represents a group of tests, test suites, and scenarios. Any combination of these items can be included, but a test suite must always contain at least one scenario and one test. Test suites are used mainly to construct reusable, packaged regression tests that can be run across multiple environments, or repeatedly within the same environment. See the following sections for more information:

- [Creating Test Suites](#)
- [Manipulating the Contents of a Test Suite](#)
- [Using Test Suite Scenarios](#)
- [Configuring Change Management Options](#)
- [Configure Results Publishers](#)
- [Continue on Fail Option](#)
- [Executing Test Suites](#)

6.4.1 Creating Test Suites

A test suite can be added to an existing operation or to an existing test suite folder, either from the context menu or the Test Factory toolbar. After providing a name for it, the test suite is opened in the editing panel to the right of the Test Factory tree.



When a test suite is first created, it is empty except for a scenario. The scenario is a container that controls how the items inside should be executed. A scenario is also used to configure probes that are available within the test suite.

A test suite can contain one or more tests, test suites, or scenarios – including any combination thereof.

NOTE: When executed, a test suite's steps are processed from top to bottom.

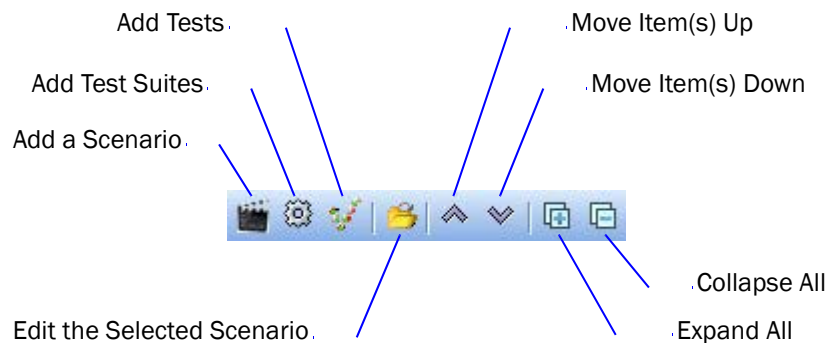
6.4.2 Manipulating the Contents of a Test Suite

Scenarios, test suites, and tests can be added to or removed from an existing test suite, and they can be moved up or down within the scenario that contains them. Scenarios can be editing within a test suite, and the item trees within a test suite can be expanded or collapsed. All of the test suite actions can be performed using the test suite toolbar or by right-clicking an existing item and using the context menu.

NOTE: When adding an item to a test suite, the new item is added below the currently selected item. In the case of a scenario, new items are added within the scenario.

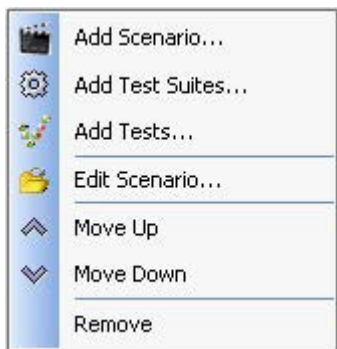
The Test Suite Toolbar

The test suite toolbar is displayed above the contents of a suite in the editing panel. The toolbar is shown below, including a description of the icons it contains.



The Context Menu

The context menu, available by right-clicking any item within a test suite, contains most of the same actions as the toolbar – the “expand” and “collapse” actions are not available, and the Remove action is included.



Adding an Item to a Test Suite

Tests and test suites can be dragged into an open suite directly from the Test Factory tree in the Test Factory. Alternatively, you can use the steps below to add items to a test suite.

1. Select the item in the test suite below which you want to add one or more new items.
2. Click the icon of the item to add in the test suite toolbar or select the item to add from the context menu.
3. Select the item(s) to add from the project resource dialog (use **Ctrl** or **Shift** to select multiple items).

The new items will be added to the test suite below the item that had been selected.

Moving Items in a Test Suite

The execution order of items in a test suite (scenarios, test suites, and tests) can be modified by moving them before or after other items in the suite.

NOTE: Items can only be moved within the scenario that contains them.
Items can not be moved from one scenario to another.

Use the “move” options in the test suite toolbar or context menu to move an item up or down within its scenario.

Removing Items from a Test Suite

One or more items can be removed from a test suite by selecting them (use **Ctrl** or **Shift** to select multiple items) and using the toolbar/context menu options, or by pressing the **Delete** key.

NOTE: If a test suite resource is removed outside of the suite (for example, one of the suite’s tests is deleted from the project tree in the Test Factory), that resource will be highlighted using a red box in the suite. The suite will still run but the missing resource can not be executed.

6.4.3 Using Test Suite Scenarios

In a test suite, a scenario controls how to execute the items it contains. A scenario can be added to a test suite (for information about this, refer to [Manipulating the Contents of a Test Suite](#)).

NOTE: If you are using Rational Integration Tester 8.0.1 (or later) and Rational Test Control Panel 8.0.1 (or later), you can copy scenarios from Rational Test Control Panel to Rational Integration Tester's Scenario Editor. (For information about creating and running scenarios in Rational Test Control Panel, refer to *IBM Rational Test Virtualization Server Reference Guide*.)

The Scenario Editor enables you configure a scenario. To open the Scenario Editor, double-click the scenario on the **Config** tab of the Test Suite window. Alternatively, right-click the scenario on the **Config** tab of the Test Suite window, and click **Edit Scenario** on the shortcut menu.

Configuring a Scenario's Name and Runtime Options

The **Config** tab on the Scenario Editor dialog box enables you to configure a scenario's name and runtime settings.

The following table describes how to use the controls on the **Config** tab.

Control	Description
Name	Enter a descriptive name for the scenario, up to 64 characters.
Environment	<p>This drop-down list enables you to specify the Rational Integration Tester environment where the currently selected scenario will run, namely the "execution environment".</p> <p>The execution environment is the environment where the test suite and the scenario are running, and it can be selected by using this drop-down list or it can be selected at run time.</p>
Lock the environment in Rational Test Control Panel	This check box enables you to lock the execution environment in Rational Test Control Panel, thus preventing other users from using the environment. (For more information about environment locking, refer to <i>IBM Rational Test Virtualization Server Reference Guide</i> .)
Run tests in parallel	Enable this option to execute tests in the scenario in parallel, which can speed up the overall execution of the test suite. Only test that qualify for it (for example, tests not dependent on the results of other tests) will be run in parallel mode.
Pacing	<p>Control how frequently the tests in a suite will run.</p> <p>Options are as follows:</p> <ul style="list-style-type: none">• pause between tests for: pause for n seconds after executing one test before executing the next (it will not pause after the last test)• run tests no more frequently than once every: if a test completes in less than n seconds, pause for the remaining time (for example, if n = 5 seconds and a test completes in 3 seconds, pause for 2 seconds before executing the next test). <p>NOTE: If a scenario contains one or more test suites, the pacing for any child suites is not inherited from the parent (that is, pacing must be specified in the scenario that contains the child suite).</p>
Fail the scenario if remote stubs fail to stop	This check box enables you to specify whether the currently selected scenario should be failed if any remote stubs fail to stop.

NOTE: You can run a scenario created in one environment on any other environment to which you have access. (For more information about

environments, refer to *IBM Rational Test Control Panel System Administration Guide*.)

Configuring Scenario References Settings

The **References** tab on the Scenario Editor dialog box enables you to configure references or dependencies from the operations that are part of the test suite.

NOTE: The operation containing the test suite references itself.

The following table describes how to use the controls on the **References** tab.

Control	Description
Load scenario from Rational Test Control Panel	<p>Clicking this button opens the Load scenario from Rational Test Control Panel dialog box, which enables you to select the configuration of a scenario saved on Rational Test Control that will be copied into the currently selected scenario, thus overwriting any conflicting stubs that are already configured.</p> <p>NOTE: Only scenarios that are available under the domain configured for the project will be available.</p> <p>If any operations are not satisfied by the selected scenario, you can specify how they are handled while being loaded into Rational Integration Tester.</p> <p>Options are as follows:</p> <ul style="list-style-type: none">• Do not change: Any operations that are not assigned a stub by the scenario loaded from Rational Test Control Panel will keep their existing selection.• Clear selection: Any operations that are not assigned a stub by the scenario loaded from Rational Test Control Panel will have their selection cleared.• Set to Live System: Any operations that are not assigned a stub by the scenario loaded from Rational Test Control Panel will have their selection changed to “Live System” so that all traffic goes to the live system. <p>Clicking OK on the Load scenario from Rational Test Control Panel dialog box loads any stubs for the operations listed on the References tab from the currently selected scenario.</p> <p>Scenarios from Rational Test Control Panel can be loaded into a test suite scenario from any environment. The scenario can then be executed in any environment as long as the stubs it contains exist at the correct version in that environment.</p> <p>Stub response time configuration and agent attributes are imported from the Rational Test Control Panel scenario but other stub configuration is not imported.</p> <p>NOTE: If Rational Test Control Panel is not running or the Rational Integration Tester project does is not correctly configured with the correct Rational Test Control Panel URL, this button will be unavailable.</p>
Reference	<p>This column displays any referenced operations for the tests within this scenario.</p> <p>NOTE: By default, the References window is sorted in ascending order using the Reference column. To change the sort, or to sort by any other column, click the title of the desired column (sorting will cycle through ascending, descending, and no sort as you click).</p>

Control	Description
Satisfied by	<p>If a referenced operation can be stubbed, clicking the cell under Satisfied by displays a drop-down list containing the following options:</p> <ul style="list-style-type: none"> • Blank: Selecting this option means that no changes will be made to anything currently running in Rational Test Virtualization Server. <p>NOTE: This option is particularly useful if you are working with nested scenarios and you want to prevent the tearing down of any stubs in a parent scenario.</p> <ul style="list-style-type: none"> • Live system: Selecting this option stops any stubs running for this operation either in Rational Integration Tester or in the execution environment in Rational Test Control Panel when the scenario is executed. <p>An option for each of the stubs available for the appropriate operation.</p> <p>For other rows displayed on the References window, clicking the operation under Satisfied by displays a drop-down list containing the following options:</p> <ul style="list-style-type: none"> • Blank • Live system
Location & version	<p>If a stub has been selected (under Satisfied by) for a referenced operation, clicking the cell under Location & version displays a drop-down list containing the following options:</p> <ul style="list-style-type: none"> • Local • Remote <Stub version number> <p>A remote stub is a stub that has been previously published from Rational Integration Tester to Rational Test Control Panel. The stub version number displayed refers to the version number specified when it was published. Only stub versions that have been published to the domain configured for the current project will be available. (For information about publishing stubs, refer to <i>IBM Rational Test Virtualization Server Reference Guide</i>.)</p>

Control	Description
Config	<p>If a stub has been selected (under Satisfied by) for a referenced operation, clicking the cell under Config displays the Stub Configuration dialog box.</p> <p>Using the Response time tab on the Stub Configuration dialog box, you can override response time settings that are configured in the currently selected stub's properties. (For information about these response time settings, refer to <i>IBM Rational Test Virtualization Server Reference Guide</i>.)</p> <p>NOTE: The Stub Configuration dialog box cannot be used to import or modify a stub's input tags, behaviour properties, or logging settings.</p>

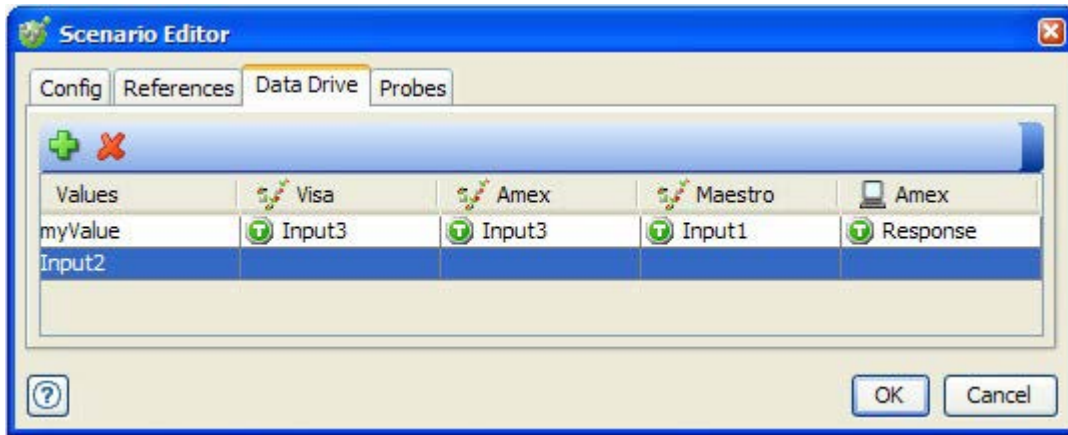
After making any desired changes on the References tab, click **OK** to close the Scenario Editor dialog box.

When you run the modified test suite containing the modified scenario, the following will happen:

- Any stubs that were running on the selected environment in Rational Test Control Panel will stop for any operations in the scenario where **Live System** or a stub was selected.
- The selected environment in Rational Test Control Panel will be locked if you clicked the **Lock the environment in Rational Test Control Panel** check box on the References tab.
- Any stub or stubs linked to the modified scenario will execute.
- When the scenario finishes, all stubs (both remote and local) that it started will be stopped.

Configuring Scenario Data


Click the **Data Drive** tab to configure static values that can be passed (at runtime) into any of the input tags used by items in a test suite scenario.



NOTE: The current test suite and the tests or stubs that it contains must be saved before editing the contents of the **Data Drive** tab.

By default, the rows in the **Data Drive** table are listed in the order in which they were added. If desired, the data can be sorted by column – in ascending or descending order – by clicking on the column title. Click once to sort in ascending order, click again to sort in descending order, and click a third time to restore the default order.

NOTE: The width of the columns displayed under the **Data Drive** tab can be adjusted by dragging the column separator to the left or right, and customized widths will be stored with the scenario. If new columns are added (that is, if a new test or stub is added to the scenario), they will be added to the right while maintaining the width of the existing columns.


To add an entry to the table, click the  icon. A new row is created with a **Values** column and a column for each test and referenced stub in the scenario. You can paste data into the table by pressing **Ctrl + V** or by right-clicking an existing row in the table and selecting **Paste** from the context menu. A new row is created with the pasted text as the value. Pasting multiple lines will create multiple rows.

NOTE: A tab character in the pasted text will create a new row in the table. If the pasted text contains **both** tabs and multiple lines, each line will create a new row in the table, but tab characters will be preserved (that is, they will not create a new row).

Double-click the **Value** field in a row to enter or edit the value to be passed in. Under the desired test or stub, select the tag (previously defined for the Input interface of the test or stub – see [Test Properties](#)) into which the value should be passed.

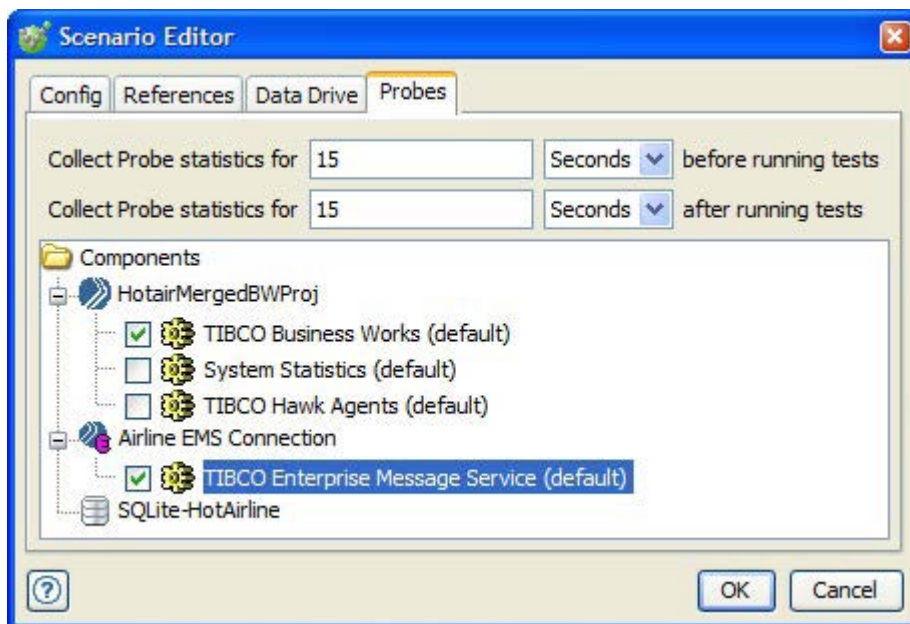
NOTE: If a test is listed to be executed more than once in the scenario, it will be configured the same way in each case.

When the scenario is executed as part of the containing test suite, the selected values will be passed into the specified tags, overriding their current values.

To remove a row from the table, select the row and click the  icon.

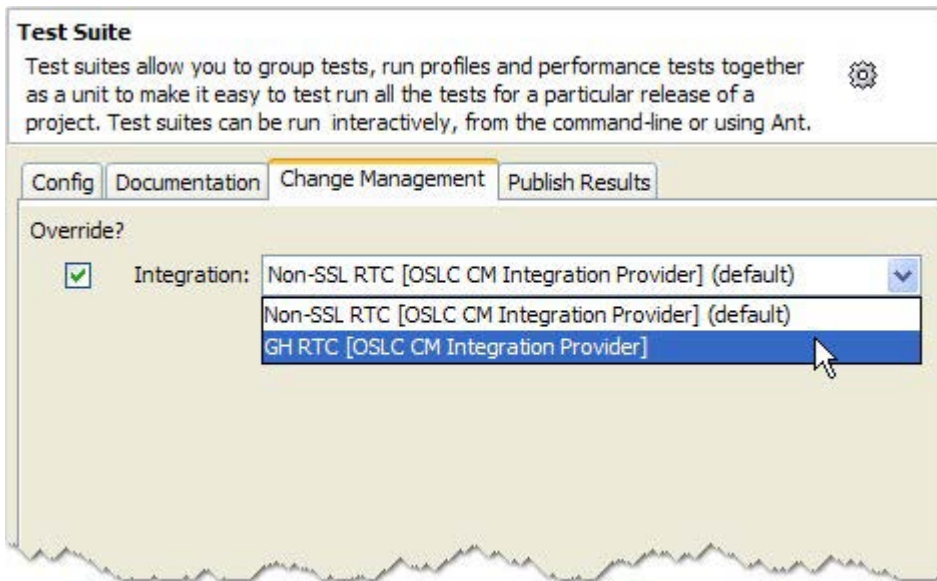
Configure Scenario Probes

Performance statistics can be collected in the scenario using the **Probes** tab. Enter the amount of time to collect statistics (before and after running tests) and enable different statistics by ticking the check box next to the desired probe. See the *IBM Rational Performance Test Server Reference Guide* for more information about probes.



6.4.4 Configuring Change Management Options

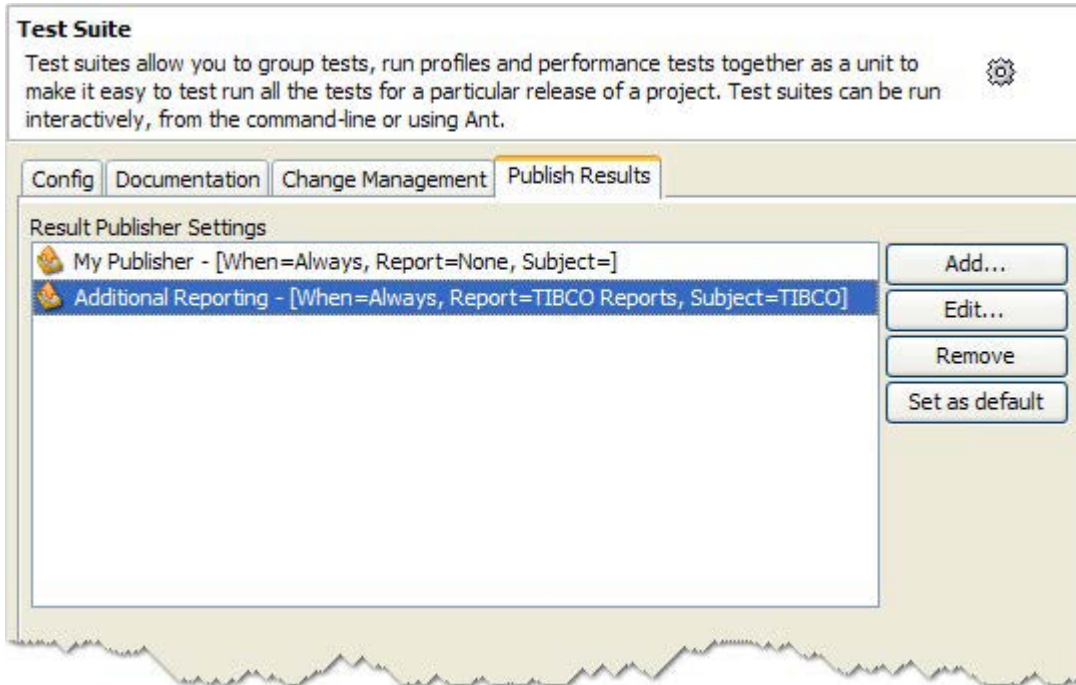
Rational Integration Tester can be integrated with OSLC compliant change management systems for raising defects within the context of a Rational Integration Tester test asset. If multiple change management integrations have been configured in the current project, you can override an inherited integration (from an operation or service component higher up the project resource tree) or the default integration under the **Change Management** tab, letting you specify a different integration to be used.



To specify a non-default integration for the current test suite, enable the **Override** option and select the existing integration from the list of those available. Ensure that you save the test suite after making any changes.

6.4.5 Configure Results Publishers

Results Publishers, created under the Project Settings, can be used to automatically publish the results of a suite after it is executed. The publishers to use and the details of what to publish and how can be configured under the **Publish Results** tab.



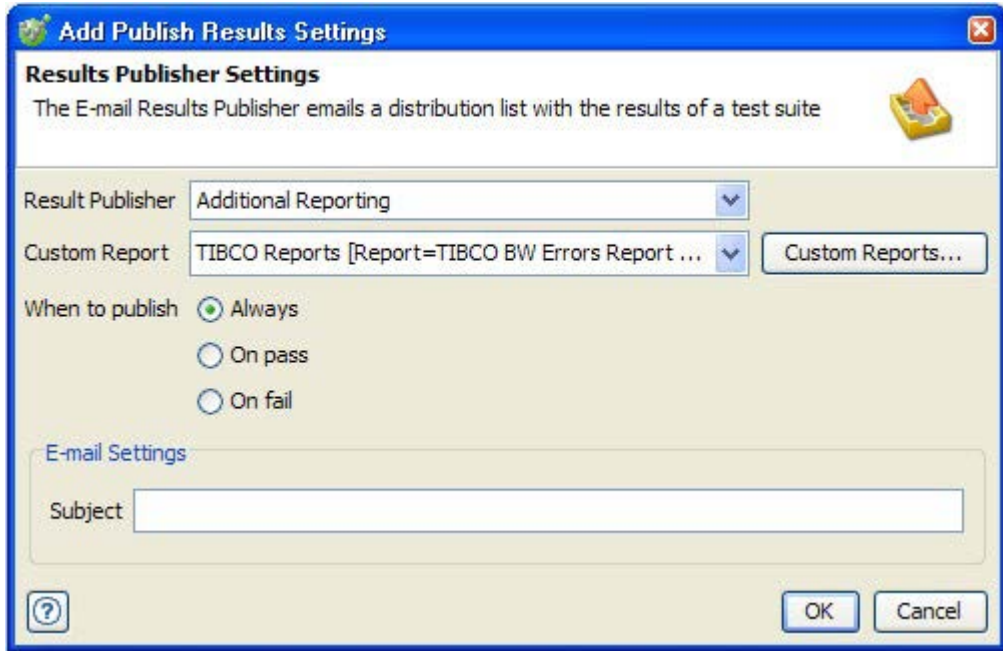
Suite execution results will be published by all publishers added to the suite if they are enabled under the Project Settings and if results publishing is enabled in the project.

Click **Add** to add a publisher to the suite, click **Edit** to modify a selected publisher (see [Add or Modify Results Publisher Settings](#)), or click **Remove** to delete the selected publisher from the suite.

To set a publisher as a default for all test suites (that is, the selected publisher and its settings will be applied to all new test suites when they are created), select the desired publisher and click **Set as default**.

Add or Modify Results Publisher Settings

When adding a publisher to a test suite or editing the settings for an existing publisher, the **Add/Edit Publish Results Settings** dialog is displayed.



When creating settings for a new publisher, select one of the available publishers from the **Result Publisher** combo box. If you are editing the settings for an existing publisher entry, the selected publisher can not be changed.

If any custom reports have been configured, select one to be published from the **Custom Report** combo box. For more information about custom reports and how to create them, see [Custom Reports](#).

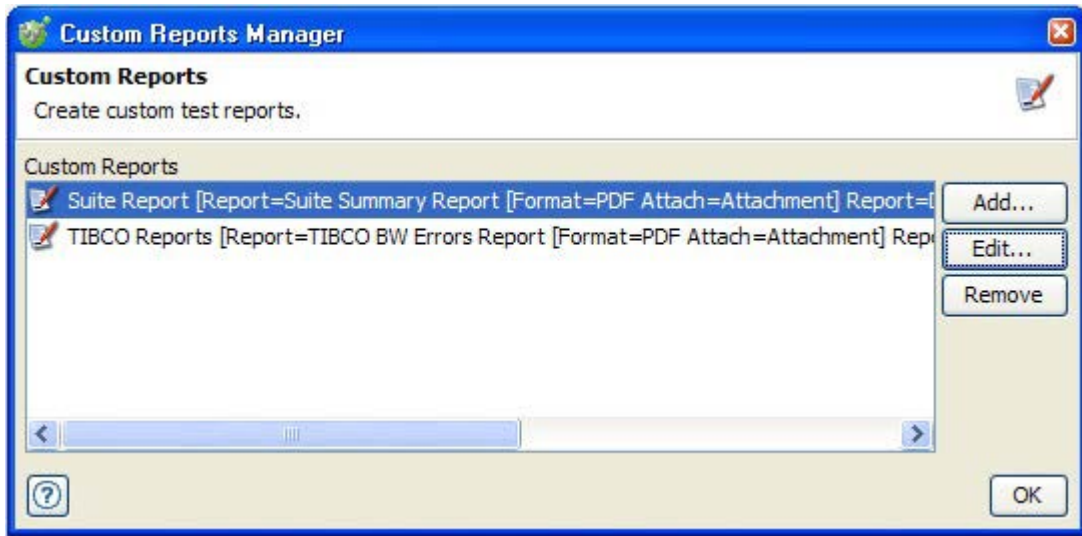
Next to **When to publish**, select the desired option for when the execution results should be published – always, when the suite passes, or when the suite fails.

If you want to override the email subject that has been configured for the selected publisher (under Project Settings), enter the new subject under **E-mail Settings**, in the **Subject** field.

When the publisher settings are finished, click **OK** to return to the test suite editor.

Custom Reports

Custom reports can be managed in the **Custom Reports Manager**, accessible from the **Add/Edit Results Publisher Settings** dialog (see [Add or Modify Results Publisher Settings](#)) or by selecting **Custom Reports** from the **Tools** menu in Rational Integration Tester.

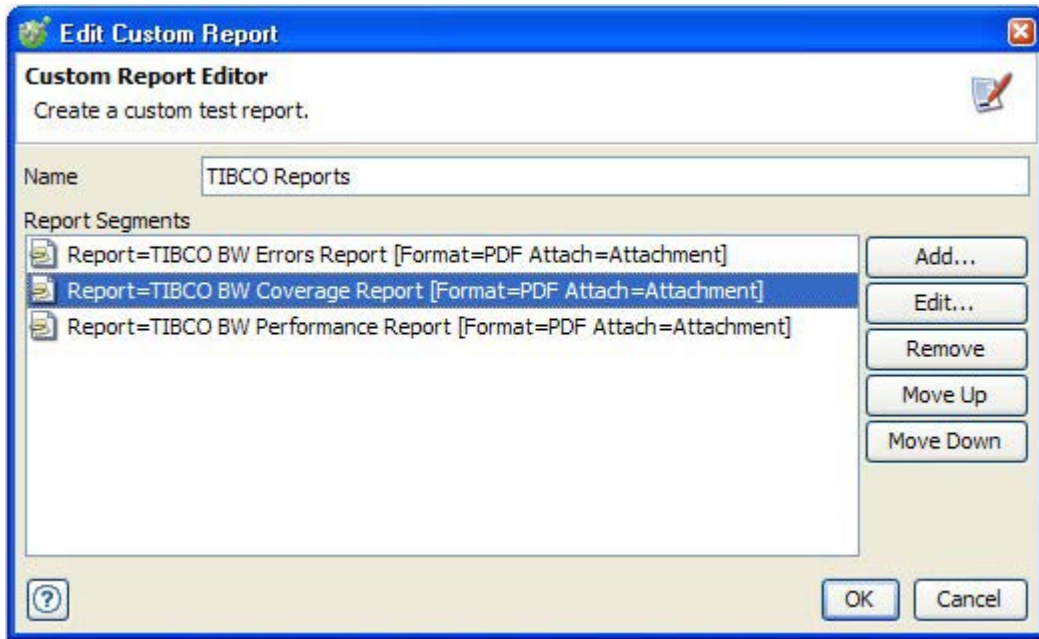


A custom report is a user-defined report containing one or more of the reports that are available in Rational Integration Tester's Results Gallery (report segment). Each report segment contains a built-in Rational Integration Tester report and a format and attachment configuration.

Click **Add** to create a new custom report, or click **Edit** to modify the settings of a selected custom report (see below). To delete a custom report from the list, select it and click **Remove**.

Add or Modify a Custom Report

The **Add/Edit Custom Report** dialog is used for adding or editing a custom report.



In the Name field, add or modify a meaningful name for the custom report.

Under **Report Segments**, you can add, edit, remove, and reorder individual report segments that make up the overall custom report.

Click **Add** to create a new report segment or click **Edit** to modify a selected segment, click **Remove** to delete a selected report segment, and click **Move Up** or **Move Down** (when available) to reorder the selected report segment within the list.

Report Segments

Report segments are the building blocks of custom reports, containing an existing Rational Integration Tester report, some optional descriptive text that precedes the segment in the overall custom report, a report format, and an attachment option.

From the **Report** combo box, select the existing Rational Integration Tester report that should make up the current segment.

The contents of the **Additional Text** field will be printed in the custom report at the start of the selected report segment. A default entry is provided (this field is tag-aware), and this entry can be modified or deleted as desired.

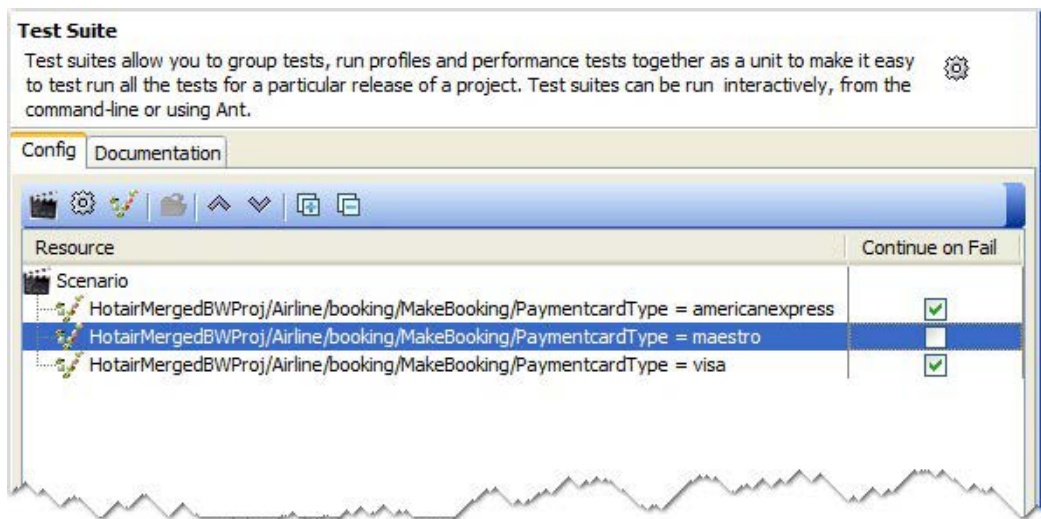
Under **Format**, select the desired format of the selected report segment (HTML or Link). The Link option is a hyperlink to the report on the Results Server, but is only available for the Suite Summary Report.

Under **Attach**, select how the selected report segment should be included in the message (inline for HTML and Link option or as an attachment for HTML option). If you select the **Separate Attachment** option (for HTML), you can specify the filename that will be assigned to the report segment (the .html extension will be added automatically when the segment is generated).

6.4.6 Continue on Fail Option

By default, all items in a test suite are executed when the suite is executed, regardless of whether previous items pass or fail. Depending on the nature of the tests it contains, however, you may want the execution of a test suite to be terminated if one or more specific test items fail.

This behavior is controlled with the **Continue on Fail** option, which can be set independently for each item in a suite.



For each item in the test suite, enable (select) or disable (clear) the **Continue on Fail** option according to how the suite should be executed.

- When enabled, the test suite will continue its execution beyond the selected item, even if the selected item fails.
- When disabled, the test suite will fail and its execution will be stopped if the selected item fails.

6.4.7 Executing Test Suites

Test suite execution can be initiated in the Test Factory, but the suite will actually run in the Test Lab perspective. After starting a suite (as described below), the Test Lab perspective will be displayed to let you monitor the suite's execution. See [Test Lab](#) for more information.

NOTE: Depending on the current “Run Resources” preferences, the suite will be executed from its current state (including any unsaved changes) or from its last saved state. See [Changing Preferences](#) for more information.

Executing Test Suites in the Current Environment

Test suites can be executed in the currently selected environment in any of the following ways:

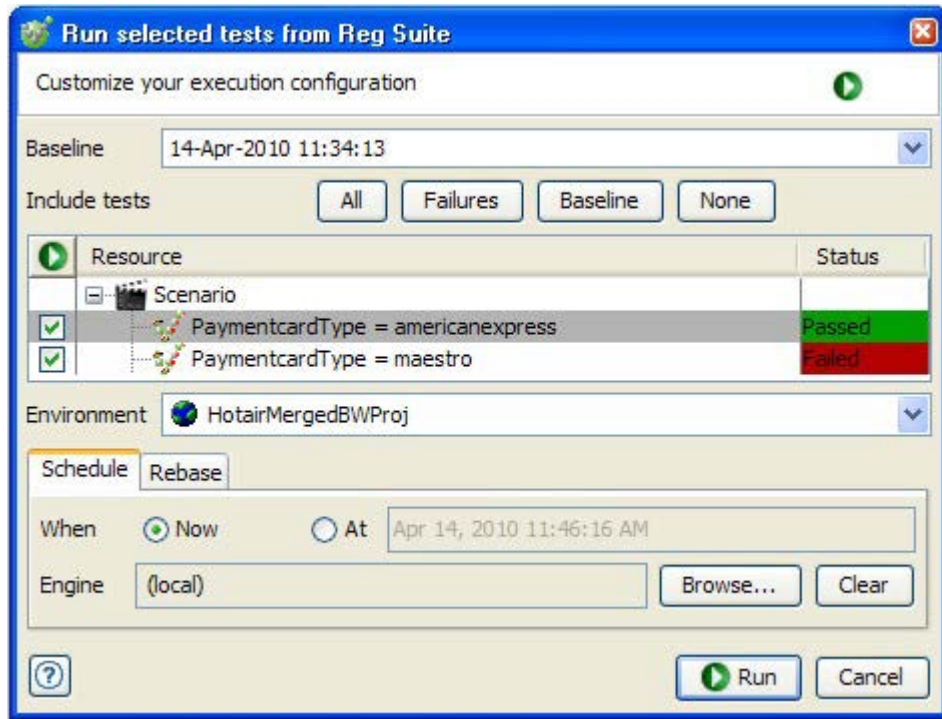
- Select one or more test suites and press **F5**.
- Right-click one or more test suites and select **Run** from the context menu.
- Select one or more test suites and click the **Run** icon in the main Rational Integration Tester toolbar.
- Click the small arrow next to the **Run** icon in the main Rational Integration Tester toolbar and select the test suite you want to run (if the test suite had been run recently).

Customizing the Execution of a Test Suite

You can customize the execution of a selected test suite (for example, run a subset of included tests or run in a different environment) by using the custom execution dialog. The results of the suite will be stored as a new instance and can be viewed as such in the Results Gallery.

1. Open the custom execution dialog in one of the following ways:
 - Select a test suite and press **Shift + F5**.
 - Right-click a test suite and select **Run...** from the context menu.
 - Select a test suite, click the small arrow next to the **Run** icon in the main Rational Integration Tester toolbar, then select the **Run...** option.

The custom execution dialog is displayed with the previously executed suite instance selected under **Baseline**. All of the tests that were previously executed will be selected for execution.



2. Select a previously executed suite instance from the **Baseline** dropdown. The contents of the selected instance and individual test results will be displayed.
3. Select the tests to execute in the new instance by ticking/unticking the checkboxes next to each test. Using the buttons next to **Include tests**, you can select all tests in the selected instance (**All**), only tests that failed in the selected instance (**Failures**), all tests that were part of the original selected instance (**Baseline**), or quickly remove all tests from the new instance (**None**).
4. Under the **Schedule** tab, select the environment in which the test suite should be executed from the **Environment** dropdown.

See [Create a New Suite from a Previous Run](#) for more information about creating a new suite under the **Rebase** tab.

5. To execute the suite remotely, click **Browse** next to **Engine** and select the agent that should execute the suite – only one suite per engine.
6. Click **Run** to execute the test suite.

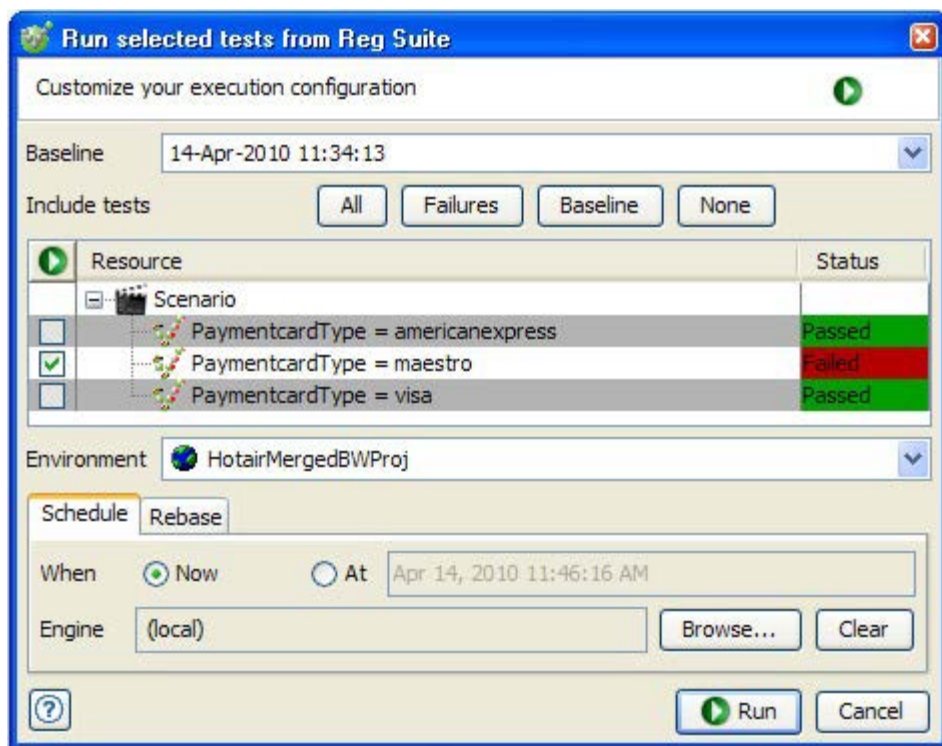
NOTE: No other custom execution options are currently supported.

Execute Failed Tests in a Test Suite

You can quickly run a new instance of a test suite that includes only the tests that failed in a previous run by using the custom execution dialog. The results of the suite will be stored as a new instance and can be viewed as such in the Results Gallery.

1. Open the custom execution dialog in one of the following ways:
 - Select a test suite and press **Ctrl + F5**.
 - Right-click a test suite and select **Re-run Failures...** from the context menu.
 - Select a test suite, click the small arrow next to the **Run** icon in the main Rational Integration Tester toolbar, then select the **Re-run Failures...** option.

The custom execution dialog is displayed with the previously executed suite instance selected under **Baseline**. Only failed tests will be selected for execution if you selected the **Re-run Failures...** option.



See [Create a New Suite from a Previous Run](#) for more information about creating a new suite under the **Rebase** tab.

2. Click **Run** to execute the test suite.

NOTE: No other custom execution options are currently supported.

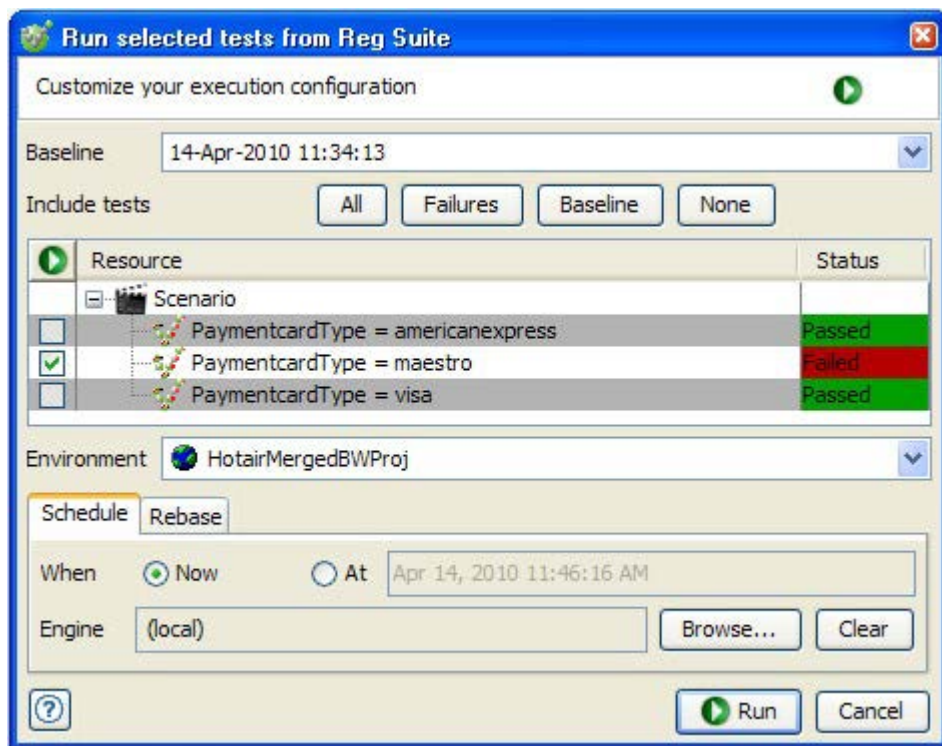
Create a New Suite from a Previous Run

You can create a new test suite (and optional tests) from an existing suite, including some or all of the tests that the original suite contained (for example, only failed tests).

Creating a new test suite can be done from the custom execution dialog.

1. Open the custom execution dialog in one of the following ways:
 - Select the desired test suite and press **Ctrl + F5**.
 - Right-click the desired test suite and select **Run...** or **Re-run Failures...** from the context menu.
 - Select the desired test suite, click the small arrow next to the **Run** icon in the main Rational Integration Tester toolbar, then select the **Run...** or **Re-run Failures...** option.

The custom execution dialog is displayed with the previously executed suite instance selected under **Baseline**. Only failed tests will be selected for execution if you selected the **Re-run Failures...** option.



2. Select the tests to include in the new suite by ticking/unticking the checkboxes next to each test, or use the buttons next to **Include tests**.

-
- Click the **Rebase** tab at the bottom of the dialog.

The screenshot shows a dialog box with a 'Rebase' tab. It includes a checked checkbox for 'Generate a new suite (with tests)'. The 'Parent' field is set to 'MakeBooking' with 'Browse...' and 'Clear' buttons. The 'Folder' field shows '14-Apr-2010 12:51:19' and the 'Suite' field shows 'Reg Suite'. An unchecked checkbox at the bottom is labeled 'At the end overwrite the expected messages with those received'.

- Tick the **Generate a new suite (with tests)** option.
- To change the location where the new suite will be created, click **Browse** next to the **Parent** field and select the desired component, folder, or operation.
- Enter or modify the name of the folder in which the new suite will be created in the **Folder** field (the current date and time is used by default).
- Enter or modify the name of the new suite in the **Suite** field.
- If desired, tick the box at the bottom of the dialog to overwrite expected messages in the executed tests with the messages that are received.
- When finished, click **Run**.

The new suite will be created and executed using the name and location specified.

6.5 Stubs

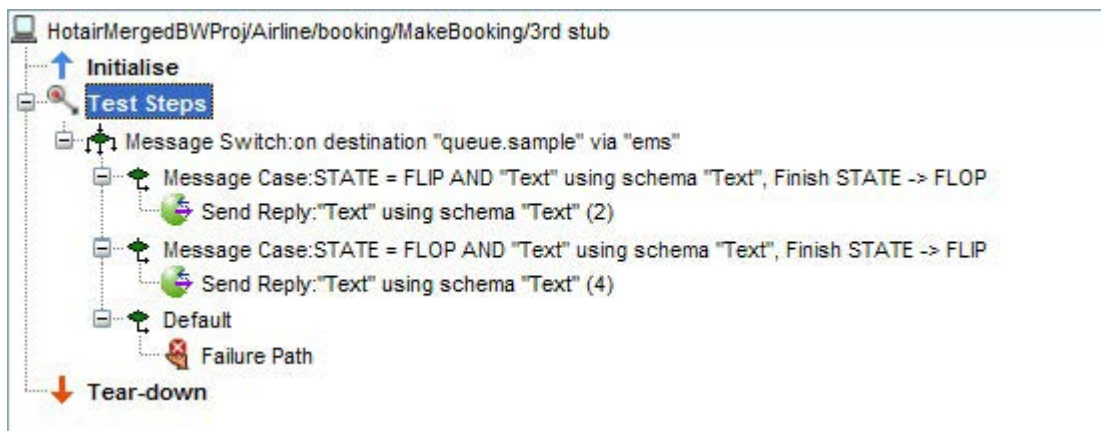
Stubs can be used to simulate services that you are unable to utilize or may otherwise be unavailable. For example, if you testing a TIBCO BusinessWorks process but you don't always have access to TIBCO Designer, you can record the process events and create a stub from them. You can then run the stub in place of Designer.

A stub is essentially a test that listens for incoming messages by means of the [Message Switch](#) action (the Message Switch is like a subscriber). Replies can be issued to specific messages or message types, matched according to incoming filters, using Message Case actions (see [Add a Message Case](#)).

A stub can be simple (for example, simulate a single service that utilizes a single message case and the default case) or complex (for example, simulate a Web Service or some other point-to-point operation that includes multiple message types or operations). In either case, the basic principles are the same (that is, send a specific reply when a specific message type is received).

At the most basic level, a stub will receive a message and (optionally) validate its contents. Based on the validation results or simply based on 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).

A more complex stub, however, can receive incoming messages and, based on the specific contents of those messages, carry out a more extensive set of test actions (for example, 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 test actions, a stub can be configured to generate responses in an intelligent manner.



NOTE: Since a stub is essentially a test, see [Test Phases](#) and [Test Views](#) for more information.

6.5.1 Creating Stubs

In the Test Factory perspective, stubs can be created in two ways: [Create an Empty Stub](#) or [Create a Stub using MEP](#). Additionally, you can create stubs from recorded events in the Recording Studio perspective (see [Creating Test Resources from Recorded Events](#) on page 180, [Create a Parameterized Stub](#), [Create a Non-deterministic Stub](#), and [Create a Deterministic Stub](#)).

Create an Empty Stub

Follow the steps below to create a new empty stub.

1. Select the operation in which you want to create the stub.
2. Click the Stub icon in the Test Factory toolbar, or right-click the operation and select **New > Stubs > Stub** from the context 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**.

3. Provide a name for the new stub when prompted, then click **OK**.

The new stub is opened for editing. See [Configuring the Contents of a Stub](#) for more information about working with the new stub.

Create a Stub using MEP

An operation's Stub properties (see [Change an Operation's Stub Properties](#)) define the binding and timeout details that should be used when creating a stub from the operation. You can use this information to quickly create a stub for an operation.

Follow the steps below to create a stub using the selected operation's MEP/Stub properties.

1. Select the operation in which you want to create the stub.
2. Right-click the operation and select **New > Stubs > Stub using MEP** from the context 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**.

3. Provide a name for the new stub when prompted, then click **OK**.

The new stub is opened for editing and includes a message switch that uses the transport defined under the operation's Stub tab and a message case with a reply. See [Configuring the Contents of a Stub](#) for more information about working with the new stub.

Create a Parameterized Stub

A parameterized stub is one that is driven by the contents of a test data source. The data source will be created based on the data fields that have been promoted to the events table (see [Adding/Removing Columns in the Event Table](#)).

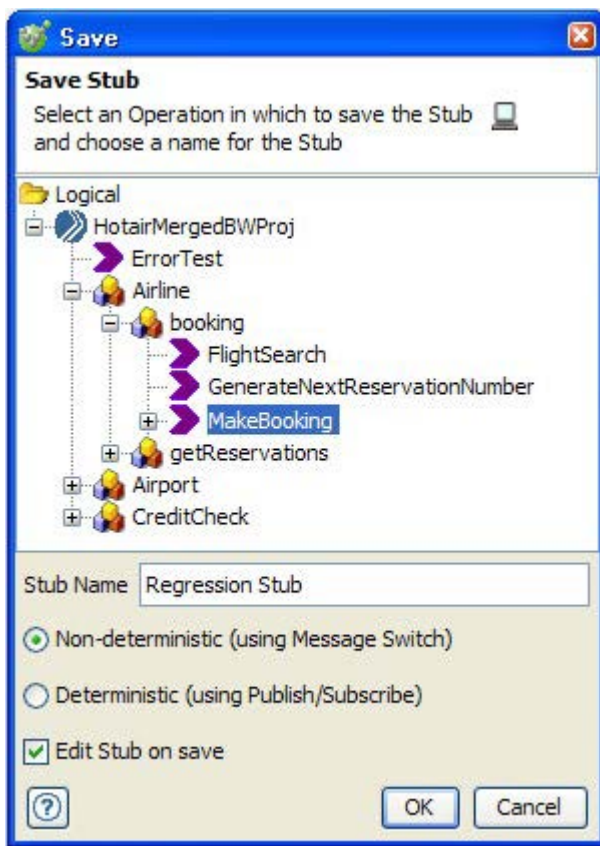
1. In Recording Studio, select one or more recorded events in the Events View.
2. Promote at least one field from the message body or header (below the Events View).
3. Select **Save Parameterized Stub** from the Events View toolbar, or right-click the messages and select **Save Parameterized Stub** from the context menu.
4. Provide a name for the stub when prompted.
5. Provide a name for the comma separated file to be created and used by the new test data source.
6. Select the promoted field to use for grouping the data and click **OK**.

The stub will be created with a Lookup Test Data action that uses the new test data source and lookup value that was selected (the grouping field).

Create a Non-deterministic Stub

A non-deterministic stub uses a message switch, where each pair of messages is used to create a Send Response or Publish case within the switch. This allows realistic system behaviour to be modelled.

1. In Recording Studio, select multiple recorded events in the Events View.
2. Click the stub icon in the Events View toolbar, or right-click the messages and select **Save Stub** from the context menu.
3. When prompted, select the operation in which the stub should be saved, provide a name for the stub, and select the **Non-deterministic** option below the stub name.

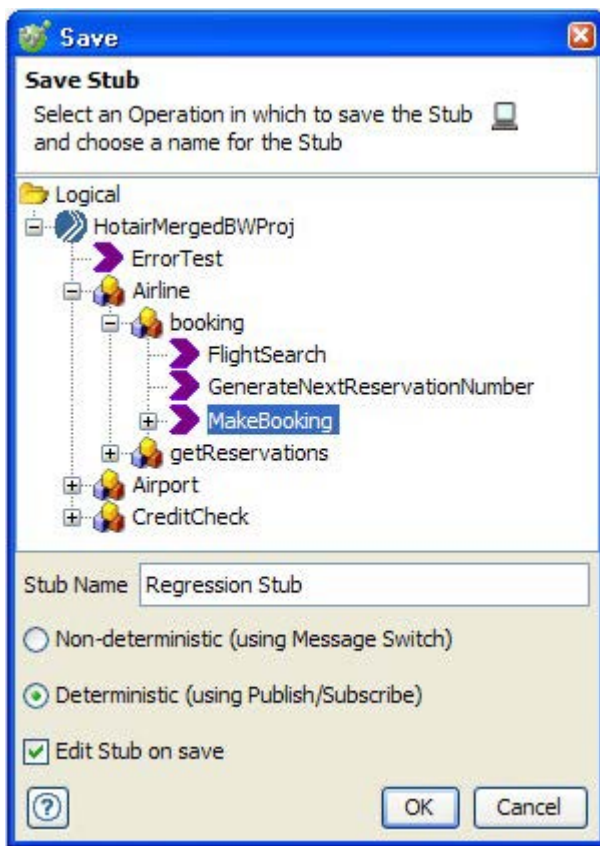


The stub will be created using a Message Switch action, based on the selected messages.

Create a Deterministic Stub

A deterministic stub uses a series of Receive Request/Send Response or Subscribe/Publish actions, and is better suited when users are trying to assert that the system under test produces requests in a certain order.

1. In Recording Studio, select multiple recorded events in the Events View.
2. Click the stub icon in the Events View toolbar, or right-click the messages and select **Save Stub** from the context menu.
3. When prompted, select the operation in which the stub should be saved, provide a name for the stub, and select the **Deterministic** option below the stub name.



The stub will be created using a series of Receive Request/Send Response or Subscribe/Publish actions, based on the selected messages.


6.5.2 Configuring the Contents of a Stub

Stubs are essentially tests, but they function in a specific way to most effectively simulate unavailable services. For this reason, refer to the following sections (in [Tests](#)) to learn more about the basic use and manipulation of stubs and their contents:


- [Manipulating the Contents of a Test](#)
- [The Tag Data Store](#)
- [Add a Message Case](#)
- [Setting Pass Paths](#)
- [Setting Failure Paths](#)

6.5.3 Configuring Stub Execution Properties

Under the **Properties** tab in the editing panel, you can define the execution properties of the current stub.

Stub
Use a Stub to provide replacement behaviour for existing or unavailable services. Stubs can be use  directly or from within Test Suites.


Actions Properties Documentation

Behaviour style: One-to-one, no looping  Instance created for every new request, message or connection

Workers 10


Session

States FLIP;FLOP

Initial State FLIP 

Keys NAME

Response Times

Delay Distribution Uniform 

Minimum Delay (ms) 200




Maximum Delay (ms) 800

Interface

Input

☐ None ☐ All Tags ☒ Select Tags




Name	Description
Response	
Input2	
Input3	
Input1	



Output

☐ None ☒ All Tags ☐ Select Tags

Name	Description
Response	
Input2	
Input3	
Input1	



The stub execution properties are described below:

- **Behaviour style** controls how the stub will execute, as follows:
 - **One-to-one, no looping:** An instance is created for every new request, message, or connection.
 - **One-to-one, looping:** An instance is created for each new connection, and the instance lives until the connection closes. This behaviour is only applicable to connection-based transports.
 - **Many-to-one:** An instance is created if no instance is currently available to process the request or messages. Worker threads (see below) are not enabled for this behaviour.
- **Workers** defines the maximum number of concurrent instances of this stub that can be running at any one time (the default is 10).
- **Session** defines the session states that should be available for the stub, the initial state for the stub, and session keys that can be used to identify a particular session.
 - One or more states (separated by a semicolon) can be entered in the **States** field. When a message case is used within the stub, the state of a stub can be checked before any actions in the case are executed, and a new state can be optionally set once the actions in the case complete. The session state is also stored in a system tag named SESSION/STATE. If desired, the session state could be set using the **setTag** function.
 - The initial state of the session can be selected under **Initial State**.
 - One or more session keys (separated by a semicolon) can be defined under **Keys**. These keys are automatically mapped to a system tag named SESSION/KEY/<Key name>. When an incoming message is received, key values should be stored in these tags, which are then used to identify the session to use.
- **Response Times** control the performance of a stub, allowing users to create realistic system conditions in a single stub (that is, without having to create multiple stub instances). A delay pattern can be specified for the stub (**Fixed**, **Uniform**, or **Gaussian**), and the minimum and maximum delay times (in milliseconds) can be set for the specified delay type.
- **Interface** defines the Input and Output tags that should be available to the stub at runtime when configured within the scenario of a test suite. See [Test Properties](#) for more information about managing input and output tags.

6.5.4 Executing Stubs

Stub execution can be initiated in the Test Factory, but the test will actually run in the Test Lab perspective. After starting a stub (as described below), the Test Lab perspective will be displayed to let you monitor the test's execution. See [Test Lab](#) for more information.

NOTE: Depending on the current “Run Resources” preferences, the stub will be executed from its current state (including any unsaved changes) or from its last saved state. See [Changing Preferences](#) for more information.

Running Stubs in the Current Environment

Stubs can be executed in the currently selected environment in any of the following ways:

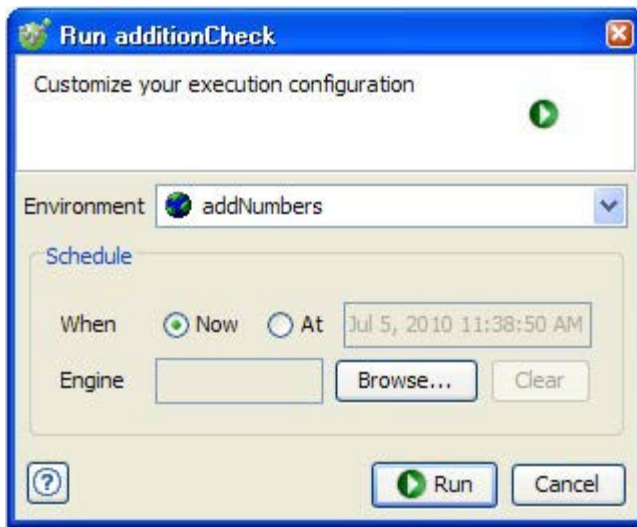
- Select one or more stubs and press **F5**.
- Right-click one or more selected stubs and select **Run** from the context menu.
- Select one or more stubs and click the **Run** icon in the main Rational Integration Tester toolbar.
- Click the small arrow next to the **Run** icon in the main Rational Integration Tester toolbar and select the stub you want to run (if the stub had been run recently).

Customizing the Execution of a Stub

A selected stub can be executed in an environment other than the current environment by using the custom execution dialog, as follows:

1. Open the custom execution dialog in one of the following ways:
 - Select a stub and press **Shift + F5**.
 - Right-click a stub and select **Run...** from the context menu.
 - Select a stub, click the small arrow next to the **Run** icon in the main Rational Integration Tester toolbar, then select the **Run...** option.

The custom execution dialog is displayed.



2. Select the environment in which the stub should be executed from the **Environment** dropdown.
3. To execute the stub remotely, click **Browse** next to **Engine** and select the agent that should execute the stub – only one stub per engine.
4. Click **Run** to execute the stub.

NOTE: No other custom execution options are currently supported.

6.6 Test Templates

To save time when creating tests, you can create a template of test sequences that can be used to quickly create a number of pre-defined tests. Like tests and other artifacts, test templates are created within operations. Therefore, you must configure at least one operation in your project before you can create a test template.

You can create a basic (empty) template, or you can create a template using the Message Exchange Pattern (MEP) that has been defined for the selected operation.

6.6.1 Create a Test Template

Follow the steps below to create a test template.

1. Select the operation or folder in which you want to create the template.
2. Right-click the operation or folder and select **New > Templates > Test Template** from the context menu, or click the arrow next to the Test icon in the Test Factory toolbar and select **Test Template**.

NOTE: If a template has already been created, you can right-click the Templates folder or one of the existing templates and select **New > Test Template**.

3. Provide a name for the new template when prompted, then click **OK**.
The new template is opened for editing.
4. Configure the contents of the template as desired (that is, add messaging actions or other test actions, configure messages, and so on).
5. When finished, click **File > Save Resource** or click the **Save** icon in the main Rational Integration Tester toolbar.

6.6.2 Create a Test Template from a Message Exchange Pattern

Follow the steps below to create a test template using the operation's MEP.

1. Select the operation or folder in which you want to create the template.
2. Select the **Test Template** option from the toolbar, or right-click the operation and select **New > Templates > Test Template using MEP**.

NOTE: If at least one template has already been created, you can right-click the Templates folder or one of the existing templates and select **New > Test Template using MEP**.

3. Provide a name for the new template when prompted, then click **OK**.

The new template is opened for editing.

NOTE: The template will contain test actions once it is created. The type and content of the test actions are determined by the Message Exchange Pattern settings of the selected operation. See [Change an Operation's Message Exchange Pattern](#) for more information.

4. Configure the contents of the template as desired (that is, add messaging actions or other test actions, configure messages, and so on).
5. When finished, click **File > Save Resource** or click the **Save** icon in the main Rational Integration Tester toolbar.

6.7 Test Data Sources

Rational Integration Tester lets you configure test data sources that can drive a test template or a test using several different instances of data. In this way you can parameterize your tests, substituting specific data from a pre-configured source in different test iterations. You can also use test data to validate the results that are returned by different test actions.

For example, you can perform a function (for example, add) on two columns of data from a database or Microsoft Excel spreadsheet, then validate the function results against a third column of data in the data source.

Rational Integration Tester supports four different test data types (database, directory, file, and Excel).

6.7.1 Creating a Test Data Source

To create a new data source, follow the steps below:

1. Select the operation in which you want to create the data source.
2. Select the data source type from the toolbar, or right-click the operation and select **New > Test Data > [data source type]** from the context menu.

NOTE: If at least one data source has already been created, you can right-click the Test Data folder or one of the existing data sources and select **New > [data source type]**.

3. Create a name for the new data source when prompted, then click **OK**.

The new data source is opened in the editing panel to the right.

4. Configure the data source as desired (see the following sections for details about each type).
5. When finished configuring the data source, click **File > Save Resource** or click the **Save** icon in the main Rational Integration Tester toolbar.

See the following sections for details about configuring each data source type:

- [Configuring a Database Data Source](#)
- [Configuring a Directory Data Source](#)
- [Configuring a File Data Source](#)
- [Configuring an Excel Data Source](#)

6.7.2 Configuring a Database Data Source

A database source can be used to supply “live” data to Rational Integration Tester. Follow the steps below to configure a database data source.

NOTE: A database source relies on a valid database connection in Architecture School. See [Databases](#) for more information about setting up a working database connection.

Database Data Source
Configure this test data set to retrieve data from an external database

Database Server: GH Tester [Browse...] [Clear]

☐ Table/View: GH.EMPLOYEE

☒ Query: select * from employee where empno = 98

☐ Loop Data

Preview (max 25 rows)

EMPNO	LASTNAME	FIRSTNAME	ROLE	SUPERVIS...	SALARY
98	Downing	Julia	Tester	388	48000

[Refresh] [Copy column names to clipboard]

1. Click **Browse** to select the desired database server from the project resource dialog.
2. Select the **Table/View** option and select a database table/view from dropdown list, or select **Query** and enter a SQL select statement to pull only specific data.

NOTE: The query field supports the use of tags for all or part of the SQL statement. Tags might typically be used to vary your SQL query according to the selected environment (that is, using an environment tag).

3. If the number of test iterations may exceed the number of matching records, enable the **Loop Data** option to force Rational Integration Tester to process the same records over and over.
4. Click **Refresh** to generate a preview of the data that will be returned.

-
5. Save the data source when finished, then see [Creating Tags from Data Source Fields](#) for details about copying column names to create tags from them.

6.7.3 Configuring a Directory Data Source

A directory data source can be used to populate tag values from the contents of complete files or their properties.

The screenshot shows the 'Directory Data Source' configuration window. It includes a title bar, a description, a directory name field with a 'Browse...' button, radio buttons for 'File System Directory' (selected), 'Project Folder', and 'Recurse Subdirectories'. There are text boxes for 'Include' (containing '*data*') and 'Exclude' (containing '*.xls'). A 'Sorting' section has radio buttons for 'No Sorting', 'By File Name' (selected), and 'By Date Last Modified', with sub-options for 'Ascending' (selected) and 'Descending'. A 'Loop Data' checkbox is present. At the bottom, a 'Preview (max 25 rows)' table shows file details, and there are 'Refresh' and 'Copy column names to clipboard' buttons.

Directory Data Source
Configure this test data set to read data from resources residing under a specified file system directory or project folder

Directory Name:

☒ File System Directory ☐ Project Folder ☐ Recurse Subdirectories

Wildcard Patterns

Include:

Exclude:

Sorting

☐ No Sorting ☒ By File Name ☐ By Date Last Modified

☒ Ascending ☐ Descending

☐ Loop Data

Preview (max 25 rows)

Entire File	File Name	Absolute Path	Relative Path	URI Path
It is possible ...	mydata.txt	C:\Docs\GreenHat...	mydata.txt	/C:/Docs/GreenHat/Suppor...
It is possible ...	mydata1.txt	C:\Docs\GreenHat...	mydata1.txt	/C:/Docs/GreenHat/Suppor...
It is possible ...	mydata2.txt	C:\Docs\GreenHat...	mydata2.txt	/C:/Docs/GreenHat/Suppor...

1. Click **Browse** to locate and select the directory to use as the data source.
2. Select whether the source is a file system directory or a project folder, and whether or not the contents of subdirectories should be used.
3. Configure wildcard patterns to use or exclude files matching the patterns.
4. Select the sorting options to control the order in which files are processed.
5. If the number of test iterations may exceed the number of matching files, enable the **Loop Data** option to force Rational Integration Tester to process the same files over and over.
6. Click **Refresh** to generate a preview of the files and properties that will be returned.

-
7. Save the data source when finished, then see [Creating Tags from Data Source Fields](#) for details about copying column names to create tags from them.

6.7.4 Configuring a File Data Source

A file data source can be used to populate tag values from the contents of complete files. Two file types – delimited and fixed width – are available when using a file data source. For details about configuring each file type, see the following sections:

- [Delimited Files](#)
- [Fixed Width Files](#)

Delimited Files

Delimited files contain “columns” of data separated by a specific character (for example, a comma or a tab), and the configuration of a file source is similar to an Excel source.

The screenshot shows the 'File Data Source' configuration window. At the top, it says 'Configure this simple data source to use data from flat files as tag values in tests.' Below this, the 'File Name' field contains the path '%%PROJECT/ROOT_DIRECTORY%%\TestSubs\tab.txt' with a 'Browse...' button to its right. There are two radio buttons: 'Delimited' (selected) and 'Fixed Width'. A 'Format Configuration' section contains a checked checkbox 'A row of the file contains column names', two input fields for 'Rows to skip before column names' and 'Rows to skip after column names' (both set to 0), and a 'Delimiter Options' section. This section has radio buttons for 'Comma', 'Tab' (selected), 'Space', 'Semi-colon', and 'Other' (with an empty text field). It also has a checked checkbox 'Ignore delimiters within quoted strings', a 'Column count' input field, and a 'Calculate' button. Below this are two more checkboxes: 'Treat empty strings as null' (unchecked) and 'Treat text as null' (checked) with a text field containing 'b'. A 'Loop Data' checkbox is checked. At the bottom, there is a 'Preview (max 25 rows)' section showing a table with two columns, 'Column1' and 'Column2', and three rows of data. Below the preview are 'Refresh' and 'Copy column names to clipboard' buttons.

Column1	Column2
1	a
2	
3	c

-
1. Click **Browse** to locate and select the file to use as the data source.
 2. Select the **Delimited** option under the file location.
 3. Use the **Format Configuration** to define the contents of the file (that is, if it contains a header row, and if there are rows that should be skipped before the header row).
 4. Under **Delimiter Options**, set the type of delimiter that is used in the file and indicate whether the delimiter should be ignored within quoted text (that is, Rational Integration Tester will treat delimiter characters enclosed in a quoted string as the character and not a delimiter).
 5. Enable the **Other** option to specify a different character or string of characters as the delimiter.

NOTE: If multiple characters are used as the delimiter, the “Ignore” option is disabled.

6. If desired, you can test the delimiter settings by clicking **Calculate** next to **Column count** to see how many columns are recognized in the file.
7. Enable the **Treat empty strings as null** option if you want to treat empty fields in the data source as null.
8. Enable the **Treat text as null** option if you want to specify a string field value that should be replaced with null. For example, the file may contain “NULL” for fields with no value instead of using empty fields. In this case you would enter “NULL” in the text field, instructing Rational Integration Tester to return a null value for such fields.

NOTE: Treating empty fields or other field contents as null can be useful when utilising repeating elements in a data source, or when filtering test data.

When previewing data, null values (empty fields or text specified as such) will be highlighted if enabled, as shown in the image preceding these steps (the highlight color can be specified in Rational Integration Tester’s preferences).

9. If the number of test iterations may exceed the number of rows in the file, enable the **Loop Data** option to force Rational Integration Tester to process the same rows over and over.
10. Click **Refresh** to generate a preview of the data that will be returned.

-
11. Save the data source when finished, then see [Creating Tags from Data Source Fields](#) for details about copying column names to create tags from them.

Fixed Width Files

Fixed width files contain columns of data that are in fixed positions within the file (for example, “Emp/Last Name” uses the first 15 characters, “Emp/First Name” uses the next 15 characters, and so on). When using fixed width files, you will need to define the contents of the file so that Rational Integration Tester knows how to extract the data from each line.

File Data Source
Configure this simple data source to use data from flat files as tag values in tests.

File Name

☐ Delimited ☒ Fixed Width

Format Configuration

☒ Offset/Width ☐ Start Char/End Char ☒ Trim cell padding

Name	Offset	Width
Emp/Last Name	0	15
Emp/First Name	15	15
Emp/Position	30	15
Emp/Salary	45	10

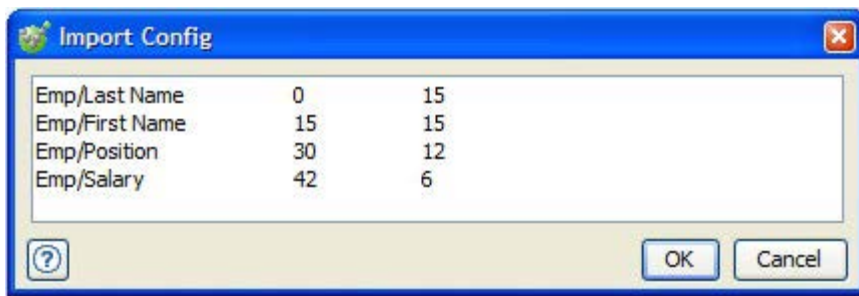
☐ Loop Data

Preview (max 25 rows)

Emp/Last Name	Emp/First Name	Emp/Position	Emp/Salary
Thompson	Aaron	Developer	72000
Miller	James	Developer	68000
Bryant	Samuel	Tester	54000
Jones	Richard	Tester	56500

1. Click **Browse** to select the data source file and select the **Fixed Width** option.

-
2. Indicate how you will define the data field locations in the source file:
 - **Offset/Width:** “offset” equals the number of characters that precede the field and “width” equals the number of characters used by the field.
 - **Start Char/End Char:** “start char” is the position directly before the field starts and “end char” is the last position of the field. “Start char” for the first field is always zero, which is one less than the first position in the file (that is, one). “End char” for all fields equals the length of the field plus its starting value, and this value is carried to “Start char” for the next field.
 3. Click **Import** to set the rules for determining the data locations. The **Import Config** dialog is displayed.



4. Enter the rules or paste rules that have been previously copied to the clipboard. Rules must be entered in the following format: Field Name<tab>value<tab>value

As an example, take the following file content:

Thompson	Aaron	Developer	72000
Miller	James	Developer	68000
Bryant	Samuel	Tester	54000
Jones	Richard	Tester	56500

In this file, “Emp/Last Name” uses the first 15 characters, “Emp/First Name” uses the next 15 characters, “Emp/Position” uses the next 12 characters, and “Emp/Salary” uses the last six characters. The fields are “padded” with spaces.

The corresponding rules for this file are shown below:

Offset/Width Rules

Emp/Last Name	0	15
Emp/First Name	15	30
Emp/Position	30	42
Emp/Salary	42	48

Start Char/End Char Rules

Emp/Last Name	0	15
Emp/First Name	15	30
Emp/Position	30	42
Emp/Salary	42	48

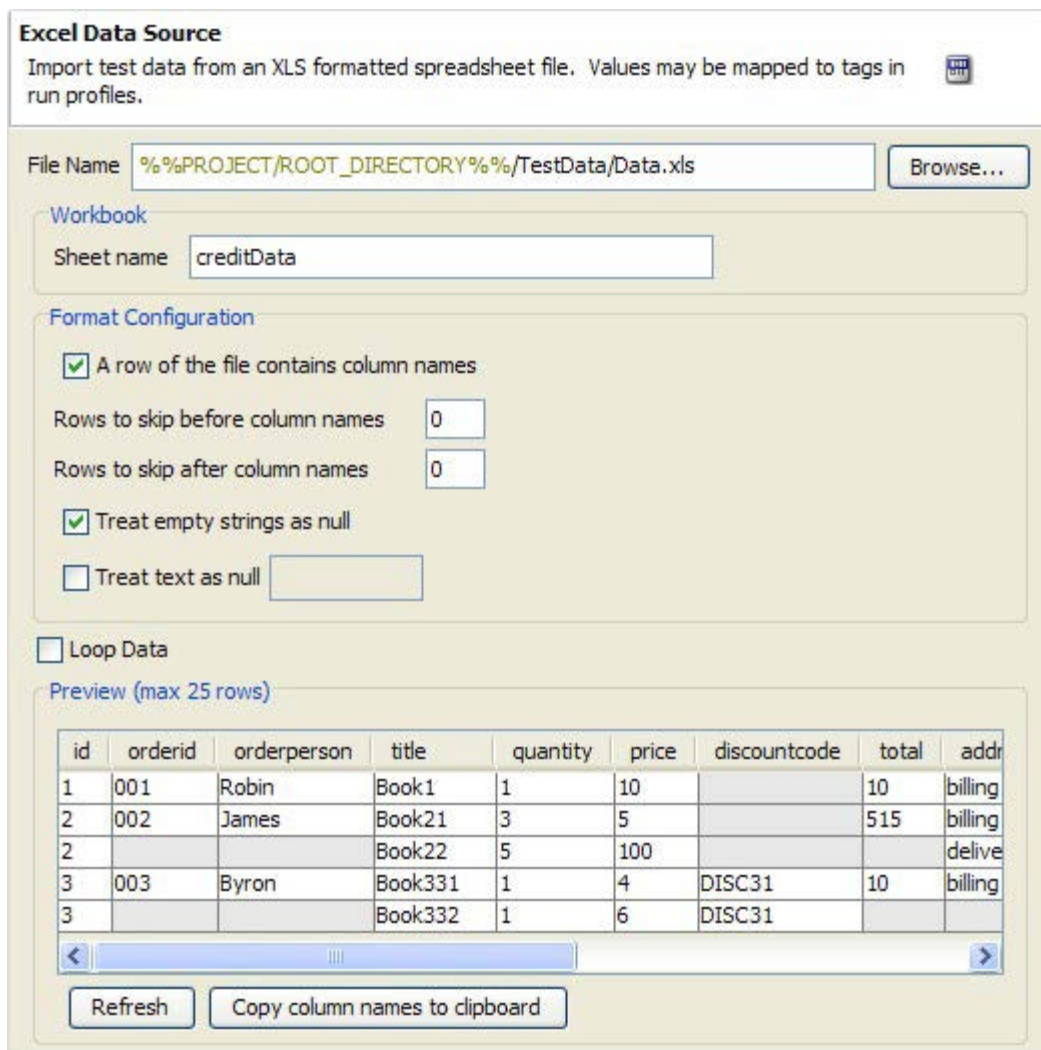
NOTE: Start Char/End Char rules are converted to Offset/Width figures when displayed in the data source configuration dialog.

5. If the fields in your files are padded with extra characters so that they fill their space in the line, select the Trim cell padding option and specify the padding character – this character will be removed when data is extracted.
6. If the number of test iterations may exceed the number of rows in the file, enable the **Loop Data** option to force Rational Integration Tester to process the same rows over and over.
7. Click **Refresh** to generate a preview of the data that will be returned.
8. Save the data source when finished, then see [Creating Tags from Data Source Fields](#) for details about copying column names to create tags from them.

6.7.5 Configuring an Excel Data Source

An Excel data source can be used to populate tag values from a Microsoft Excel (.xls) file.

NOTE: At this time, not all formulas (used in spreadsheet cells) supported by Microsoft Excel are supported by Rational Integration Tester. If your spreadsheet uses formulas, you should verify how data is being parsed by Rational Integration Tester.



The dialog box is titled "Excel Data Source" and contains the following sections:

- Import test data from an XLS formatted spreadsheet file. Values may be mapped to tags in run profiles.** (with an icon)
- File Name:** A text field containing "%PROJECT/ROOT_DIRECTORY%/TestData/Data.xls" and a "Browse..." button.
- Workbook:** A section with a "Sheet name" text field containing "creditData".
- Format Configuration:** A section with several options:
 - ☒ A row of the file contains column names
 - Rows to skip before column names: 0
 - Rows to skip after column names: 0
 - ☒ Treat empty strings as null
 - ☐ Treat text as null (with a text field)
- ☐ Loop Data
- Preview (max 25 rows):** A table showing data from the spreadsheet.
- Buttons: "Refresh" and "Copy column names to clipboard".

id	orderid	orderperson	title	quantity	price	discountcode	total	addr
1	001	Robin	Book1	1	10		10	billing
2	002	James	Book21	3	5		515	billing
2			Book22	5	100			delive
3	003	Byron	Book331	1	4	DISC31	10	billing
3			Book332	1	6	DISC31		

1. Click **Browse** to locate and select the Excel file to use as the data source.
2. If the file contains multiple worksheets, enter the name of a valid worksheet to use in the **Sheet name** field. If this field is left blank, the first worksheet in the file is used.

-
3. Use the **Format Configuration** to define the contents of the file (that is, if the worksheet contains a header row, and if there are rows to be skipped before and/or after the header row).
 4. Enable the **Treat empty strings as null** option if you want to treat empty fields in the data source as null.
 5. Enable the **Treat text as null** option if you want to specify a string field value that should be replaced with null. For example, the file may contain “NULL” for fields with no value instead of using empty fields. In this case you would enter “NULL” in the text field, instructing Rational Integration Tester to return a null value for such fields.


NOTE: Treating empty fields or other field contents as null can be useful when utilising repeating elements in a data source, or when filtering test data.

When previewing data, null values (empty fields or text specified as such) will be highlighted if enabled, as shown in the image preceding these steps (the highlight color can be specified in Rational Integration Tester’s preferences).

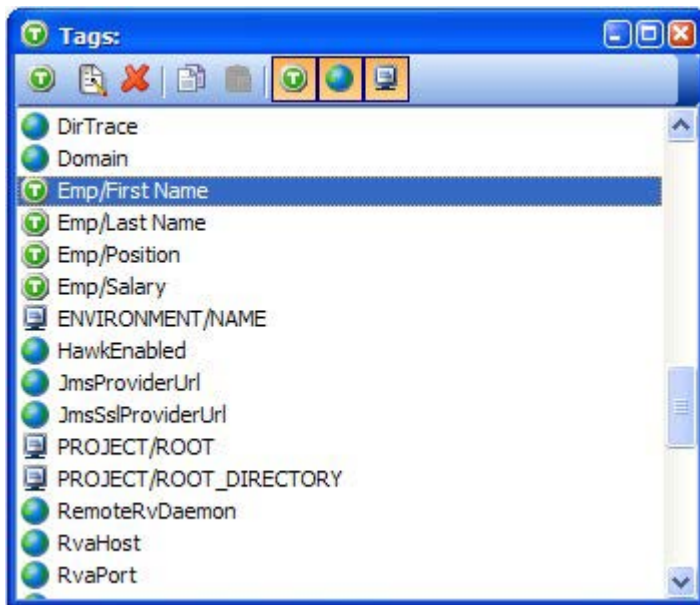
6. If the number of test iterations may exceed the number of data rows, enable the **Loop Data** option to force Rational Integration Tester to process the same rows over and over.
7. Click **Refresh** to generate a preview of the data that will be returned.
8. Save the data source when finished, then see [Creating Tags from Data Source Fields](#) for details about copying column names to create tags from them.

6.7.6 Creating Tags from Data Source Fields

Before you can use the data from any configured source, a tag must be defined for each value. The steps below provide a quick method to create these tags.

1. Configure the specific data source as needed and generate a preview of the source data (click **Refresh**).
2. When the data is displayed properly and the column headings appear to be correct, click the **Copy column names to clipboard** button.
3. Open (or create) the test in which you will use the data source.
4. In the test editor, open the Tag Data Store (right-click a test step or test phase and select **Tag Data Store** from the context menu).
5. Click the **Paste tag** icon  in the Tag Data Store toolbar.

The column headings from the clipboard will be pasted as new tags in the Tag Data Store.

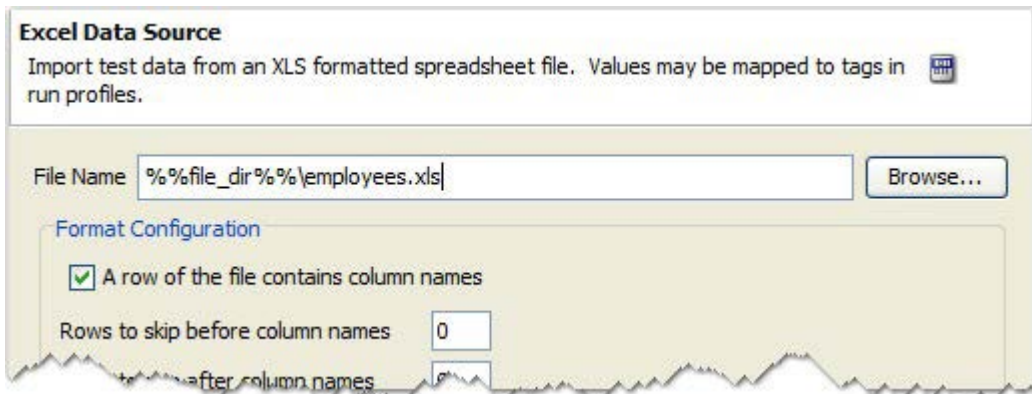


NOTE: By naming the fields “Emp/<field>,” the tag names are automatically created with the “Emp” prefix or folder name in the data store.

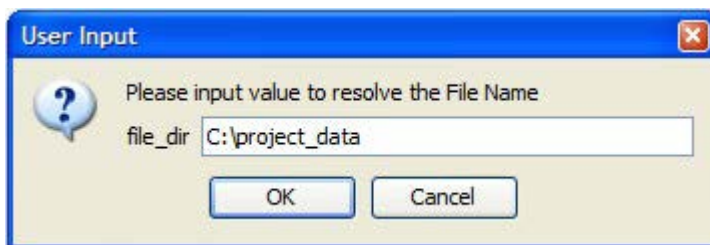
6.7.7 Using Tags to Define Test Data

Tags can be used in test data set definitions to provide dynamic data access. These tags can be used within file locations, file names, and database queries. You can select environment or system tags from the context menu.

You can also reference test tags, but these have to be entered manually since the test data set has no knowledge of the tags when it is being configured. When tagging data source locations, bear in mind that the test data set may be used with many tests.



When you click **Refresh** to preview the test data, you will be prompted for a tag value.



NOTE: The test tag must be defined within the test that is accessing the data source, and the tag must be given a default value.

Test Lab

Contents

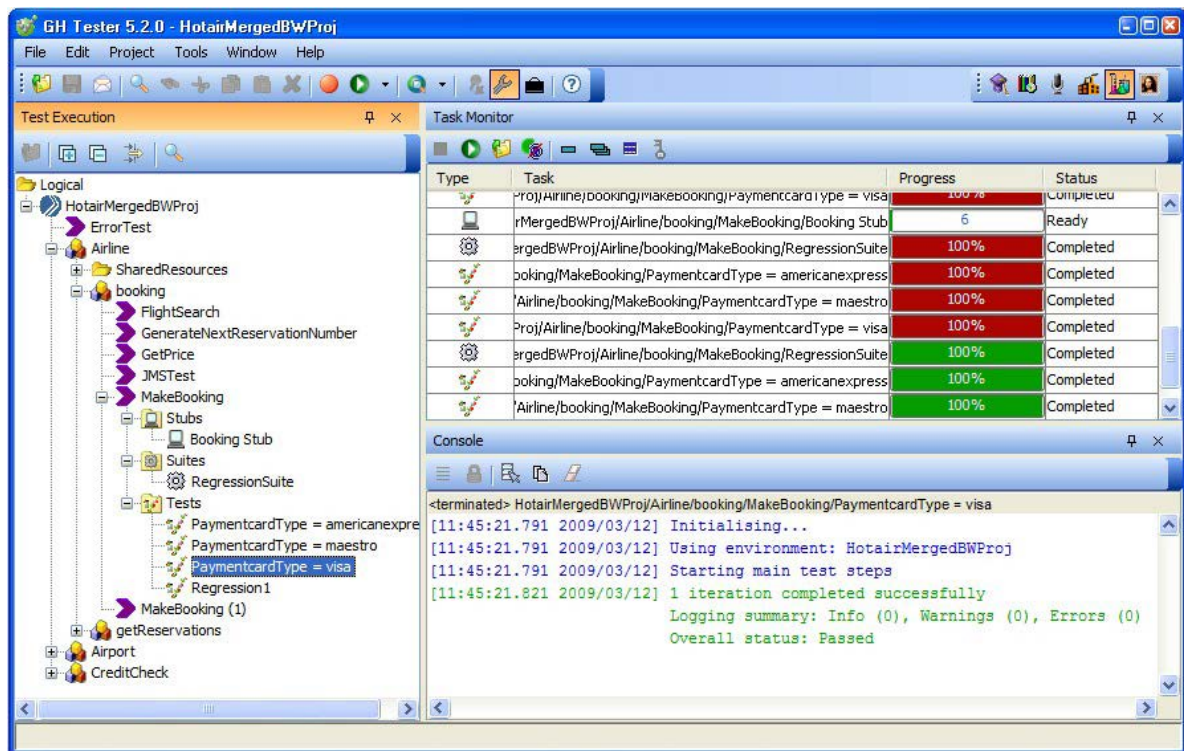
Overview

Working in the Test Lab

This chapter provides an overview of Rational Integration Tester's Test Lab perspective, including details about how to run tests, test suites, and stubs. Information is also included about using the test repair wizard.

7.1 Overview

The Test Lab, another main perspective in Rational Integration Tester, is where you can execute all of the test, test suites, and stubs that have been created in the Test Factory.




The Test Lab contains three main views: the Test Execution tree, the Task Monitor, and the Console. Each of these views are described in the following sections:

- [The Test Execution Tree](#)
- [The Task Monitor](#)
- [The Console](#)

7.1.1 The Test Execution Tree

The Test Execution tree displays all of the test items in your Rational Integration Tester project (that is, all of the same items that can be seen in the Test Factory).

NOTE: The only items that can be executed in the Test Lab are tests, performance tests, test suites, and stubs.

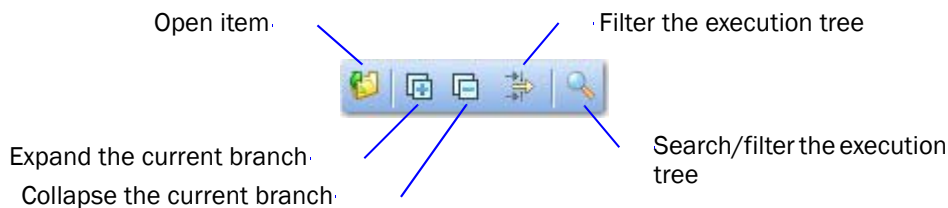
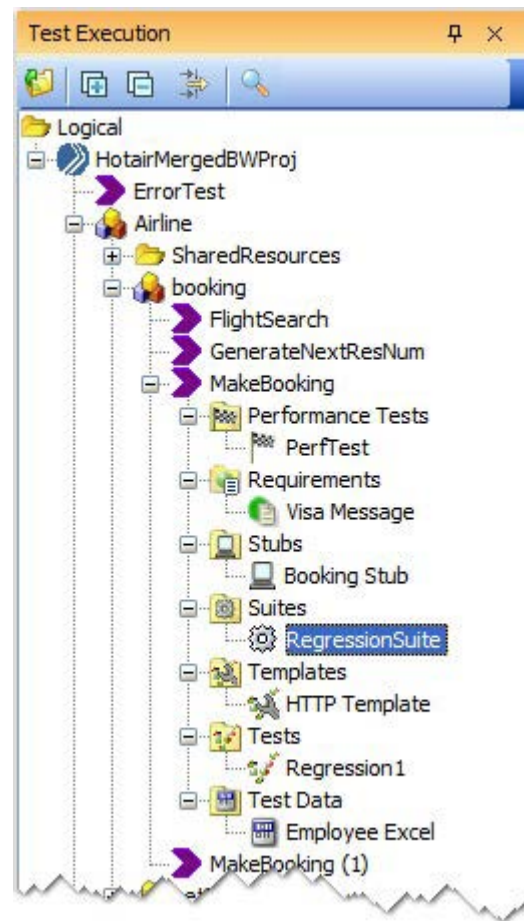
You can open any executable item displayed in the tree by right-clicking on it and selecting **Open** from the context menu. Alternatively, you can select any item and click the **Open** icon  in the Test Execution tree toolbar.

You can run any of the executable items in the tree by right-clicking and selecting **Run** from the context menu. To execute all items within a folder, right-click the folder and select **Run All** from the context menu.

More information about executing specific test items can be found later in this chapter.

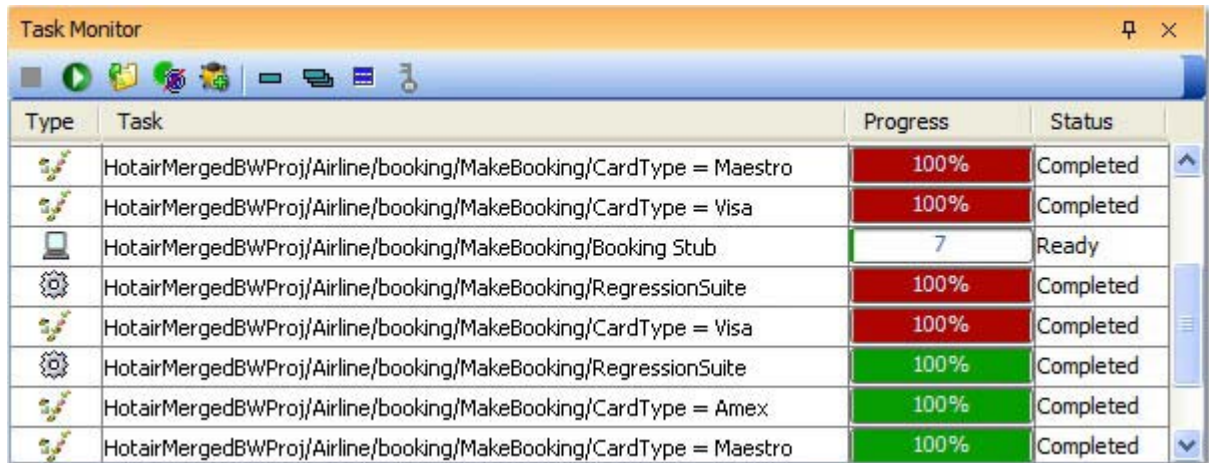
To focus the view on a specific item, right-click it and select **Promote to Root** from the context menu. To restore the normal view, click the root of the tree and select **Demote from Root** from the context menu.









The toolbar in the Test Execution tree contains shortcuts for opening selected items, manipulating what is displayed in the tree, and for searching test resources.



7.1.2 The Task Monitor










The Task Monitor displays all of the test items that have been executed in the current session.



Type	Task	Progress	Status
	HotairMergedBWProj/Airline/booking/MakeBooking/CardType = Maestro	100%	Completed
	HotairMergedBWProj/Airline/booking/MakeBooking/CardType = Visa	100%	Completed
	HotairMergedBWProj/Airline/booking/MakeBooking/Booking Stub	7	Ready
	HotairMergedBWProj/Airline/booking/MakeBooking/RegressionSuite	100%	Completed
	HotairMergedBWProj/Airline/booking/MakeBooking/CardType = Visa	100%	Completed
	HotairMergedBWProj/Airline/booking/MakeBooking/RegressionSuite	100%	Completed
	HotairMergedBWProj/Airline/booking/MakeBooking/CardType = Amex	100%	Completed
	HotairMergedBWProj/Airline/booking/MakeBooking/CardType = Maestro	100%	Completed

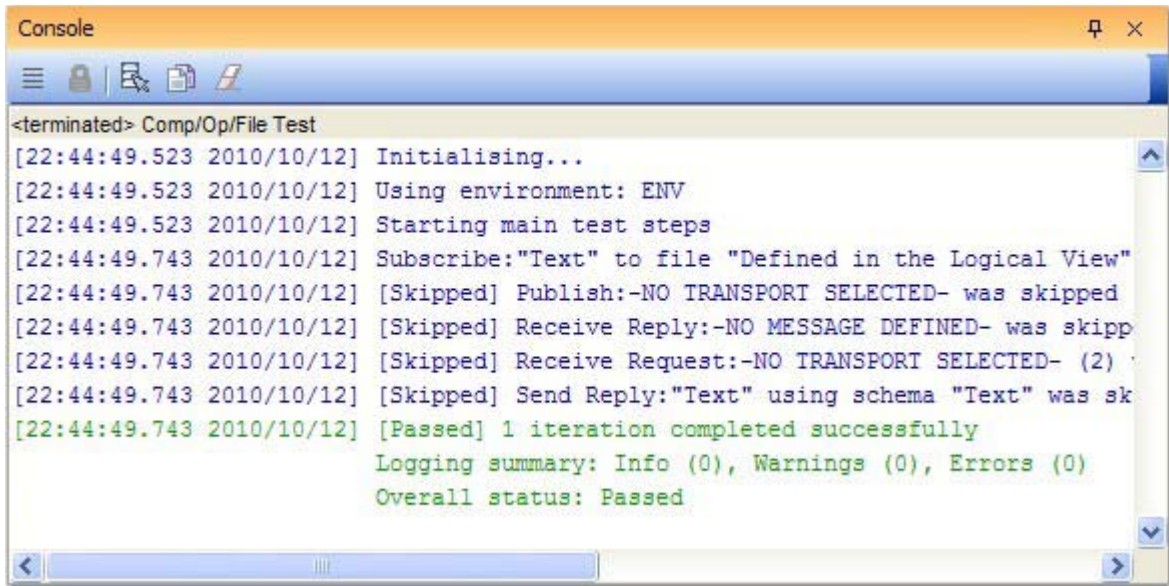
Test assets are displayed in the monitor view with an icon representing their type, the full path to the asset within the project, the progress (percent complete) and pass/fail status of the asset, and the overall execution status of the asset (for example, running, completed, torn down, and so on).

The toolbar at the top of the Task Monitor provides shortcuts for manipulating the contents of the view.






-
- | | |
|---|---|
|  | Stop the selected asset. |
|  | Run the selected asset(s) again. |
|  | Open the selected asset for editing. |
|  | Open the selected asset(s) in the Test Repair wizard |
|  | Raise a defect against the executed test or test suite in the associated change management system (see Change Management). |
|  | Remove the selected asset from the monitor view. |
|  | Remove all terminated assets from the monitor view. |
|  | Select all assets in the view. |
|  | Lock the current view (that is, the contents of the monitor will not scroll when new assets are run). |
-

7.1.3 The Console

The Console displays the detailed execution status and results of the test asset that is selected in the Task Monitor.



The toolbar at the top of the Console provides shortcuts for manipulating the contents of the view.

-
-  Toggle line wrapping on or off.
 -  Lock or unlock scrolling in the Console.
 -  Select all text in the Console.
 -  Copy the selected text to the clipboard.
 -  Clear the contents of the Console.
-

7.2 Working in the Test Lab


The main tasks you will carry out in the Test Lab will consist of executing test assets. You can also, however, repair tests and open test assets for editing directly from this perspective.

This section contains information about performing the various tasks that are available in the Test Lab perspective, as follows:

- [Open a Test Asset for Editing](#)
- [Execute a Test Asset](#)
- [Stop a Test Asset's Execution](#)
- [Raise a Defect Against a Test Asset](#)
- [View Message Differences](#)
- [Open a Failed Test Action](#)
- [Repair Validation Failures](#)
- [Run the Test Repair Wizard](#)
- [Overwriting Messages that Use Validation Rules](#)

7.2.1 Open a Test Asset for Editing

Test assets can be opened for editing (in the Test Factory) directly from the Test Execution tree, or from a test assets entry in the Task Monitor.

From within either view, you can right-click the asset and select **Open** from the context menu, or select the asset and click the **Open** icon  in the toolbar.

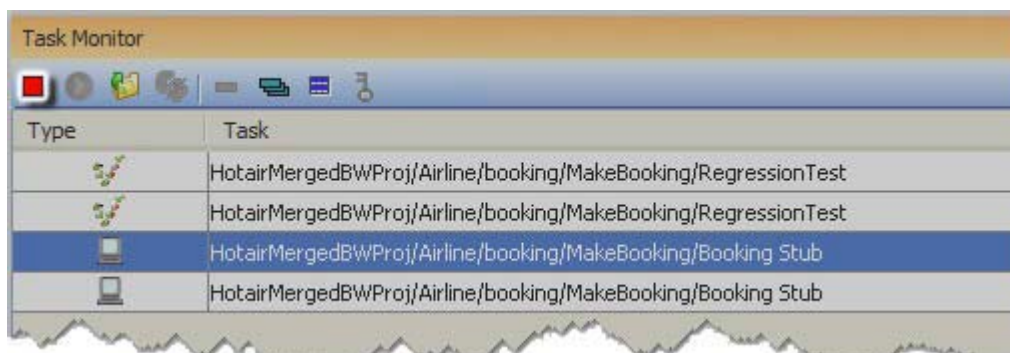
7.2.2 Execute a Test Asset

Test assets can be executed in the Test Lab in several ways:

- Right-click the asset in the Test Execution tree and select **Run** to use the current environment or **Run...** to customize execution.
- Select the asset and click the **Run** icon in the main toolbar to use the current environment or click the small arrow next to the icon to use the **Run...** option for customized execution.
- Click the small arrow next to the **Run** icon in the main toolbar and select the desired test asset to run (if it had been executed recently).
- If the asset has been executed once already, select its entry in the Task Monitor and click the **Launch again** icon in the Task Monitor toolbar.
- For more information about execution of specific test resources, see [Executing Tests](#), [Executing Test Suites](#), and [Executing Stubs](#).

7.2.3 Stop a Test Asset's Execution

If the execution of a test asset is in progress, you can stop it at any time by clicking the **Stop selected** icon in the Task Monitor toolbar.



NOTE: You must select the entry of a running asset in the Task Monitor before the **Stop selected** icon becomes available.

7.2.4 Copy the Resource Execution URL of an Asset


For any test or test suite that has an entry in the Task Monitor of the Test Lab, you can right-click on the entry to copy the resource execution URL for the asset (select **Copy Link** from the context menu). This link can be saved or shared and used to open the execution results (in the Results Gallery) of the specified asset, either from the Rational Integration Tester Welcome dialog (see [Open a Resource Execution](#)

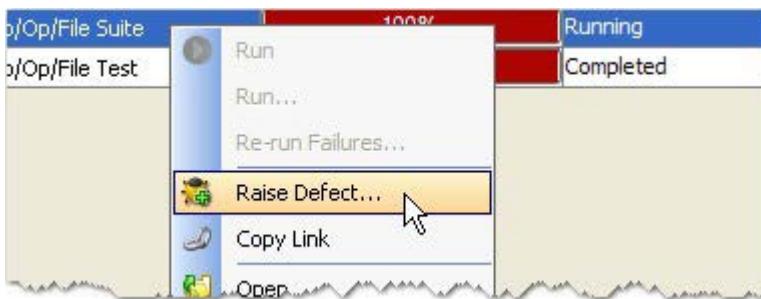
[URL](#)), or by pasting the URL into Internet Explorer or Firefox.

7.2.5 Raise a Defect Against a Test Asset

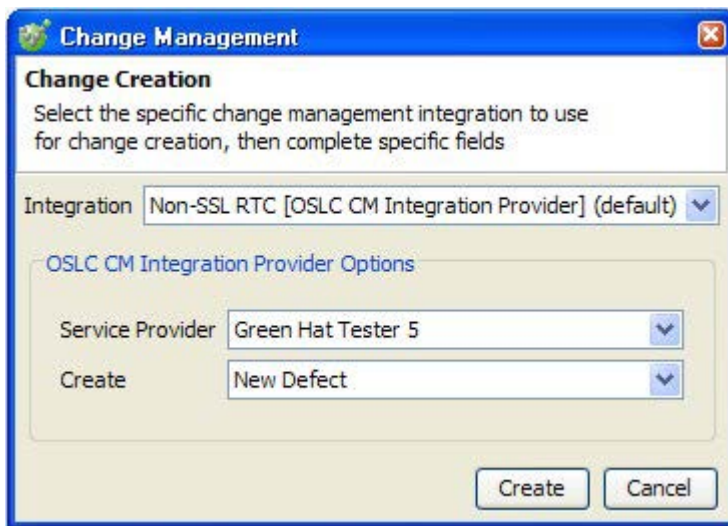
After a test or test suite has been executed in the Test Lab, users have the option of raising a defect against it using an existing change management integration (configured under Project Settings, see [Change Management](#)).

To raise a defect, follow the steps below:

1. Select the test or test suite entry in the Task Monitor and click the **Raise Defect**  icon in the toolbar. Alternatively, you can right-click the desired entry in the Task Monitor or right-click an error message within the console and select **Raise Defect** from the context menu.



The **Change Management** dialog is displayed.



2. To select a different change management integration than the default, select one from the **Integration** combo box.

OSLC Integrations

1. For OSLC integrations, select a target service provider (that is, a project area in the selected OSLC instance) from the **Service Provider** options.
2. From the **Create** options, select whether you want to create a **New Defect** or a **New Plan Item**.
3. When the change management options are correct, click **Create**.

A delegated creation window is displayed.

NOTE: Before creating the defect, you may be required to log in to the selected OSLC change management provider.

The screenshot shows a dialog box titled "OSLC CM Delegated Creation" with a timestamp "<02:13:25>". The dialog contains several input fields and sections:

- Summary:** A text field containing "File Test".
- State:** A dropdown menu.
- Resolution:** A dropdown menu.
- Severity:** A dropdown menu with a yellow smiley face icon, currently set to "Normal".
- Found In:** A dropdown menu set to "Unassigned".
- Filed Against:** A dropdown menu set to "Unassigned".
- Owned By:** A dropdown menu set to "Unassigned".
- Priority:** A dropdown menu set to "Unassigned".
- Planned For:** A dropdown menu set to "Unassigned".
- Quick Information:** A section on the right with the text "No Links..".
- Description:** A text area containing:
 - GH Tester Project: Defects
 - GH Tester Resource: Comp/Op/File Test
 - GH Tester Result URL: ghtester://resourceexecution/?
- Discussion:** A section at the bottom with the text "No Comments".

At the bottom right of the dialog are "OK" and "Cancel" buttons.

4. Fill in the fields required for creating the new defect or plan item in the selected change management system and click **OK**.
5. The new defect will be created as requested in the selected change management system.

Atlassian JIRA Integrations

For JIRA integrations, the defect is created directly from Rational Integration Tester.

Change Management

Change Creation
Select the specific change management integration to use for change creation, then complete specific fields

Integration: Jira Local [Atlassian Jira CM Integration Provider] (default)

Atlassian Jira CM Integration Provider Options

Project: TEST - Test Project

Issue type: Bug - A problem which impairs or prevents the functions of th...

Priority: Major - Major loss of function.

Summary: Defect for Simple File Test

Description:

Please fix this ASAP.

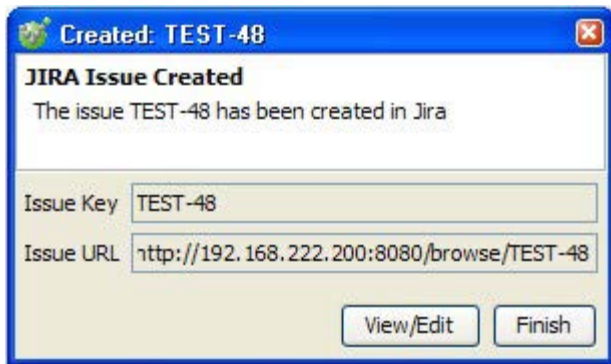
GH Tester Project: Defects
GH Tester Resource: Comp/Op/Simple File Test

GH Tester Link URL: ghtester://link/?MD0wMDAwMDEyYi04YzM0LTMwMTgtMDAwMC1hMWI1MTNiNGVzdF9yZXNvdXJjZSY2PUZpbGUgVGZdCAyJjk9L3RleHQmYT1mYWxzZSY4PS00MUY0ZDJiNzoxMmJiMDM2Zjc3MzotN2UyNQ==#2414431218

Create Cancel

1. If desired, you can modify the default project, issue type, or priority from the available fields – the default options were are configured when the integration is created.
2. Verify that the default **Summary** text for the new defect is correct – the default text is created automatically according to the name of the Rational Integration Tester resource.
3. In the **Description** field, you can add custom text to the default defect description, which includes a Rational Integration Tester link to be included in the defect.
4. When all of the provider options are correct, click **Create** to create the defect.

The issue will be created and a confirmation dialog is displayed.

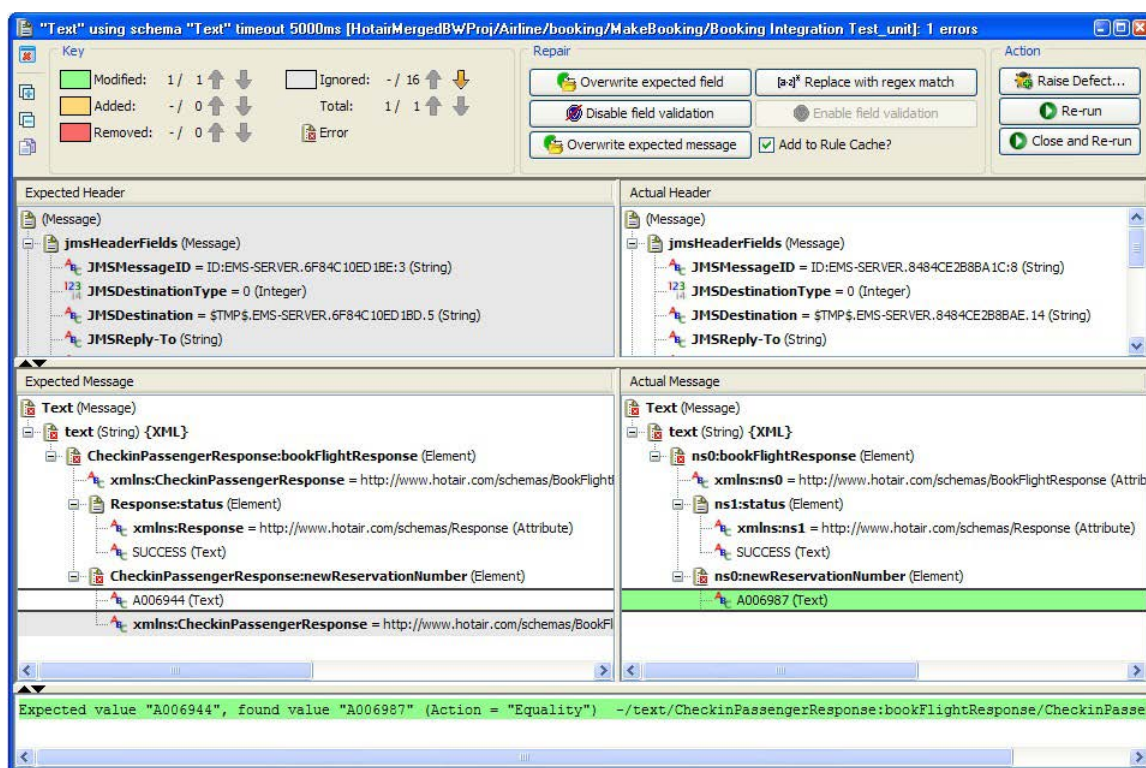


5. Click **View/Edit** to open the defect in JIRA, or click **Finish** to close the dialog.

7.2.6 View Message Differences

After a message passes or fails validation, you can view the expected message and the received message in the Message Differences window. To open the dialog, click the entry in the Console that details the difference or the passed assertion, or right-click the Console entry and click **Show Differences** from the context menu.

NOTE: If you click a publishing action, the Message Differences window displays the original (configured) message and the received message.



NOTE: If you are using Rational Integration Tester 8.0.1 (or later) to create and run message-based stubs, you can use the Message Differences window to debug a stub without having to open the Test Factory perspective and analyze each expected message individually. (For more information about this, refer to *IBM Rational Test Virtualization Server Reference Guide*.)



Overview

The header and body of the messages are displayed side-by-side, with corresponding nodes and elements in each message lined up next to one another. If you select a line in one message, the corresponding line in the other message is also selected. The details of any selected differences are displayed below the message bodies, and they are highlighted according to the type of difference. Line wrapping in the differences detail area will be the same as what is configured in the test console.





NOTE: Added or missing namespace identifiers will be ignored if they contain no attributes or elements, and they will not be highlighted in the Message Differences window.


Differences between the messages are highlighted in different colors, according to the difference type (modified, added, or removed). The highlight colors can be customized in the Rational Integration Tester preferences (**Window > Preferences** menu).

When clicking on a highlighted line in the console (below the displayed messages), the applicable node is expanded (if necessary) and highlighted in the messages above. To copy a highlighted portion of the console, right click and select **Copy selection to clipboard** or press **Ctrl + C**. You can also copy the entire contents of the console by right clicking and selecting **Copy all to clipboard**.

Next to the desired category under **Key**, click  to select the next difference or  to select the previous difference. As you move through the differences, the number of the selected difference will be displayed next to the total number (for example, 4 / 16).

NOTE: See [Message Comparison](#) for more information about preferences to control the **Next difference** and **Previous difference** buttons, and looping within the Message Differences window.

To display only differences, click . Use  and  to expand and collapse selected nodes within the messages, and click  to copy the contents of the selected line in the received header or message body.

NOTE: When expanding all nodes in large messages, Rational Integration Tester will only process the message for up to five seconds. After that time, all nodes that could be expanded will be expanded. If necessary, you can click  again to continue expanding any collapsed nodes.

NOTE: If you are using Rational Integration Tester 8.0.1 (or later), there are Expected Message controls (two navigation buttons and two option buttons) on the upper-left side of the Message Differences window. These controls relate to message-based stubs and are not available for individual tests. (For more information about the Expected Message controls, refer to *IBM Rational Test Virtualization Server Reference Guide*.)

Options for repairing validation errors, in the **Repair** panel, are explained in [Repair Validation Failures](#). If you want to be prompted to save any repairs as a rule, enable the **Add to Rule Cache?** option (see [The Rule Cache](#) for more information). Applicable repair options are also available from the context menu.

NOTE: After repairing one of multiple errors using the **Overwrite expected field**, **Disable field validation**, or **Replace with regex match** buttons, the next error in the message will be selected.

If validation has been disabled (from the console, test repair wizard, or using the message differences window) for any field in the expected message (that is, affected fields will be ignored and highlighted in grey), you can re-enable it by selecting the desired field and clicking the **Enable field validation** button.

NOTE: If validation has been disabled in the message editor, it can not be enabled again from the message differences dialog.

If desired, you can raise a defect for the selected test asset directly from the Message Differences dialog by clicking the **Raise Defect** button. See [Raise a Defect Against a Test Asset](#) for more information.

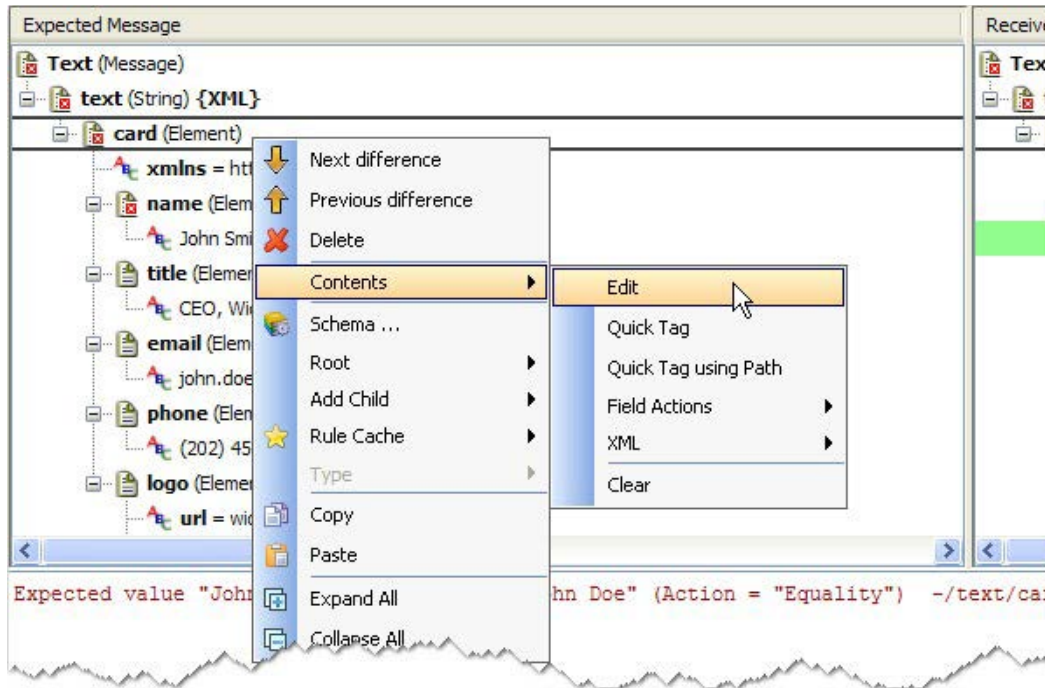
To run the test again (for example, after editing the expected message, modifying validation options, and so on), simply click the **Re-run** button. The Message Differences window will be closed, the test will be executed again, and the Message Differences window will open again, displaying the updated results.

If you want to run the test again without re-displaying the results, you can click the **Close and Re-run** button. The Message Differences window will be closed and the test will be executed in the Test Lab. The results will be displayed in the console, but the Message Differences window will not open again.

Editing Expected Messages

When viewing message differences, the expected message can be edited. Once modifications are made, the validation will be refreshed and the contents of the window will be updated.

To edit the contents of a message element, simply double-click to open the **Field Editor**, or you can right-click the field and select **Contents > Edit** from the context menu.



The expected message can be modified just as it can from a message editor, again using the options available in the context menu.

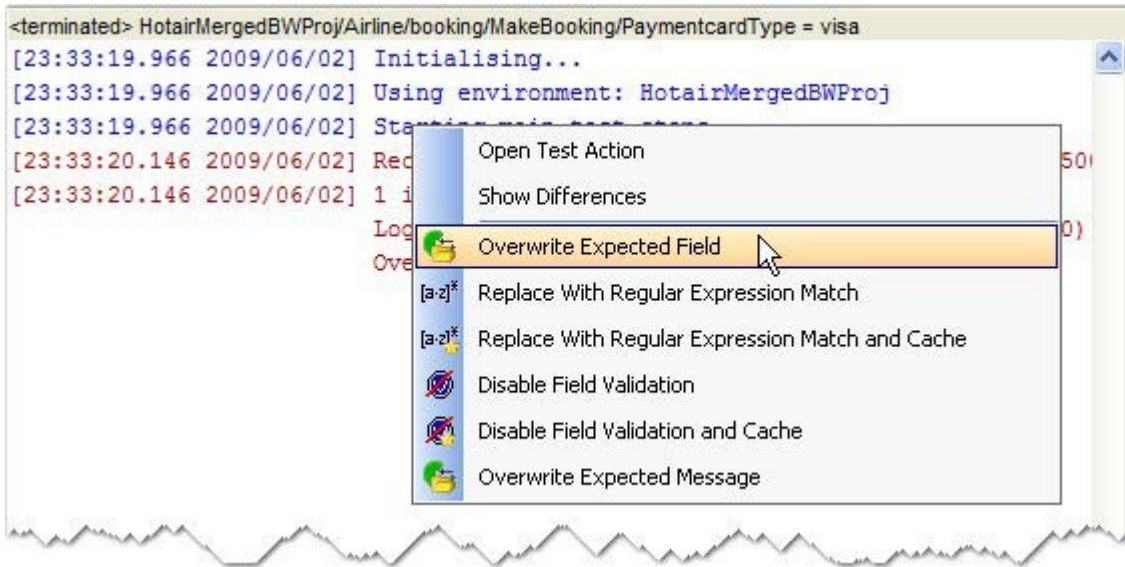
Additional information about the editing options available in the **Message Differences** window can be found throughout this guide.

NOTE: If changes are made to the expected message and the **Message Differences** window is closed, the user is prompted to save/apply the changes, discard the changes, or cancel the action and return to the window.

The Rational Integration Tester preference for saving/applying changes automatically can be enabled by checking the **Always save changes** option. See [Changing Preferences](#) for more information.

7.2.7 Repair Validation Failures

If a message fails validation, you can repair the differences directly from the Console by right-clicking the Console entry that details the failure and selecting one of the available repair options.

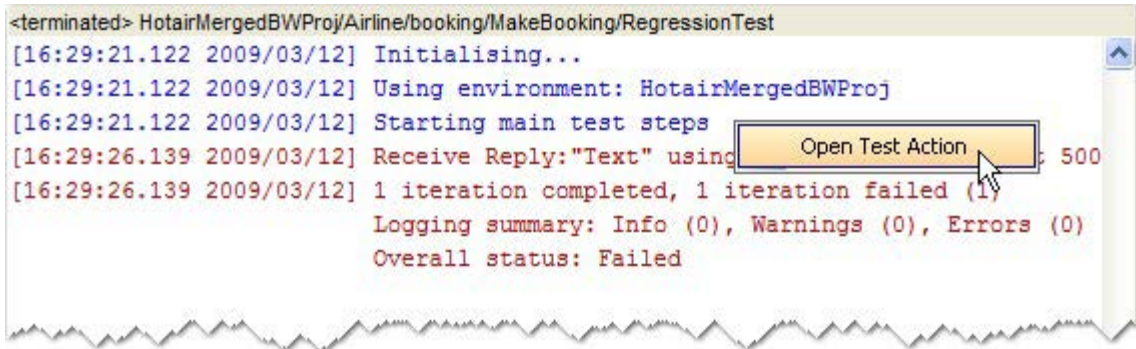


- **Overwrite Expected Field** will replace the value in the expected message with the value from the corresponding field in the received message.
- **Replace With Regular Expression Match** replaces the value in the expected message with a regular expression (for example, A004532 would be replaced by `^A\d{6}`, which would match the letter “A” followed by any six digits). If a regex match would not apply to the selected message fields, the option is not displayed.
- **Replace With Regular Expression Match and Cache** replaces the value with a regular expression and creates a rule to apply the same expression when a similar field is encountered (based on its path relative to the root of the message). If a regex match would not apply to the selected message fields, the option is not displayed. A star is displayed in the affected field in the message editor. See [The Rule Cache](#) for more information about managing rules.
- **Disable Field Validation** will modify the test action, disabling validation of the field that was different.
- **Disable Field Validation and Cache** disables field validation and creates a rule to disable validation when a similar field is encountered. See [The Rule Cache](#) for more information about managing rules.
- **Overwrite Expected Message** will replace the entire expected message with the contents of the message that was received.

7.2.8 Open a Failed Test Action


If a test fails because of a failed test action (for example, a message is not received within the timeout period), you can open the failed test action directly from the Console.

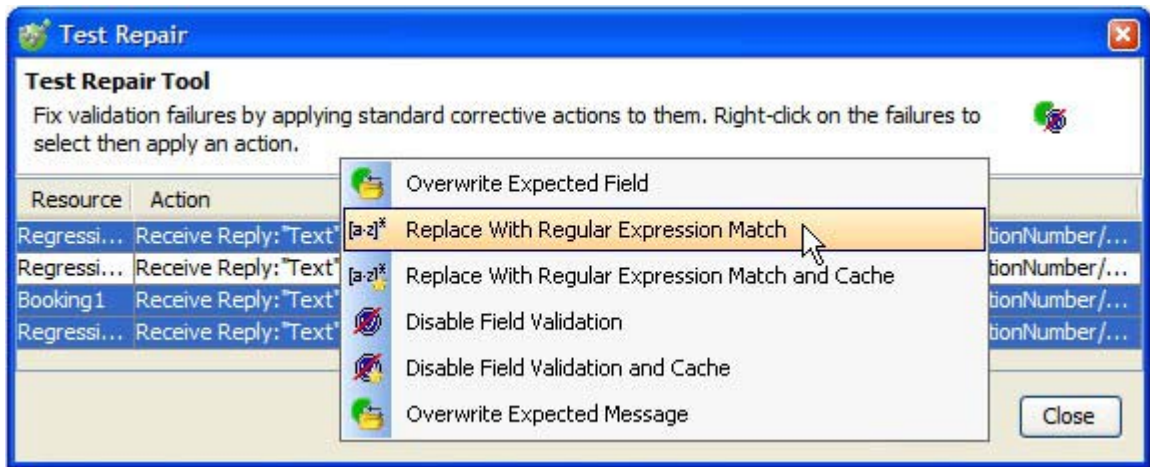
To open the test action, click on the entry in the Console that details the failure, or right-click the Console entry and select **Open Test Action** from the context menu.



The action that failed within the selected test asset will be opened in the Test Factory.

7.2.9 Run the Test Repair Wizard

The Test Repair wizard can be used to fix validation errors in one or more executed tests. To launch the wizard, select the entries of the tests to repair in the Task Monitor and click the **Launch Test Repair Wizard** icon  in the toolbar.

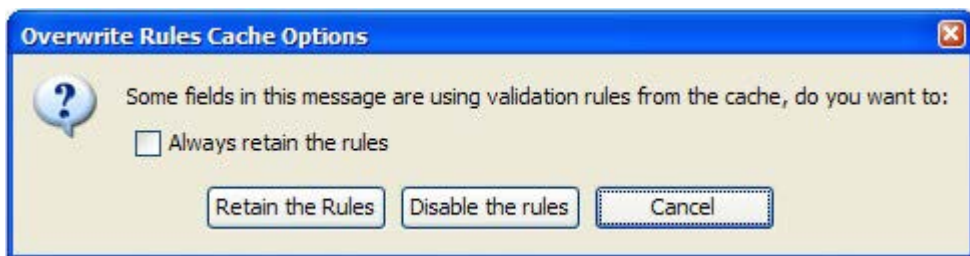


All of the validation errors from the selected tests are displayed. Select the tests you want to repair (use Ctrl or Shift to select multiple tests), then right-click one of the

tests and select the desired repair option (described in [Repair Validation Failures](#)).

7.2.10 Overwriting Messages that Use Validation Rules

If you try to overwrite an expected message (that is, in the [View Message Differences](#) dialog, when you [Run the Test Repair Wizard](#), or directly from the console) that contains one or more fields using validation rules (see [The Rule Cache](#)), you will be prompted to decide how to handle the rules.



- If you click **Retain the Rules**, the message is overwritten but the status of any rules on the message is not changed.
- If you click **Disable the Rules**, the message is overwritten and all cached rules in use by the selected message will be disabled.
- If you click **Cancel**, the message is not overwritten and the operation is cancelled. In this case, further investigation is probably required.
- If you check the **Always retain the rules** option, taking the same action (that is, overwriting a message that uses validation rules) in the future will result in the message being overwritten and the status of any rules on the message being left unchanged.

NOTE: Checking the **Always retain the rules** option sets an application preference that can be modified under the Message Comparison area of the Preferences dialog (see [Changing Preferences](#) in Chapter 2).

Results Gallery

Contents

Overview

Test Suite Results

Test Case Results

Test Cycle Results

Detailed Reports

Managing Reports, Console Output, and Notes

Notes

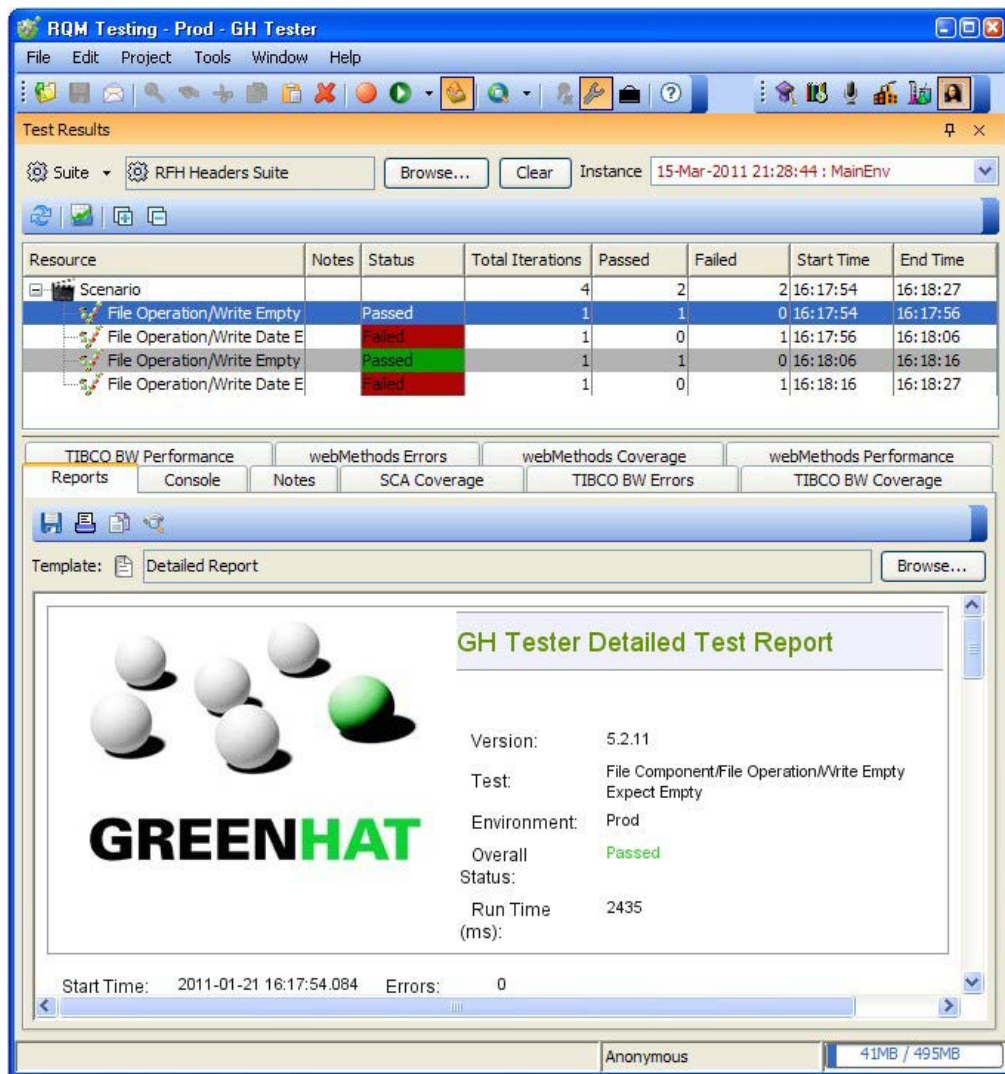
Deleting Test Suite Results

Populating Suite Details for Past Test Runs

This chapter provides an overview of Rational Integration Tester's Results Gallery perspective, including details about how to view summary and detailed coverage reports for specific test suite execution instances and for tests that were executed within a test suite.

8.1 Overview

The Results Gallery perspective lets users view an execution summary and detailed coverage, error, and performance reports for executed test suites and for test cases that have been executed within a suite or as part of a test cycle.



The Results Gallery contains the Test Results view, which is divided into an execution summary on top and the report viewer on the bottom. In the report viewer area, users can select from numerous report templates in which results can be displayed.

NOTE: When browsing for execution results, click the resource-type button (options: Suite, Test, Test Cycle, or Performance Test). Clicking this button opens the Select a Resource dialog box.

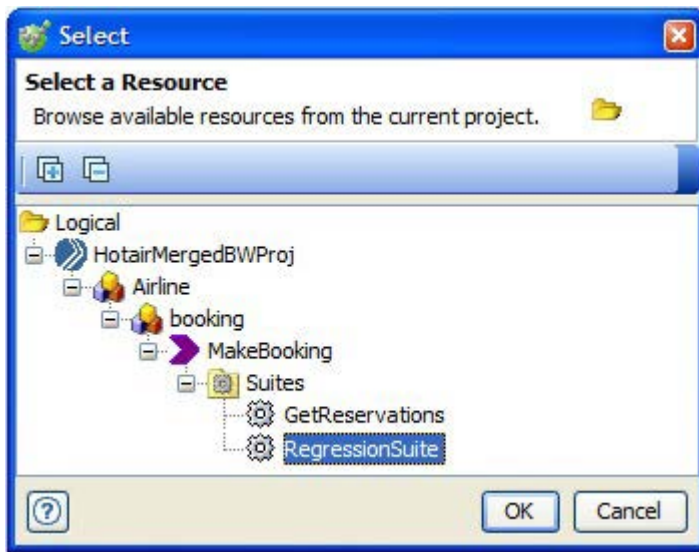
8.2 Test Suite Results

Summary and detailed reports can be displayed for one test suite instance at a time.

8.2.1 Selecting a Test Suite Instance

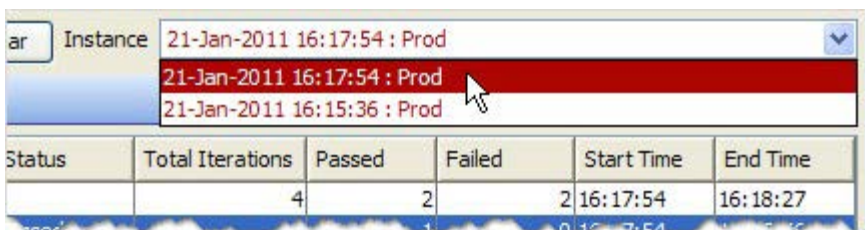
Follow the steps below to select a test suite instance in the Results Gallery.


1. Select **Suite** as the resource type, click **Browse** next to the **Resource** field, locate and select the desired test suite in the project resource dialog, then click **OK**.



2. For the specified suite, select the execution instance from the **Instance** dropdown.

The date, time, and environment of each instance is displayed, and the status (passed or failed) of the instance is indicated by the color (green or red) of its entry.

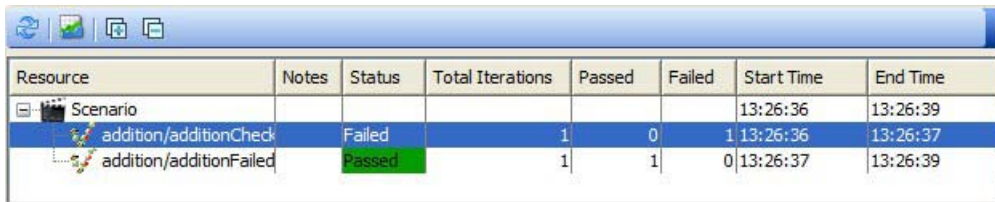


NOTE: If you don't see an expected instance, click the **Reload** icon  to refresh the contents of the dropdown.

The execution summary of the selected instance is displayed (top), and detailed reports for the instance can be selected in the report viewer area (below).

8.2.2 Test Suite Execution Summary





An execution summary for the selected test suite instance is displayed at the top of the Results Gallery perspective.



Resource	Notes	Status	Total Iterations	Passed	Failed	Start Time	End Time
Scenario		Passed				13:26:36	13:26:39
addition/additionCheck		Failed	1	0	1	13:26:36	13:26:37
addition/additionFailed		Passed	1	1	0	13:26:37	13:26:39

The summary report contains the following information:

Column	Description
Resource	Displays the contents of the selected test suite, with details for each suite resource provided in the remaining columns.
Notes	Indicates whether or not a note has been saved for the selected scenario or suite resource.
Status	The overall status of the selected resource, either Passed or Failed .
Total Iterations	The total number of iterations of the selected resource that were executed in the test suite. For a scenario, the total of all iterations for child tests is displayed. If a scenario contains child scenarios, the total of all child resources is displayed.
Passed	The number of iterations of the selected resource that passed. For a scenario, the total number of passed child tests is displayed.
Failed	The number of iterations of the selected resource that failed. For a scenario, the total number of failed child tests is displayed.
Start Time	The time when the selected resource's execution began.
End Time	The time when the selected resource's execution ended.
Exec Time (ms)	The overall time (in milliseconds) that the selected resource took to complete its execution. For the overall scenario, this time includes set up and tear down phases.

To reload the summary, click the **Reload** icon  in the toolbar. To view Probe results for the selected resource, click the **Probe Results** icon . To expand the selected node, click the **Expand Selection** icon . To collapse the selected node, click the **Collapse Selection** icon .

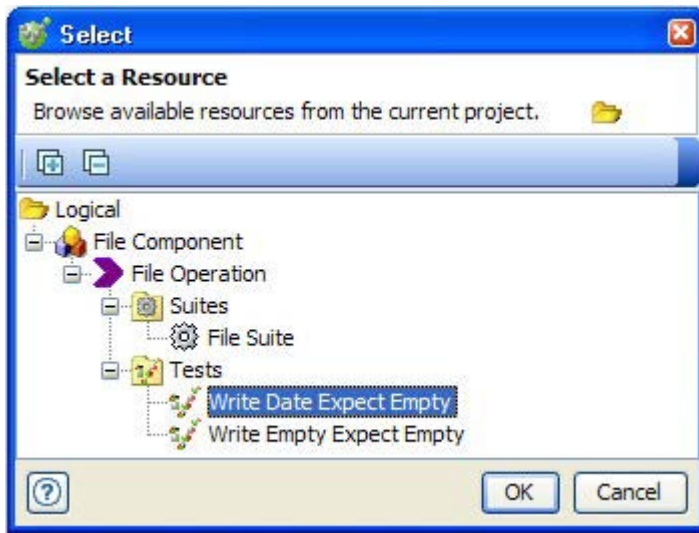
8.3 Test Case Results

Execution results can be displayed for test cases that were executed as part of a test suite. Summary and detailed reports can be displayed for one test case at a time.

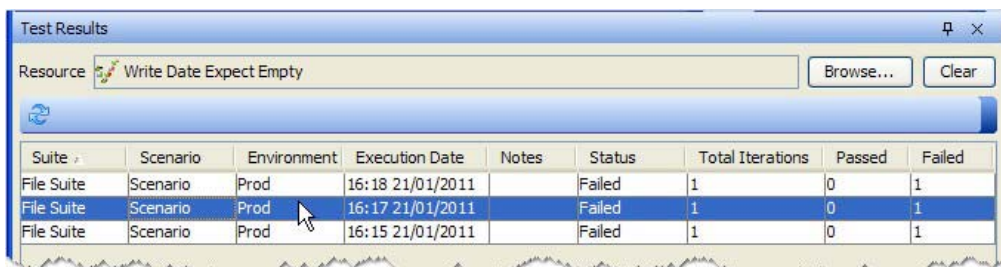
8.3.1 Selecting a Test Case Instance

Follow the steps below to select a test case instance in the Results Gallery.


1. Select **Test** as the resource type, click **Browse** next to the **Resource** field, locate and select the desired test case in the project resource dialog, then click **OK**.



2. For the selected test case, select the specific execution instance from the execution summary table below the resource field.



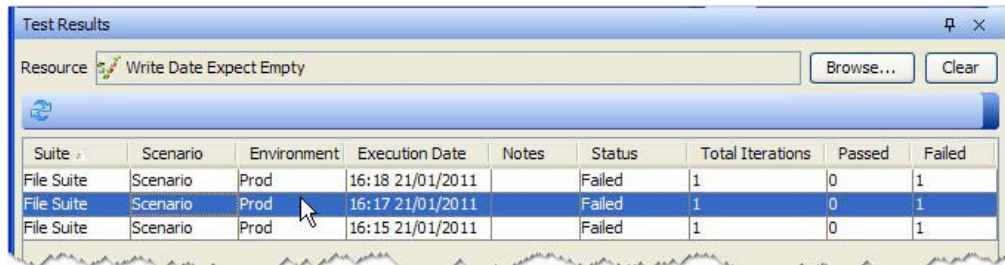
Suite	Scenario	Environment	Execution Date	Notes	Status	Total Iterations	Passed	Failed
File Suite	Scenario	Prod	16:18 21/01/2011		Failed	1	0	1
File Suite	Scenario	Prod	16:17 21/01/2011		Failed	1	0	1
File Suite	Scenario	Prod	16:15 21/01/2011		Failed	1	0	1

NOTE: If you don't see an expected instance of the selected test case, click the **Reload** icon  to refresh the contents of the table.

The detailed reports for the specified test case instance can be selected in the report viewer area (below).

8.3.2 Test Case Execution Summary

An execution summary for the selected test case is displayed at the top of the Results Gallery perspective.




The screenshot shows a window titled "Test Results". At the top, there is a "Resource" field with a dropdown menu showing "Write Date Expect Empty", a "Browse..." button, and a "Clear" button. Below this is a table with the following columns: Suite, Scenario, Environment, Execution Date, Notes, Status, Total Iterations, Passed, and Failed. The table contains three rows of data, all with a status of "Failed".

Suite	Scenario	Environment	Execution Date	Notes	Status	Total Iterations	Passed	Failed
File Suite	Scenario	Prod	16:18 21/01/2011		Failed	1	0	1
File Suite	Scenario	Prod	16:17 21/01/2011		Failed	1	0	1
File Suite	Scenario	Prod	16:15 21/01/2011		Failed	1	0	1

The summary report contains the following information:

Column	Description
Suite	Displays the name of the executed test suite or test cycle that contained the selected test case at runtime.
Scenario	Displays the name of the scenario in which the test case was executed. For test cycle results, "Test Cycle" is displayed.
Environment	Displays the name of the environment in which the test case was executed.
Execution Date	The time and date when the selected test case was executed.
Notes	Indicates whether or not a note has been saved for the selected test case or containing test suite.
Status	The status of the selected test case, either Passed or Failed .
Total Iterations	The total number of iterations of the selected test case that were executed in the test suite.
Passed	The number of iterations of the selected test case that passed.
Failed	The number of iterations of the selected test case that failed.

- To reload the test case summary table, click the **Reload** icon  in the toolbar.
- To open the suite (in the Test Factory) in which the test case was executed, highlight and right-click the test case instance and select **Open Suite**.
- To copy the Rational Integration Tester link to the selected test case results, highlight and right-click the test case instance and select **Copy Link**.
- To view the results for the test suite in which the test case was executed, highlight and right-click the test case instance and select **View Suite Results**.

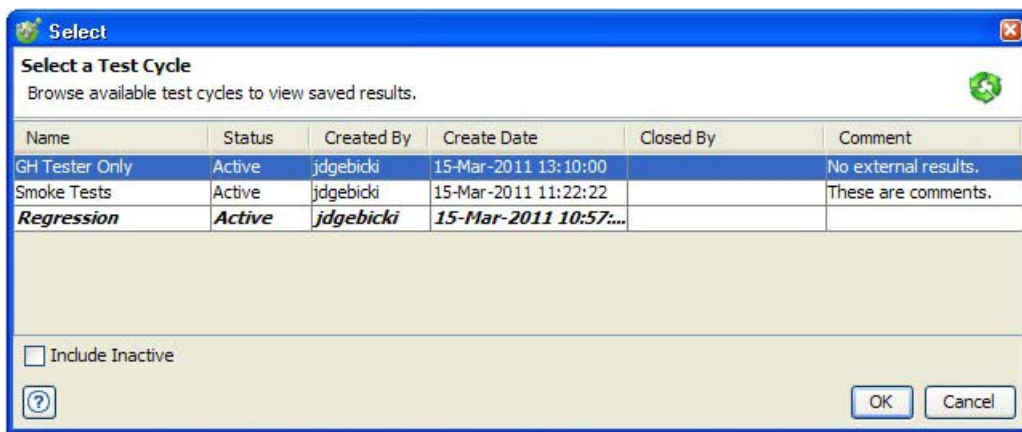
8.4 Test Cycle Results

Summary and detailed reports can be displayed for one test cycle at a time.

8.4.1 Selecting a Test Cycle


Follow the steps below to select a test cycle in the Results Gallery.

1. Select **Test Cycle** as the resource type and click **Browse** next to the **Resource** field.
2. Locate and select the desired test suite in the project resource dialog – enable the **Include Inactive** option to see inactive test cycles – then click **OK**.








The selected test cycle is displayed, containing results for all resources that were executed against it.

Resource	Notes	Status	Total Iterations	Passed	Failed
Test Cycle			21	14	
FileTests/SimpleDataTest		Failed	1	0	
Short Last Segment Suite			2	1	
File Operation/Simple File		Passed	1	1	
Validating an Empty File -		Passed	1	1	
Validating an Empty File -		Passed	1	1	
10135/1 - Validate Correc		Passed	1	1	
10220/01 - Single Value T		Failed	1		

NOTE: If you don't see an expected instance, click the **Reload** icon  to refresh the contents of the dropdown.





8.4.2 Test Cycle Summary

An execution summary for the selected test cycle is displayed at the top of the Results Gallery perspective.

Resource	Notes	Status	Total Iterations	Passed	Failed	Start Time	End Time
 Test Cycle			21	14	7	10:57 15/03/2011	
 FileTests/SimpleDataTest		Failed	1	0	1	13:58 15/03/2011	13:58 15/03/2011
 Short Last Segment Suite			2	1	1	15:26 15/03/2011	15:27 15/03/2011
 File Operation/Simple File		Passed	1	1	0	15:26 15/03/2011	15:26 15/03/2011

The summary contains the following information:

Column	Description
Resource	Displays the contents of the selected test cycle, with details for each resource provided in the remaining columns.
Notes	Indicates whether or not a note has been saved for the selected resource.
Status	The overall status of the selected resource, either Passed or Failed – there is no status for the test cycle.
Total Iterations	The total number of iterations of the selected resource that were executed in the test cycle. For the test cycle, the total of all iterations is displayed. For a scenario, the total of all iterations for child tests is displayed. If a scenario contains child scenarios, the total of all child resources is displayed.
Passed	The number of iterations of the selected resource that passed. For a scenario, the total number of passed child tests is displayed.
Failed	The number of iterations of the selected resource that failed. For a scenario, the total number of failed child tests is displayed.
Start Time	The time when the selected resource's execution began.
End Time	The time when the selected resource's execution ended.
Exec Time (ms)	The overall time (in milliseconds) that the selected resource took to complete its execution. For the overall scenario, this time includes set up and tear down phases.

To reload the summary, click the **Reload** icon  in the toolbar. To view Probe results for the selected resource, click the **Probe Results** icon . To expand the selected node, click the **Expand Selection** icon . To collapse the selected node, click the **Collapse Selection** icon .

8.5 Detailed Reports

Below the suite execution summary is a collection of more detailed reports (accessed by their named tabs) that can be viewed for any selected test instance.

NOTE: If you are using Test Cycles and want coverage reports, the applicable tests must still be executed from within a test suite (that is, coverage reports will not be produced for tests that are executed outside of a test suite).

Reports

This report provides execution details for a selected test suite. Summary information is included, plus an overall suite status, the percentage of tests that passed and failed, and the number of errors and warnings that were caught in each test in the suite.

For a test case, the report displays similar execution results as for a suite, but it also includes details for each of the test steps that were executed.

NOTE: If additional report templates are available, they can be selected by clicking **Browse** next to the **Template** field.

Console

The **Console** tab displays the console output (from the Test Lab) for the selected test.

SCA Coverage

This report provides a summary of the test activities that were performed for selected Oracle Fusion resources (SCA Domains).

TIBCO BW Errors

This report provides a summary of the BusinessWorks processes that ran and the number of activities within each process that errored. The total number of errored activities is also provided.

TIBCO BW Coverage

This report provides a summary of the number of activities that were run and missed for each BusinessWorks process that ran in the suite. Based on the run/missed numbers, a coverage percentage is generated. An overall summary of run/missed/coverage is provided for all processes.

For each BusinessWorks process that was executed, a detailed report is generated that shows which activities were covered and which ones were missed. For covered

activities, details are also provided regarding the number of times each activity was called, as well as the minimum, maximum, average, and total time for each activity.

TIBCO BW Performance

This report provides a summary of the details contained within the TIBCO BusinessWorks Coverage Report, listing all of the BusinessWorks activities that were executed and the process that contains them. For each activity, the same details are provided (times called and execution time results).

webMethods Errors

This report provides a summary of the processes that ran and the number of activities within each process that errored. The total number of errored activities is also provided.

webMethods Coverage

This report provides a summary of the number of activities that were run and missed for each webMethods process that ran in the suite. Based on the run/missed numbers, a coverage percentage is generated. An overall summary of run/missed/coverage is provided for all processes.

For each webMethods process that was executed, a detailed report is generated that shows which activities were covered and which ones were missed. For covered activities, details are also provided regarding the number of times each activity was called, as well as the minimum, maximum, average, and total time for each activity


webMethods Performance

This report provides a summary of the details contained within the webMethods Integration Server Coverage Report, listing all of the activities that were executed and the process that contains them. For each activity, the same details are provided (times called and execution time results).

8.6 Managing Reports, Console Output, and Notes


Rational Integration Tester allows you to save, print, and copy reports using the icons displayed above the current report.

8.6.1 Save a Report

The displayed report, console output, or notes can be saved as a web page or text file by clicking the **Save** icon  in the toolbar above the displayed report.

When prompted, specify the location and name of the output file, then click **Save**.


8.6.2 Print a Report

The displayed report, console output, or notes can be printed to a previously configured printer by clicking the **Print** icon  in the toolbar above the displayed report.


NOTE: For best results when printing reports, Rational Integration Tester will prompt you to save the report and print it from a browser. To do this, click **No** to cancel and refer to [Save a Report](#). To continue printing, click **Yes**.

When prompted, select the desired printer and click **OK** to print.

8.6.3 Copy a Report

The displayed report, console output, or notes can be copied to the clipboard by clicking the **Copy** icon  in the toolbar above the displayed report. Rational Integration Tester will copy the report code or console/note text to the clipboard, which can be pasted into another application for further use.

8.6.4 Open a Report Externally

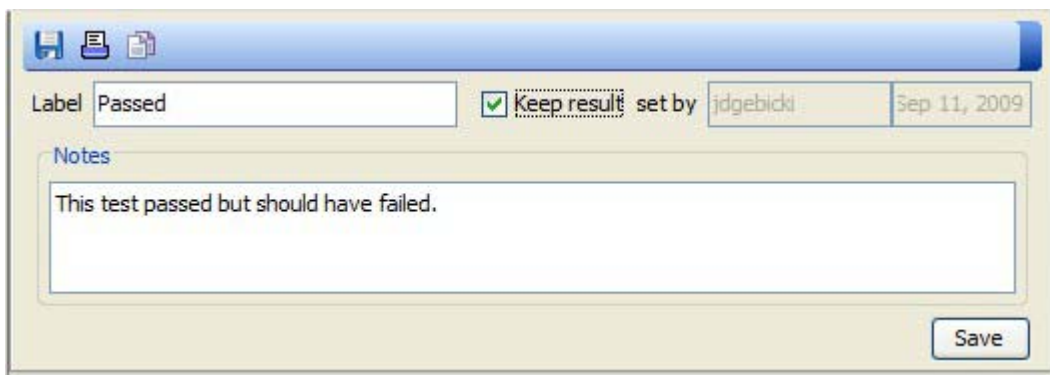
The displayed report can be opened externally (in the default browser on the local system) by clicking the **Open Externally** icon  in the toolbar above the displayed report.

NOTE: Files for externally displayed reports will be generated in a temporary directory within the current user's profile.

8.7 Notes

Under the **Notes** tab you can apply a label to the selected test suite instance, making it easier to identify from the list of available instances. You can also enable the **Keep result** option for the selected instance, which designates that results for the selected instance should not be deleted. The ID of the current user and the current date are stored when this option is enabled.

Finally, you can create a note (for example, additional information about the entry or details that might be useful to other users) to be saved with the selected scenario or test suite resource. A single note can be saved with each scenario or resource.



The screenshot shows a dialog box titled "Notes" with a blue header bar containing icons for save, print, and copy. Below the header, there is a "Label" field with the text "Passed". To the right of the label field is a checkbox labeled "Keep result" which is checked, followed by the text "set by". To the right of "set by" is a text field containing the username "jdgebicki". To the right of the username field is a date field containing "Sep 11, 2009". Below these fields is a large text area with the text "This test passed but should have failed." and a "Save" button at the bottom right.


If any of the options are changed, be sure to click **Save** before selecting another instance or scenario resource. Otherwise, the changes will be lost.

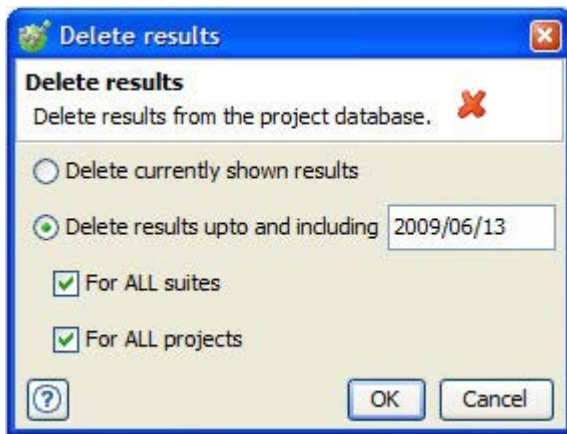
8.8 Deleting Test Suite Results

The results of one or more test suite instances can be deleted from the project database directly within the Results Gallery or from the command line.

NOTE: If a suite has been run in a test cycle, the suite's past results cannot be deleted as described here. Current results can be deleted, and deleting the test cycle itself will delete the test suite results.

8.8.1 Deleting Results from the Results Gallery

Test suite results can be deleted from the Results Gallery by clicking the **Delete** icon  in Rational Integration Tester's main toolbar. After clicking the icon, the **Delete Results** dialog is displayed, providing you several options for how to delete result data.



- To delete only the displayed results (that is, those of the currently selected test suite instance), enable the **Delete currently shown results** option.
- To delete multiple result sets for the selected suite, enable the **Delete results up to and including** option and enter the date for which all results and all earlier results should be deleted. When deleting multiple results, wildcard options can delete a broader range of data:
 - **For ALL suites** lets you delete results for all test suites before and including the selected date.
 - **For ALL projects** lets you delete all test suite results in all projects within the database before and including the selected date.

NOTE: If any test suites have been designated to be kept (see [Notes](#)), they will not be removed by the delete action.

8.8.2 Deleting Results from the Command Line

Test suite results can be deleted from the command line using the **GHTesterCmd** executable, found in the root of the Rational Integration Tester installation (C:\Program Files\IBM\RationalIntegrationTester, by default). The command has the following syntax:

```
GHTesterCmd -p <project> delete-all-results-keeping PnYnMnD
```

- <project> specifies the full path to the Rational Integration Tester project (*.ghp) from which results should be deleted.
- PnYnMnD specifies how much data should be kept, represented in years (nY), months (nM), or days (nD).

For example, to delete all results older than 90 days, the following command might be issued:

```
GHTesterCmd -p C:\GHProjects\HTTP\http.ghp delete-all-results-keeping P90D
```

8.9 Populating Suite Details for Past Test Runs

Detailed reports for tests that were executed within a suite will not be available in the Results Gallery if those tests were executed against a database schema prior to version 1.9.24.b. If you are now using schema version 1.9.24.b or higher, the containing suite details for such tests can be populated in the database using the `backfill-results` argument with the **GHTesterCmd** executable, found in the root of the Rational Integration Tester installation (C:\Program Files\IBM\RationalIntegrationTester, by default). The command has the following syntax:

```
GHTesterCmd -p <project> backfill-results
```

<project> specifies the full path to the Rational Integration Tester project (*.ghp) for which test results should be updated.

NOTE: Ensure that the specified project has been configured to use the correct database (that is, the one you wish to populate with suite details for applicable test runs).

After the command is finished, the total number of rows that were updated in the database will be displayed. Tests that were executed before using schema 1.9.24.b will display their root suite in the Results Gallery, and the detailed reports for those tests will now be available.

Getting Started

Contents

Starting the Program

The Welcome Dialog

Opening an Existing Project

Opening an Existing Project

Creating a New Project

Cloning a Project

**Fetching a Project from Source
Control**

Project Errors

This chapter provides information about how to get started with Rational Integration Tester, including how to launch the application, and how to open, create, and clone a project.

9.1 Starting the Program

This section describes how to launch Rational Integration Tester, depending on the installed platform:

- [Rational Integration Tester on Windows](#)
- [Rational Integration Tester on UNIX](#)

9.1.1 Rational Integration Tester on Windows

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

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

9.1.2 Rational Integration Tester on UNIX

To launch Rational Integration Tester on the UNIX platform, you must set the display environment and then execute the application script.

1. Set the DISPLAY environment variable to point to your screen. For example:

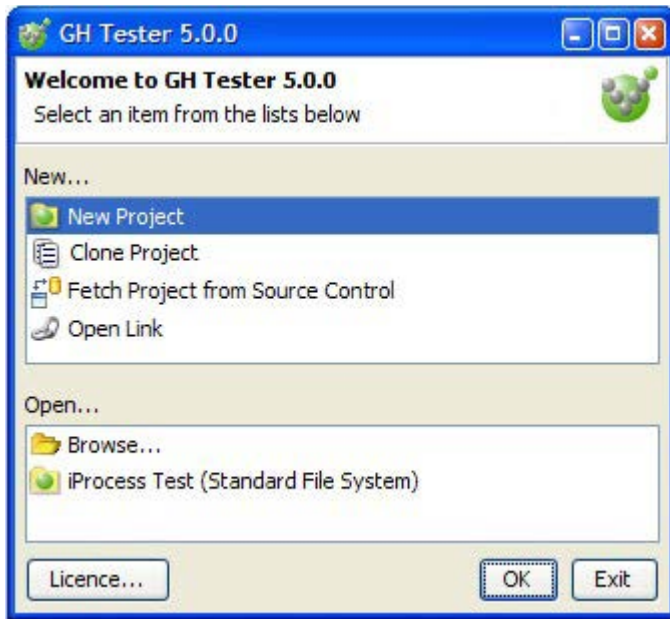
```
csh> setenv DISPLAY myserver:0 (where myserver is the host name of the  
workstation running Rational Integration Tester)
```

2. Change to the “bin” directory of the Rational Integration Tester installation and execute the **ghtester.sh** script:

```
csh> ghtester
```

9.2 The Welcome Dialog

When you launch Rational Integration Tester, a welcome dialog is displayed that lets you create new project or open an existing one.



- If this is the first time you are using Rational Integration Tester, click on the **Licence** button to enter your licence key (see [Opening an Existing Project](#)).
- To create a new project, select **New Project** and click **OK** (see [Creating a New Project](#)).
- To clone an existing project, which creates a new project containing the same structure and objects from the original project, select **Clone Project** and click **OK** (see [Cloning a Project](#)).
- To check out an existing project that has been shared in Rational Integration Tester's Source Control application, select **Fetch Project from Source Control** and click **OK** (see [Fetching a Project from Source Control](#)).
- To open an existing project, select its entry in the list and click **OK**.

NOTE: If you are unsure of a project's location, place the mouse over the project name and the project directory will be displayed.

- To open an existing project that is not displayed, click **Browse**, then click **OK**. Locate the Rational Integration Tester project (*.ghp) file and click **Open**.

-
- To open a resource execution link, select **Open Link** and click **OK** (see [Open a Resource Execution URL](#)).

9.3 Opening an Existing Project

All work in Rational Integration Tester is managed in the context of individual projects, which makes it easier to organise complex test scenarios. Rational Integration Tester projects do not need to be saved, but the resources they contain do.

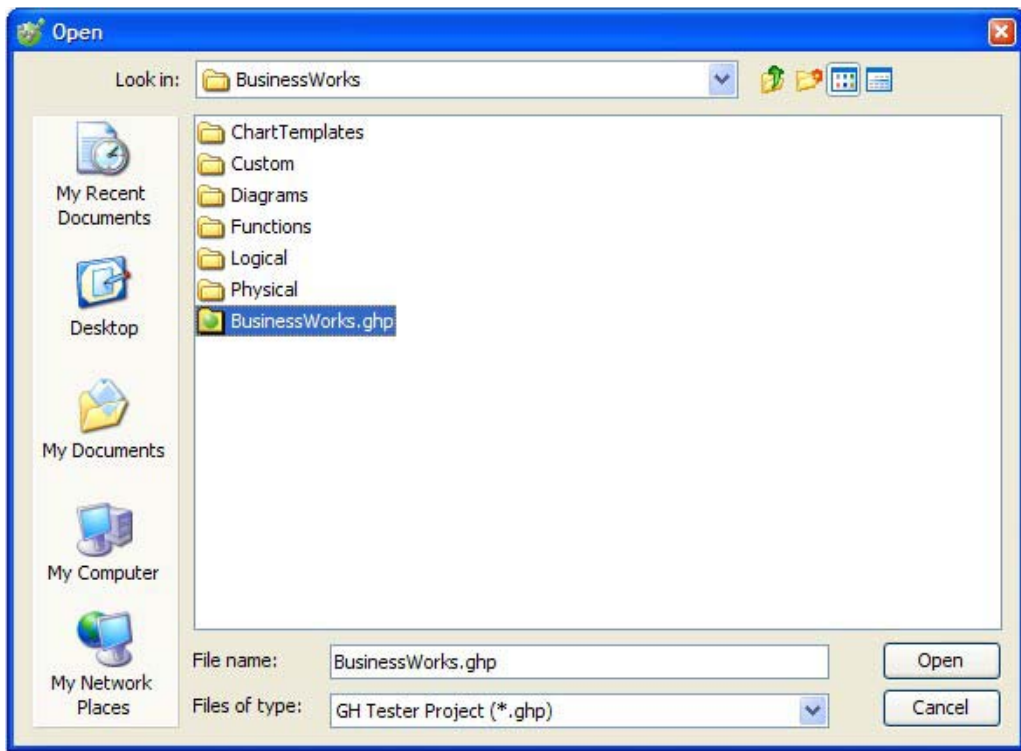
1. Launch Rational Integration Tester (see [Starting the Program](#)).

The Rational Integration Tester welcome screen is displayed.



2. If the project is listed under **Existing**, select the project name and click **OK**.
The project is opened in Rational Integration Tester.
3. If the project is not listed under **Existing**, select **Browse** and click **OK**.

The project **Open** dialog is displayed.



4. Browse for and select the desired project file (*.ghp), then click **Open**.
5. If permissions have been enabled on the project, enter a valid domain user/password combination, or enable the “Logon as project administrator” option and enter the admin password, then click **OK**.



If the login credentials are accepted, or if permissions are not enabled on the project, the project resources are loaded and the project is opened in Rational Integration Tester.

9.4 Creating a New Project

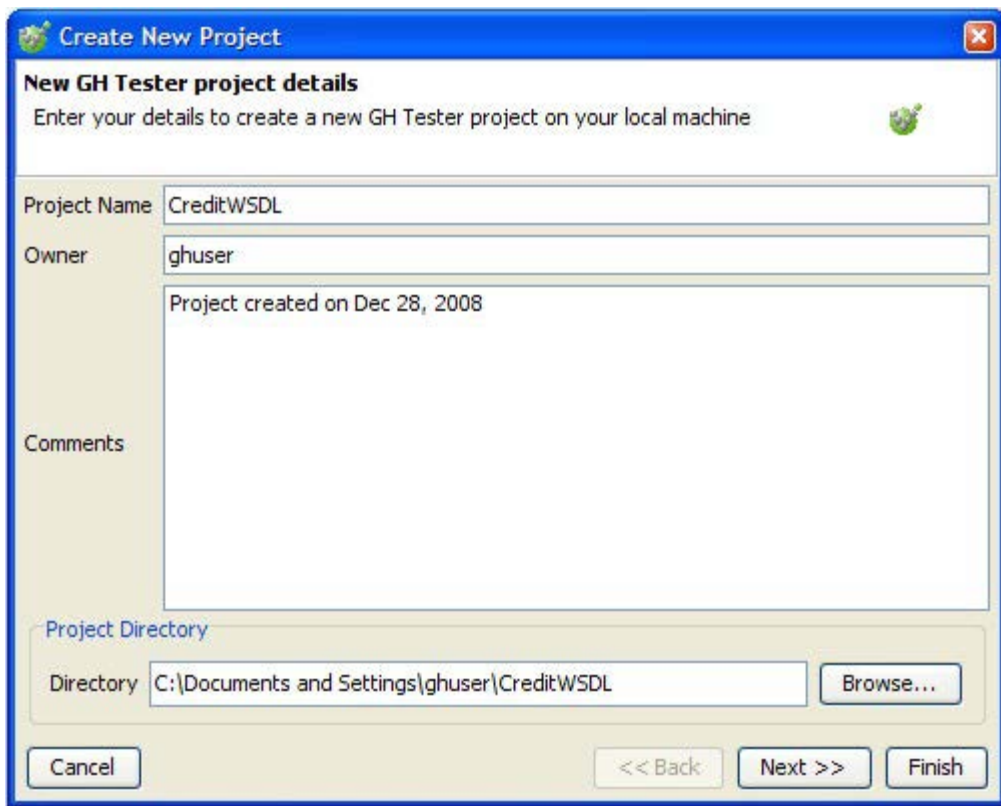
Creating a new project is simplified through the use of a wizard, which is launched when you choose the **New Project** option in the application.

1. Launch Rational Integration Tester (see [Starting the Program](#)).
2. Select **New Project** in the Rational Integration Tester welcome screen.



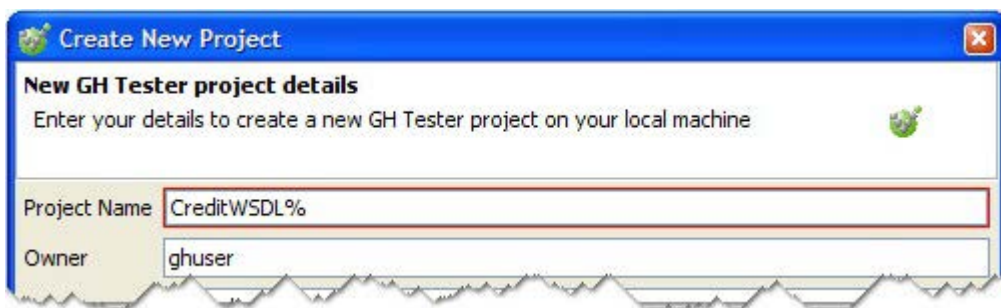
3. Click **OK**.

The **Create New Project** wizard is displayed.



4. Enter a name for the project in the **Project Name** field (the project directory changes according to the project's name). The name should be specific enough to identify the project's intended purpose.

NOTE: If any of the following characters are used in the project name, the field is highlighted in red and you can not continue with the wizard: @ \$ % ^ & * + =



-
5. If desired, modify the **Owner** and **Comments** fields (these are saved with the project and can be modified later), as well as the project **Directory** (enter the full path to the directory where the project should be created or click **Browse** to locate the directory).

NOTE: The selected directory can not contain an existing Rational Integration Tester project.

6. When you are satisfied with the project details, click **Next**.

The **Project Database** dialog is displayed, which lets you configure the connection to the repository that stores all of the test data collected by Rational Integration Tester.

Create New Project

Project Database
Configure the project database for the performance testing components.

Database Provider: MySQL

Connection Details

Database URL: jdbc:mysql://localhost:3306/<database name>

User Name:

Password:

Test Connection

Database Details

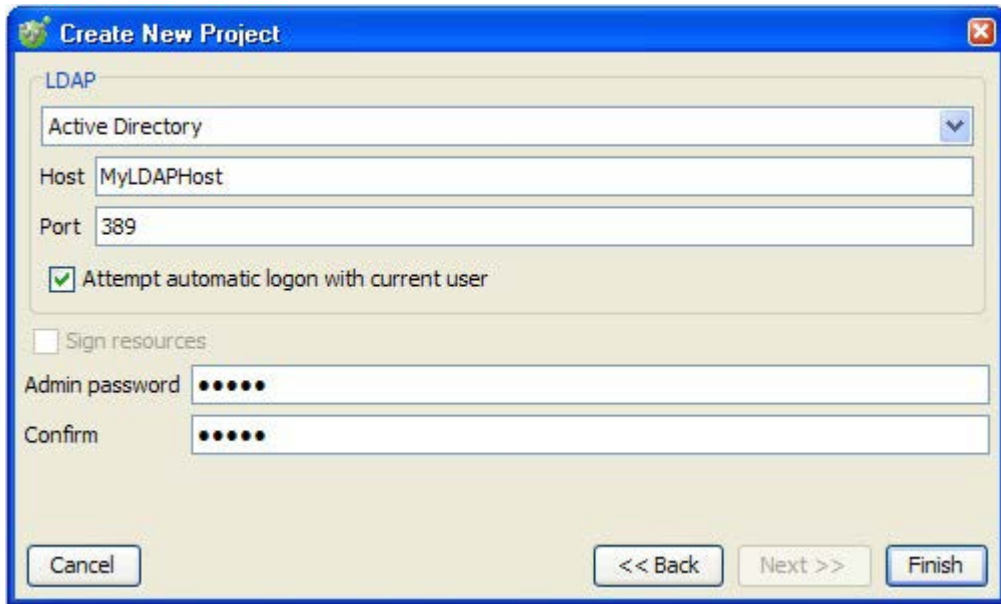
Property	Value

Cancel << Back Next >> Finish

NOTE: If desired, you can click **Finish** to skip the database configuration until later. However, a valid database and working connection are required to store or view any historical results in Rational Integration Tester.

7. Enter the database connection details (**Database URL**, **User Name**, and **Password**) and click **Test Connection** to verify them.

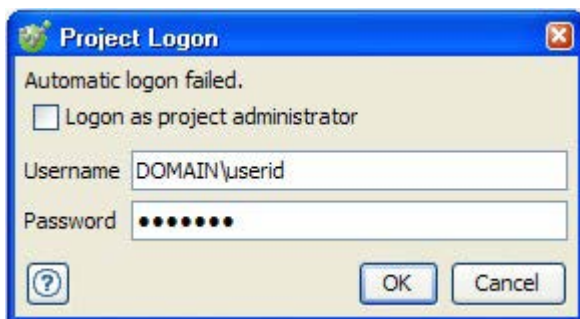
-
8. To enable project permissions, select an LDAP provider and enter its host name and port, then set the password of the project administrator. For information about this, refer to *IBM Rational Integration Tester Installation Guide*.



The "Create New Project" dialog box is shown. It has a blue title bar with the text "Create New Project" and a close button. The main area is light yellow. At the top, there's a section labeled "LDAP" in blue. Below it, a dropdown menu shows "Active Directory". Below that, there are text boxes for "Host" (containing "MyLDAPHost") and "Port" (containing "389"). A checkbox labeled "Attempt automatic logon with current user" is checked. Below that, there's an unchecked checkbox labeled "Sign resources". At the bottom, there are two text boxes for "Admin password" and "Confirm", both containing five dots. At the very bottom, there are four buttons: "Cancel", "<< Back", "Next >>", and "Finish".

NOTE: The password for the project administrator should be set up at this time as the administrator is the only user that can modify project permissions or add/remove users and groups for the project.

9. When you are satisfied with the project details, click **Finish** to open the new project in Rational Integration Tester.
10. If the project was configured to use LDAP for permissions, you must log into the project as a valid domain user or as the project administrator.



The "Project Logon" dialog box is shown. It has a blue title bar with the text "Project Logon" and a close button. The main area is light yellow. At the top, it says "Automatic logon failed." Below that, there's an unchecked checkbox labeled "Logon as project administrator". Below that, there are text boxes for "Username" (containing "DOMAIN\userid") and "Password" (containing five dots). At the bottom left, there's a question mark icon in a square. At the bottom right, there are two buttons: "OK" and "Cancel".

11. Enter a valid domain user/password combination, or enable the "Logon as project administrator" option and enter the admin password, then click **OK**.

9.5 Cloning a Project

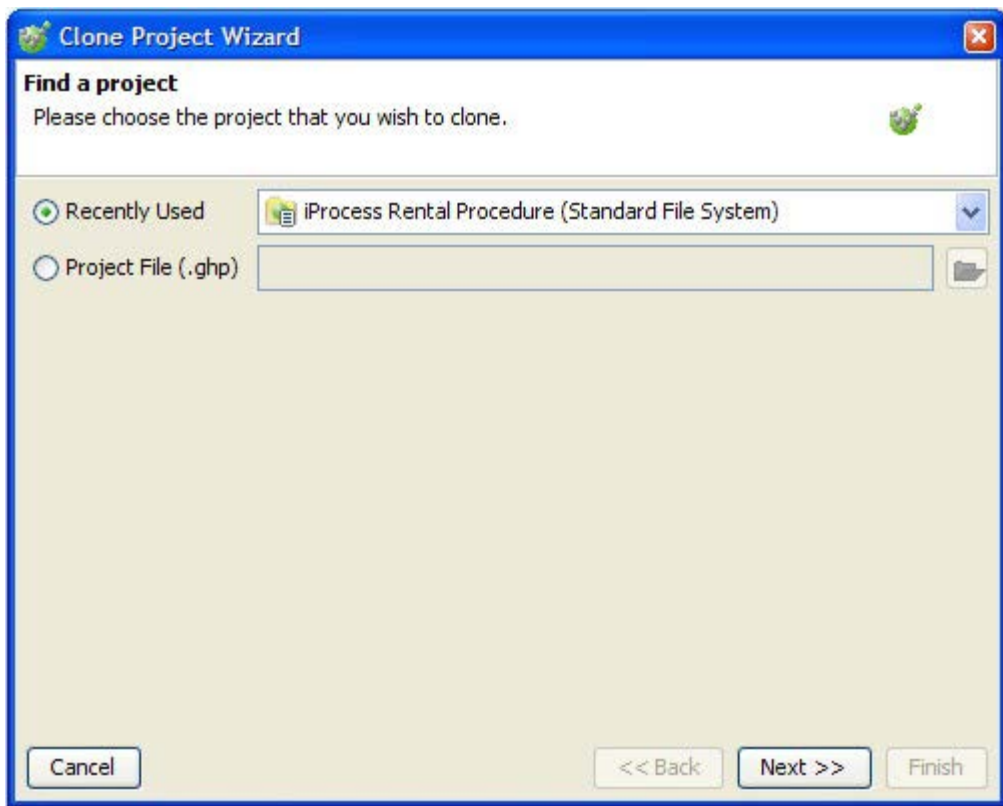
“Cloning” creates a copy of an existing Rational Integration Tester project, letting you work with a new project that already contains the resources that you want.

1. Launch Rational Integration Tester (see [Starting the Program](#)).
2. Select **Clone Project** in the Rational Integration Tester welcome screen.



3. Click **OK**.

The **Clone Project** wizard is displayed.



4. Select **Recently Used** to clone a recently used project, or select **Project File** to browse for the project (.ghp file) you want to clone.
5. After selecting the desired option and project, click **Next**.

The project details dialog from the **Create New Project** wizard is displayed.

The screenshot shows a Windows-style dialog box titled "Create New Project" with a close button (X) in the top right corner. The main heading is "New GH Tester project details" with a subtitle "Enter your details to create a new GH Tester project on your local machine". The dialog contains several input fields: "Project Name" with the value "CreditWSDL", "Owner" with the value "ghuser", and a "Comments" text area containing "Project created on Dec 28, 2008". Below these is a "Project Directory" section with a "Directory" field containing "C:\Documents and Settings\ghuser\CreditWSDL" and a "Browse..." button. At the bottom are four buttons: "Cancel", "<< Back", "Next >>", and "Finish".

6. Enter a name for the cloned project in the **Project Name** field (the project directory changes according to the project's name). The name should be specific enough to identify the project's intended purpose.

NOTE: If any of the following characters are used in the project name, the field is highlighted in red and you can not continue with the wizard: @ \$ % ^ & * + =

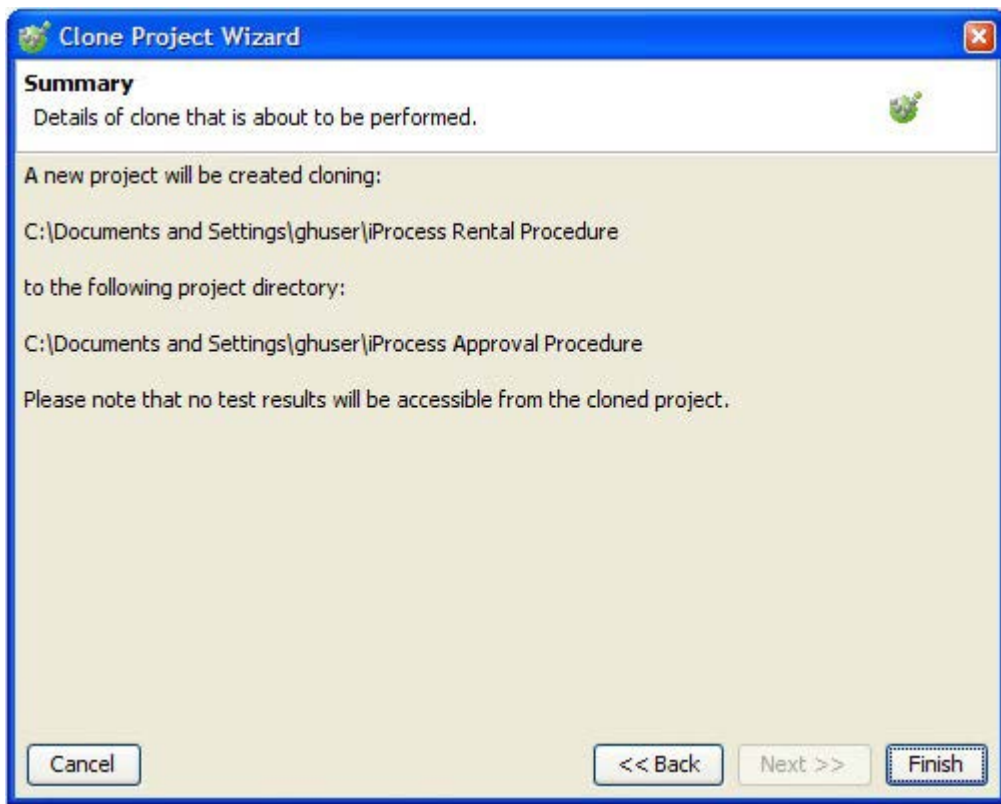
This screenshot shows the same "Create New Project" dialog box, but the "Project Name" field now contains "CreditWSDL%". The text "CreditWSDL%" is highlighted with a red border, indicating it is an invalid input. The "Owner" field still contains "ghuser". The "Project Directory" section and the bottom buttons are also visible.

-
7. If desired, modify the **Owner** and **Comments** fields (these are saved with the project and can be modified later), as well as the project **Directory** (enter the full path to the directory where the project should be created or click **Browse** to locate the directory).

NOTE: The selected directory can not contain an existing Rational Integration Tester project.

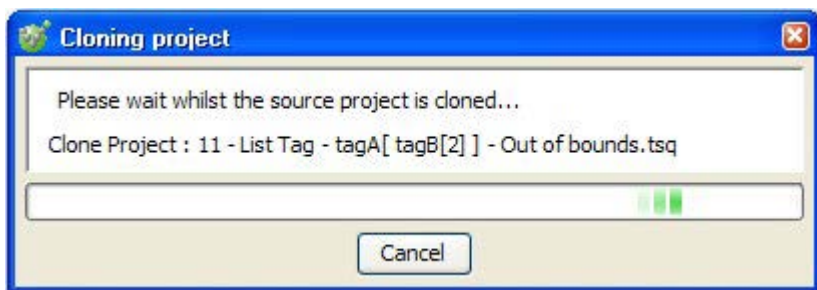
8. When you are satisfied with the project details, click **Next**.

A summary of the proposed action is displayed.



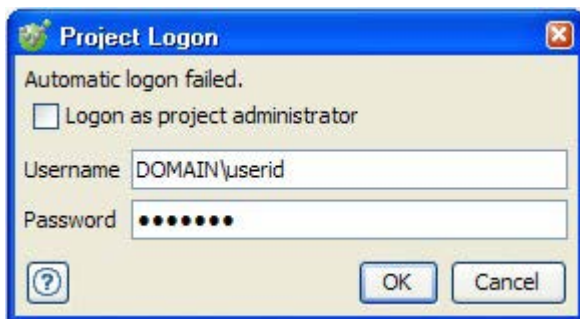
9. If the summary is correct, click **Finish** to close the wizard and open the new (cloned) project in Rational Integration Tester. Otherwise, click **Back** to make changes or click **Cancel** to close the wizard without cloning any projects.

-
10. Depending on the number of artefacts in the source project, a progress dialog will be displayed while the project is being cloned.



Once the project has been cloned/created, it will be loaded in Rational Integration Tester.

11. If the project was configured to use LDAP for permissions, you must log into the project as a valid domain user or as the project administrator.



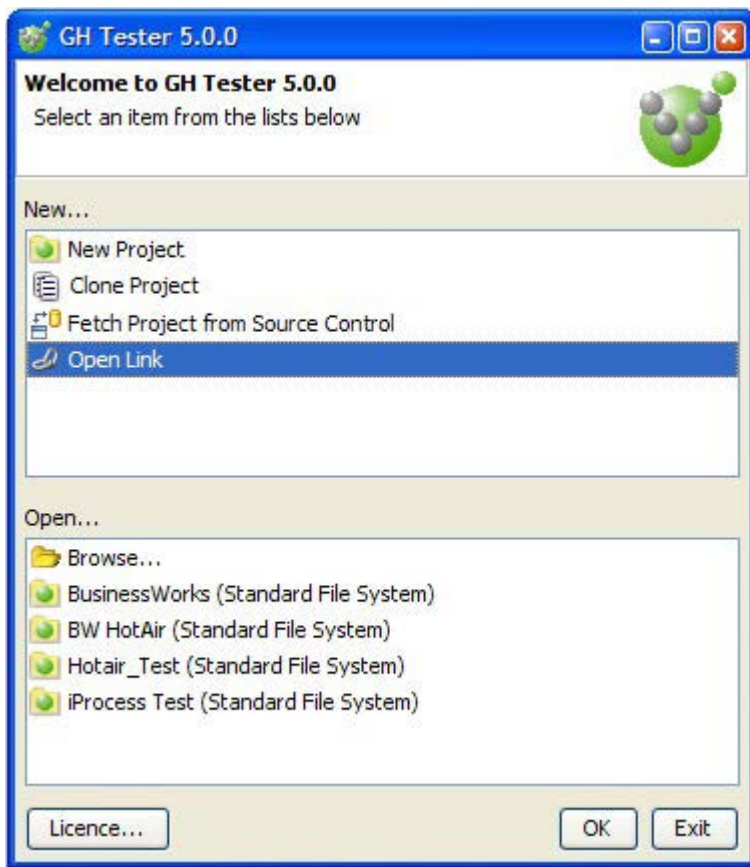
12. Enter a valid domain user/password combination, or enable the “Logon as project administrator” option and enter the admin password, then click **OK**.

9.6 Fetching a Project from Source Control

If an existing Rational Integration Tester project has been shared using the Source Control Management application, it can be checked out and opened like any other project.

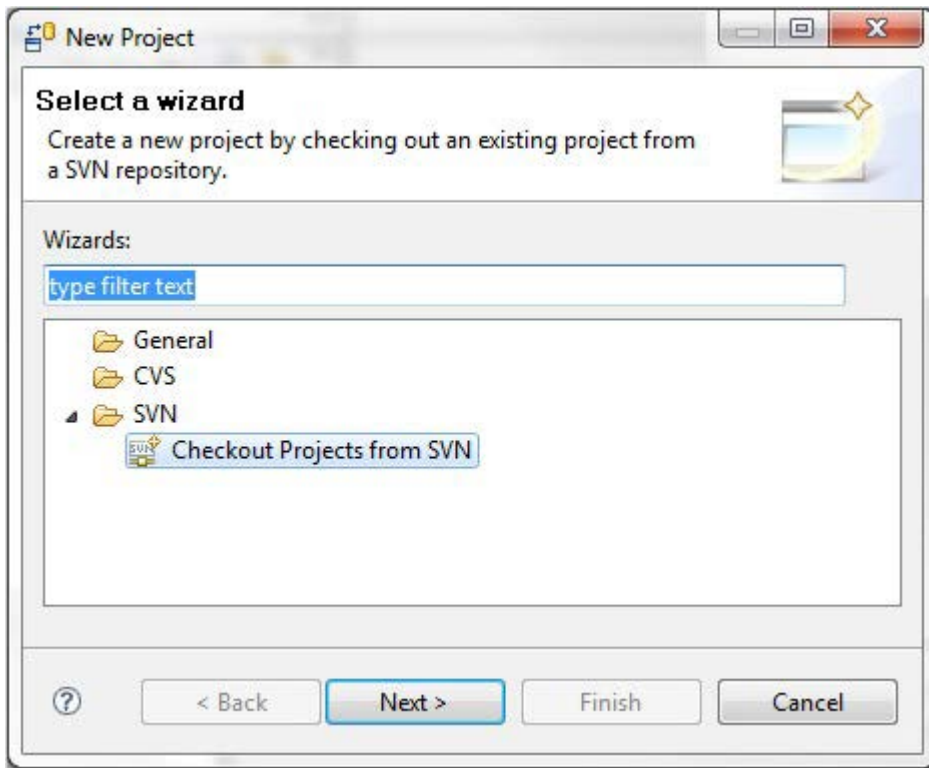
NOTE: Before you can fetch a project, source control and the desired team provider must be configured on the Rational Integration Tester client in use. See the *IBM Rational Integration Tester SCM Application Guide* for more information.

1. Launch Rational Integration Tester (see [Starting the Program](#)).
2. Select **Fetch Project from Source Control** in the Rational Integration Tester welcome screen.



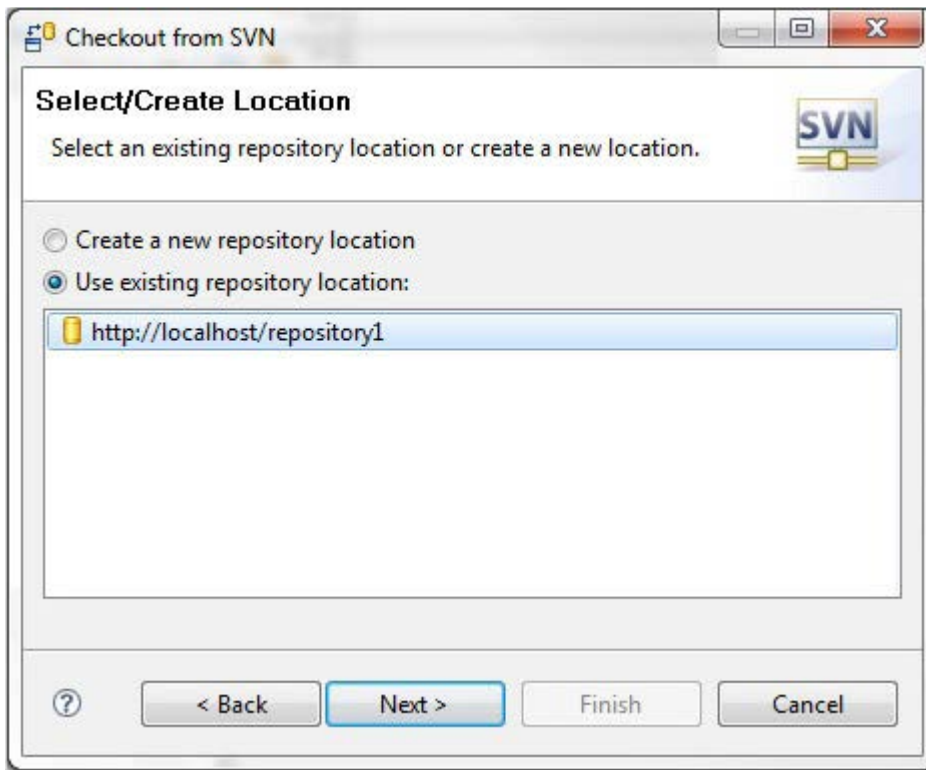
3. Click **OK**.

The SCM application (**Tools.exe**) is launched and the **New Project** dialog is displayed.



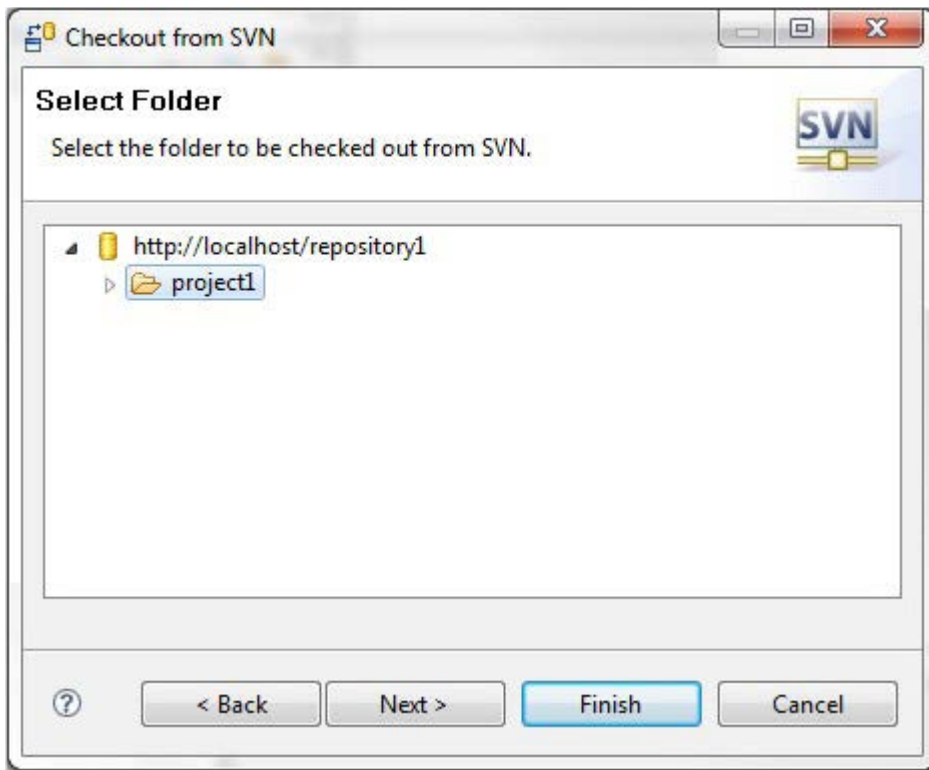
4. Select the **Checkout Projects...** option under the team provider that was used to share the desired project, then click **Next**.

The **Select/Create Location** dialog is displayed.



5. Select the **Use existing repository location** option and select the desired repository, then click **Next**.

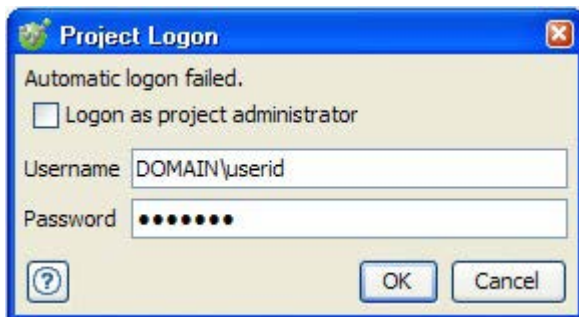
The **Select Folder** dialog is displayed.



6. From the repository selected previously, select the desired Rational Integration Tester project folder and click **Finish**.

The SCM application will check out the selected project and it will be opened in Rational Integration Tester.

7. If the project was configured to use LDAP for permissions, you must log into the project as a valid domain user or as the project administrator.

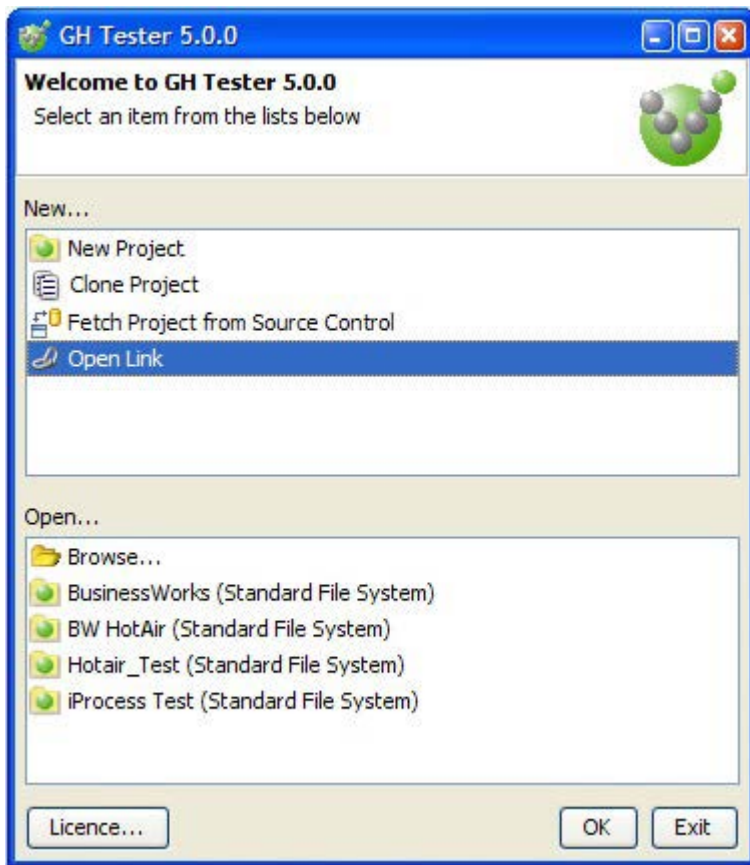


-
8. Enter a valid domain user/password combination, or enable the “Logon as project administrator” option and enter the admin password, then click **OK**.

9.7 Open a Resource Execution URL

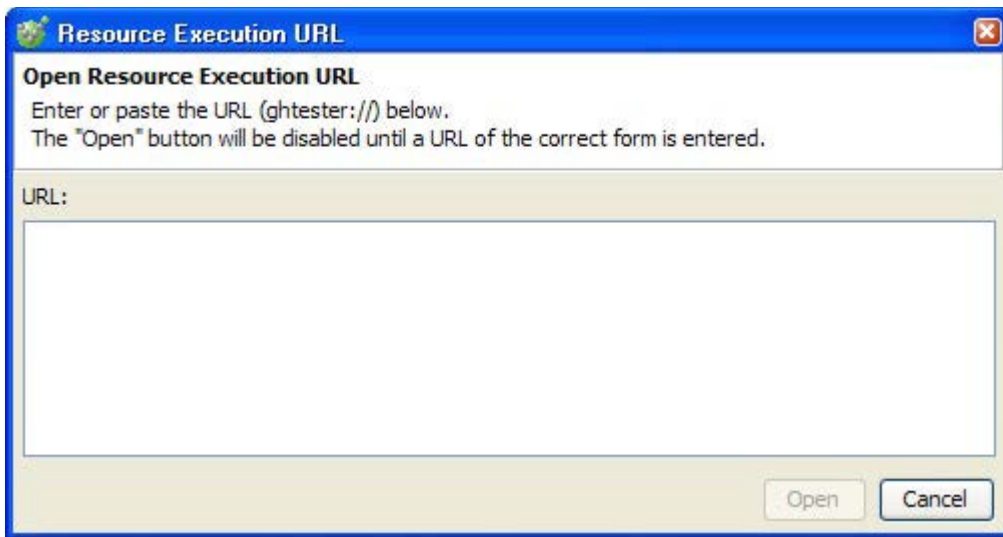
A resource execution URL is a direct link to the execution results of a test suite or a test within a suite. The URL can be copied from the asset's entry in the Test Lab's Task Monitor, and it can be opened later to navigate within Rational Integration Tester to the Results Gallery report for the execution of the selected resource.

1. Launch Rational Integration Tester (see [Starting the Program](#)).
2. Select **Open Link** in the Rational Integration Tester welcome screen.



3. Click **OK**.

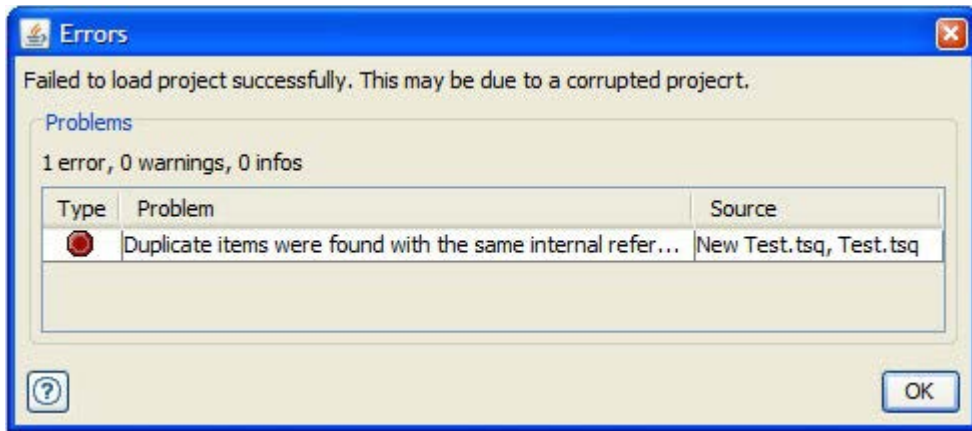
The **Resource Execution URL** dialog is displayed.



4. Paste a valid resource execution URL into the **URL** field and click **Open** when enabled – the **Open** button is only enabled after a valid URL has been entered.
5. If permissions are enabled on the project, log in when prompted.
6. The Results Gallery will be opened and the execution report specified by the selected URL will be displayed.

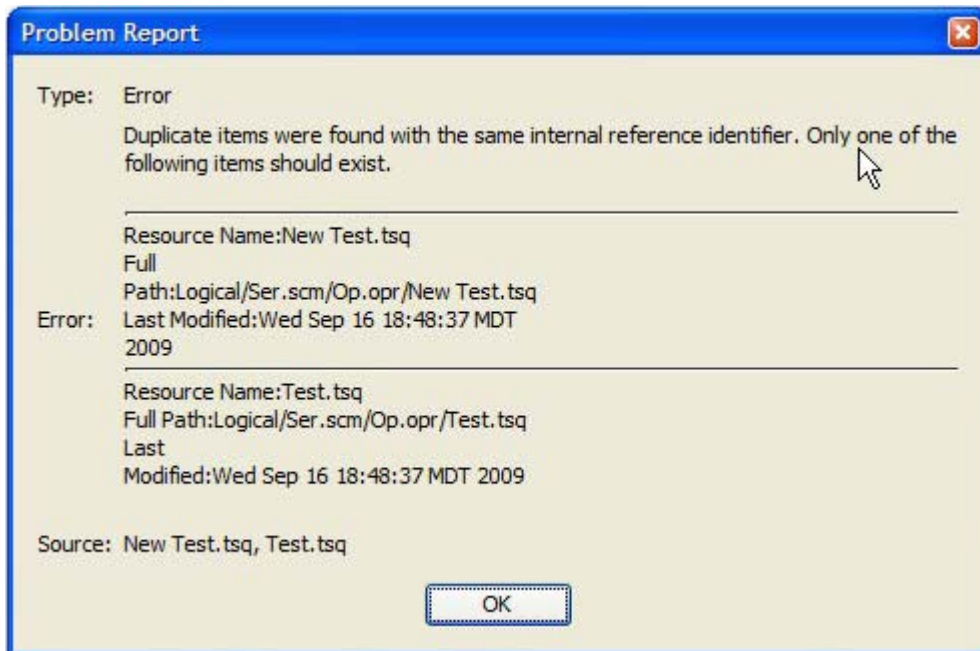
9.8 Project Errors

If any errors are encountered when opening a project, they will be displayed in the **Errors** dialog, as shown below:



A description of the error is displayed under **Problem**, and the source of the error is displayed under **Source**. For more details about an error, double-click its entry.

The **Problem Report** dialog is displayed, providing additional details.



This page intentionally left blank for two-sided printing.

Depending on the error that was encountered, the project may be lost or it may be repaired manually. In the example shown above, one of the duplicate tests must be

deleted manually from the project before it can be opened in Rational Integration Tester.

Transports and Formatters

Contents

Transports

Formatters

This chapter provides information about messaging transports and formatters, including how to create and select them in Rational Integration Tester.

10.1 Transports

Rational Integration Tester employs a decoupled, plug-in architecture to provide maximum flexibility with regard to the messaging transport software in use by a system under test. A messaging transport provides the information the Rational Integration Tester needs to communicate with specific 3rd party systems (for example, TIBCO EMS, WebSphere MQ, and so on).

The program code for each transport is contained in one or more JAR files (libraries), which are stored in Rational Integration Tester's "plugins" directory. Any libraries upon which these plugins depend must be accessible to Rational Integration Tester. For example, if Rational Integration Tester needs to work with TIBCO EMS messages, the EMS JAR files must be installed somewhere on the same machine as Rational Integration Tester.

The Library Manager is used to configure these 3rd party libraries. Please see *IBM Rational Integration Tester Installation Guide* for information about using Library Manager.

10.1.1 Creating a Transport

Transports are created in Rational Integration Tester when the corresponding logical resources (for example, an HTTP connection, a TIBCO EMS Domain, and so on) are created in Architecture School's Logical View). The logical resources are then bound to testable, physical resources (created in Architecture School's Physical View) using one or more environments.

NOTE: See [Architecture School](#) for more information about creating logical and physical resources.

Transports are configured in the physical resources that represent them. For example, the properties of an HTTP transport are determined by the web server configuration to which the transport's logical resource is bound.

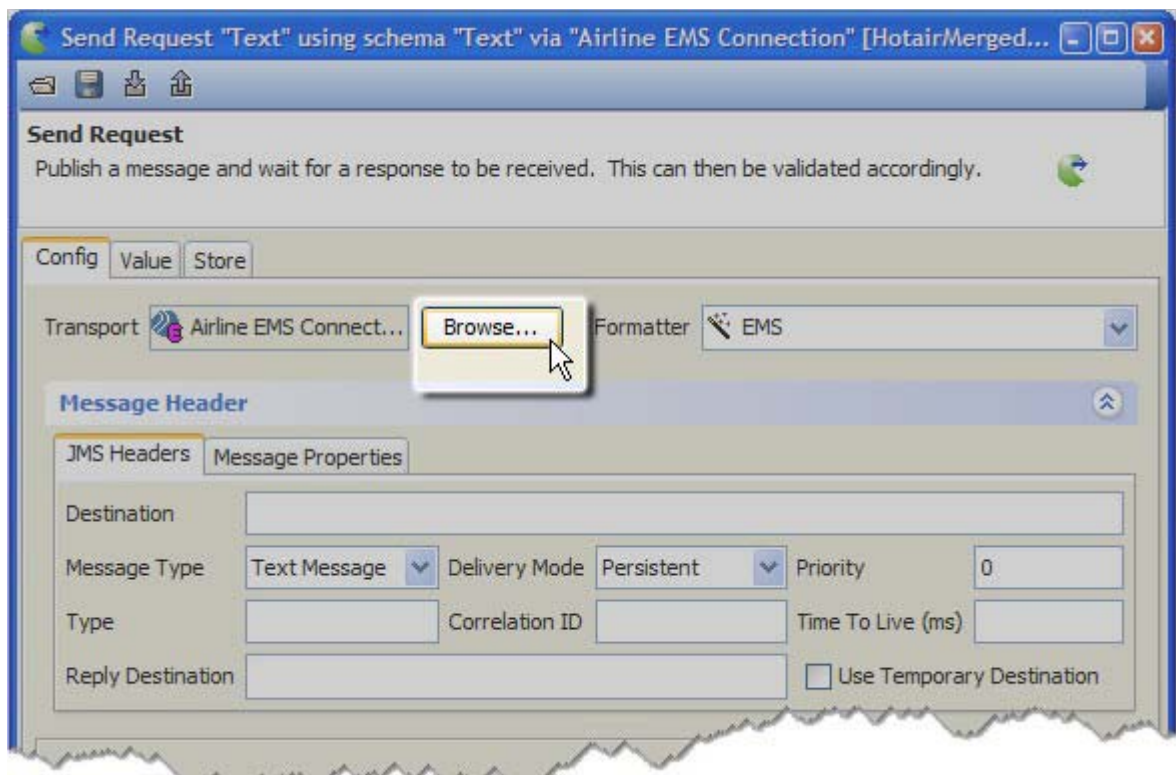
NOTE: The File transport is an exception, as you can not create a physical resource to which the logical File Contents resource is bound.

10.1.2 Selecting a Transport

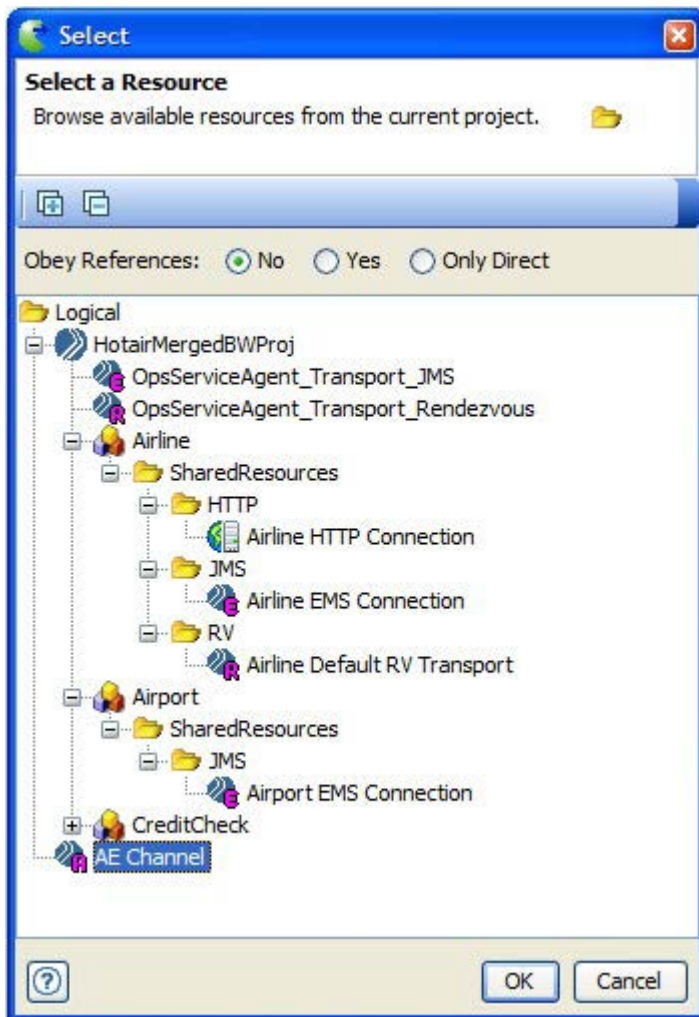
Transports determine the structure and content of the messages that use them. The details about configuring message headers and bodies for specific transports can be found in the reference guide for each transport.

To select a transport in messaging actions (for example, Publish, Subscribe, and so on), follow the steps below:

1. After opening the messaging action, click **Browse** next to the **Transport** field.



-
2. In the **Select a Resource** dialog, you can select any of the transports that have been created in the project (that is, by having created resources in Architecture School).



3. Next to **Obey References**, select the desired option for locating a transport, as follows:
- If you select **No**, you can choose any transport in the project.
 - If you select **Yes**, you can choose only transports that are referenced (directly or indirectly) by the operation containing the test in which you are editing the messaging action.
 - If you select **Only Direct**, you can choose only transports that are referenced directly by the operation containing the test in which you are editing the messaging action.

-
4. After selecting the desired transport, click **OK** to return to the message editor.

10.2 Formatters

A formatter translates messages from the native transport format (for example, TIBCO Rendezvous) to the internal format understood by Rational Integration Tester, known as A3 (Accelerated Adaptor Architecture).

For some formats, the specific message type (for example, Text, Map, Bytes, and so on) is selected within the message body. In these cases, it is the Message Type field – in conjunction with the formatter – that determines the content/structure of the message body.

- [Available Formatters](#)
- [Selecting a Formatter](#)

10.2.1 Available Formatters

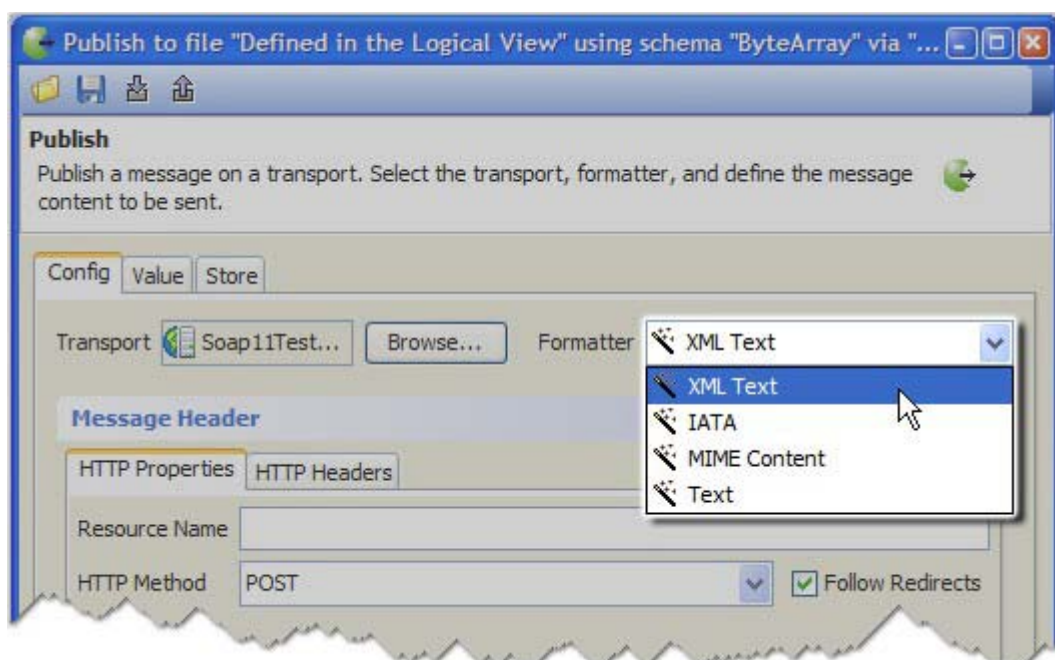
The following table lists the formatters that are available for each messaging transport in Rational Integration Tester:

Transport	Formatters
File	Byte Array, IATA, Text
HTTP Connection	HTTP Message, Text, IATA, MIME Content
JMS	JMS (message type configured in body)
Sonic	Sonic (message type configured in body)
TCP/UDP	Byte Array
TIBCO EMS Domain	EMS (message type configured in body)
TIBCO Rendezvous Bus	TIBCO Rendezvous, TIBCO BusinessWorks Log
WebSphere MQ Broker	Text, Byte Array, XML Text, IATA
webMethods Broker Domain	webMethods Broker

10.2.2 Selecting a Formatter

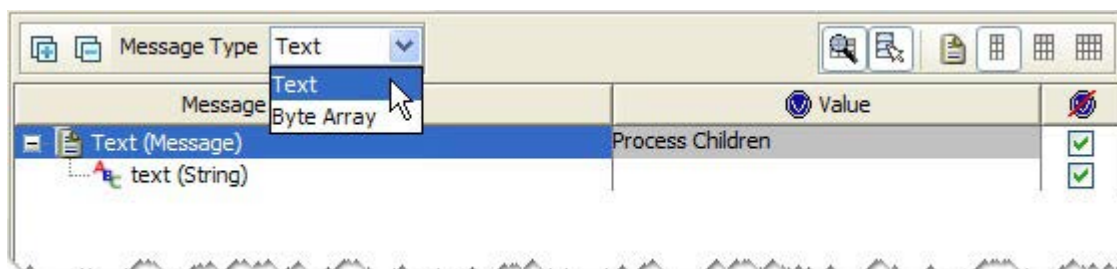
Formatters and message types determine the structure and content of the message body, specifically the elements that the body contains. To select a formatter in messaging actions (for example, Publish, Subscribe, and so on), follow the steps below:

1. Open the messaging action and select the desired transport as described in [Selecting a Transport](#).
2. Select one of the available formatters from the **Formatter** dropdown menu.



NOTE: As you select different formatters, the content of the message body changes.

3. If available (depending on the selected transport and formatter), select the type of message from the **Message Type** dropdown menu, above the message body.



10.3 Dynamic Formatters

JMS-based and HTTP transports utilize dynamic formatters. For JMS-based transports, only one formatter is available at the top level, but the specific message type can be set within the message body. For HTTP transports, the **HTTP Message** formatter enables the use of **Byte Array** or **Text** message types to be selected within the body.

For both types, the selection of these top level formatters enables Rational Integration Tester to select the appropriate message type (for example, in Recording Studio) based on the content of the incoming message.

Messages

Contents

Overview

Selecting a Transport and Formatter

Constructing Message Headers

Constructing Message Bodies

Repeating Elements

MIME, DIME, and Multipart Content

Using Attachments

Applying Specific Formats to a Message

Apply Integra Message Handlers

XML Message Properties

The Field Editor

This section describes how to construct and manipulate messages that Rational Integration Tester can publish and subscribe to as part of a test.

This information is useful in other parts of the application, as there are many parts of Rational Integration Tester that contain generalized message editors.

11.1 Overview

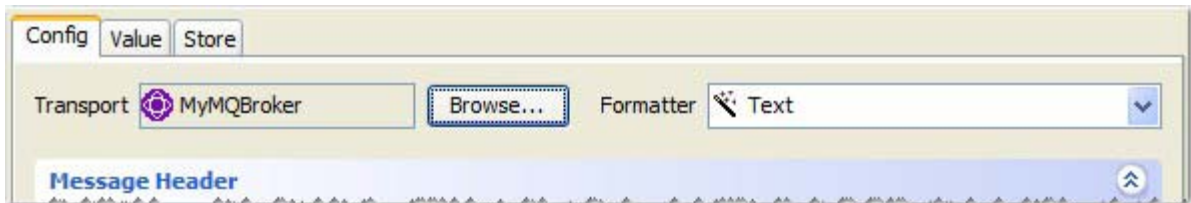
A message normally consists of two parts: a header and a body. The header usually adheres to a predefined logical construction, while the structure of the body can be more arbitrary.

This chapter provides information about the different message parts, message fields, and how messages are compared and processed.

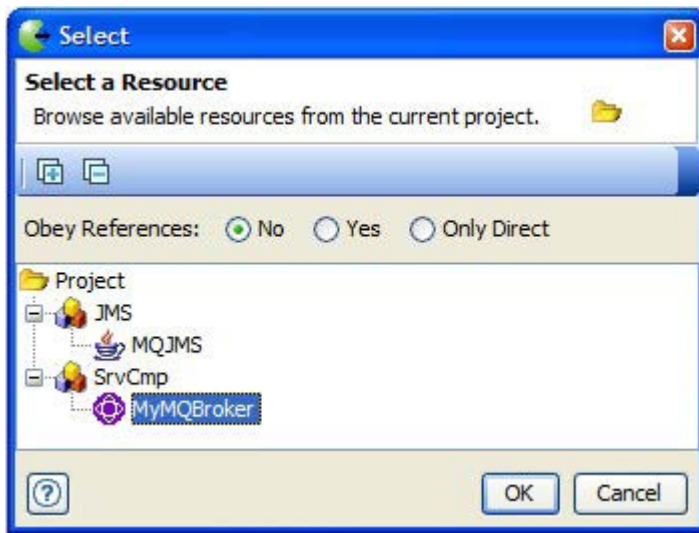
- [Selecting a Transport and Formatter](#)
- [Constructing Message Headers](#)
- [Constructing Message Bodies](#)
- [Repeating Elements](#)
- [MIME, DIME, and Multipart Content](#)
- [Using Attachments](#)
- [Applying Specific Formats to a Message](#)
- [Apply Integra Message Handlers](#)
- [XML Message Properties](#)
- [The Field Editor](#)

11.2 Selecting a Transport and Formatter

The first step in building a message is to select a transport and formatter, available at the top of the message **Config** tab.



1. Click **Browse** to select a transport in the **Select a Resource** dialog.



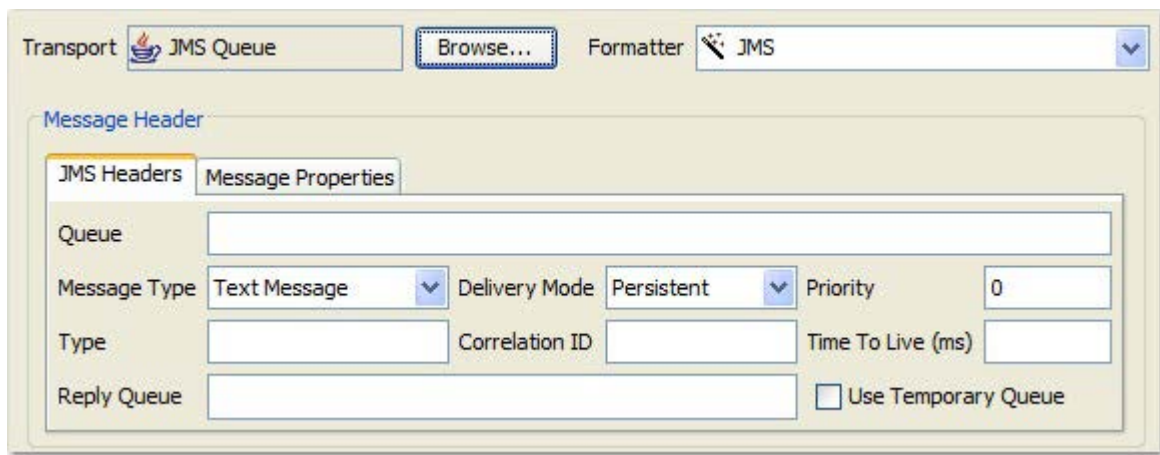
2. Select the option for obeying references:
 - If you select **No**, you can select any transport in the project.
 - If you select **Yes**, you can select only transports that are referenced (directly or indirectly) by the operation containing the test.
 - If you select **Only Direct**, you can select only transports that are referenced directly by the operation containing the test.
3. Locate and select the desired transport in the resource tree.
4. Click **OK** when finished.
5. Select the format of the message from the **Formatter** menu.

11.3 Constructing Message Headers

Message headers contain metadata about the message itself (for example, sender and destination details, message properties, message type, and so on). Headers for different transports will contain different types of information.

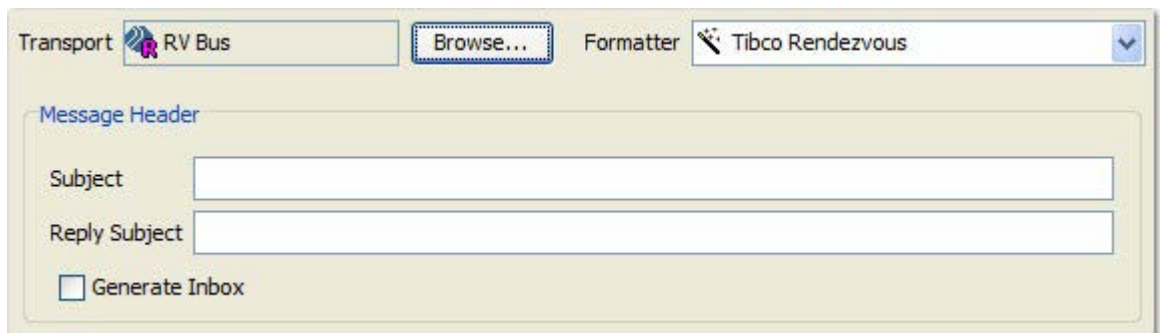
The images below illustrate a few different transport configuration dialogs, each containing different message header fields.

JMS



The JMS Queue configuration dialog shows the 'Transport' set to 'JMS Queue' and the 'Formatter' set to 'JMS'. The 'Message Header' section has two tabs: 'JMS Headers' (selected) and 'Message Properties'. Under 'JMS Headers', there are fields for 'Queue', 'Message Type' (set to 'Text Message'), 'Delivery Mode' (set to 'Persistent'), 'Priority' (set to '0'), 'Type', 'Correlation ID', 'Time To Live (ms)', and 'Reply Queue'. There is also a checkbox for 'Use Temporary Queue'.

TIBCO Rendezvous

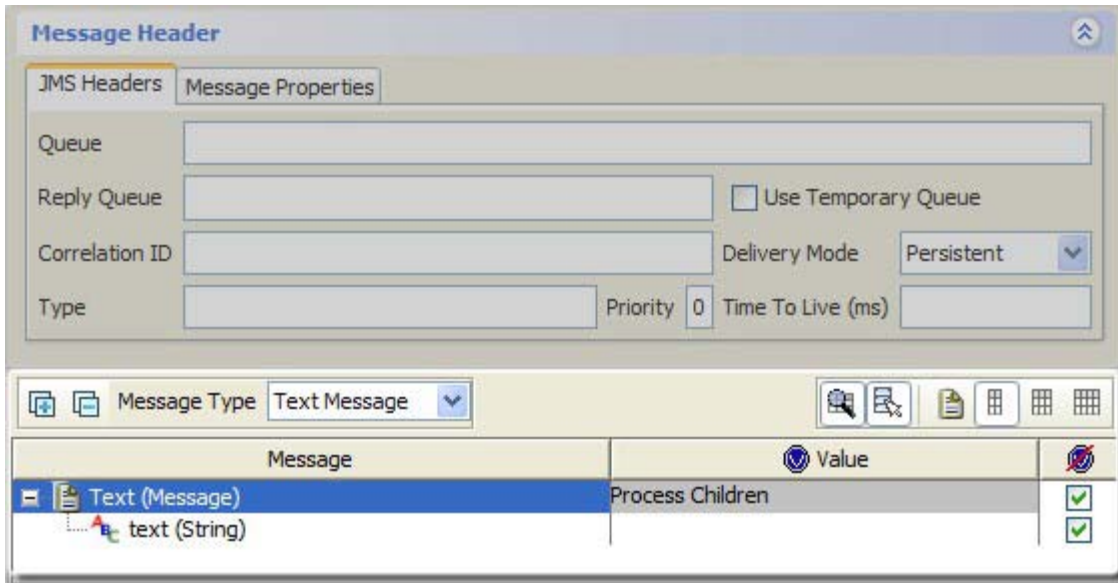


The Tibco Rendezvous configuration dialog shows the 'Transport' set to 'RV Bus' and the 'Formatter' set to 'Tibco Rendezvous'. The 'Message Header' section has two tabs: 'Message Properties' (selected) and 'JMS Headers'. Under 'Message Properties', there are fields for 'Subject' and 'Reply Subject', and a checkbox for 'Generate Inbox'.

Details about configuring protocol specific transports/headers can be found in the various Rational Integration Tester plugin guides.

11.4 Constructing Message Bodies

The message body is constructed below the message header in publishers and subscribers.




This section provides information about how to construct message bodies in Rational Integration Tester.

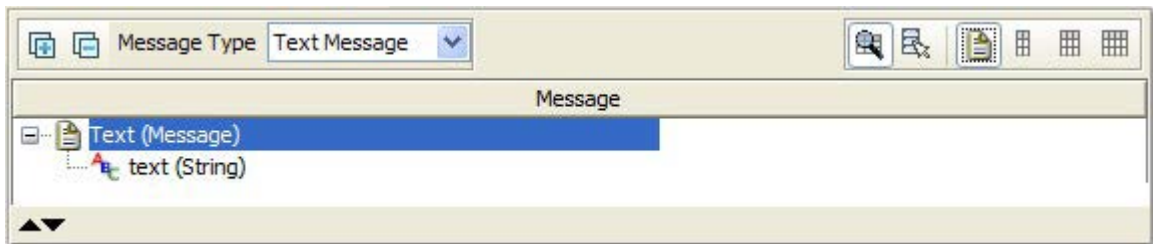
- [Publisher/Subscriber Views](#)
- [Import and Export Messages](#)
- [Quick Tags](#)
- [Message Types](#)
- [Validation Rules \(Comparison Ignores\)](#)
- [Message Bodies with Non-schema Transports](#)
- [Example: Publish a Simple Message Body](#)
- [Example: Simple Message Body for Subscription](#)
- [Converting from Flat XML to Structured XML](#)
- [Editing XML Message Structures](#)

11.4.1 Publisher/Subscriber Views

The structure of a message is defined when using the Publish/Subscribe actions (see [Appendix A: Test Actions](#)). In each action type, different views of the message can be selected (that is, for displaying information relating to the storage and/or validation of the message).

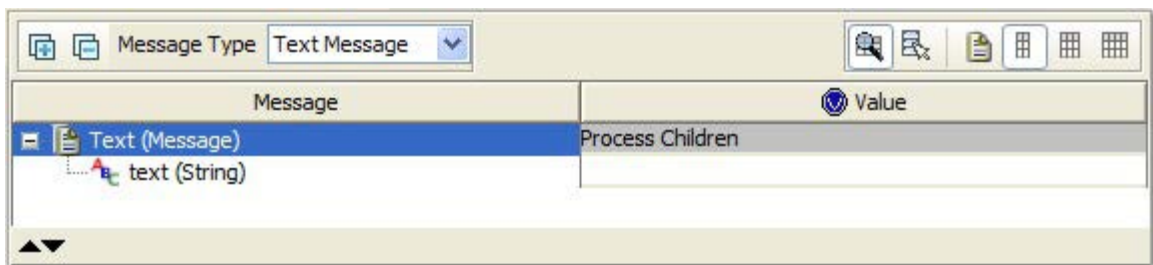
The view is controlled by the view selector panel .

In the example shown below, the Schema View  has been selected:



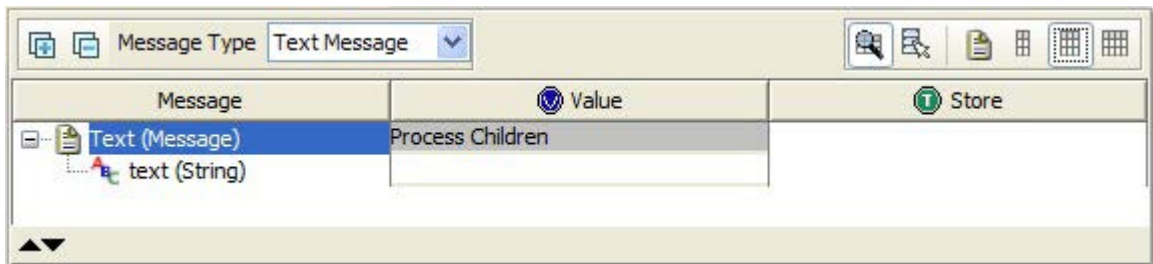
In this view, only the message structure (element/field names) is shown under the **Message** heading.

In the example shown below, the Simple View  has been selected:



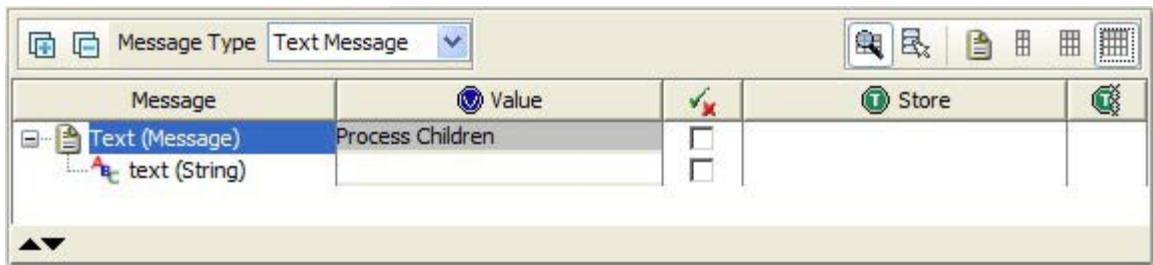
The message structure (element/field names) is shown under the **Message** heading and the content of each element/field is shown under the **Value** heading.



In the example shown below, the Simple Editing View  has been selected for the same Publisher as above:



In the Simple Editing view, the tag in which the element/field value is being stored is shown under the **Store** heading. In the previous example, the **Store** field is blank as the **Value** field is not being stored in a tag.



In the example shown below, the Advanced View  has been selected for the same Publisher as above:



In the Advanced View, the Enable Validation  and Enable Store Actions  checkboxes are displayed.

The Enable Validation check box toggles between full validation (Equality, Name and Type) of the selected element/field and no validation at all. The validation can also be set using the Field Editor (see [The Field Editor](#)).

The Enable Store Actions check box toggles between enabling and disabling all the tag storage actions for the corresponding element/field. The tag storage configuration can also be set using the Field Editor (see [The Field Editor](#)).

The Show/Hide Action Pane  and Show/Hide Enable States  buttons can be activated separately or at the same time, and with any of the other message body

views. In the example below, the same Publisher is shown in the Advanced View with the Enable States being shown:

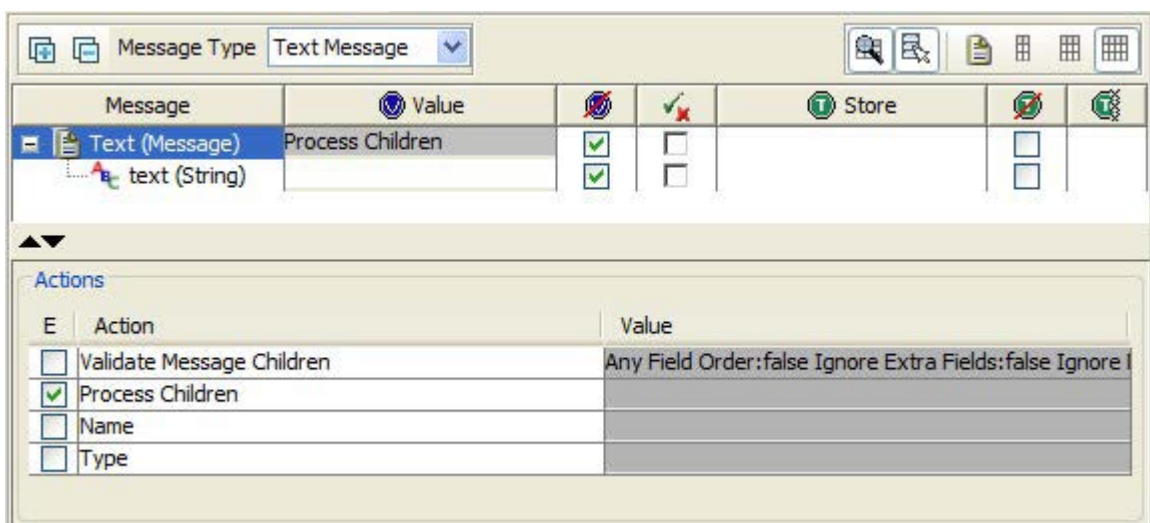


When Enable States are shown, the Enabled Value and Enable Store Action checkboxes are displayed.

The Enabled Value check box toggles between enabling and disabling the value specified for the content of a particular element/field. The value configuration can also be set using the Field Editor (see [The Field Editor](#)).

The Enable Store Action check box is similar to the Enable Store Actions check box, except that this check box only takes effect for a single tag storage action for the corresponding element/field. If there is more than one tag storage action, then this check box is disabled.



In the example below, the same Publisher as above is shown with the Actions pane shown:



The Actions pane shows the detailed validation and tag storage configuration for the selected element/field. This provides another method of altering the configuration, in

addition to the check boxes shown above and the Field Editor (see [The Field Editor](#)).

11.4.2 Import and Export Messages

At the top of all message editors are the import  and export  icons.



To export the current message, click the export icon, select a name, type, and location for the file, then click **Export**. To import a previously saved or exported message, click the import icon, select the desired message type (message or publisher), locate and select the desired message (*.ghm or *.gtp file), then click **Import**.

11.4.3 Quick Tags

A fast way to insert tags into fields is to use quick tags, available by right-clicking a message field and selecting the **Contents > Quick Tag** or **Contents > Quick Tag using Path** option.



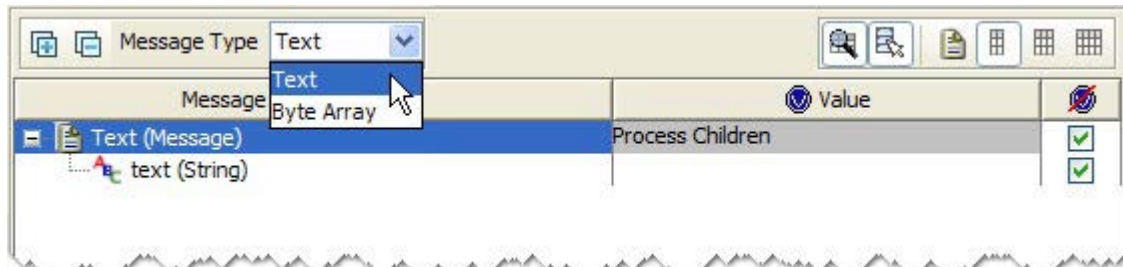
The new tag will obtain the value of the field from an automatically generated tag, except in the case of Subscribe actions (in this case, the tag will be used to store the value of the field in received messages).

If the selected field has a name, the quick tag will be given the same name. If not, the name of the parent field is used. If the parent field also has no name, the tag will be called *newTag*.

NOTE: Selecting the **Quick Tag using Path** option creates a tag named according to the path location (for example, “Order/Details/Id” instead of just “Id”). This prevents two tags from having the same name.

11.4.4 Message Types

Depending on the transport in use, the message type can be selected using the **Message Type** dropdown menu, above the message body.



The message type and content will change according to the format that is selected.

11.4.5 Validation Rules (Comparison Ignores)

Rules can be added to one or more message fields to define default actions to apply when processing similar fields. To add a rule, select the desired fields, right-click one of the fields, and select **Rule Cache > Add to Cache** from the context menu.



Any field actions can be copied as part of the rule. In the **Add to Rule Cache** dialog, only the actions that are currently enabled will be selected, as follows:

- **Value** – the Enable action box must be checked
- **Filter** – a filter action must be present and checked
- **Validate** – at least one validate action must be checked
- **Store** – a store action must be present and checked

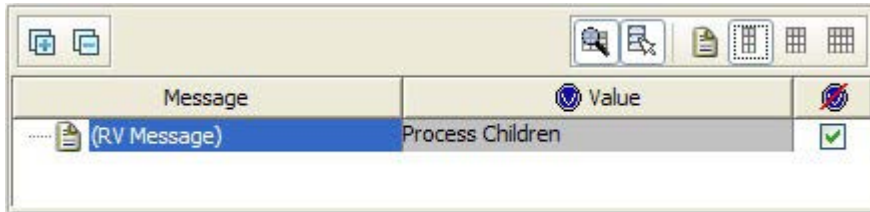
To create a rule for the message field that is based only on the structure of the message, enable the **All schemas** option. To create a rule for the message field that would only be applied when the selected schema is in use, select the **Just this schema** option. To add a comment to the rule (making it easier to identify in the Rule Cache view), enter it in the **Comment** field.

If a **single** selected field is already using a rule, you can open the rule for editing (in the Rule Cache view of the Architecture School perspective) by selecting **Edit Rule** from the context menu. Additionally, you can enable or disable the rule on selected fields by selecting **Enable** or **Disable** from the context menu.

NOTE: The context actions available depend on the state of the selected fields and rules. For example, if some fields use a rule but some do not, only the **Enable** and **Disable** options are available, and the **Add to Cache** option is only available if none of the selected fields are using a rule.

11.4.6 Message Bodies with Non-schema Transports

When using transports without a defined message schema (for example, TIBCO Rendezvous), the body of the message must be created manually.



Right-click on the message root to view the context-menu and begin constructing the message body. Using these commands, arbitrarily complex tree structures can be built. The menu for all objects in the message-tree is essentially the same.

- **Contents** provides access to field and XML actions, quickly creating a Tag action, and opening up the Field Editor for the selected part of the message. The field editor is a powerful tool enabling not only the type and value of the field to be edited, but also field validation and tag storage (see [The Field Editor](#)).
- **Schema** invokes the schema wizard for creating schema-based messages.
- **Root** selects the root of a message from a schema.
- **Add Child** (if applicable) adds a child to the end of the message tree.
- **Rule Cache** provides options for creating rules or managing existing rules.
- **Type** allows the field type to be reselected.
- **Move Up** and **Move Down** move the current element up or down, relative to the other elements, if possible.
- **Delete** deletes the selected object.
- **Cut**, **Copy**, and **Paste** work like they do in other applications (that is, cutting, copying, or pasting the selected item to/from the clipboard).
- **Expand All** and **Collapse All** will expand or collapse the selected message node.

NOTE: When expanding all nodes in large messages, Rational Integration Tester will only process the message for up to five seconds. After that time, all nodes that could be expanded will be expanded. If necessary, you can expand again to continue expanding any collapsed nodes.

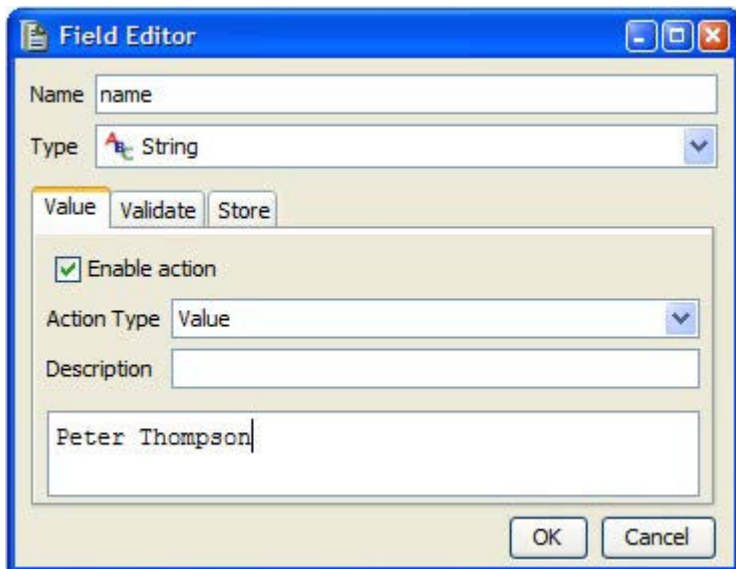
- **Add Attachment** lets users add a file attachment to the selected message node.

11.4.7 Example: Publish a Simple Message Body

In the following example, we will construct a simple message consisting of the following structure.

Field	Type	Value
name	(String)	Peter Thompson
age	(Integer)	42
position	(String)	CEO
otherData	(Message)	
libraryCardNo	(Integer)	765432463
creditRating	(Integer)	0
carRegistration	(String)	A227 TWR

1. Create a new publisher, as described in [Appendix A: Test Actions](#). Select a transport and formatter that allow the creation of arbitrary messages (for example, TIBCO Rendezvous). If you choose a JMS Topic or Queue, as a simple example select XML String as the formatter. The XML message editor is covered in [Editing XML Message Structures](#).
2. Right click on the root message node (RV Message, in this example) and select **Add Child < (String)**, and the Field Editor appears. In the **Name** field, enter “name.” In the value editor (the empty field below the description), enter “Peter Thompson”.



-
3. Click **OK** when finished and the structure of the message should appear as follows:

Message	Value
(RV Message)	Process Children
name (String)	Peter Thompson

4. Add two more child elements in the same way, one of type Integer and one of type String. For the Integer child, enter “age” as the name and “42” as the value. For the String element, enter “position” as the name and “CEO” as the value.

Message	Value
(RV Message)	Process Children
name (String)	Peter Thompson
age (Integer)	42
position (String)	CEO

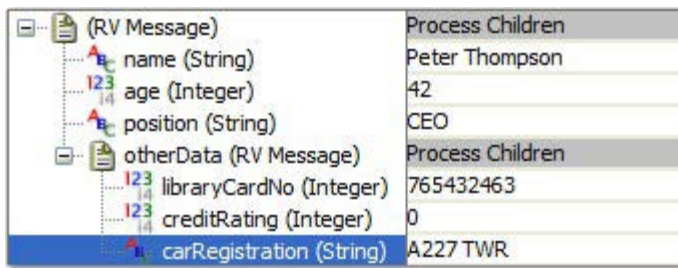
5. Add the **otherData** element, a subtree consisting of two Integer fields and a String. This can be achieved by using the type Message. Add a child of type Message (RV Message, in this case). In the field editor, enter “otherData” in the name field. The message structure should now appear as follows:

Message	Value
(RV Message)	Process Children
name (String)	Peter Thompson
age (Integer)	42
position (String)	CEO
otherData (RV Message)	Process Children

Since the **otherData** element is of type Message, the **Add Child** option is enabled in its context menu. Add the following three child elements to **otherData**:

- libraryCardNo, an Integer with a value of 765432463
- creditRating, an Integer with a value of 0
- carRegistration, a String with a value of A227 TWR

The message structure is now complete and can be published, saved, and so on



The screenshot shows a message structure editor. On the left is a tree view of the message structure, and on the right is a table of values for each field. The tree view shows a root node '(RV Message)' with children: 'name (String)', 'age (Integer)', 'position (String)', 'otherData (RV Message)', and 'carRegistration (String)'. The 'otherData (RV Message)' node has children: 'libraryCardNo (Integer)', 'creditRating (Integer)', and 'carRegistration (String)'. The table on the right lists the values for each field: 'Process Children' for 'name', 'Peter Thompson' for 'age', '42' for 'position', 'Process Children' for 'otherData', '765432463' for 'libraryCardNo', '0' for 'creditRating', and 'A227 TWR' for 'carRegistration'.

Field	Value
(RV Message)	
name (String)	Process Children
age (Integer)	Peter Thompson
position (String)	42
otherData (RV Message)	CEO
libraryCardNo (Integer)	Process Children
creditRating (Integer)	765432463
carRegistration (String)	0
	A227 TWR

11.4.8 Example: Simple Message Body for Subscription

In this example, we will create the same message structure, but begin to use the more powerful features of the field editor - the ability to validate the structure of a message when it is received and the ability to capture field values and save them to tags.

Validation is a key aspect of deciding whether a test passes or fails. To learn more about the power of validation, see [Validation for Scalar Fields](#) and [Validation for Message Fields](#) later in this same chapter.

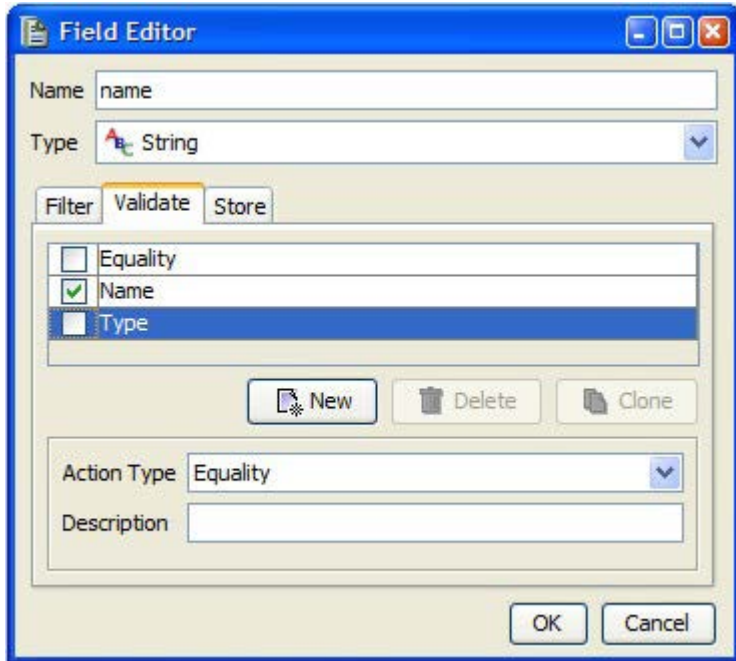
In the case of our example, we can assume that validity will be fulfilled if the message that arrives meets the following requirements:

- Having at least the four parts: name, age, position and otherData -> creditRating.
- The value in the creditRating field must be greater or equal to 10.

The full course of validation features are available only within the context of tests. Therefore, we will start by creating a new subscriber action within the test steps of a Test (see [Test Factory](#) for more information).

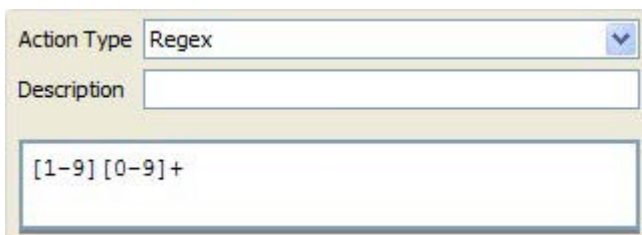
1. Begin by creating a new Subscribe action and set its transport, formatter, and message header to be the same as those of our Publish action.
2. Next, create the same body structure, except for **libraryCardNo** and **carRegistration**, as we are not interested in these fields.

-
- Double-click on the first name (String) element (Peter Thompson) to open the field editor. Click on the Validate tab and uncheck the Equality and Type options, since we are only interested in validating the attribute's name.



- Click **OK** to save the changes and repeat the same step for the **age** and **position** elements. If desired, you can disable the Equality and Type actions in the Actions pane (if displayed), beneath the message body.
- Now open the creditRating element (under **otherData**) in the field editor and select the **Validate** tab.

We wish to accept messages where this field's value is 10 or greater. A convenient way of doing this is by specifying a regular expression. Select the Equality action and change its Action Type to Regex. Next, enter `[1-9][0-9]+` in the empty value field. This is a Perl regular expression which accepts numbers from 10 upwards.

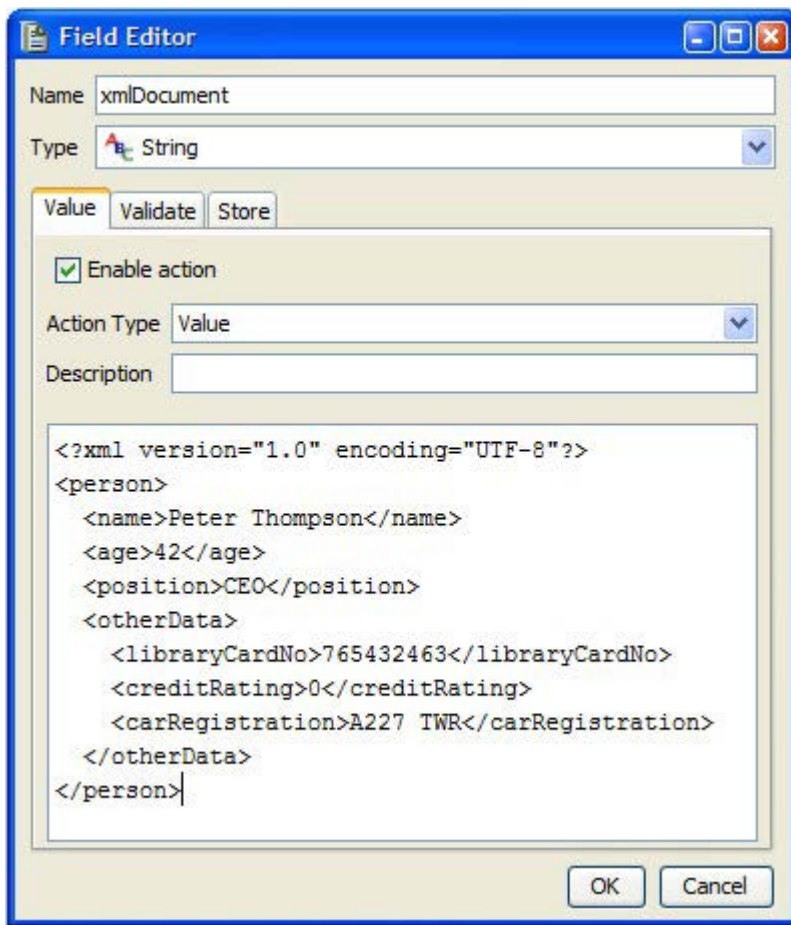


11.4.9 Converting from Flat XML to Structured XML

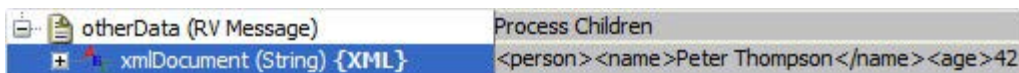
A very powerful feature of the Rational Integration Tester message editor is the ability to enter XML in **String** type fields and then edit the XML in a structured way.

NOTE: The performance of the Field Editor (shown below) will be degraded if a single line of XML is larger than 50,000 bytes. This is not an issue if the XML is on multiple lines (within reason).

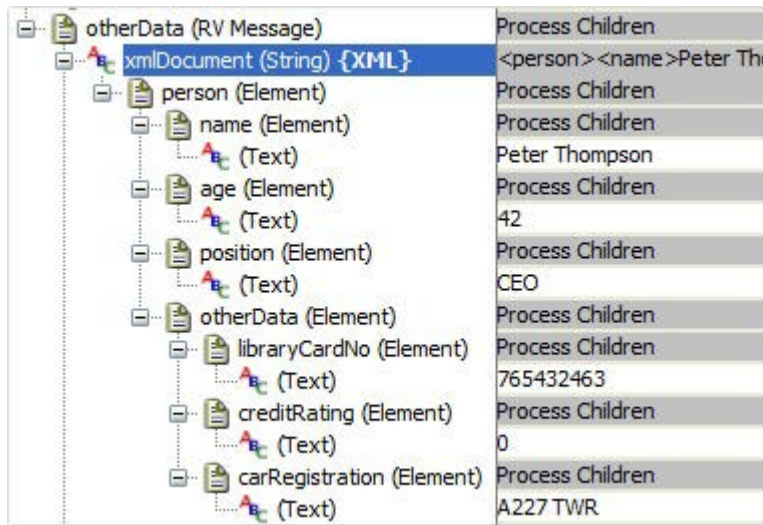
In the following example, a field named **xmlDocument** of type **String** has been created, and some XML entered in the field editor as its content.



Clicking **OK** saves the changes, leaving the message structure like this:



This tree can then be manipulated in the usual way; selecting **Expand All** from the context menu will yield the following:



The screenshot displays a hierarchical tree view of an XML message. The root node is 'otherData (RV Message)'. It contains a child node 'xmlDocument (String) {XML}', which is expanded to show a 'person (Element)'. The 'person' element has several children: 'name (Element)' (Text: Peter Thompson), 'age (Element)' (Text: 42), 'position (Element)' (Text: CEO), 'otherData (Element)', 'libraryCardNo (Element)' (Text: 765432463), 'creditRating (Element)' (Text: 0), and 'carRegistration (Element)' (Text: A227 TWR). Each node in the tree has a context menu icon (a small 'A' with a right-pointing arrow) next to it. The right pane of the interface shows the corresponding XML structure, with the 'person' element expanded to show its children.

Node Name	Value
otherData (RV Message)	Process Children
xmlDocument (String) {XML}	<person><name>Peter Thompson
person (Element)	Process Children
name (Element)	Process Children
(Text)	Peter Thompson
age (Element)	Process Children
(Text)	42
position (Element)	Process Children
(Text)	CEO
otherData (Element)	Process Children
libraryCardNo (Element)	Process Children
(Text)	765432463
creditRating (Element)	Process Children
(Text)	0
carRegistration (Element)	Process Children
(Text)	A227 TWR

NOTE: When expanding all nodes in large messages, Rational Integration Tester will only process the message for up to five seconds. After that time, all nodes that could be expanded will be expanded. If necessary, you can expand again to continue expanding any collapsed nodes.

The message hierarchy can now be edited as normal XML. Each individual node within its structure can now be manipulated by the field editor.

If you are dealing with content that follows the XML structure but would prefer to have it manipulated in a plain text manner, you can right-click the base element and select **Contents > Treat as plain text**.

11.4.10 Editing XML Message Structures

Editing XML is similar to editing normal message structures, but with a few enhancements. The following options in the element context menu apply specifically to editing XML:

- The **Schema** option launches the schema wizard which lets you select a schema, select the message root from the schema, and apply various validation options. You can also include/exclude optional fields and text nodes. The schema can come from an XSD, WSDL, DTD, or other custom formats in the project. See [Schemas](#) for more information.
- The **Root** option lets you select the root of the schema if there is more than one root.
- The **Add Child** option allows six types of XML document structure items to be added, as follows:
 - **Attribute** adds an XML attribute
 - **Comment** adds an XML comment
 - **Doc Type** adds an XML doc type element
 - **Element** adds an XML element
 - **Processing Instruction** adds an XML processing instruction
 - **Text** adds an XML text node

For information about the various functions of the different parts of XML documents, please see the W3C's XML pages at <http://www.w3c.org/XML>.

11.5 Repeating Elements

Rational Integration Tester supports the driving of repeating XML elements (and other array type objects) from a test data set, letting users create repeating XML elements on the fly. With the creation of repeating elements, users can also populate tags with lists of data (see [Using Tags](#)).

Consider the following “item” element in an XML schema that can be repeated multiple times in the message.

```
<xs:element name="item" maxOccurs="unbounded" type="itemtype"/>
```

Within the item element, there is another (repeating) “discountcode” element that is also optional and may not be needed.

```
<xs:element name="discountcode" type="stringtype" minOccurs="0"
maxOccurs="unbounded" />
```

When the entire XML schema is matched with the following test data (in the form of an Excel data set), five orders can be obtained:

id	order	user	title	quantity	price	discount	total	addresstype
1	001	Robin	Book1	1	10		10	billing
2	002	James	Book21	3	5		515	billing
2			Book22	5	100			delivery
3	003	John	Book331	1	4	DISC31	10	billing
3			Book332	1	6	DISC31		
3						DISC32		
4	004	Richard	Book4441	4	45	DISC41	180	billing
4						DISC42		
5	005	Steve	Book5551	3	15		177	billing
5			Book55552	2	66	DISC51		delivery

Previously, a user would need to manually create five tests, one for each structure. Currently, however, the user can define a single message that makes use of repeating elements to dynamically build the message from the test data set at the time of publishing or subscribing.

11.5.1 Using Tags

The values of repeating elements can be written to tags, stored as tags, and reported in test output.

Lists and Scalars

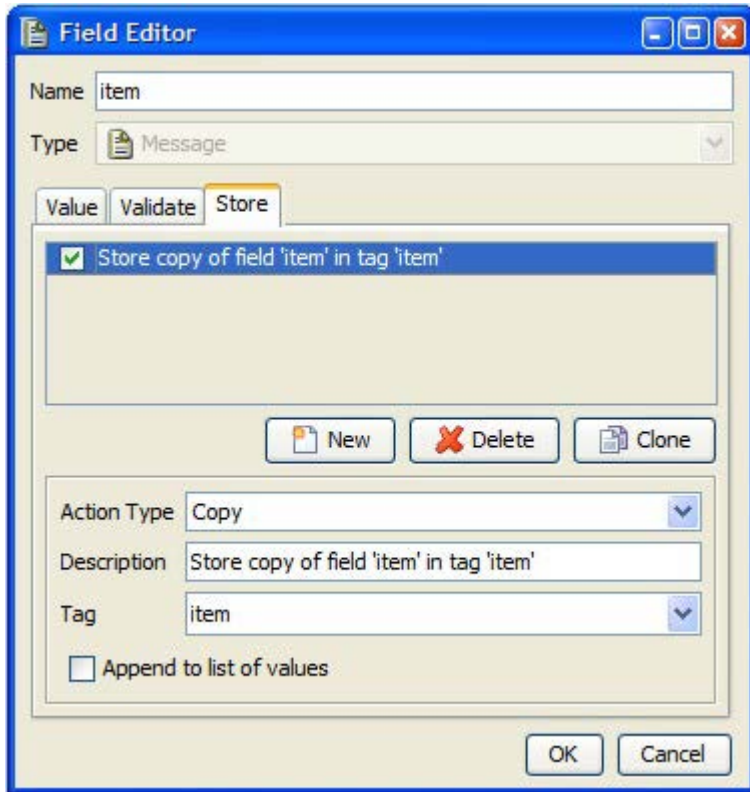
Repeating elements can be written to a tag in list mode or scalar mode. In scalar mode, any setting of a value will replace the current value of that tag, and the value returned from the tag will be the scalar object.

In list mode, if the tag contains its default value, any setting of a value will replace the default value. If the tag contains a non-default scalar, the tag will be converted to a list containing its current value as the first element and the new value as the second element.

The value returned from the tag will depend on the context in which it is evaluated. Message processing will read a particular value from the list if the message contains repeating elements and that tag determines the structure of the message.

Store Actions

Store actions on repeating fields can write to tags as lists or scalars. In the field editor, the “Append to list of values” option will write values to the tag as a list instead of a scalar.



Reporting Tags

When reporting the contents of a tag that is a list, it will be written in the following form: {scalar1, scalar2, ...}

Extracting a Value from a List of Values

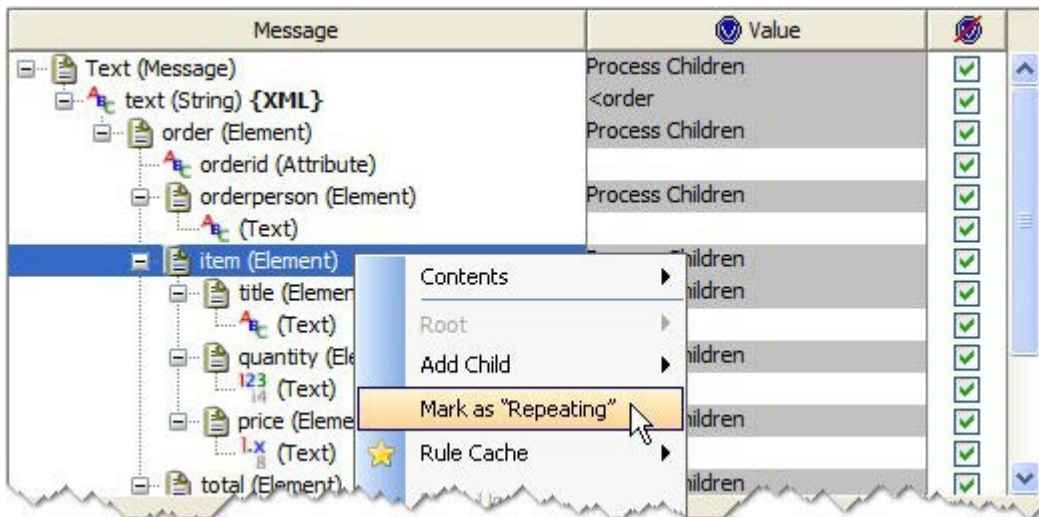
In some test actions (for example, Log, SQL Query) you can extract a specific value from a multi-value tag using the `%%tag[index]%%` notation. The values within a tag list are stored using a zero index (for example, to extract the first value in the list you would use `%%tag[0]%%`, to extract the third value you would use `%%tag[2]%%`, and so on).

NOTE: If the selected tag is a scalar or if the tag does not exist, an error will be produced.

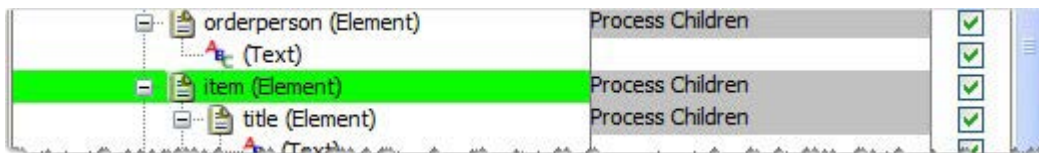
11.5.2 Repeating Elements in Messages

When a message schema describes one or more repeating elements (for example, an array or sequence), Rational Integration Tester can generate multiple children within a single element with the repeating elements being driven from a received message or a test data set.

In the message editor, you can right-click on a repeatable node and select **Mark as “Repeating”** from the context menu.



The selected element will be highlighted to indicate that it has been marked as repeating.



To revert the element to its original state, right-click it and select **Unmark as “Repeating”** from the context menu.

11.5.3 Test Data

The test data must be organized in a normalised form (see the example test data shown in the beginning of this section). The first column should contain an identifier that groups all the data from a particular record. Each subsequent column should be populated according to the following rules:

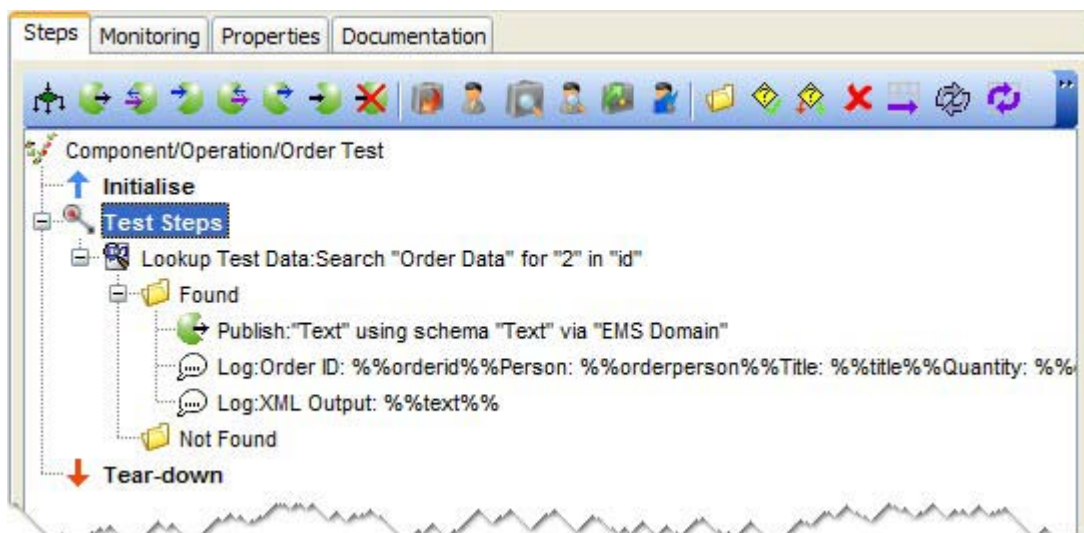
- Each piece of data for a given level of the hierarchy should exist on the same row, including the first child
- Subsequent children should exist on new rows
- A child should not appear on the same row as the child of its parent's sibling
- The end of the data for a particular depth should be padded out with null values

NOTE: When creating the test data set, ensure that the **Allow null values** option is enabled, specifying the string that designates a NULL within the data set. In the case of an Excel spreadsheet, an empty cell would be permitted by enabling the option and leaving the value field blank. If some other string is used to indicate NULLS, then that string should be specified.

11.5.4 Example

A common use of repeating elements would be to populate tags with lists of data or generate repeating XML elements. The following example illustrates how to do both using the example data and XSD shown earlier.

The test used to write tag lists and generate repeating XML elements is shown below:



Now, each component of the test will be described to help illustrate the actions taking place in each one.

Test Data Set

An Excel test data set is created using the spreadsheet data shown earlier. The data preview is shown below:

Preview (max 25 rows)

id	orderid	orderperson	title	quantity	price	discountcode	total	addresstype
1	001	Robin	Book1	1	10		10	billing
2	002	James	Book21	3	5		515	billing
2			Book22	5	100			delivery
3	003	John	Book331	1	4	DISC31	10	billing
3			Book332	1	6	DISC31		
3						DISC32		
4	004	Richard	Book4441	4	45	DISC41	180	billing
4						DISC42		
5	005	Steve	Book55551	3	15		177	billing
5			Book55552	2	66	DISC51		delivery

NOTE: The tags you will use in the test must be present in the Tag Data Store. They can be entered manually. In this example, however, the column names are copied from the data set and pasted into the Tag Data Store.

Lookup Test Data Action

The Lookup Test Data action scans the selected test data set and looks up values matching an “id” of 2. The action is configured to return all matching results, which means it will write tag lists instead of single values. Additionally, all of the columns from the test data set are mapped to tags of the same name.

Lookup Test Data
Use a value from the current run to extract information from a test data set.

Dataset:

Lookup Values

Column Key	Lookup Value
id	2

☒ Return all matching results

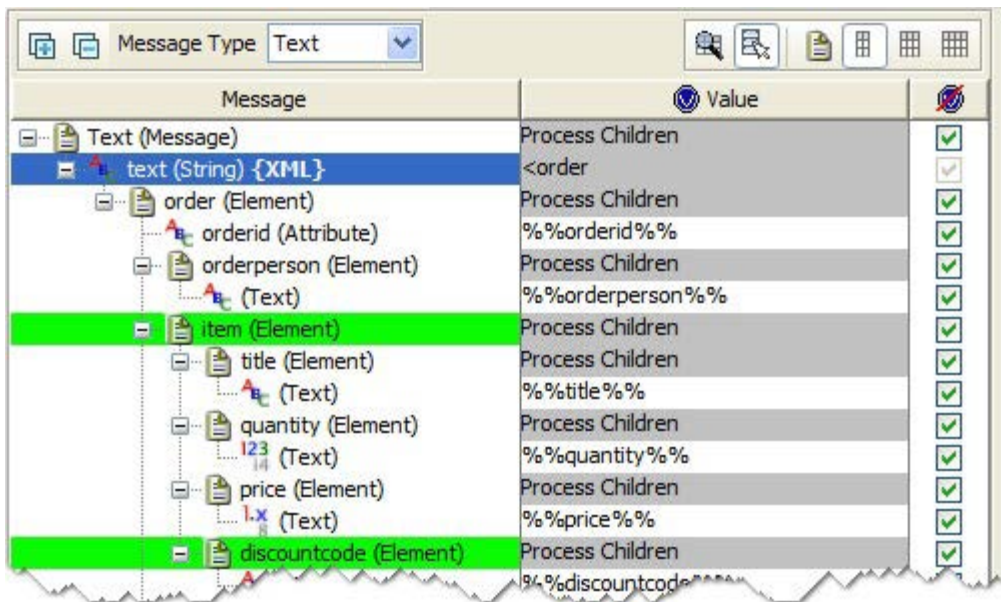
Mappings

Tag name	Data
total	total
discountcode	discountcode
addresstype	addresstype

In the case of the example, two rows of data having an “id” of 2 will be returned.

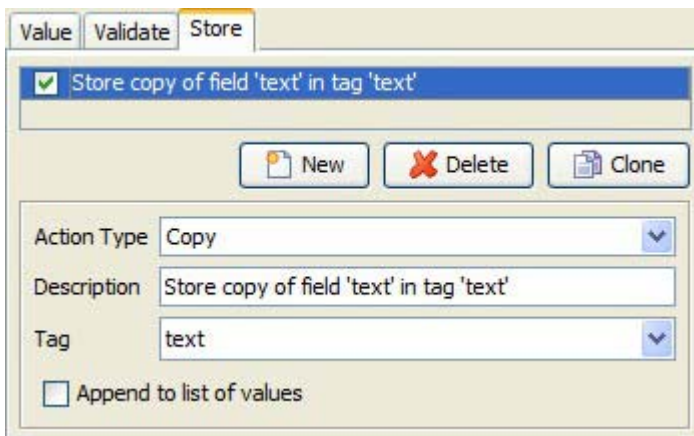
Publish

The Publish action contains an XML message to which the our schema has been applied. The transport used for publishing is not important for the purpose of the example. Each available field in the message has been “quick tagged,” meaning the contents of the field have been set to a tag matching the name of the element. The fields within the data set that contain repeating elements have been marked as such.



Message	Value	
Text (Message)	Process Children	✓
text (String) {XML}	<order	✓
order (Element)	Process Children	✓
orderid (Attribute)	%%orderid%%	✓
orderperson (Element)	Process Children	✓
(Text)	%%orderperson%%	✓
item (Element)	Process Children	✓
title (Element)	Process Children	✓
(Text)	%%title%%	✓
quantity (Element)	Process Children	✓
123 (Text)	%%quantity%%	✓
price (Element)	Process Children	✓
1.x (Text)	%%price%%	✓
discountcode (Element)	Process Children	✓
	%%discountcode%%	✓

Additionally, the contents of the entire message are being stored in a tag named %%text%% (created on the root of the XML message, the **text (String)** element).



Value	Validate	Store
<input checked="" type="checkbox"/> Store copy of field 'text' in tag 'text'		

New Delete Clone

Action Type: Copy

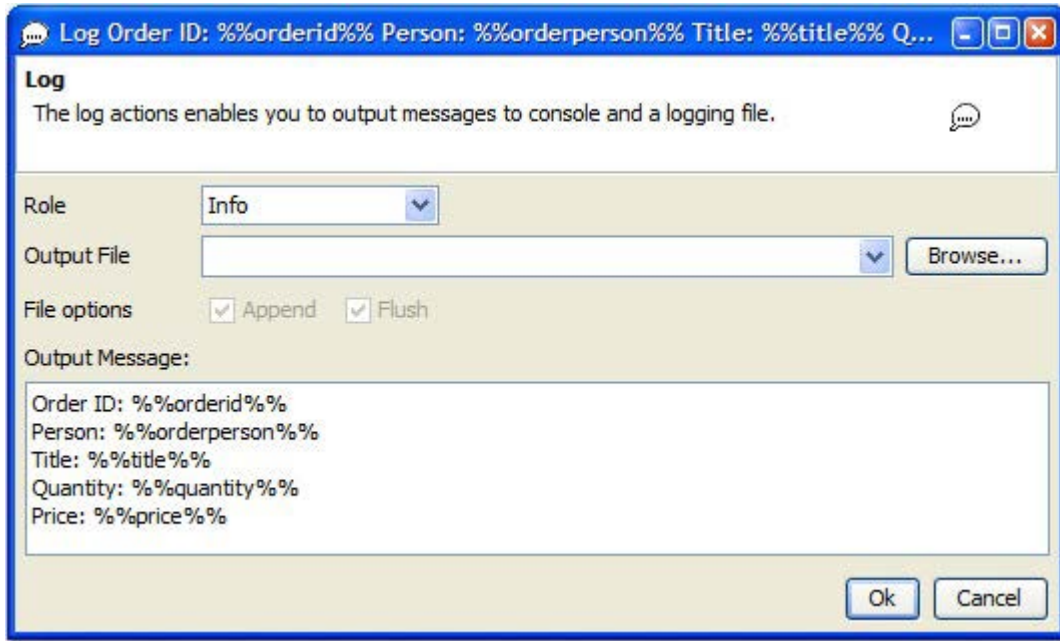
Description: Store copy of field 'text' in tag 'text'

Tag: text

☐ Append to list of values

Log Actions

The two log actions are used simply to show us (in the console) what is being mapped into the specified tags and the XML that is being generated from the lookup.



Log
The log actions enables you to output messages to console and a logging file.

Role: Info

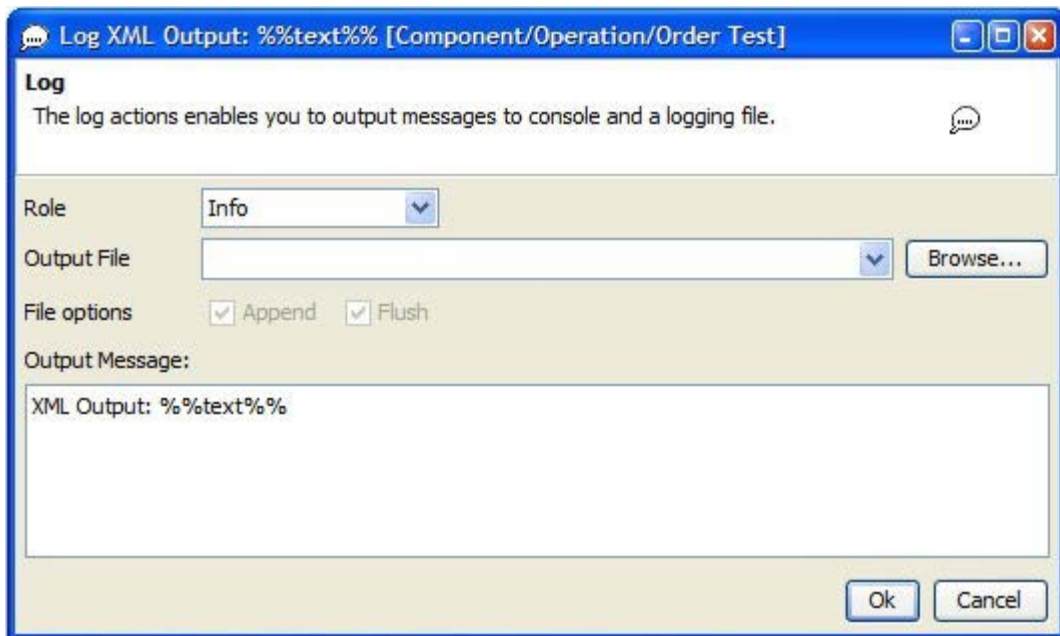
Output File: Browse...

File options: ☒ Append ☒ Flush

Output Message:

```
Order ID: %%orderid%%  
Person: %%orderperson%%  
Title: %%title%%  
Quantity: %%quantity%%  
Price: %%price%%
```

Ok Cancel



Log
The log actions enables you to output messages to console and a logging file.

Role: Info

Output File: Browse...

File options: ☒ Append ☒ Flush

Output Message:

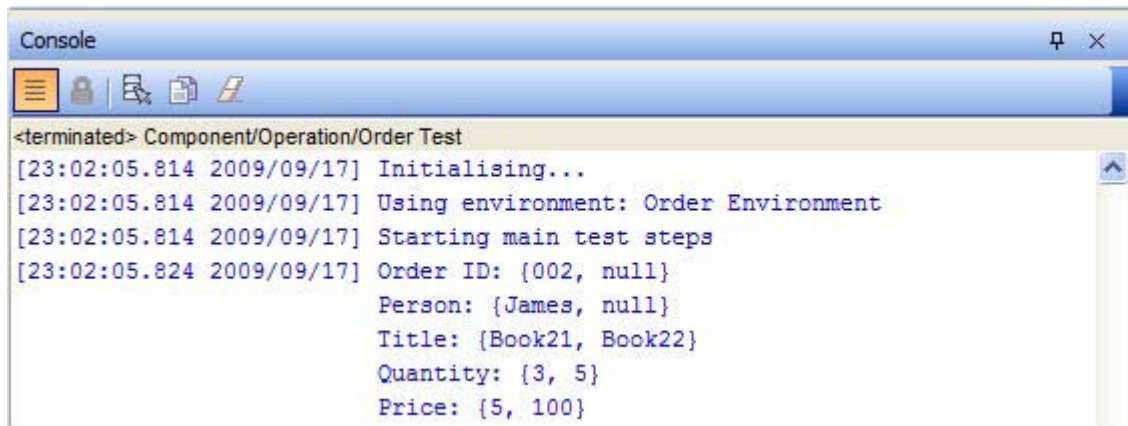
```
XML Output: %%text%%
```

Ok Cancel

Output

When the test is executed, the fields in the data set in the rows matching an “id” of 2 are mapped to the specified tags, and the XML of the full message (based on the matching rows) is mapped to the %%text%% tag.

The output of the first log (containing the tag lists) is shown below:



As you can see, all of the tags were written as lists, and Order ID and Person contained null values in the second row of data.

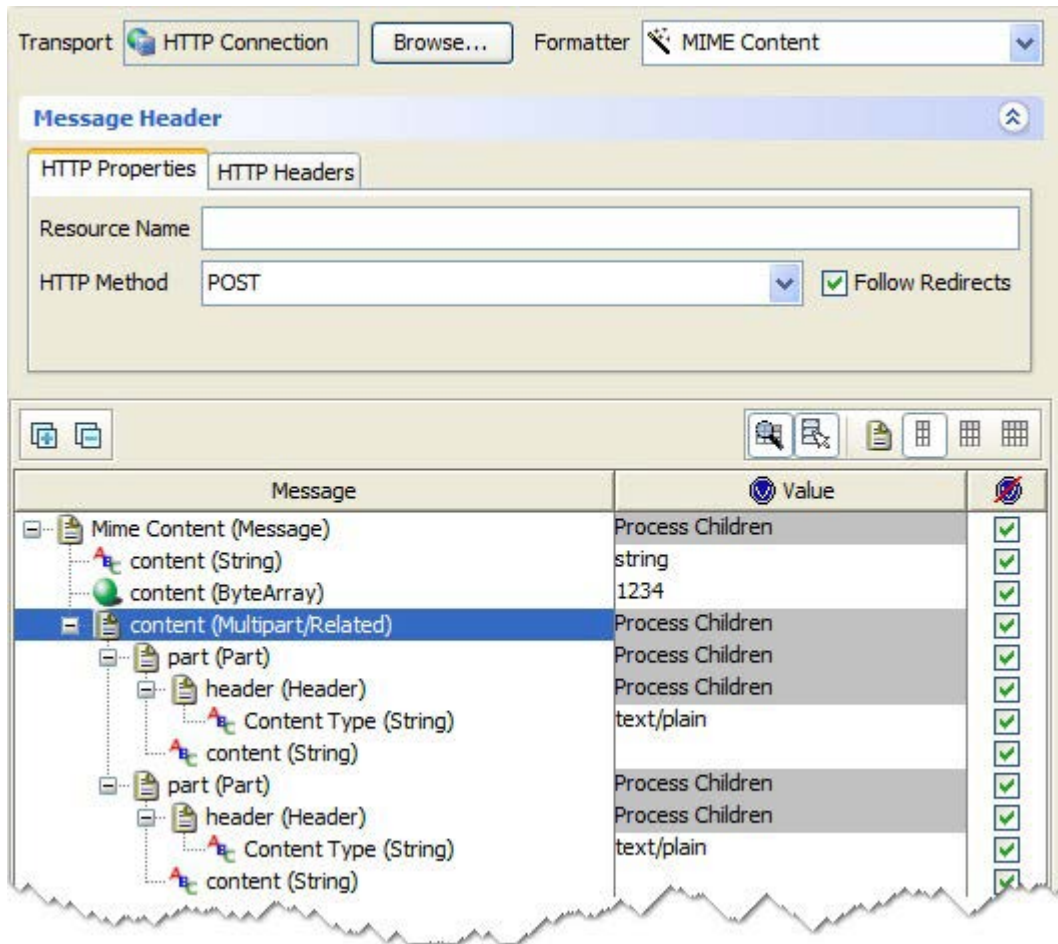
The dynamic XML, generated in the message according to the matching rows in the data set, is shown below:

```
<order orderid="002">
  <orderperson>James</orderperson>
  <item>
    <title>Book21</title>
    <quantity>3</quantity>
    <price>5.0</price>
  </item>
  <item>
    <title>Book22</title>
    <quantity>5</quantity>
    <price>100.0</price>
  </item>
  <total>515.0</total>
  <address addresstype="billing">
    <town>luton</town>
  </address>
</order>
```

Note the two “item” elements, created from the two rows of data matching an “id” of 2 from the Lookup Test Data action.

11.6 MIME, DIME, and Multipart Content

When using the HTTP transport, the message format can be set to **MIME Content**, which enables the addition of one of three child elements – Byte Array, Multipart/related, and String – from the context menu. For multipart nodes, multiple parts can then be added to the message (also from the context menu).

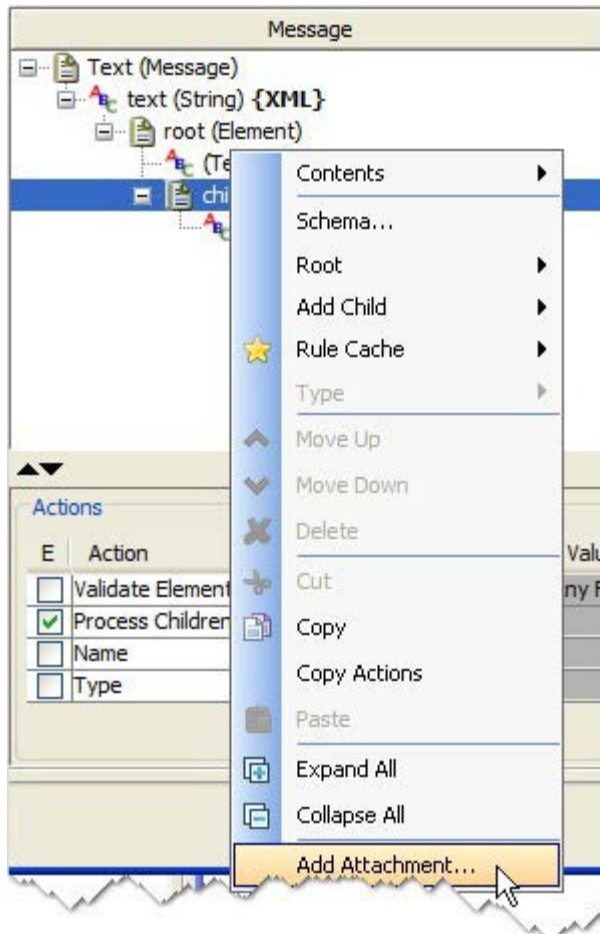


A message format can be converted to MIME or DIME by adding the appropriate attachment type or by starting to add the attachment and canceling the attachment wizard (see [Using Attachments](#)). Once the format is set to MIME or DIME, multiple parts or records can be added using the context menu.

- MIME parts can be added to the **content (multipart)** node. Part types can be binary, multipart, or text.
- Dime records can be added to the **(Message)** node. Record types can be binary or text.

11.7 Using Attachments

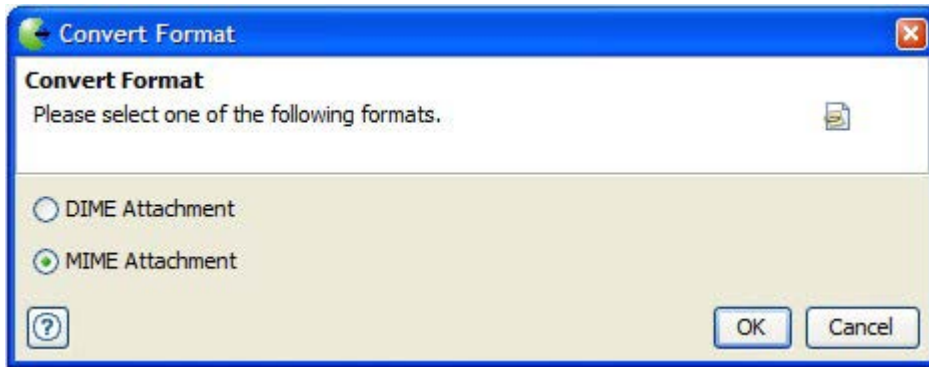
Attachments can be added to parts of a message by right-clicking the desired node and selecting **Add Attachment** from the context menu.



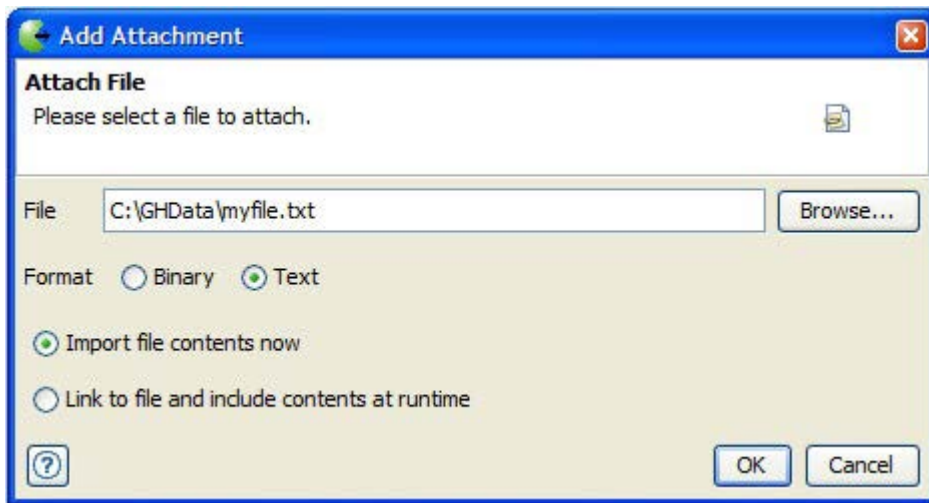
When using the HTTP transport and the HTTP message formatter, MIME or DIME attachments can be added to any node, but the content of the selected node may need to be changed accordingly (that is, Byte Array for DIME and Text for MIME). The option to convert the selected node will be provided when adding the attachment.

When using non-HTTP transports, the type of attachment available will depend upon the selected node's type.

After selecting **Add Attachment** from the context menu, the user must select the attachment type.



Once the attachment type is selected, a dialog is displayed for selecting the attachment, its format, and whether or not the attachment should be imported now or at runtime.

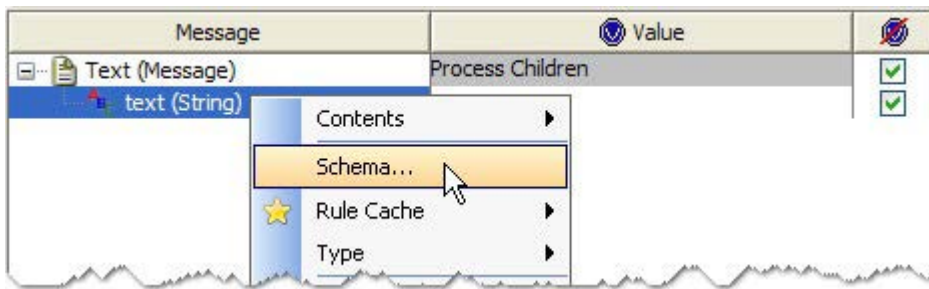


For example, using tags and a test data set that includes attachment details, you can link the attachments at runtime and use different attachments for different messages.

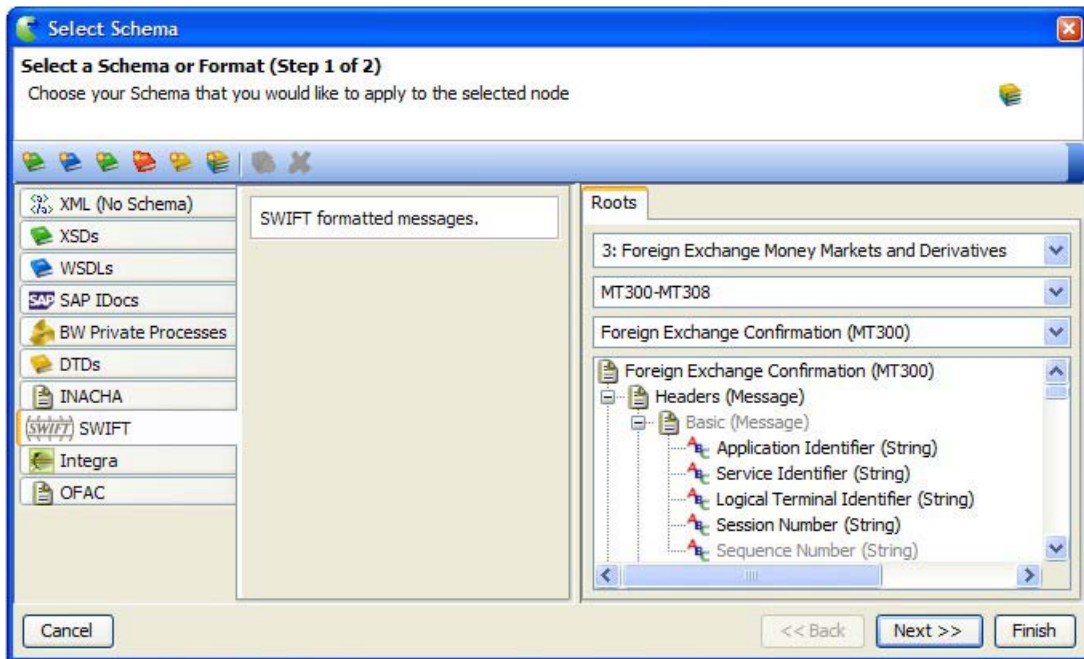
11.8 Applying Specific Formats to a Message

Built in or custom message formats can be applied to any message within Rational Integration Tester the same way (and using the same mechanism) that schemas can be applied (see [Schemas](#)).

1. Right-click the message in an editor and select **Schema** from the context menu.

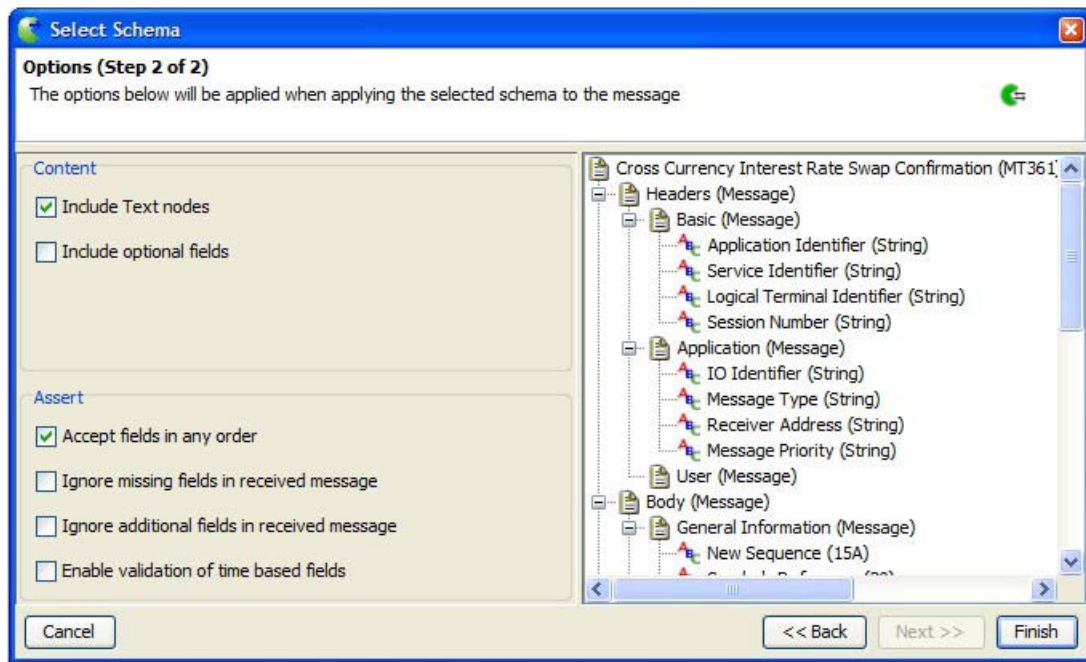


The **Select Schema** wizard is displayed.



2. Select the desired message format on the left and any additional format options under the **Roots** tab.
3. Click **Next** to configure additional message and validation options.

The second page of the wizard is displayed.



4. Select the desired content and assertion options, then click **Finish** to apply the format and close the wizard.
5. If prompted, select whether or not you want to retain existing message content (that is, if existing content can not be mapped to the new format).

The basic message structure and required fields are populated in the message editor, and the message type is indicated.

Message	Value	
Text (Message)	Process Children	✓
text (String) {SWIFT}	Expanded Content	✓
Request for Transfer (MT101)	Process Children	✓
Headers (Message)	Process Children	✓
Basic (Message)	Process Children	✓
Application Identifier (String)		✓
Service Identifier (String)		✓
Logical Terminal Identifier (String)		✓

NOTE: Any optional fields defined by the format will be available from the **Add Child** context menu option.

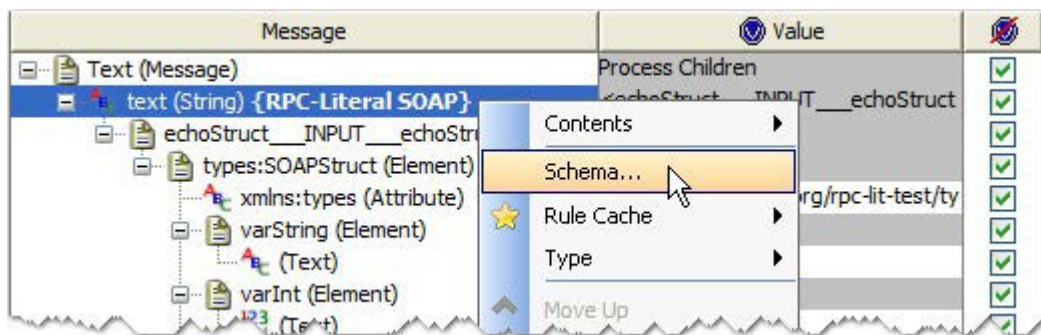
11.9 Apply Integra Message Handlers

Users who have migrated to Rational Integration Tester from Solstice Integra Suite can apply Integra message handlers to Rational Integration Tester messages. Once the “Integra” format has been applied to a message, one or more of the available message handlers can be applied using the **Pipeline** dialog (when viewing message properties).

NOTE: Before message handlers can be applied, they must be added to the Library Manager by means of a new custom provider. For more information, refer to *IBM Rational Integration Tester Installation Guide*.

Follow the steps below to apply Integra message handlers to a message:

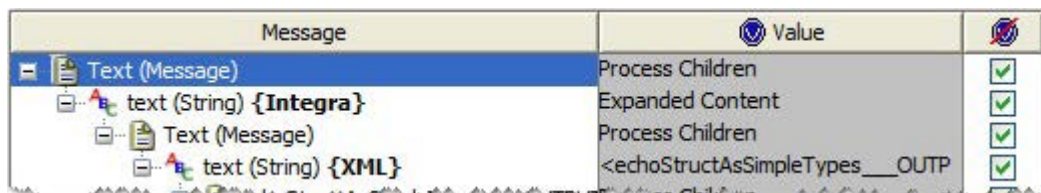
1. Right-click the message in an editor and select **Schema** from the context menu.



The **Select Schema** wizard is displayed.

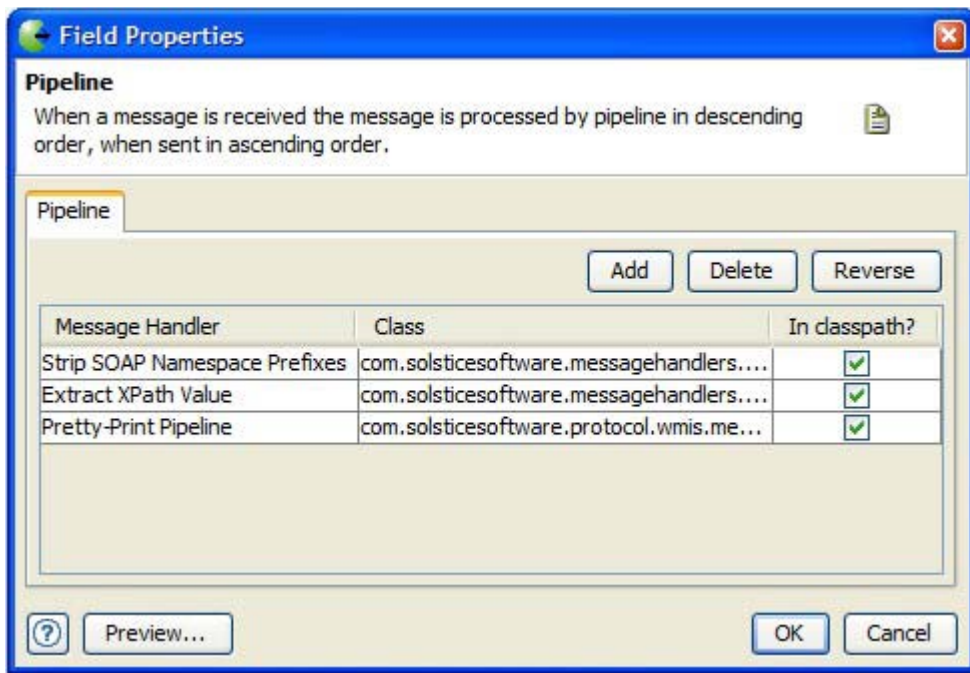
2. Select the **Integra** tab on the left and click **Finish**.
3. When prompted to retain or discard the message's current content, select the **Yes** option to retain it (otherwise the message content will be cleared).

The format of the message will now be **Integra**.



4. Right-click the **{Integra}** node and select **Properties** from the context menu.

The **Pipeline** dialog is displayed.



5. Click **Add** to add a message handler to the pipeline and select the desired message handler from those available under **Message Handler** column.

The message handler class is displayed under **Class**, and the **In classpath?** column indicates whether or not the required JAR files are available to Rational Integration Tester.

Message handlers can be reordered by dragging them to a new position in the list, and the order of the entire list of message handlers can be reversed by clicking **Reverse**. If you want to delete one of the current message handlers, select it and click **Delete**.

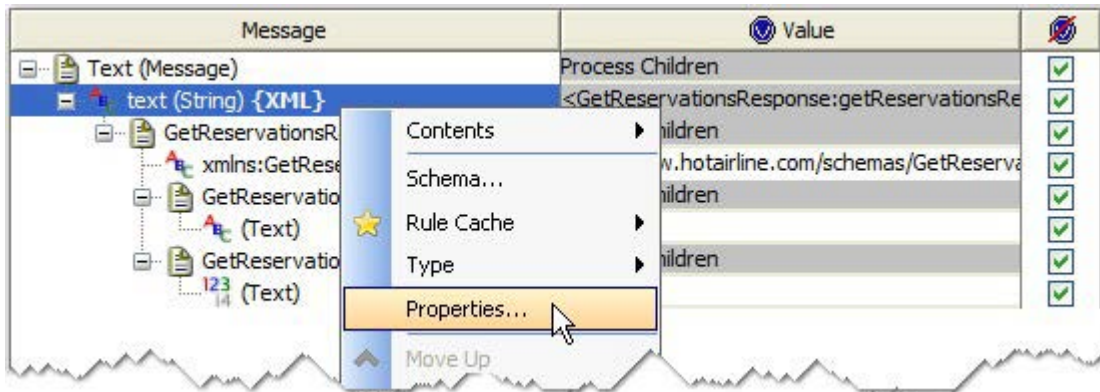
To preview the “handled” message (that is, the original message after being transformed by the listed message handlers), click the **Preview** button. The transformed message is displayed in a simple text dialog (click **OK** to close the preview).

6. When you are finished adding and ordering the desired message handlers, click **OK** to finish and close the dialog.

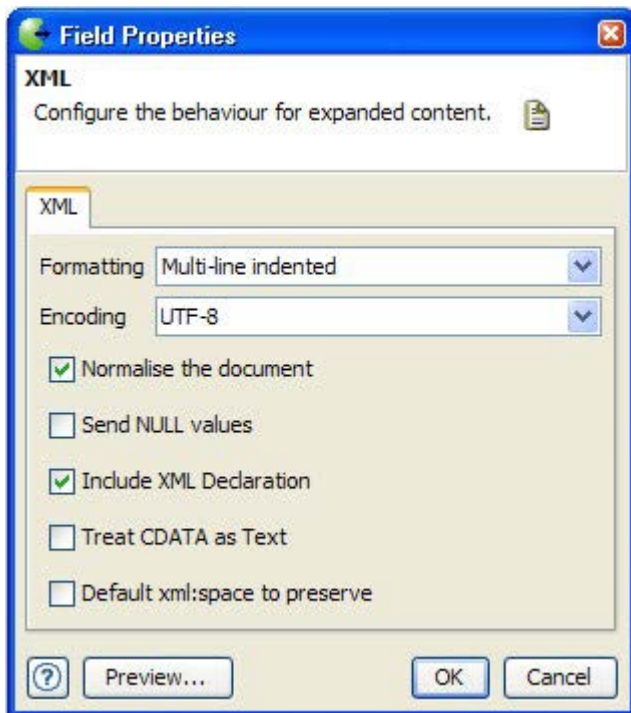
When the test is executed, the specified message handlers will be applied to outgoing or incoming messages in the order in which they were listed.

11.10 XML Message Properties

Message properties in Rational Integration Tester let you configure how message content is treated in test actions (for example, Publish, Subscribe, and so on). To view or modify message properties for an XML message, right-click on the root of the message in the requirement or in the body of the message editor.



Use the **Field Properties** dialog to view or edit the message properties.



NOTE: Click **Preview** at any time to view the message as it will be applied using the properties that you have modified.

The options available for handling XML are described below:

Formatting	Select how XML should be formatted, either Single-line or Multi-line indented .
Encoding	Select the encoding to use, either UTF-8, UTF-16, or ISO-8859-1.
Normalize empty text nodes	If enabled, extra spaces will be removed from the XML.
Send NULL values	Enables or disables the sending of XML elements whose text contains only a Tag, the value of which is NULL (not empty string ""). If Send NULL values is enabled, this element would be sent. If Send NULL values is not enabled, this element would not be sent.
Include XML Declaration	Enable this option to force the inclusion of the XML declaration at the beginning of the message.
Treat CDATA as Text	Enable this option to treat CDATA fields in the XML as text.
Default xml:space to preserve	Enables or disables the preservation of white space in XML.

NOTE: The default values for handling XML can be modified by means of the Rational Integration Tester preferences, under the **XML** section.

11.11 The Field Editor

The Field Editor, opened by double-clicking on a field or element name within a message or right-clicking the field and selecting **Contents > Edit** from the context menu, is a powerful way of specifying that a message field (data attribute) be populated using, or must fulfil, certain criteria.

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

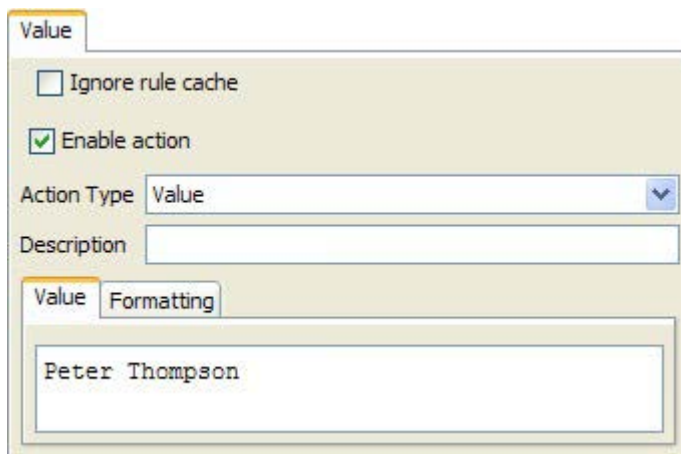
The standard field editor consists of a window with three action groups provided under their own tabs: **Value**, **Validate** and **Store** (for outgoing messages) or **Filter**, **Validate**, and **Store** (for incoming messages).

NOTE: Within each tab of the Field Editor, the “Ignore rule cache” option can be enabled if you want to ignore a specific action type within a rule that has been set on the field.

11.11.1 The Value Tab

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

NOTE: Under the **Formatting** tab, various types of formatting can be applied to field values to generate random values at runtime. See [Generating Random Data from Field Values](#) for more information.

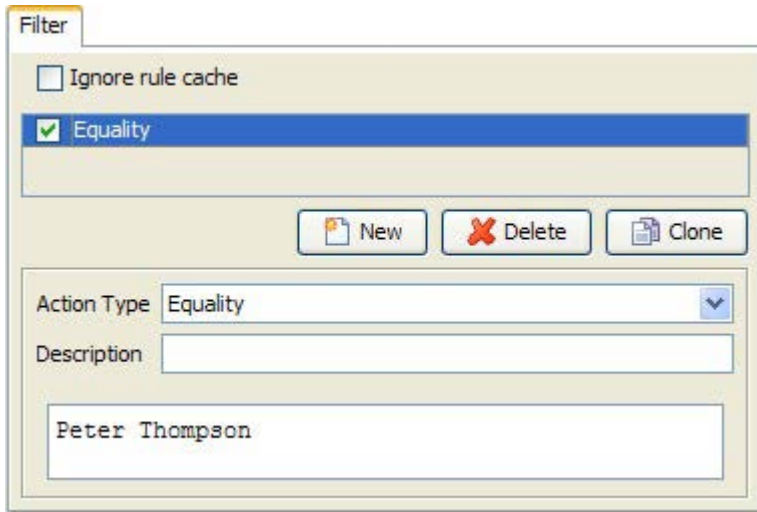


The screenshot shows the 'Value' tab of the Field Editor. It features a 'Value' tab label at the top left. Below it are two checkboxes: 'Ignore rule cache' (unchecked) and 'Enable action' (checked). There is an 'Action Type' dropdown menu currently set to 'Value'. Below that is a 'Description' text field. At the bottom, there are two sub-tabs: 'Value' and 'Formatting'. The 'Value' sub-tab is active, showing a text input field containing the text 'Peter Thompson'.

See [Setting a Field Value](#) for more information.

11.11.2 The Filter Tab

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



The screenshot shows a window titled "Filter" with a tab labeled "Filter". Inside the window, there is a checkbox labeled "Ignore rule cache" which is unchecked. Below this is a list box containing a single item "Equality" with a green checkmark to its left. Under the list box are three buttons: "New" (with a document icon), "Delete" (with a red X icon), and "Clone" (with a document icon). Below the buttons are two input fields: "Action Type" with a dropdown menu showing "Equality" and a downward arrow, and "Description" with a text box containing the text "Peter Thompson".

See [Filtering Fields](#) for more information.

11.11.3 The Validate Tab

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

The screenshot shows a 'Validate' tab in a software interface. At the top, there is a tab labeled 'Validate'. Below it, there is a checkbox labeled 'Ignore rule cache'. Underneath, there is a list of validation rules: 'Equality' (selected), 'Name', and 'Type'. Each rule has a checkbox to its left. Below the list, there are three buttons: 'New' (with a plus icon), 'Delete' (with an X icon), and 'Clone' (with a document icon). At the bottom, there is a section with 'Action Type' set to 'Equality' (in a dropdown menu) and a 'Description' text area.

See [Validation for Scalar Fields](#) and [Validation for Message Fields](#) for more information.

11.11.4 The Store Tab

The **Store** tab lets you 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).

The screenshot shows a 'Store' tab in a configuration window. At the top, there is a checkbox labeled 'Ignore rule cache' which is unchecked. Below it, a list box contains one item: 'Store copy of field 'text' in tag 'text'', which is selected and highlighted in blue. Under the list box are three buttons: 'New' (with a document icon), 'Delete' (with a red X icon), and 'Clone' (with a document and plus icon). Below these buttons are three input fields: 'Action Type' with a dropdown menu showing 'Copy', 'Description' with a text box containing 'Store copy of field 'text' in tag 'text'', and 'Tag' with a dropdown menu showing 'text'. At the bottom, there is a checkbox labeled 'Append to list of values' which is unchecked.

If the “Append to list of values” option is enabled, the tag will be updated as a list of values (for example, {val1, val2, val3}) when new values are stored.

See [Storing Field Data in Tags](#) for more information.

11.11.5 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.

The actions available for scalar fields are as follows.

Value	Validate	Store
Value	Equality	Copy (Value)
File	Length	Regular Expression
Function	Regex	XPath Query
Decrement	Xpath	
Increment	Schema	
List	IsNull	
Null	Not Null	
	Assert using function	

The actions available for message-based fields are as follows:

Value	Validate	Store (in Tag)
Process Children	Validate Message Children	Copy (Value)
Process Tag	Validate Using Tag	Regular Expression
Null	Validate Using Message from File	XPath Query
	IsNull	
	Not Null	
	Assert using function	

NOTE: In all cases, the availability of action types will depend upon the Transport, Formatter, and Field Type that has been selected.

11.11.6 Available Field Types

The basic field types available in Rational Integration Tester are shown below. When using some schema-based formatters, the range of types will change according to those supported by the formatter.

Boolean - True or False	Byte - 0 ... 255
Date-Time - 12/02/2005 5:15pm	Double - 8-byte floating point
Float - 4-byte floating point	Integer - 4-byte integer
IPAddress - a valid IP address	IPPort - 0 ... 65535
Long - 8-byte integer	Opaque - Array of bytes
Message - Agglomeration of fields	Short - 2-byte integer
String - Arbitrary length string (Default)	XML - XML content

11.11.7 Common Context Menu

You can right-click in any of the editable text fields in the Field Editor to display the following context menu:

Open Field Value: Replaces the contents of the editor with the contents of a user-selected file.

Save Field Value: Saves the contents of the text editor to a file.

Word Wrap Field Value: If enabled, the editor “wraps” words to the next line instead of scrolling horizontally.

Format XML: Format and indent the contents of the text editor if it contains XML. The formatting rules are defined in application preferences.

Flatten XML: Flattens the contents of the editor into a single string if it contains XML.

Search: Opens a dialog that lets you search for (and optionally replace) a specific string in the editor

Copy/Cut/Paste: Copies, cuts, or pastes the selected contents of the editor to/from the clipboard.

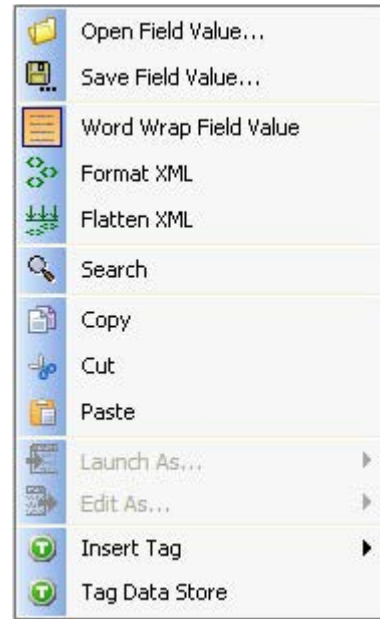
Launch As: Launches the contents of the editor as a file of the specified type.

Edit As: Edits the contents of the text editor as a file of the specified type, opening the editor specified by the system file associations.

NOTE: The contents of the **Launch As** and **Edit As** submenus can be redefined by changing options in Rational Integration Tester Preferences. See [Rational Integration Tester Overview](#) for more information.

Insert Tag: Adds a tag at the current cursor position. You can create a new tag with the **New** option, or select from a list of existing tags (split into Environment, Test Scope, or System menus).

Tag Data Store: Provides access to the tags that are available in the current test.



11.11.8 Setting a Field Value

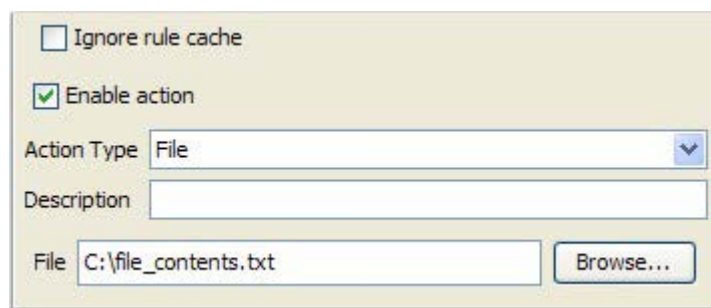
The **Value** tab lets you change the value of the selected field. Various actions may be performed on this field, which are available in the **Action Type** menu (see [Action Types](#) for more information).



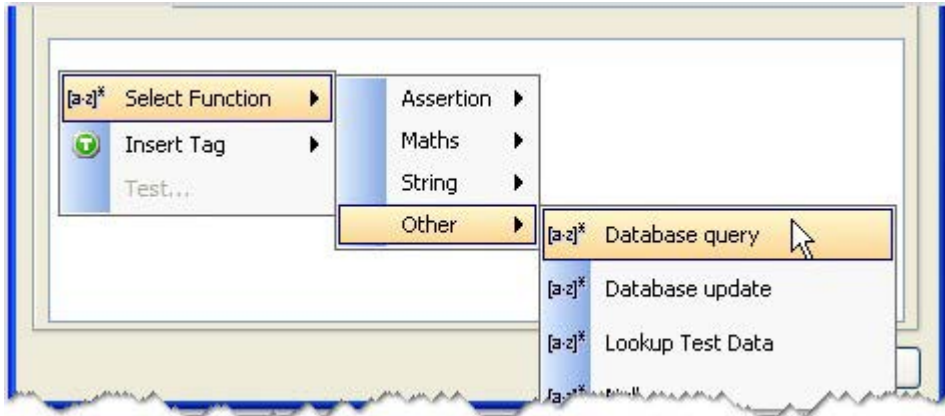
The simplest type of action is to provide a value. Enter the value manually into the edit box, paste it from the clipboard, or select **Open Field Value** from the context menu (described in [Common Context Menu](#)).

The **File** action will populate the contents of the field with the contents of a selected file.

To select a file, click **Browse** next to the **File** field and locate the desired file.



The **Function** action allows you to configure a Rational Integration Tester function (including custom functions) to be executed. The result of the function is used as the field value. Right-click in the editor to select the function and tags desired, as shown.



If **Decrement** or **Increment** is selected, the editor changes slightly, as shown below.

Initial Value is the numeric value that the field will take when the message is first sent.

Step Value is the amount by which the value will be increased or decreased on each subsequent iteration.

If desired, you can reset the field to its original value after some number of iterations. To do this, tick the box under **Reset** and enter the desired number of iterations in the field provided.

A screenshot of the configuration dialog for the 'Decrement' action. The dialog includes checkboxes for 'Ignore rule cache' and 'Enable action'. The 'Action Type' is set to 'Decrement'. The 'Description' field is empty. The 'Initial Value' is 1427, the 'Step Value' is 4, and the 'Reset' checkbox is checked with 'Every N Iterations' set to 2.

When using the **List** action , Rational Integration Tester treats each entry in the specified input as a separate value. Values can be separated using any of the available delimiters – new line, comma, tab, or full stop.

The first time the message is sent, the field will be set to the first value. For each subsequent iteration, the next value in the sequence will be used.

If a message is published more times than the number of available values, the cycle will start from the first value again.

The screenshot shows a configuration window for an action. At the top, there is a checkbox labeled 'Enable action' which is checked. Below it is a dropdown menu for 'Action Type' set to 'List'. A 'Description' text field is empty. The main input field contains the text '1427,129,12,3458,9324,652'. Below this is a 'Delimiter' dropdown menu set to 'Comma'. At the bottom, there is a 'Reset' button and a checkbox labeled 'Every N Iterations' which is unchecked, followed by a text field containing the number '2'.

To reset the action and restart from the first value, tick the box under **Reset** and enter the desired number of iterations in the field provided.

The **Null** action simply clears the contents of the field, setting its value to null. There are no further options or properties to configure for this action.

11.11.9 Generating Random Data from Field Values

When editing a field value, you can use the **Formatting** tab to apply different formats (**Date Time**, **String**, **Number**, or **Currency**) to field values at runtime, allowing you to generate random data from the field's current value.

To enable the conversion of the field value, select the **Formatting** tab and tick the **Enable** option. Next, select the desired data output category from the **Category** list.

The screenshot shows the 'Formatting' tab of a software interface. At the top, there are three tabs: 'Value', 'Validate', and 'Store'. Below them, there is a section with a checked 'Enable action' checkbox. Underneath, 'Action Type' is set to 'Value' and 'Description' is empty. The 'Formatting' tab is active, showing a checked 'Enable' checkbox. A 'Category' list on the left includes 'Date Time' (selected), 'String', 'Number', and 'Currency'. The 'Sample' field displays 'Apr 1, 1971 2:03:00 AM'. To the right of the sample is a 'Tag Value...' button. Below the sample, 'Input Format' is 'mmddddd' and 'Output Format' is 'MMM d, yyyy h:mm:ss a'. There is an unchecked checkbox for 'Use Current Date and Time'. A 'SimpleDateFormat Summary' section provides a link to the 'GH Tester Reference Guide (Appendix D)'. At the bottom, a table titled 'Letter Date or Time Component Presentation' shows 'G' for 'Era designator'.

NOTE: For **Date Time**, the input format of the value must match the format specified in **Input Format**. Alternatively, you can apply the current date and time by ticking the **Use Current Date and Time** option.

The generated value (based on the field value and selected category/output format) is displayed in the **Sample** field.

If the field value contains a tag, the **Tag Value** button is enabled. The “tag value” button is only enabled if a tag is in the field. Click this button to display the current tags, then click **Refresh** to update their values in the **Sample** field, allowing you to see the results of the conversion that would take place at runtime.

NOTE: For the **Number** and **Currency** categories, if the contents of a tag are being formatted, then the tag must contain a valid format (see the Decimal Format Summary, displayed for each category).

Selecting the **Custom** check box – not available for **Date Time** – will enable the custom output field for the selected category, letting you customize the output to the format of your choice. The **Sample** field will update as you edit the custom format. For some categories, the custom field uses a drop-down menu that will remember previously entered data.

NOTE: The output format formula is similar to what you would use in a Microsoft Excel formula.

Below the format options for each category is a summary pane that describes the format and may help in configuring the input and output settings.

SimpleDateFormat Summary

To learn more about Simple Date Formats please refer to the [GH Tester Reference Guide \(Appendix D\)](#)

Letter	Date or Time Component	Presentation	Examples
G	Era designator	Text	AD
y	Year	Year	1996; 96
M	Month in year	Month	July; Jul; 07
w	Week in year	Number	27
W	Week in month	Number	2
d	Day in year	Number	1

At the bottom of the **Formatting** tab is the **Alignment** pane, which allows you to edit the maximum length of a value, configure its justification, add a repeating character to the left or right of the value to fill in the space up to the maximum, or trim the value if it exceeds the maximum length.

Alignment

Length: ☐ None ☐ From Schema ☒ Custom

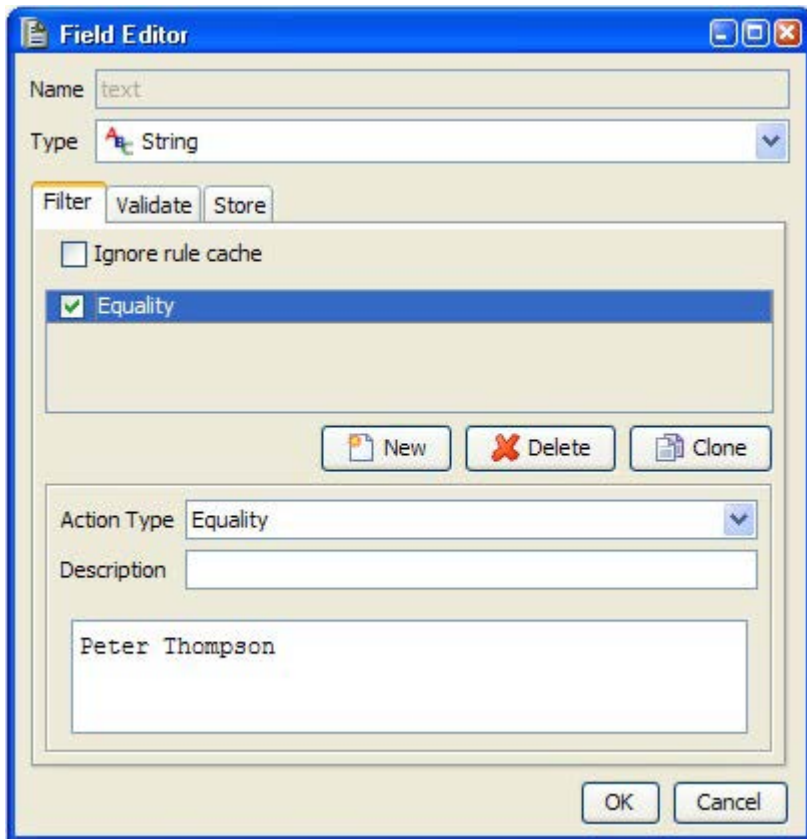
Justification: ☒ Left ☐ Right ☐ Centered

Character: Left Right

☒ Trim if too long?

11.11.10 Filtering Fields

Under the **Filter** tab, you can restrict which messages a subscriber will receive based on the content of both the header and body of the message.



Only those messages that conform to the filter specified will be validated or stored. Messages that do not conform will be ignored.

For scalar fields, the same action types that are available for validation (see [Validation for Scalar Fields](#)) can be used for filtering.

For message based fields, additional types are available, as follows:

Does Exist can be used to accept or ignore messages based on whether or not the selected field actually exists in the message. **Name** and **Type** can be used to accept or ignore messages based on the name or type of the message field. **Is Null**, **Not Null**, and **Assert using Function** are also available (these are described under [Validation for Scalar Fields](#)).

11.11.11 Validation for Scalar Fields

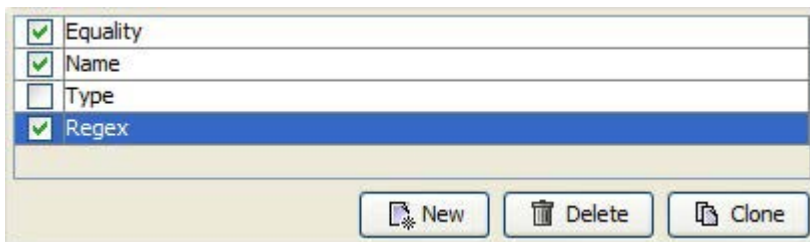
Validation allows decisions about sent and received messages to be made automatically at test time. A validation is made up of one or more actions, as described earlier. Under the **Validate** tab, the default configuration contains three validation actions, as follows:

Name	When checked, enables name validation (that is, valid when the <i>name</i> field is the same).
Type	When checked, ensures that the field <i>type</i> is the same as what is specified (that is, String, Integer, and so on)
Equality	When checked, validates that the value of the field equals what is expected. Equality is the default action, but it can be changed as described in Action Types .

You can add more validations by clicking **New**. To delete a validation that you have added, select it and click **Delete**.

NOTE: The default validations can not be deleted.

To duplicate any validation (except for the **Name** and **Type** validations), select it and click **Clone**. The validations that are in the editor can be enabled and disabled by checking/unchecking the box next to them.



Details about the different validation actions are provided in the following sections:

- [Equality](#)
- [Length](#)
- [XPath and Regex](#)
- [Schema](#)
- [Is Null and Not Null](#)
- [XSD Type](#)
- [Assert using Function](#)

Equality

If **Equality** is chosen, the value to validate should be entered in the text field. The value can be pasted or loaded from a file, as described earlier.

A screenshot of a software dialog box for the 'Equality' validation type. It features a dropdown menu for 'Action Type' set to 'Equality', an empty 'Description' field, and a large text area containing the text 'Peter Thompson'.

Length

If the required validation type is **Length**, specify the minimum and maximum lengths of the field value in the appropriate fields.

A screenshot of a software dialog box for the 'Length' validation type. It includes a dropdown menu for 'Action Type' set to 'Length', an empty 'Description' field, and two numeric input fields: 'Minimum Length' with the value '0' and 'Maximum Length' with the value '1,000,000'. Both numeric fields have up and down arrow buttons for adjustment.

XPath and Regex

If **XPath** or **Regex** is selected as the action type, the XPath expression or the regular expression used to validate the field must be entered in the text field.

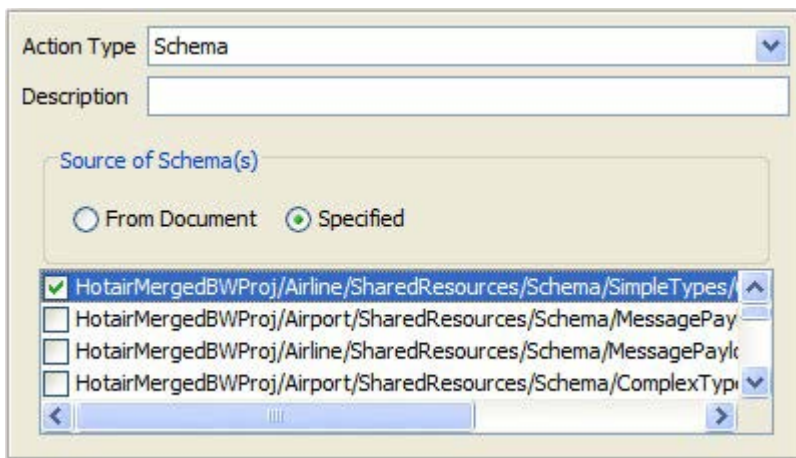
A screenshot of a software dialog box for the 'Regex' validation type. It features a dropdown menu for 'Action Type' set to 'Regex', an empty 'Description' field, and a large text area containing the regular expression '[1-9] [0-9]+'. At the bottom, there is a 'Document' field with an empty text box and a 'Browse...' button, and a 'Result' field showing 'false' next to a 'Test' button.

To help in building regular expressions and XPath expressions, a testing utility is available. By clicking **Test**, you can verify your expression against any arbitrary content without waiting for a full test run to show the results.

Values to validate against can be entered in the **Document** field. Alternatively, you can click **Browse** to locate and select a test document, the contents of which will be loaded in the **Document** field.

Schema

To validate the field value against a schema, select the **Schema** action type.



The screenshot shows a configuration window for the 'Schema' action type. At the top, 'Action Type' is set to 'Schema' and 'Description' is empty. Below, the 'Source of Schema(s)' section has two radio buttons: 'From Document' (unselected) and 'Specified' (selected). A list box below shows four schema paths, with the first one selected and checked: 'HotairMergedBWProj/Airline/SharedResources/Schema/SimpleTypes/'. The other three paths are 'HotairMergedBWProj/Airport/SharedResources/Schema/MessagePay...', 'HotairMergedBWProj/Airline/SharedResources/Schema/MessagePayl...', and 'HotairMergedBWProj/Airport/SharedResources/Schema/ComplexTyp...'. The list box has scroll arrows on the left and right.

Select the **Specified** option and select one or more schemas from the list of those displayed.

NOTE: Before a schema can be used, it must have already been imported into Rational Integration Tester. See [Schemas](#) for more information.

Is Null and Not Null

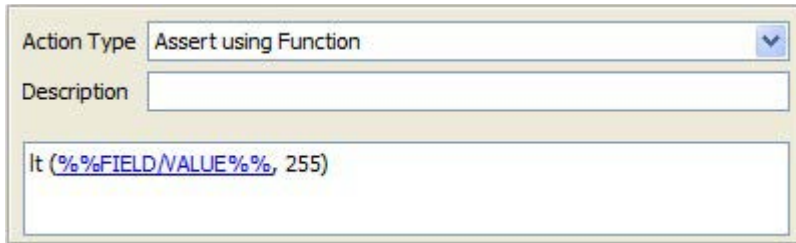
The **Is Null** and **Not Null** validations simply check whether the specified field is null or has some value.

XSD Type

The **XSD Type** validation simply checks whether the field is of the correct type as specified by the XSD used to build the message.

Assert using Function

To validate using a Rational Integration Tester function, select the **Assert using Function** action.



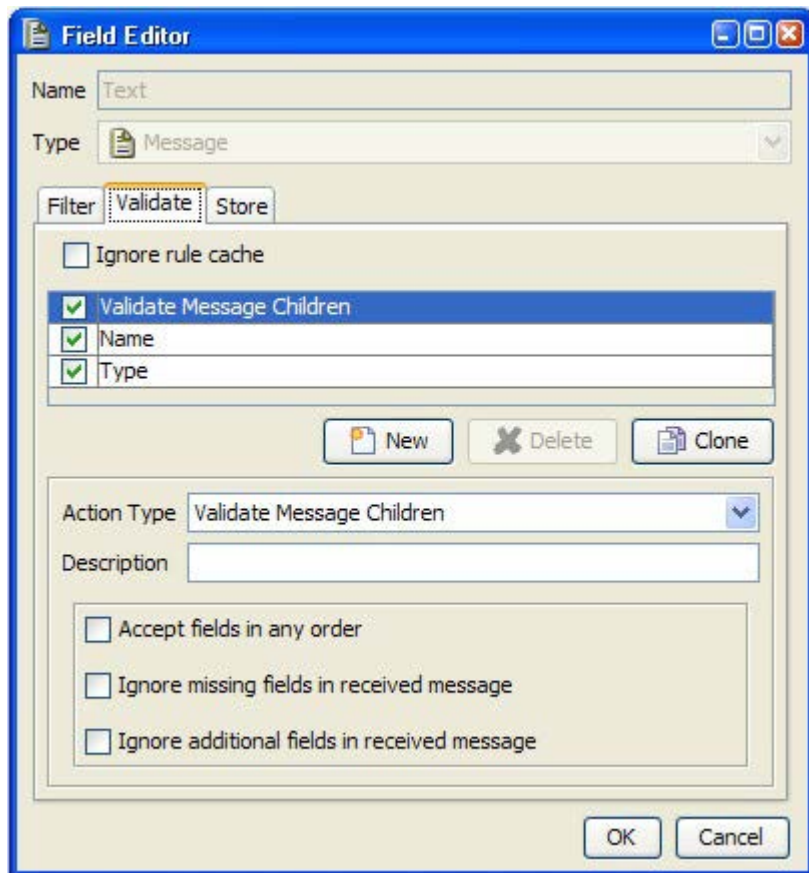
The screenshot shows a configuration dialog for the 'Assert using Function' action. It has a title bar and a light beige background. At the top, there is a label 'Action Type' followed by a dropdown menu that currently displays 'Assert using Function'. Below this is a label 'Description' followed by a text input field. The input field contains the text 'It (%%FIELD/VALUE%%, 255)'. The text '%%FIELD/VALUE%%' is highlighted in blue, indicating it is a tag or function name.

Use the context menu to select any function that return a Boolean type, which will determine whether the assertion passes or fails. You can access the value of the field within the message using the **FIELD/VALUE** tag, as shown above.

NOTE: Message validation is performed **before** the evaluation of Store actions. This means that although it is possible to reference tags within the same message, they will not contain data from the incoming message at the point when the function is evaluated (that is, it is not possible to compare two fields from the incoming message using the **Assert using Function** action).

11.11.12 Validation for Message Fields

For fields that are of the type *Message*, the contents of the validate tab changes. The new options let users validate the structure of a message (or its child elements) rather than its individual attributes.



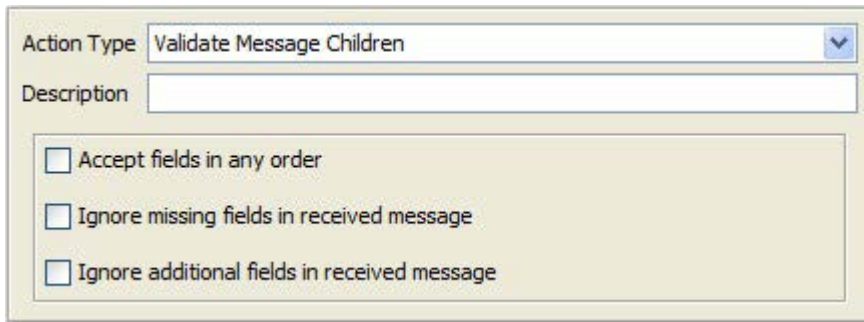
NOTE: Some of the same validations that are available for scalar fields can be used on *Message* fields. These are **Is Null**, **Not Null**, and **Assert using Function**.

The new validation actions available for *Message* fields are described in the following sections:

- [Validate Message Children](#)
- [Validate Using Tag](#)
- [Validate Using Message from File](#)

Validate Message Children

If you want to validate the children of a message element, select the **Validate Message Children** option.



Action Type: Validate Message Children

Description:

☐ Accept fields in any order

☐ Ignore missing fields in received message

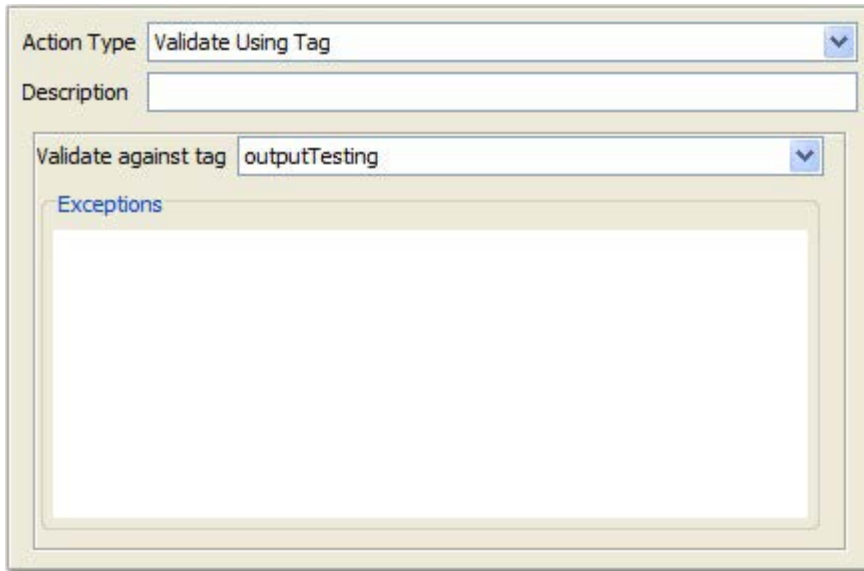
☐ Ignore additional fields in received message

Select the additional options, as follows:

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 message	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 message	When enabled, any fields present in the received message but absent in the expected message structure will not cause invalidation.

Validate Using Tag

To compare a received value against a message or sub-message that has been previously tagged, select the **Validate Using Tag** option.



The image shows a configuration dialog box for the 'Validate Using Tag' action. It has a title bar and a light beige background. At the top, there is a dropdown menu labeled 'Action Type' with 'Validate Using Tag' selected. Below this is a text field labeled 'Description'. Underneath the description field is another dropdown menu labeled 'Validate against tag' with 'outputTesting' selected. At the bottom of the dialog is a section titled 'Exceptions' in blue text, followed by a large, empty rectangular area for entering XPath representations of fields to be ignored.

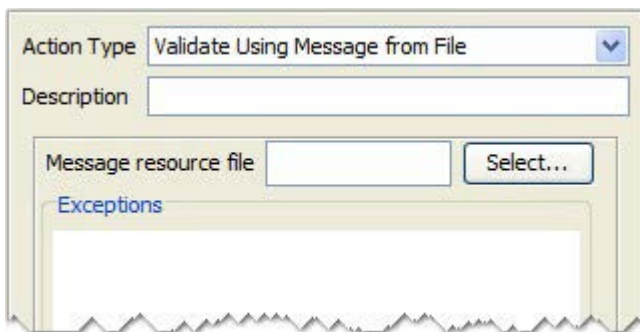
Select the existing tag against which the comparison should occur from the **Validate against tag** menu. If there are any fields that should be ignored in the comparison, you can enter an XPath representation of them under Exceptions.

Validate Using Message from File

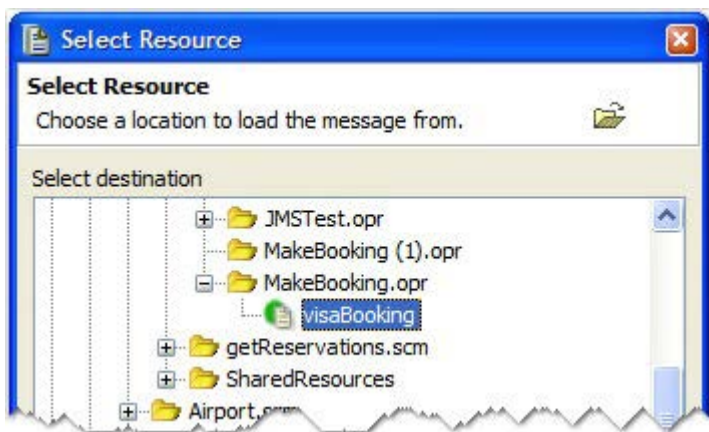
It is possible to validate a message against a previously saved version of the message.

In the following example, we have created a two step test where the first step publishes data from a file and the second step subscribes to the published data and compares the data to messages that have previously been saved. The publish step provides us with the tagged data that allows us to specify the name of the file we wish to compare the data with in the subscribe step.

To enable validation against a saved message file, edit the message in the subscribe step and double-click the top (Message) node of the message. Next, select the **Validate Using Message from File** action type.

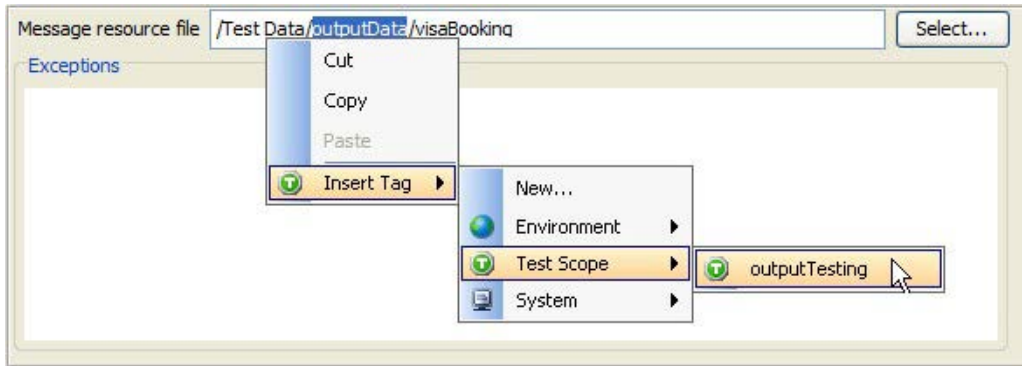


Click **Select** to choose a message resource file. Select the desired message from the **Select Resource** dialog and click **OK** when finished.



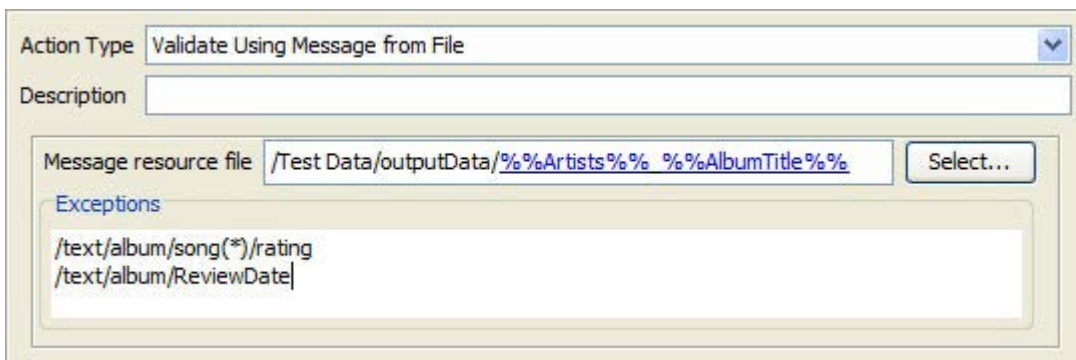
The message location is displayed as a path in the field editor (for example, /Test Data/Output Data/myMessage).

It is possible to make the filename dynamic (that is, generated from data received in the incoming messages) by substituting a tag for a portion of the message path. In the example shown below, only the message folder name is being substituted.



You are now ready to run the test. For each message to which the test subscribes, a comparison will be made to the message in the file that matches the resource file name constructed by the published tag.

If you are not interested in comparing some of the fields (for example, date fields which are likely to have changed), validation exceptions can be added. Paths to fields that should not be included in the validation should be entered in the **Exceptions** field.



In the example shown above, validation will not be carried out against the *ReviewDate* attribute or against any of the *rating* elements for each of the songs.

11.11.13 Storing Field Data in Tags

Storing the value of a field in a tag is most useful when a message has been received. You can, however, store a field in a tag when a message has been published with dynamically populated field values.



There are three differing ways (under **Action Type**) in which the value can be stored: **Copy**, **XPath Query**, and **Regular Expression**.

NOTE: If the “Append to list of values” option is enabled, the tag will be updated as a list of values (for example, {val1, val2, val3}) when new values are stored.

The **Copy** action simply stores the untransformed value in the specified tag.

If **Regular Expression** or **XPath Query** are specified as action type, additional details are required.

The screenshot shows a configuration dialog box for a 'Regular Expression' action. It includes fields for 'Action Type' (set to 'Regular Expression'), 'Description' (set to 'Store copy of field 'name' in tag 'FieldName''), and 'Tag' (set to 'FieldName'). There is a large empty text area for the 'Expression:'. Below this, the 'Result is:' section has two radio buttons: 'Match?' (unselected) and 'Extract Instance' (selected). To the right of the 'Extract Instance' radio button is a text field containing the number '1'. At the bottom, there is a 'Document' field with a 'Browse...' button next to it, and a 'Result' field with a 'Test' button next to it.

If you want to store only the field values that match the entered expression or query, enable the Match? option. If you want to extract a portion of repetitive data, the relevant part of which is identified only by its position, you can provide an instance number (1, by default 1).

For instance, if the value of the field is '12:02:05' and you want to extract the minutes portion (02) to be stored in a tag, you could specify the following regex:

(\d\d)

This expression specifies a group definition that matches the numeric parts of the data. In this case, since you are interested in the minutes entry, you would extract instance 2.

NOTE: When extracting data using regular expressions, you must use brackets to indicate what is to be extracted rather than what is to be matched.

To help in building regular expressions and XPath queries, a testing utility is available. By clicking **Test**, you can verify your expression against any arbitrary content without waiting for a full test run to show the results.

Values to validate against can be entered in the **Document** field. Alternatively, you can click **Browse** to locate and select a test document, the contents of which will be loaded in the **Document** field.

Schemas

Contents

Creating Schema Resources

Using Schemas

Changing Schemas

Analysing Schemas

File Schemas

Record Layouts

This chapter provides information about creating and using schema resources in Rational Integration Tester.

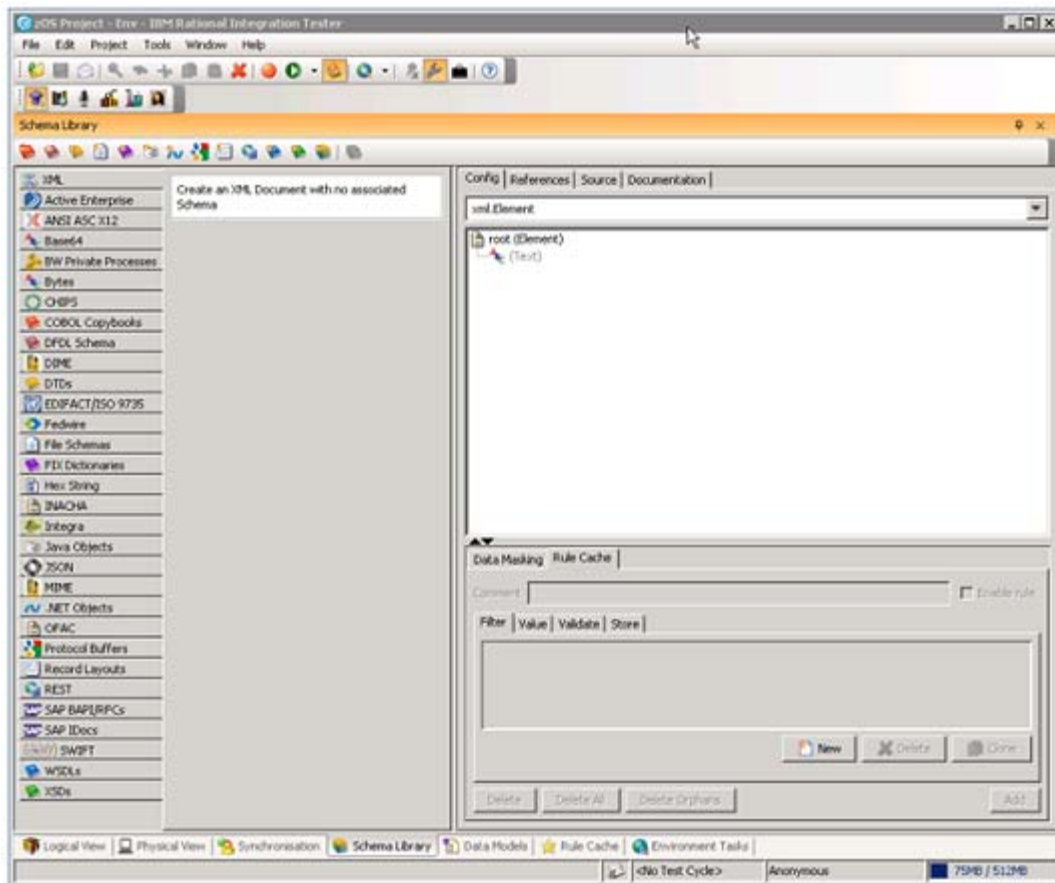
All message editors in Rational Integration Tester are schema “aware” (that is, their contents can be restricted according to a metadata description of the message structure or schema). These restrictions are either implicit from the choice of message Transport and Formatter, or explicit by applying an imported schema.

12.1 Creating Schema Resources

Schemas can be added to your Rational Integration Tester project when they are part of an externally synchronised resource (that is, WSDL, TIBCO BusinessWorks project, or webMethods Integration Server Domain), or they can be added manually in the Schema Library (see [The Schema Library](#) for more information). This section provides information about adding schemas manually.

Currently, the following schema types can be added manually to a Rational Integration Tester project:

- XML
- ANSI ASC X12
- CHIPS
- COBOL Copybooks
- DTD schemas
- Fedwire
- File Schemas
- FIX Dictionaries
- Java Objects
- JSON
- .NET Objects
- Protocol Buffers
- Record Layouts
- REST
- WSDL schemas
- XSD schemas



The following schema types can be added to a project when synchronizing with an external resource:

- SAP BAPIs/RFCs
- SAP IDocs
- TIBCO BusinessWorks Private Processes
- TIBCO Active Enterprise
- webMethods

NOTE: SAP BAPIs, RFCs, and IDocs are added after they have been imported or defined in an SAP system configured in the Logical View. Private Process and Active Enterprise schemas are made available when a TIBCO BusinessWorks project that contains them is added to a Rational Integration Tester project. webMethods schemas are added

when importing packages during the synchronization with a webMethods Integration Server.

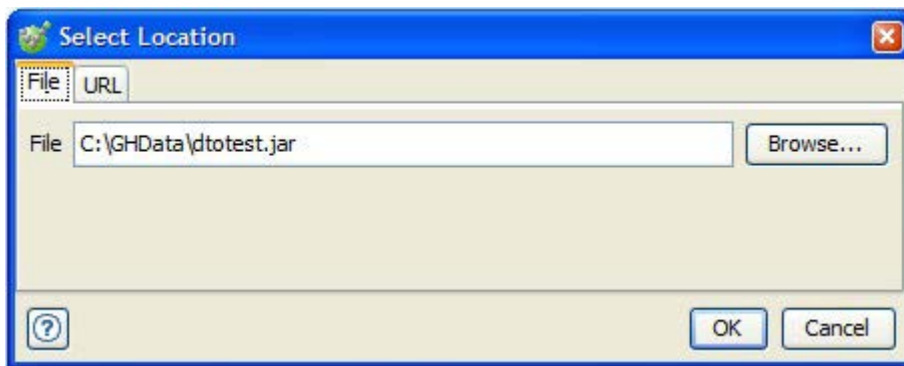
12.1.1 Add a Single Schema

Follow the steps below to add a single schema to Rational Integration Tester:

1. Open Rational Integration Tester's Architecture School perspective (**F7**) and select the Schema Library view.
2. In the toolbar at the top of the view, click the icon that corresponds to the type of schema you want to add or click the **Add schema file...** icon to browse all supported schema types.



3. In the **Select Location** dialog, click **Browse** to locate and select a local schema file of the specified type.



NOTE: If you want to add an external schema using its URL, select the URL tab and enter it there. Additionally, WSDLs can be added using an existing UDDI server (refer to *IBM Rational Integration Tester Reference Guide for HTTP & Web Services*).

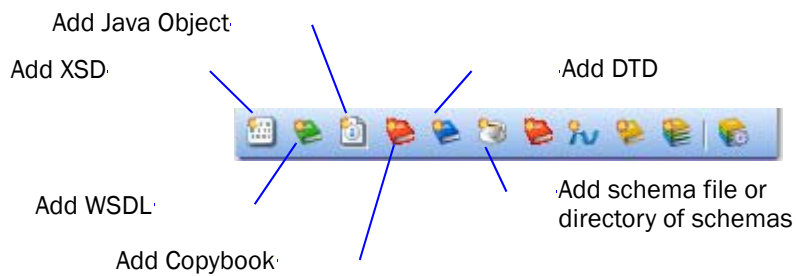
4. Click **OK** when finished.

-
5. The new schema is added to Rational Integration Tester under the appropriate tab of the Schema Library view.

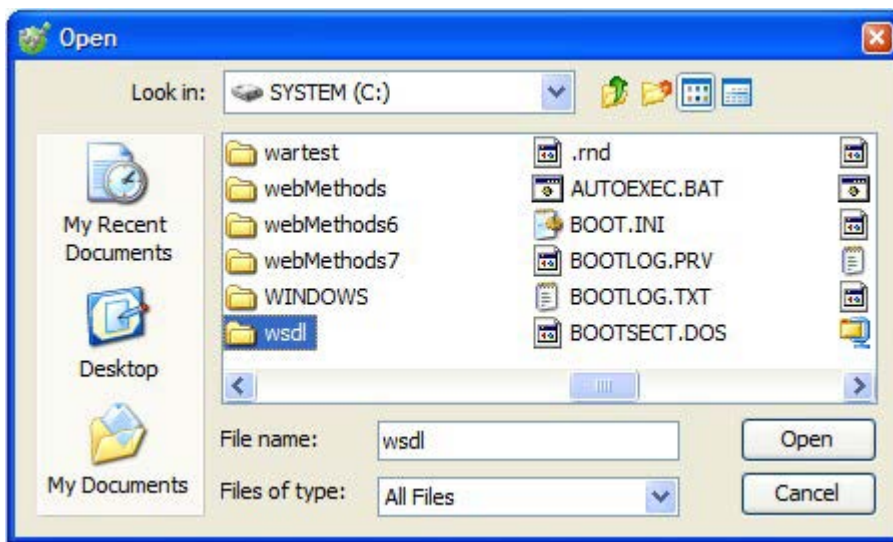
12.1.2 Add a Directory of Schemas

Follow the steps below to add a directory of schemas to Rational Integration Tester:

1. Open Rational Integration Tester's Architecture School perspective (**F7**) and select the Schema Library view.
2. In the toolbar at the top of the view, click the **Add schema file...** icon.



3. In the file browser that is displayed, locate and select the directory containing the schema files and click **Open**.



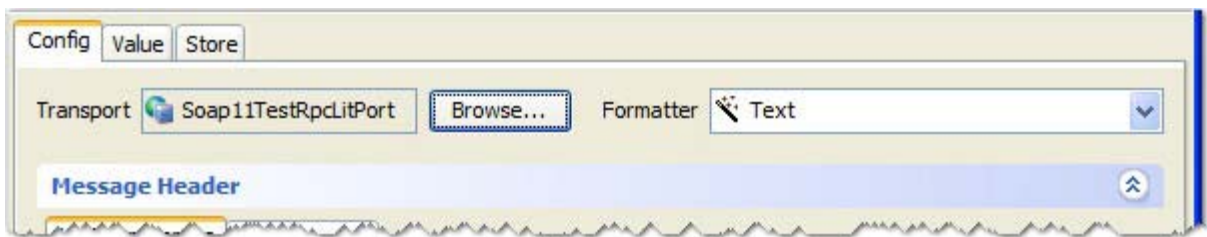
4. All valid schema types found in the selected directory will be added to Rational Integration Tester and displayed under the appropriate tab of the Schema Library view.

12.2 Using Schemas

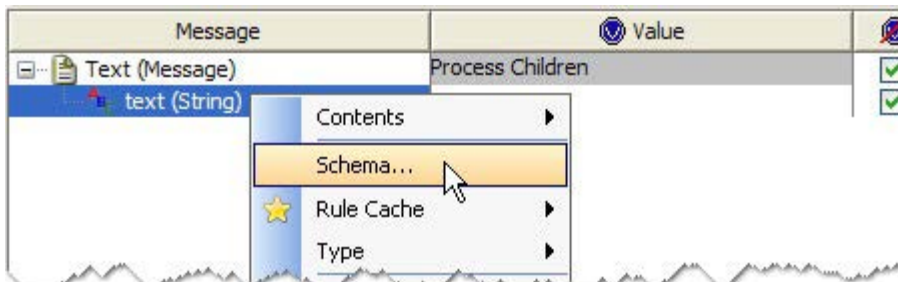
Once a schema has been added manually or as part of a synchronised resource, it can be applied to any message within Rational Integration Tester that supports its schema type. For example, you would use the HTTP transport to send a request to a Web Server hosted service.

NOTE: The following example illustrates how to apply a WSDL schema to an HTTP-based message. However, the general steps are the same for applying different schemas to other message types.

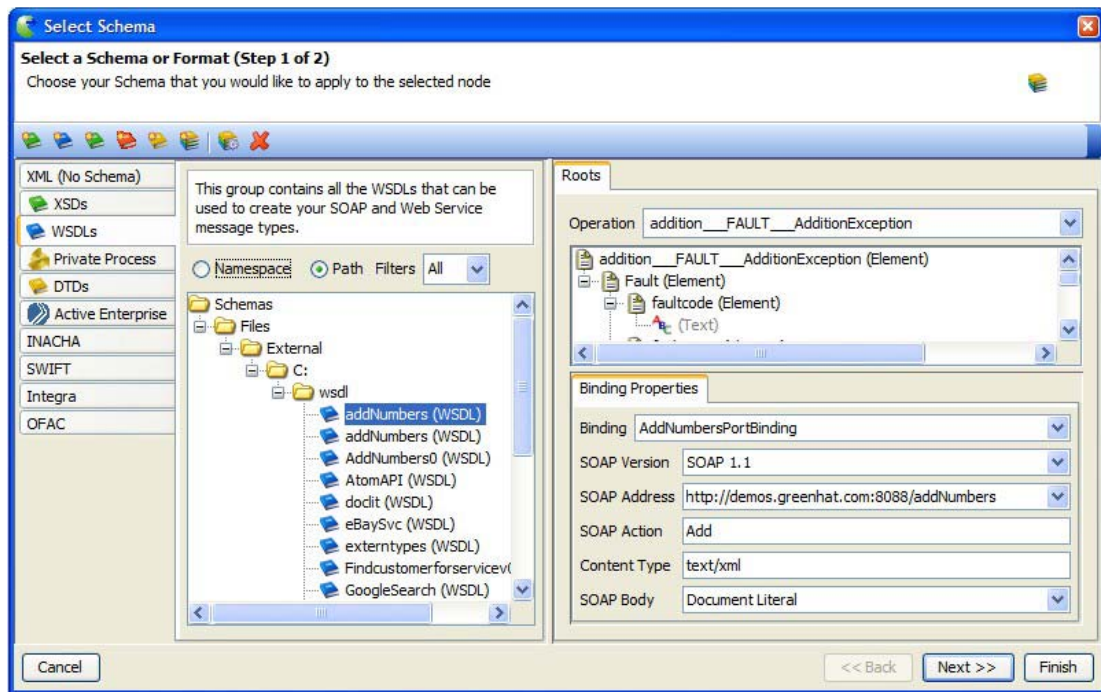
1. Create a Send Request action in a new or existing test.
2. Select an HTTP transport and the Text formatter, as shown below.



3. Apply the schema to the message by right-clicking the text node and selecting **Schema** from the context menu.



The **Select Schema** wizard is displayed.



4. On the left side of the dialog, select the WSDL tab and locate the desired schema within the Rational Integration Tester project.

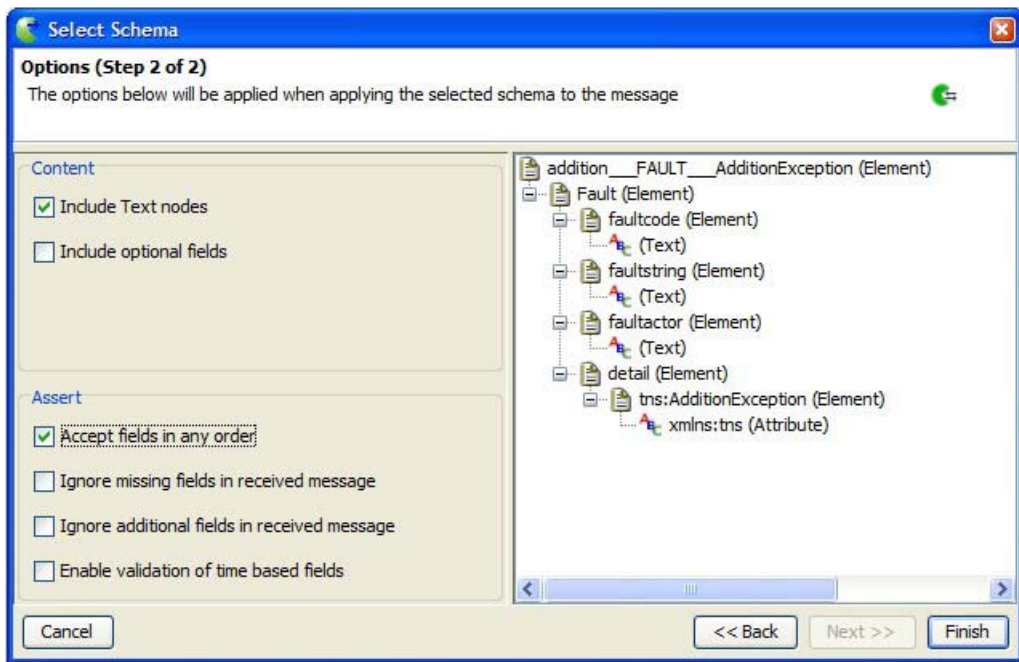
NOTE: The available schema types to select will vary according to the selected message format. For example, Java Object schemas can only be applied when the message format is Byte Array.

5. Select an operation under the **Roots** tab on the right side of the dialog.

Operations are divided into Input and Output operations. Essentially, an input message is a message that would be sent to a Web Service as a request, and the output operation is the structure of the message expected as a reply from the Web Service.

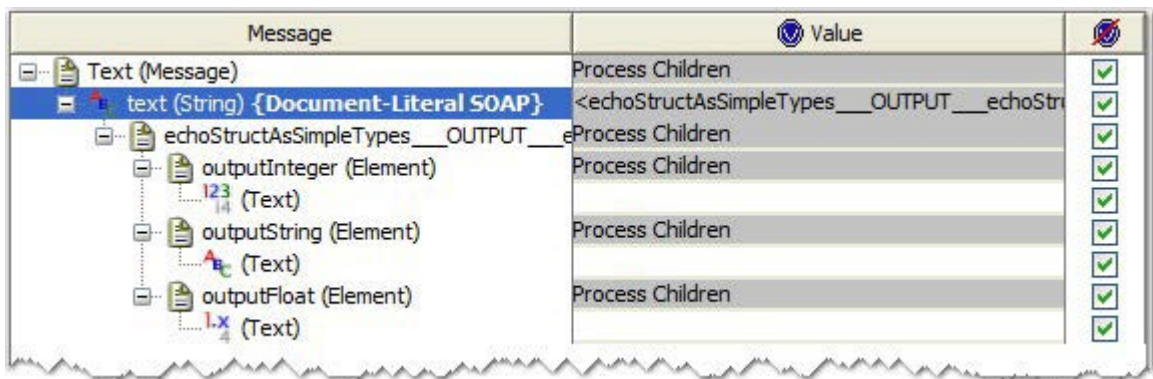
6. Select the desired operation, or select **Show All** to see all available messages. The Binding properties are automatically detected from the schema.
7. Click **Next** to configure additional message and validation options.

The second page of the wizard is displayed.



8. Select the desired content and assertion options, then click **Finish** to apply the schema and close the wizard.

The basic message structure and required fields are populated in the message editor, and the message type (for example, Document-Literal SOAP) is determined from the schema.



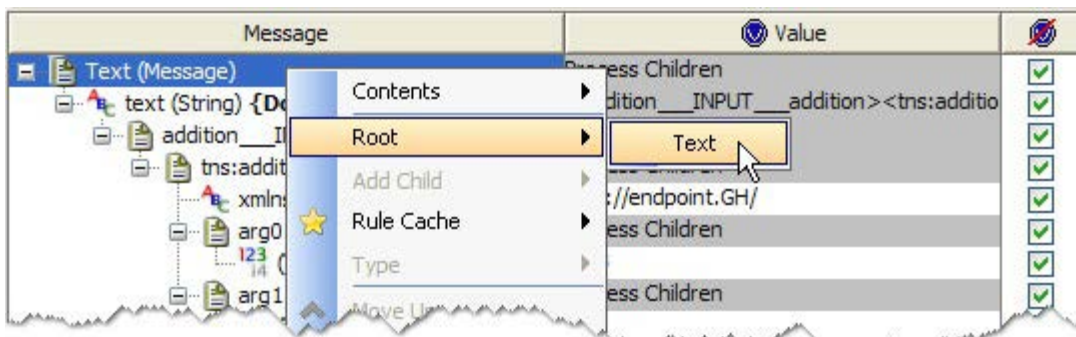
Any optional fields defined in the schema will be available from the **Add Child** context menu option, and they can be added to the message as defined by the rules in the schema.

12.3 Changing Schemas

Over time, it is likely that a schema in use for message construction will change. To update the schema in the Rational Integration Tester project, use the Synchronisation view in Architecture School (see [Synchronisation](#)).

The following example shows how an updated schema can be applied to an XML message. The process is the same for other schema-based message types.

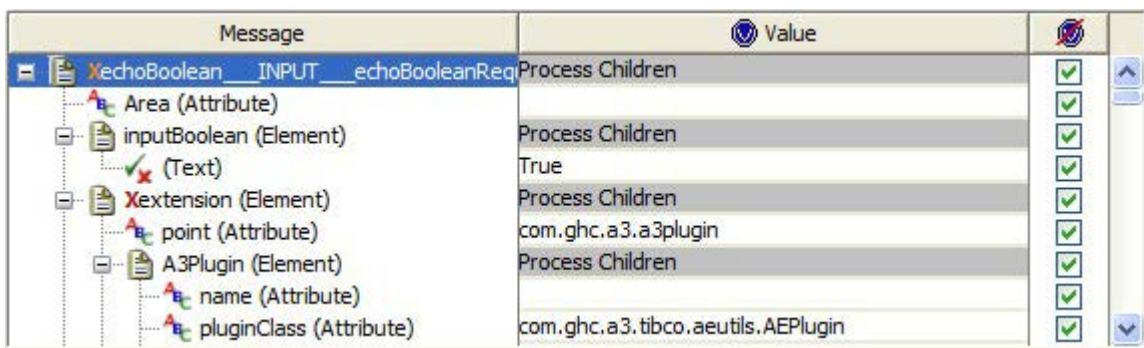
1. In the message editor, right-click the base message node and select the desired structure to apply from the **Root** option.



2. You will be prompted about how to apply the new root to the message.

If you retain the content, fields will be added to the message and existing fields will be retained with previous data left intact. If you overwrite the content, the schema will be applied and all of the data fields will be initialized as empty. To cancel the change, click **Cancel**.

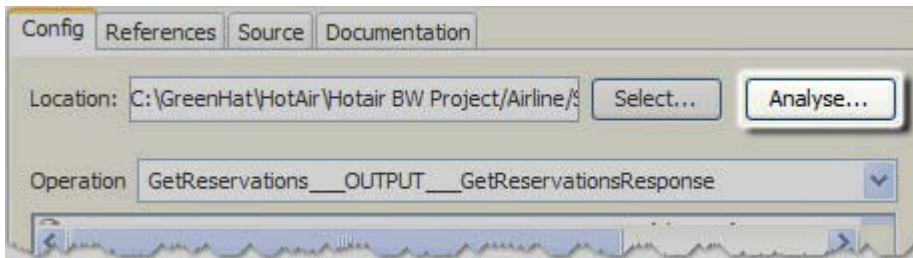
In the screenshot shown, the element **Area** is a new one so appears with no content. The element **extension** is not present in the latest schema. It has been retained with its data, but it is marked with a red “X” to show that it violates the schema. Data from this field should be copied, then the field can be removed.



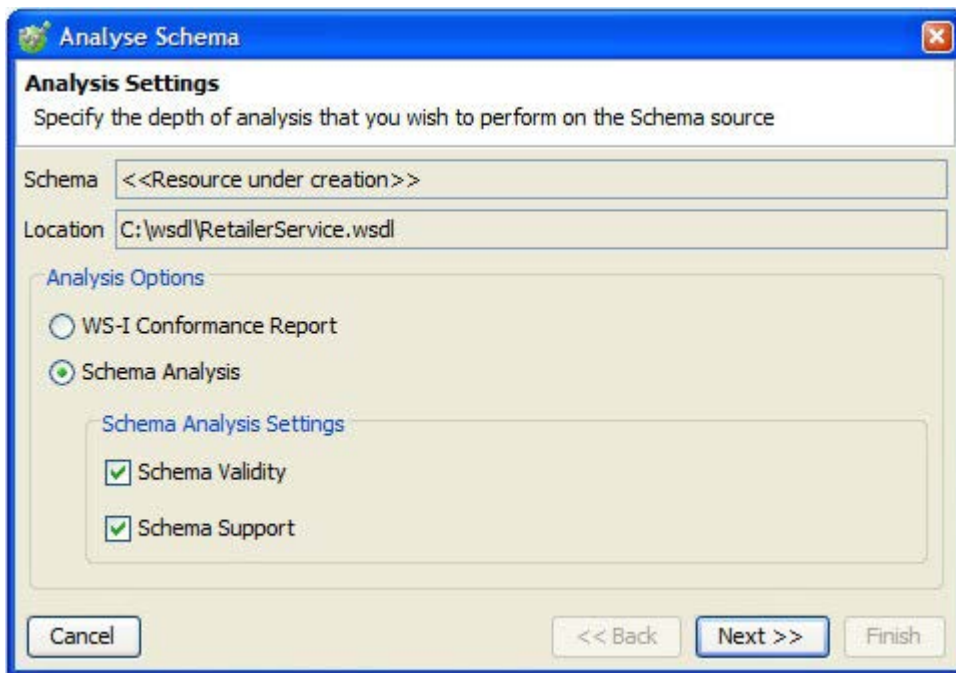
12.4 Analysing Schemas

You have the option to analyze certain schema resources (that is, XSDs, WSDLs, and DTDs) after they have been imported into Rational Integration Tester. This can be useful in ensuring that there are no problems with the resource.

Once the schema has been selected in the Schema Library, click the **Analyse** button.



For WSDLs, two analysis options are available in the **Analysis Settings** dialog. For XSDs and DTDs, only the basic analysis is available.

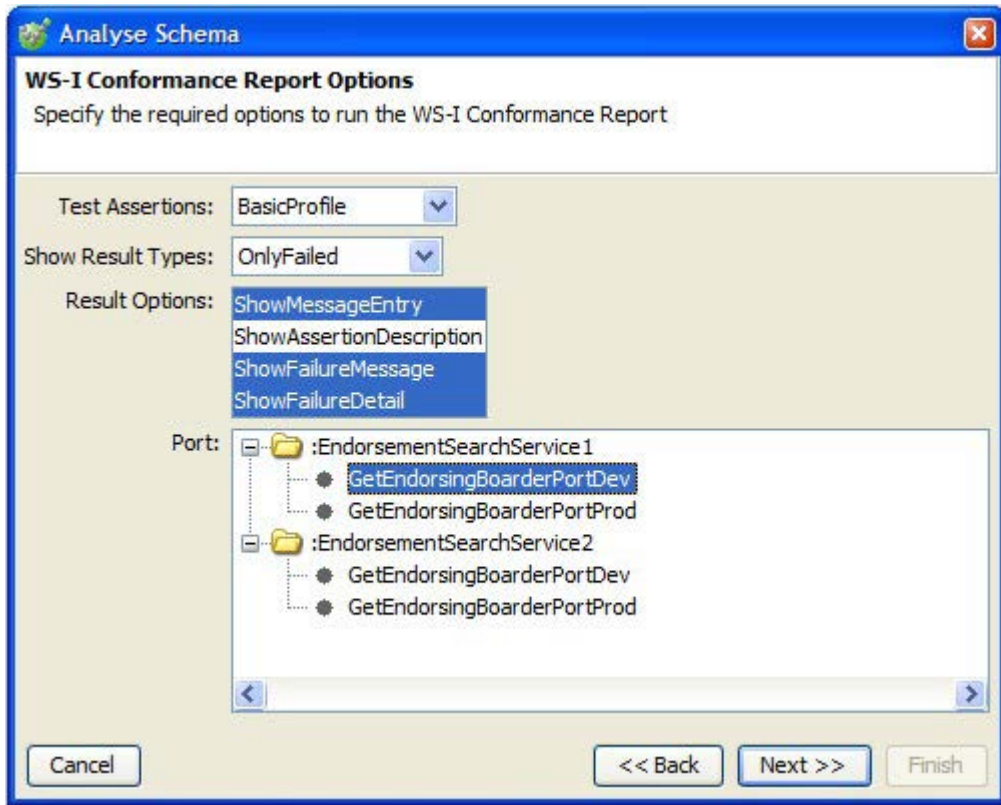


See [Running the WS-I Conformance Report](#) and [Performing Basic Schema Analysis](#) for more information.

12.4.1 Running the WS-I Conformance Report

The **WS-I Conformance Report** will generate a report that details if and how the schema complies with the Web Services Interoperability Profile guidelines.

1. After selecting the WS-I Conformance Report in the Analysis Settings dialog, click **Next** to proceed.
2. The **WS-I Conformance Report Options** dialog is displayed.



3. Select the options you want to use for generating the report.
 - **Test Assertions:** Select the type of analysis to perform
 - **Show Result Types:** Select which results to display in the report
 - **Result Options:** Select which options to display with results (use Ctrl or Shift to select multiple options)
 - **Port:** If available, select the port in the schema that you want to analyse.

-
- Click **Next** to proceed and the report is generated and displayed in the next window.



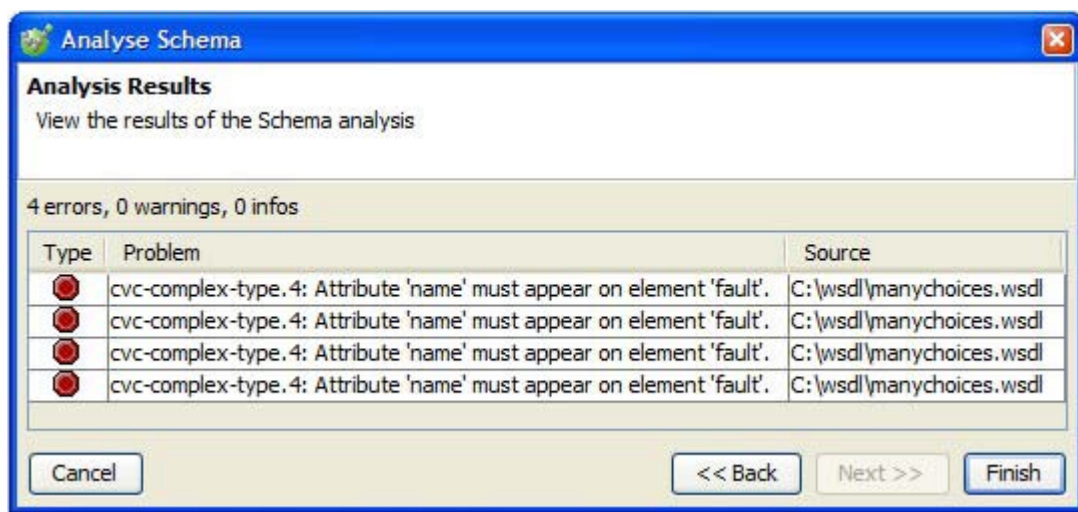
- When finished with the report, click **Finish**.

12.4.2 Performing Basic Schema Analysis

Schema Analysis provides two options for analysing the schema:

- **Schema Validity** check the schema for general errors (for example, unresolved references)
- **Schema Support** lists features within the schema that Rational Integration Tester does not currently support

1. After selecting the desired options, click **Next** to proceed.
2. The results are displayed in the **Analysis Results** dialog.



NOTE: If no errors or conflicts are found, the results dialog will be empty.

3. When finished with the report, click **Finish**.

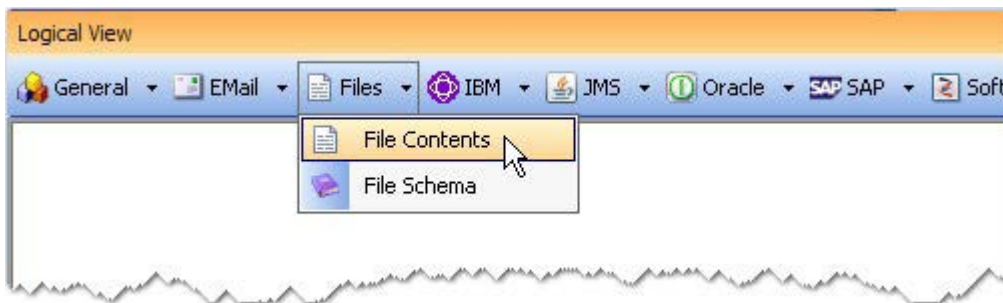
12.5 File Schemas

File schemas can be used to define the contents one or more files that you intend to write to (publish) or read from (subscribe) in Rational Integration Tester tests. For files that are made up of a number of records, using a file schema can save users from having to write to or read from a file over and over again (that is, creating a publisher or subscriber for each record in the file). The file schema lets Rational Integration Tester iterate over the entire file, reading the desired records according to the way the file is defined.

The file schema is used in conjunction with the **File Contents** resource (created in the Logical View of Architecture School), which defines the file containing the records to be processed. For more information about working with files in Rational Integration Tester, refer to *IBM Rational Integration Tester Reference Guide for Files*.


The following example illustrates how to create a file schema that defines a file contents resource.

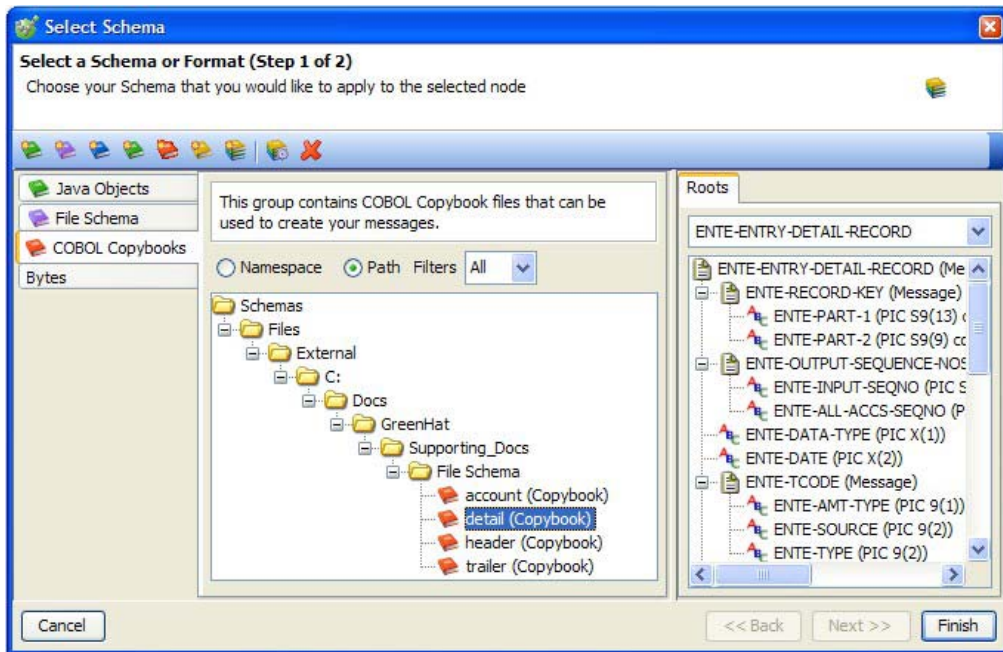
1. In Rational Integration Tester, create a new project or open an existing one.
2. Go to Architecture School (**F7**) and select the Logical View.
3. Create a new file resource by selecting **File Contents** from the **Files** menu.



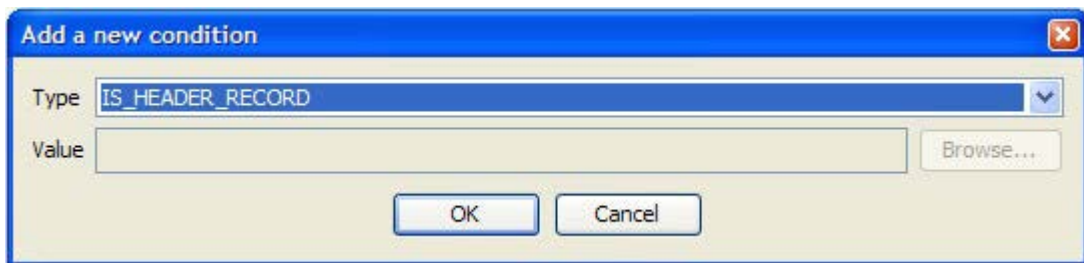
4. Provide a name for the file resource and double-click to edit it.
5. Under the **Settings** tab, click **Browse** next to **File Name** and select the file to be processed in Rational Integration Tester tests.
6. Click **OK** to save the changes and close the **File Contents** editor.
7. Select the **Schema Library** view and add the schemas that will define the records within the file to be processed.

NOTE: Records within the file are treated as byte fields. Currently, only COBOL Copybook schemas can be used for defining file records.

8. Click  to add a file schema.
9. Enter a name for the schema based on the file type with which it will be used.
10. On the right side of the Schema Library view, the file schema definition is displayed under the **Config** tab.
11. Add records (that is, the schemas imported earlier that define the file records) to the file schema using the **New Record** button.
12. In the Schema wizard, select the desired schema (or add one) and click **Finish**.



13. Add conditions to the defining records using the **New Condition** button.



Conditions are used to help Rational Integration Tester understand the structure of the file being processed. If available for a condition, you can select a field within the record to use for matching the condition in the **Value** field.

In the example file schema shown below, the file contains a header record, a number of account records, and a trailer (footer) record. Within each account record, one or more account detail records may appear.

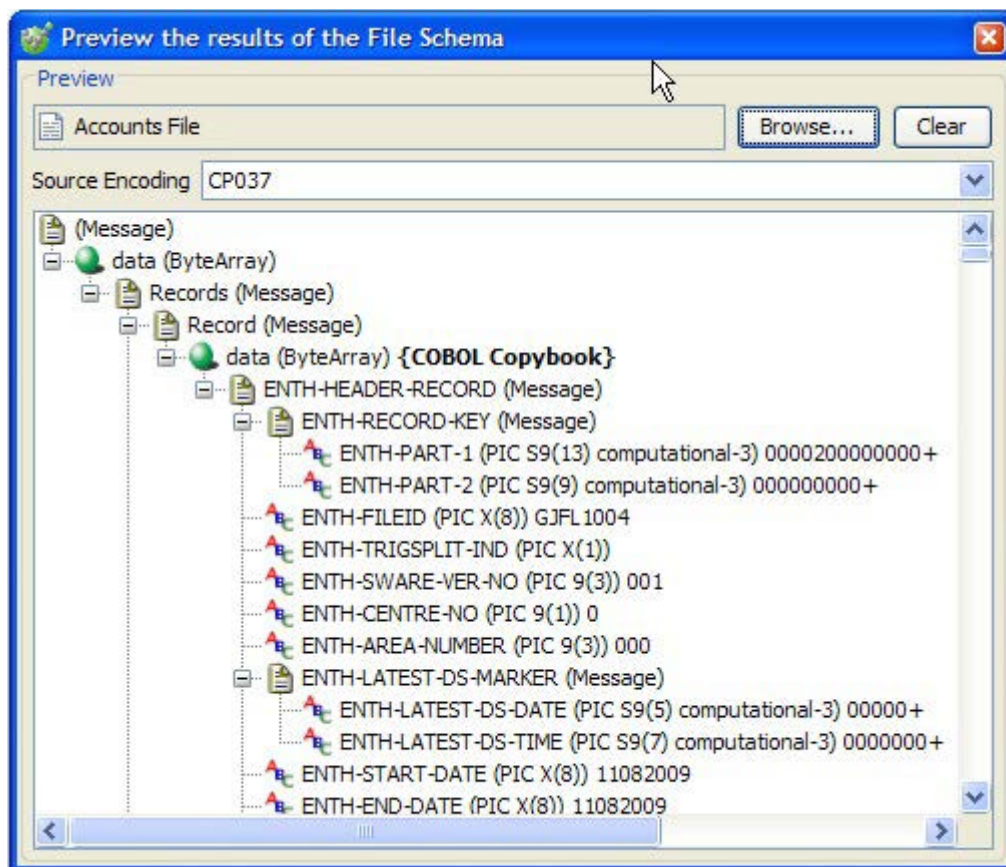


Record definitions can be moved and deleted using the **Remove Record**, **Move Up**, and **Move Down** buttons. Conditions can not be moved, but they can be deleted using the **Remove Condition** button.

When finished, or at any time, you can click the **Preview** button to apply the file schema to the file to be processed, ensuring that the structure of the file has been defined properly.

14. In the **Preview** dialog, click **Browse** to select the File Contents resource (created in the Logical View) that defines the file to be processed.
15. Select the desired encoding type to apply from the **Source Encoding** dropdown menu.

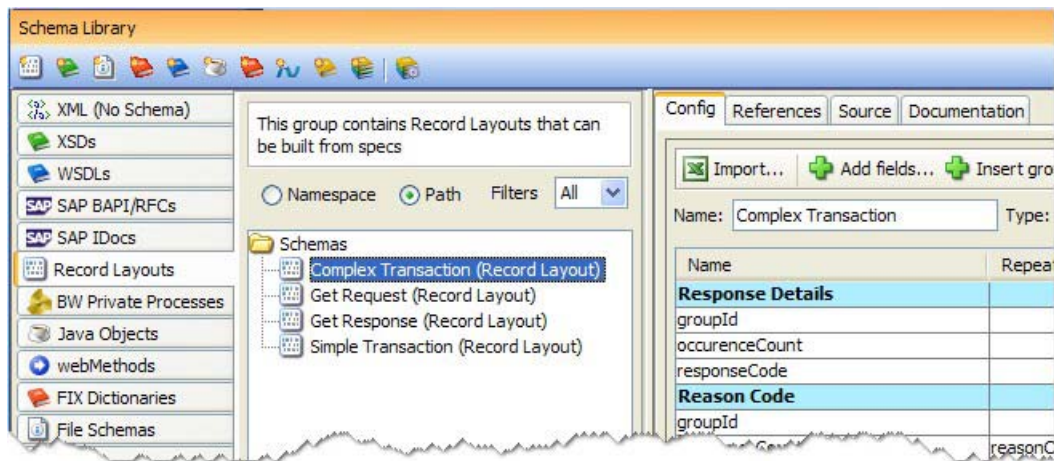
A preview of the file to be processed is displayed.



12.6 Record Layouts

While Rational Integration Tester's built in support for schema types and message formats is extensive, customers may need to process and validate messages that use a customized schema or format. For such messages, Rational Integration Tester provides record layouts, which can be used to define a file schema composed of fixed width, custom, or delimited fields. The customized schema can then be applied to a bytes field in a Rational Integration Tester message.

Record layouts are managed under the Record Layouts tab in the Schema Library.



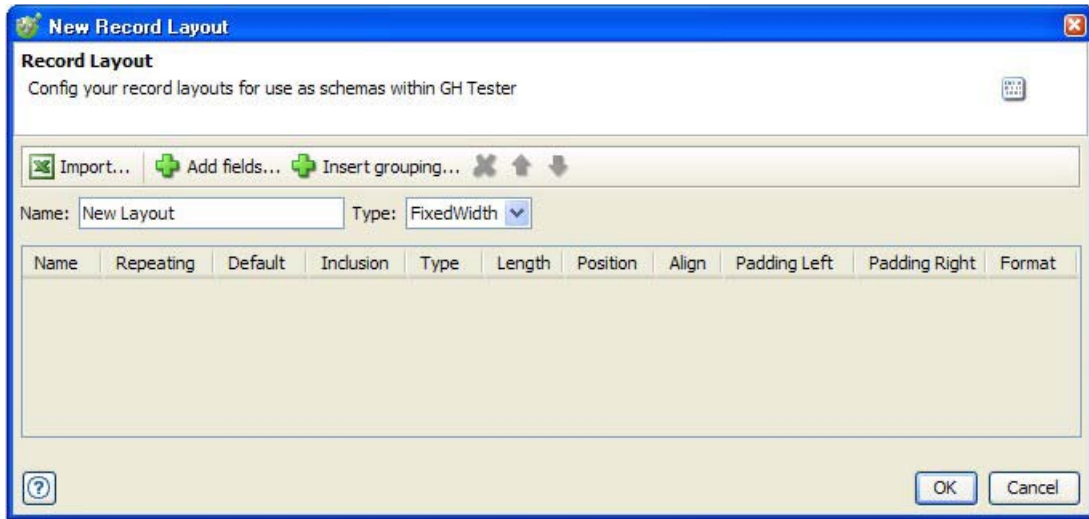
Existing schemas (record layouts) can be sorted or filtered as usual. The details of any selected schema are displayed – and can be modified – under the **Config** tab on the right side of the view.

12.6.1 Creating a New Record Layout

Follow the steps below to create a new record layout in Rational Integration Tester.

1. In the Schema Library, select the Record Layouts tab and click the **Record Layout** button  at the top of the view.

The **New Record Layout** window is displayed.



2. In the **Name** field, provide a meaningful name for the new layout. The new layout will be listed in Rational Integration Tester according to this name.
3. From the **Type** field, select whether you want to create a layout containing fixed width fields (see [Creating Fixed Width Layouts](#)), a custom schema (see [Creating Custom Layouts](#)) for messages that do not adhere to a fixed width format, or a delimited layout (see [Creating Delimited Layouts](#)).

NOTE: If desired, you can import a record layout that is defined in an Excel spreadsheet by clicking the Import button. See [Import Record Layouts](#) for more information.

Creating Fixed Width Layouts

Fixed width record layouts contain one or more fields that can be (optionally) divided into logical groupings. The groupings do not appear in the finished layout/schema, but they can be used to help organize the fields it contains.

To add a grouping to the layout, click the **Insert grouping** button in the **New Record Layout** window. When prompted, provide a name for the grouping by entering it in the field provided or by selecting a recently used name from the dropdown arrow.



To add a field to the layout, click the **Add fields** button. The field configuration window is displayed.

Create new Fields for the Record Layout

Record Field

Name:

☐ Use for grouping

Type:

Default Value:

Inclusion:

Format

Category: (List: Date Time, String, Number, Currency)

Sample:

Tag Value...

Configuration: ☐ Custom Format:

String Summary

The String Pattern will use a java.util.Formatter to format the String. The argument list will only consist of the value from this action.
A full description of the java.util.Formatter class exists at <http://java.sun.com/javase/6/docs/api/index.html?java/util/Formatter.html>

Alignment

Length: ☐ None ☐ From Schema ☒ Custom

Justification: ☒ Left ☐ Right ☐ Centered

Character:

☐ Trim if too long?

Enter the basic details about the field under the **Record Field** area. **Name** is the field name to be displayed in the layout, **Type** is the data type that the field contains (which can be any of the supported field types in Rational Integration Tester), **Default Value** is an optional default value that can be applied to the field if none is present in the message, and **Inclusion** indicates if the field must appear in the message or if it is optional.



To create a grouping in the layout, enable the **Use for grouping** option.




Under the **Format** area you can apply additional formatting to the selected field. For more information about using the format options, see [Generating Random Data from Field Values](#) in the “Messages” chapter. The options available under **Format** are

the same.


To create the field and continue with another in the same window, click **Next**. To create the field and return to the New Record Layout window, click **Finish**. If you have created a field and moved on, you can click **Back** to manage the details for that field again.

The details of each field can be modified in the **New Record Layout** window, as follows:

Name	The name of the field.
Repeating	If the selected field contains repeating elements within the grouping, they can be selected by clicking the  icon.
Default	Enter a default field value to be applied in a message if no value is present.
Inclusion	Select whether the field is required to be included (Mandatory) in the message or not (Optional). Conditional?
Type	Select the field type from those available in Rational Integration Tester.
Length	Enter the number of characters in the message that are designated for the selected field.
Position	Indicates the position in the message (starting from zero) of the selected field, based on the size of all preceding fields.
Align	Select the alignment of the data for the selected field in the message (left, center, or right).
Padding Left	Enter the character that is used in the message as padding to the left of the field data (for right- or center-aligned fields).
Padding Right	Enter the character that is used in the message as padding to the right of the field data (for left- or center-aligned fields).
Format	Click the  icon to apply and custom formatting to the field.

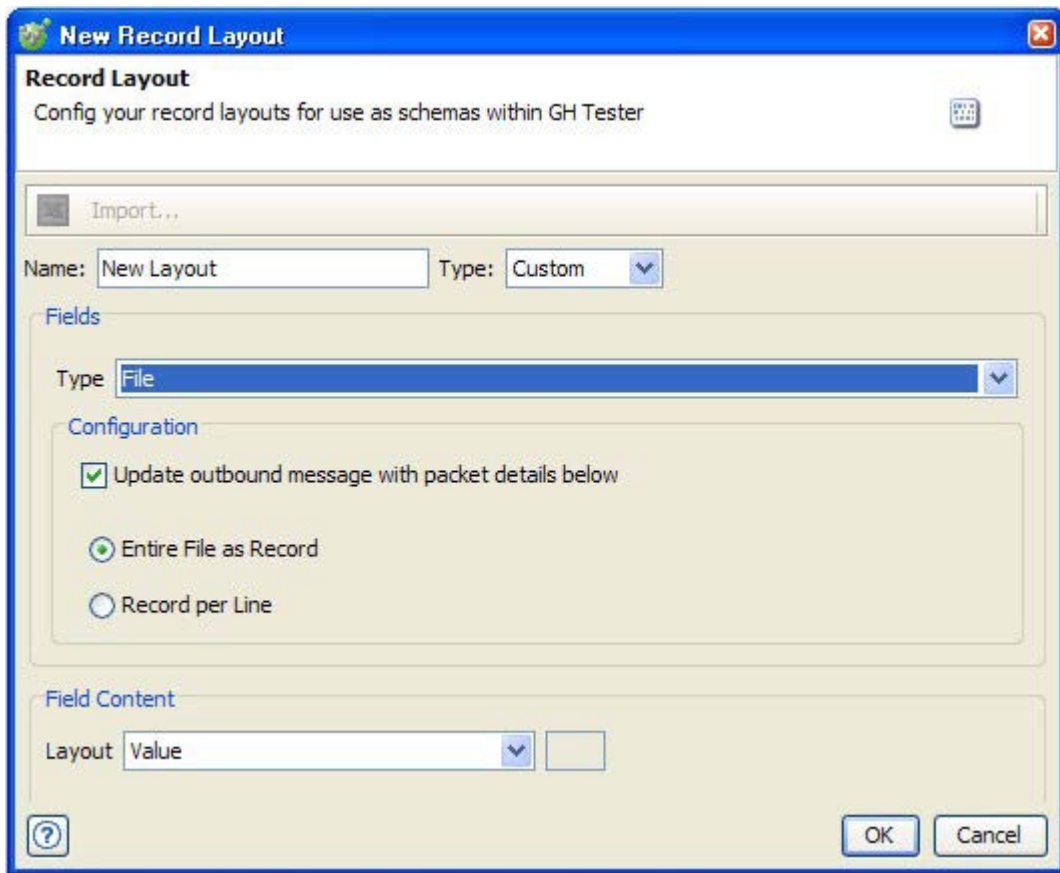
Selected fields and groupings (one or more) can be removed from the layout by clicking the  icon. To move one or more selected fields or groupings up or down within the layout, click the  or  icon.

The editable properties for each field can be modified by pressing the **F2** key. You can move around the cells within the layout using the arrow keys. To apply a change in the current cell, press **Enter**. To apply a change and move to the next cell to the right, press **Tab**. To move to the cell below the current one, press **Enter**.

After you are finished creating the layout or after making any changes, click the **Save** icon  or press **Ctrl + S** to save the layout.

Creating Custom Layouts

Custom layouts let you define how to divide a file into different records. The individual records can be configured as values, or they can be further divided into name-value pairs. For example, john | david | miller | represents three values while first,john | middle,david | last,miller represents three name-value pairs.



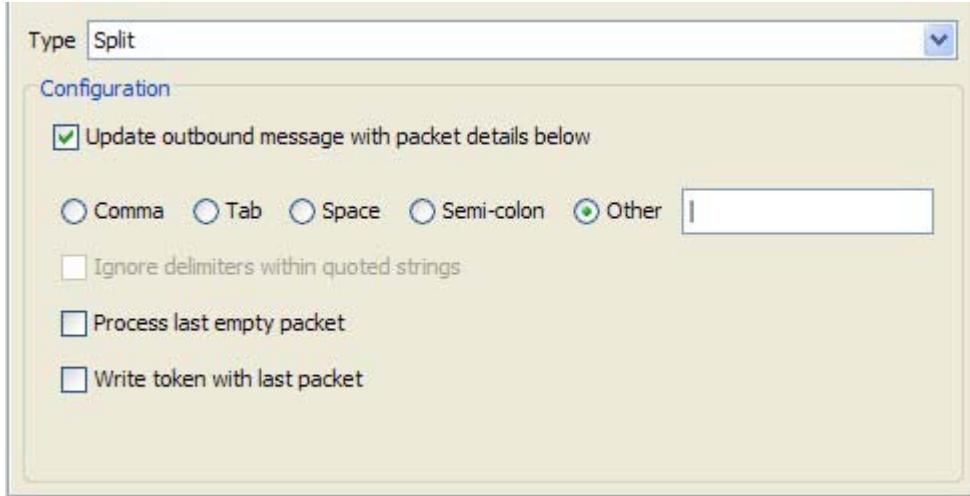
In the **Fields** panel, you select how each record can be read or should be written in the **Type** field, configure the details of each type under **Configuration**, and specify if fields contain values or name-value pairs – including the name-value delimiter – under **Field Content**.

NOTE: For all record definition types, you have the option to update outbound messages with relevant packet information (that is, the length of the remaining data) according to the record type. To include this information in outbound messages (that is, when publishing),

enable the **Update outbound message with packet details below** option.

Split

The **Split** option can packetize (split) the contents of files based on a user-defined delimiter.



The screenshot shows a configuration window for the 'Split' option. At the top, a dropdown menu is set to 'Split'. Below it, the 'Configuration' section contains several options: a checked checkbox for 'Update outbound message with packet details below', a row of radio buttons for 'Comma', 'Tab', 'Space', 'Semi-colon', and 'Other' (which is selected), and a text input field next to 'Other' containing a vertical bar character '|'. Below these are three unchecked checkboxes: 'Ignore delimiters within quoted strings', 'Process last empty packet', and 'Write token with last packet'.

You can select one of the existing delimiter types (comma, tab, space, semi-colon), or select **Other** and enter the delimiter character(s) in the field provided.

When reading records, use the **Process last empty packet** option depending how the record ends. Enable this option if the record ends with your delimiter and you want to process one more packet as an empty string “”. Disable this option if the delimiter indicate that there are no more packets.

When writing to a record, enable the **Write token with last packet** if you want the packetizer to write out the delimiter as the last character, so the record ends with a delimiter. For example: “...|myfieldvalue|” when enabled and “...|myfieldvalue” when disabled.

Length

The Length option can be used to iterate over the contents of a file to extract multiple records.

The screenshot shows a configuration window for the 'Length' type. At the top, a dropdown menu is set to 'Length'. Below it, a 'Configuration' section contains four radio buttons: 'Fixed Length' (unselected), 'Token' (unselected), 'Offset' (selected), and 'Prefix' (unselected). To the right of these buttons are input fields: '24' for Fixed Length, an empty field for Token, '16' for Offset, and an empty field for Prefix. Below the radio buttons is a 'Size' input field with the value '4' and a 'Format' dropdown menu set to 'Bytes'. At the bottom, there is a checkbox labeled 'Swap Bytes' which is currently unchecked.

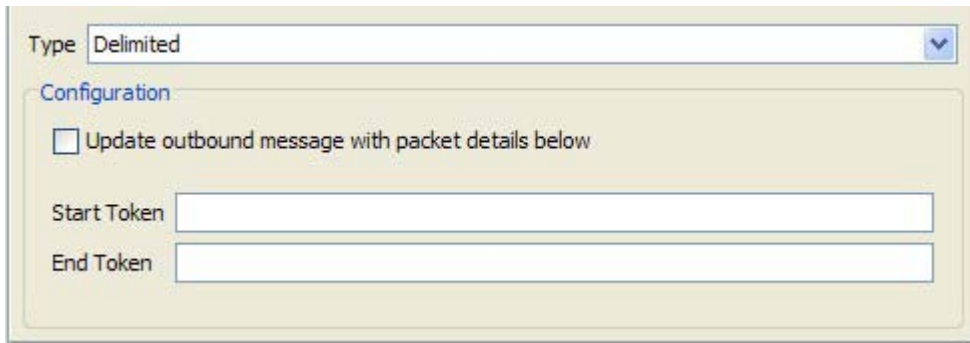
Fixed Length	Each record will be the same designated size.
Token	The string of characters that marks the start of each record.
Offset	Each record starts after the entered number of bytes. In the above example, 16 bytes of data precede the length information.
Prefix	A number of bytes / characters at the start of the record that denote the length of the record.

When using the **Token**, **Offset**, or **Prefix** modes, the following options control how the actual packet length will be read from the stream of information.

Size	The number of bytes or characters that contain the length information.
Format	Indicates whether the prefix values should be treated as raw values (Bytes) or translated from their ASCII equivalent (ASCII).
Swap Bytes	Indicates whether the transport should swap the order of the bytes before treating the data as the length of the record.

Delimited

The **Delimited** option can be used to iterate over the contents of a file to extract multiple records that are designated by a start and end token.



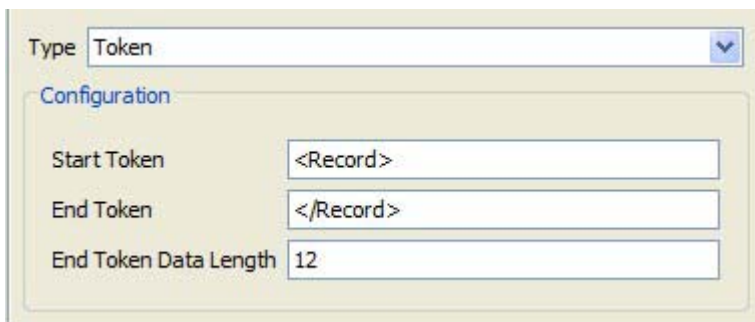
The screenshot shows a configuration window for the 'Delimited' option. At the top, a 'Type' dropdown menu is set to 'Delimited'. Below it, a 'Configuration' section contains a checkbox labeled 'Update outbound message with packet details below', which is currently unchecked. Underneath the checkbox are two text input fields: 'Start Token' and 'End Token', both of which are empty.

Start Token A series of characters that denotes the start of a record.

End Token A series of characters that denotes the end of a record.

Token

The **Token** option is similar to the Delimited option, but can be used when you need to specify additional data after the end token.



The screenshot shows a configuration window for the 'Token' option. At the top, a 'Type' dropdown menu is set to 'Token'. Below it, a 'Configuration' section contains three text input fields. The 'Start Token' field contains the text '<Record>'. The 'End Token' field contains the text '</Record>'. The 'End Token Data Length' field contains the number '12'.

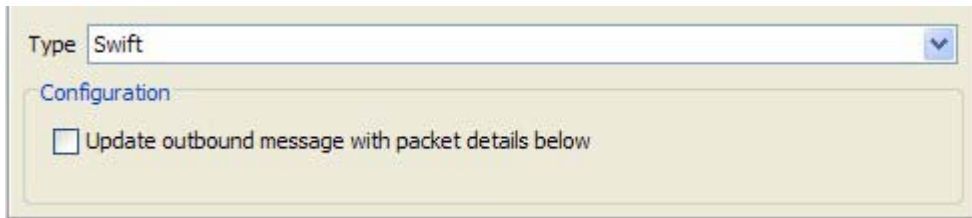
Start Token A series of characters that denotes the start of a record.

End Token A series of characters that denotes the end of a record.

End Token
Data Length An optional quantity of data that can be present following the end token.

Swift

The Swift option can be used to break up the contents of the file into a Swift message based on simple Swift rules (that is, {n:...}).



A screenshot of a configuration dialog box for the 'Swift' type. The 'Type' dropdown menu is set to 'Swift'. Below it, the 'Configuration' section contains a single checkbox labeled 'Update outbound message with packet details below', which is currently unchecked.

File

When using a file as a single record, the record definition can be configured to use the entire file as a single record, or to treat each line as a record.

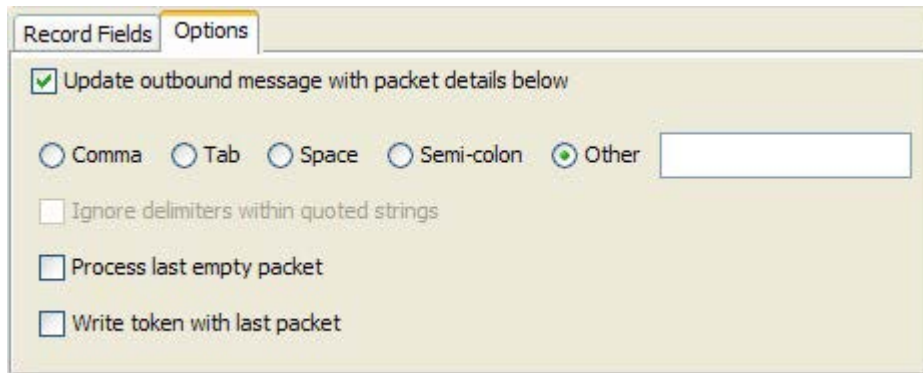


A screenshot of a configuration dialog box for the 'File' type. The 'Type' dropdown menu is set to 'File'. Below it, the 'Configuration' section contains two radio button options: 'Entire File as Record' (which is selected) and 'Record per Line'.

Creating Delimited Layouts

Delimited record layouts are used for files containing one or more records that are separated by a user-defined delimiter. Please see [Creating Fixed Width Layouts](#) for details about creating and modifying the record fields of a delimited layout as it is the same as with a fixed width layout. The only difference is the number of fields that can be configured (that is, only Name, Default, and Type are available in a delimited layout).

Under the **Options** tab, you can configure the packetizer and delimiter details.



The screenshot shows a configuration window with two tabs: 'Record Fields' and 'Options'. The 'Options' tab is active. It contains the following options:

- ☒ Update outbound message with packet details below
- Delimiter selection: ☐ Comma, ☐ Tab, ☐ Space, ☐ Semi-colon, ☒ Other (with an adjacent empty text input field).
- ☐ Ignore delimiters within quoted strings
- ☐ Process last empty packet
- ☐ Write token with last packet

NOTE: You have the option to update outbound messages with relevant packet information (that is, the length of the remaining data) for delimited records. To include this information in outbound messages (that is, when publishing), enable the **Update outbound message with packet details below** option.

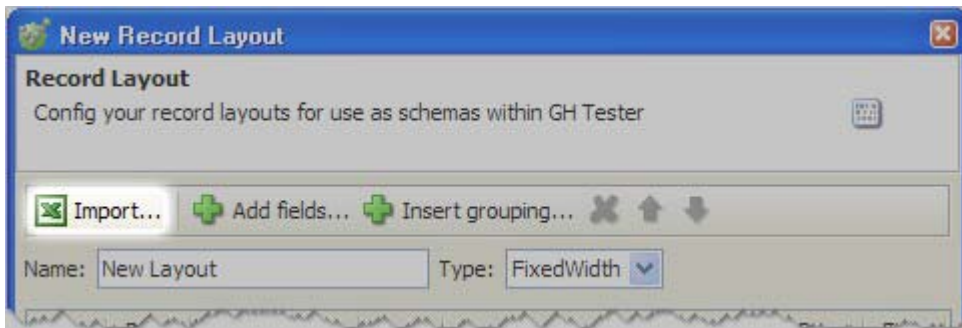
You can select one of the existing delimiter types (comma, tab, space, semi-colon), or select **Other** and enter the delimiter character(s) in the field provided.

When reading records, use the **Process last empty packet** option depending how the record ends. Enable this option if the record ends with your delimiter and you want to process one more packet as an empty string “”. Disable this option if the delimiter indicate that there are no more packets.

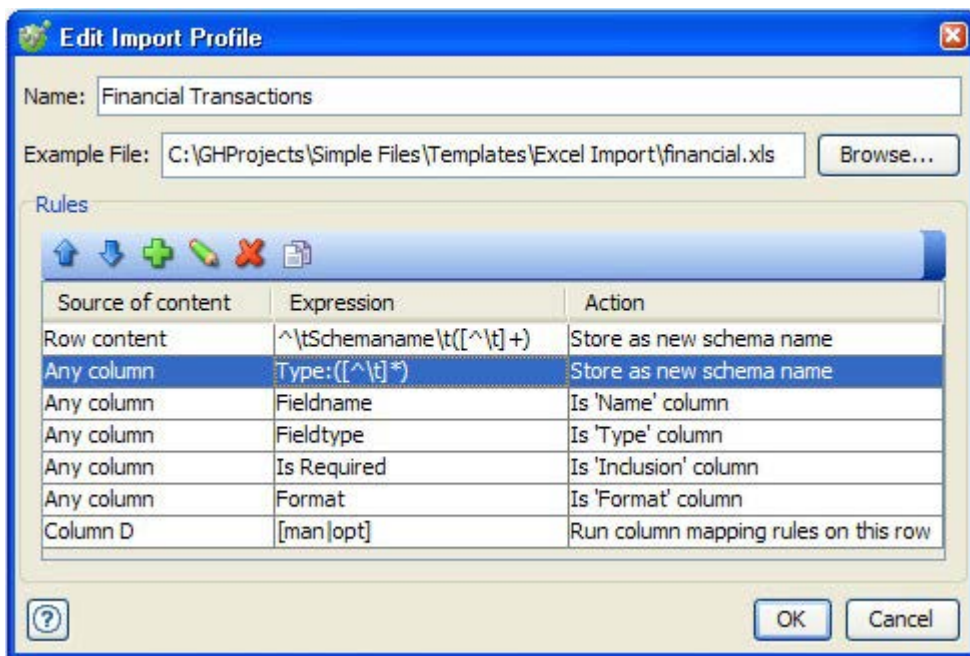
When writing to a record, enable the **Write token with last packet** if you want the packetizer to write out the delimiter as the last character, so the record ends with a delimiter. For example: “... | myfieldvalue |” when enabled and “... | myfieldvalue” when disabled.

12.6.2 Import Record Layouts

If you have fixed width record layouts that are specified in an Excel spreadsheet, you can import the specification and define rules for parsing it using the **Import** feature – for new or existing layouts.



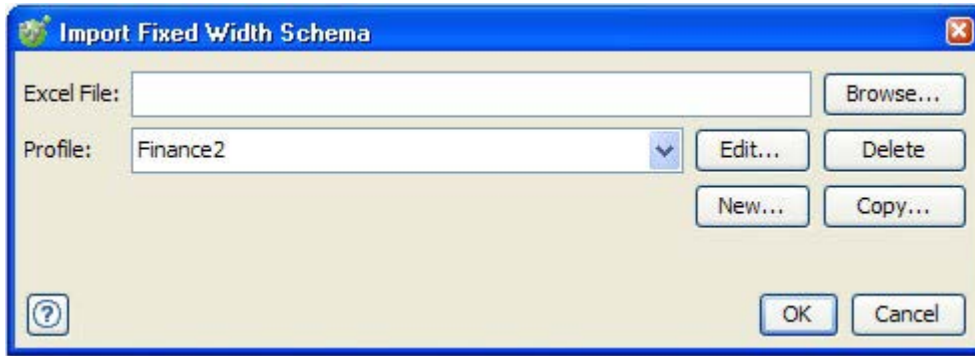
To import a fixed width layout, you need the Excel spreadsheet that defines it and an import profile that is configured in the wizard. The import profile is given a name and configured with a set of rules that will be applied to a selected spreadsheet for creating the layout.



1. To launch the wizard, click **Import** in the **New Record Layout** wizard or under the **Config** tab of an existing layout.

NOTE: The results of importing a layout will be added to any existing fields and groupings in an existing layout.

The **Import Fixed Width Schema** dialog is displayed.

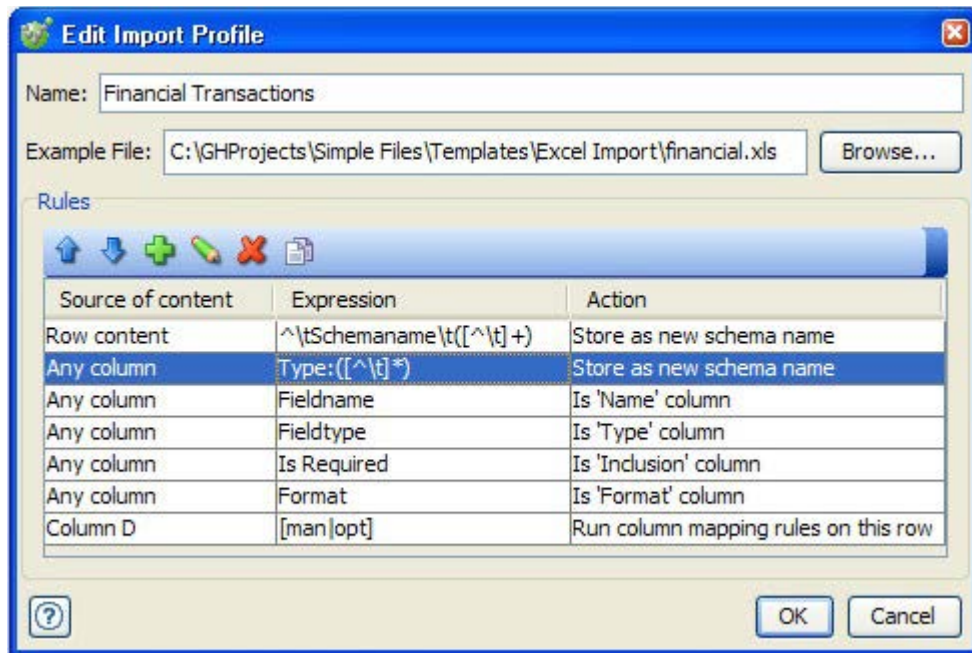


2. Select the spreadsheet that specifies the layout by clicking **Browse** next to the **Excel File** field.
3. Select an import profile from the drop-down menu in the **Profile** field.
4. Click **OK** to apply the profile to the selected spreadsheet and create the layout.

NOTE: If you need to edit the selected profile, click **Edit** (see [Editing Profiles](#)). To create a new profile, click **New** (see [Editing Profiles](#)). To copy the selected profile and open the copy for editing, click **Copy** (see [Editing Profiles](#)). If you want to delete the selected profile, click **Delete**.

Editing Profiles

When creating a new profile or editing an existing profile (including copies), the profile editor is displayed.



The profile editor manages the rules that are applied to the spreadsheet when creating the layout. Next to the **Example File** field, click **Browse** to select the specification spreadsheet, which is used in the rule editor to show how expressions will match the content. Rules are applied to the spreadsheet in the order in which they are displayed.

Rules are managed using the toolbar icons at the top of the rules table, as follows:

Icon	Use
	Move the selected rule up within the list.
	Move the selected rule down within the list.
	Create a new rule below the currently selected rule (see Editing Rules).
	Edit the selected rule (see Editing Rules) – you can also double-click a rule in the list to open it for editing.
	Delete the selected rule.
	Copy the selected rule and paste it below the current location.

Editing Rules

The rules used to extract information from the spreadsheet and generate record layouts are regular expressions.

Edit Rule

Create a new rule

Use this panel to create a rule to extract the desired information from the spreadsheet.

Source of content: Row content

Matching the expression: `^\\tSchemaname\\t([\\^\\t]+)`
(tabs '\\t' separate columns in Excel spreadsheets)

☒ Matches
☐ Must not match

Preview

Matched 1 times

Match Instance: 1 Captured text: Financial

And translate using this table

From	To
------	----

Action: Store as new schema name

☒ And don't process any more rules for this content.

OK Cancel

For each rule, the source of the content to match in the spreadsheet is selected from the **Source of content** drop-down. Available sources are **Filename** (for example, to verify that the filename selected is a valid file), **Sheet name** (if the spreadsheet contains multiple sheets with content, one or more sheets can be skipped by name), **Row content** (searches row by row), **Any column** (searches column by column), and **Column *n*** (searches one of the specific columns in the spreadsheet containing data).

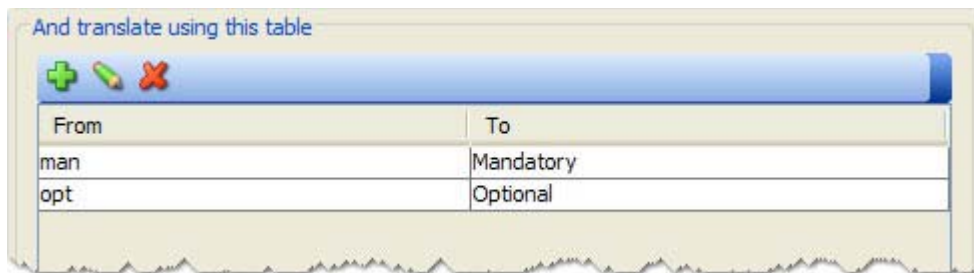
The expression to be applied to the selected content source is entered in the **Matching the expression** field – expressions use groupings, so the content to be extracted is what is found in the first grouping.

NOTE: The tab character in the expression (\t) is used to separate columns in the spreadsheet.

You must specify whether the entered expression should be used to include (**Matches**) or exclude (**Must not match**) the content of a cell in the spreadsheet.




The Preview pane displays the number of matches for the entered expression (**Matched *n* times**), and the text that will be captured by the expression is displayed in the **Captured text** field. When multiple matches are found, you can use the up and down arrows to move through the matches in the spreadsheet. The position of the current match in the spreadsheet is displayed in the **Match Instance** field.

You can translate matched content by adding rules in the translation table.



From	To
man	Mandatory
opt	Optional

In the example shown above, when “man” is found in a matched cell in the spreadsheet, it will be translated to “Mandatory” (for example, for the Inclusion column) in the record layout.

Click  to add a rule to the table, click to  edit a selected rule, or click  to delete a selected rule. When adding or editing a rule, enter the value found in the spreadsheet to the **From** field and the value to which it should be translated in the **To** field.



Add Translation

From: man

To: Mandatory

? OK Cancel

From the **Action** drop-down, select the action to take on the text that is matched with the expression. The available rules are described in the following table:

Action	Result
Start running actions	Starts running the actions of rules that match the current content, used to turn on action processing after it has been turned off with Stop running actions until next start .
Stop running actions until next start	Stops running the actions until Start running actions is encountered. This can be used to skip portions of a spreadsheet that are not relevant.
Ignore	Used to ignore matched content in the spreadsheet.
Store as new schema name	Stores the matched text as the name of the new record layout. NOTE: This action is required – the layout cannot be created without a name.
Store as new record grouping	Stores the matched text as a grouping record in the layout.
Is '<field>' column	These actions designate which columns in the spreadsheet contain values for specific columns of the record layout. Content for all columns in the layout can be specified except for the “Position” column, which is updated automatically. NOTE: These actions are important – if columns in the spreadsheet are not matched to layout columns, the content of those columns in the layout cannot be populate.
Run column mapping rules on this row	This action is used to actually create content in the record layout. Once you have specified certain columns to be included as layout fields, this action extracts the data found in those matched columns and applies it to fields in the record layout. NOTE: This action is required – none of the fields in the record layout will be populated without it.

For all new rules, the “And don’t process any more rules for this content” option is enabled. When enabled, the wizard will only use the matched text for the specified rule and will not try to match it in any other rules. If you want the matched text to be evaluated by other rules in the profile, you can disable this option.

When the rule is configured as desired, click **OK** to return to the profile editor.

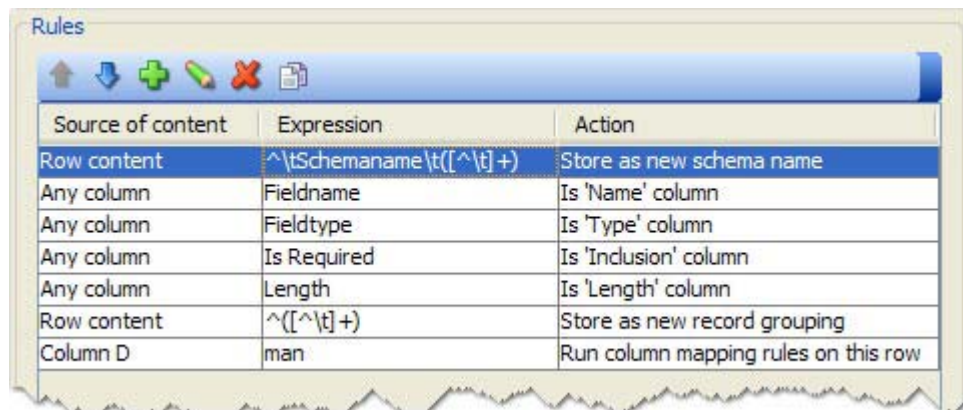
Example

The following example illustrates how an import profile can be used to create a record layout from a simple schema specification.

The following specification (spreadsheet) is used:

	Schemaname	Financial		
	Fieldname	Fieldtype	Is Required	Length
customerData				
	nameLast	String	man	22
	nameFirst	String	man	16
accountData				
	acctNum	String	man	12
	acctType	String	man	12
	acctBal	Double	man	10
transactionData				
	transID	String	man	9
	transDate	Date	man	8
	transAmount	Double	man	10
	transType	Double	man	3

And the following rules are applied:



Source of content	Expression	Action
Row content	^\tSchemaname\t([^\t]+)	Store as new schema name
Any column	Fieldname	Is 'Name' column
Any column	Fieldtype	Is 'Type' column
Any column	Is Required	Is 'Inclusion' column
Any column	Length	Is 'Length' column
Row content	^\t([^\t]+)	Store as new record grouping
Column D	man	Run column mapping rules on this row

The rules shown above will take the following actions on the content of the spreadsheet:

- The content of the cell found to the right of “Schemaname” will be used as the name of the new layout.
- The column containing the “Fieldname” cell is specified as containing the “Name” fields the layout.
- The column containing the “Fieldtype” cell is specified as containing the “Type” fields the layout.
- The column containing the “Is Required” cell is specified as containing the “Inclusion” fields the layout.
- The column containing the “Length” cell is specified as containing the “Length” fields the layout.
- The Row Content = $^{\wedge}([^{\wedge}\backslash t]^+)$ will match the first column and create record groupings that use the matched text (three found).
- Every row in the spreadsheet that should be used to create a field in the layout contains “man” in the “Is Required” column. This field will be used to designate that the column mapping rules should be run on every row containing “man” in that column.

The resulting layout, created according to the import profile and selected spreadsheet, is shown below:

Name	Rep...	Def...	Ind...	Type	Length	Posi...	Align	Pad...	Pad...	For...
custo...										
nameL...	...		Manda...	String	22	0	LEFT		SPACE	...
name...	...		Manda...	String	16	22	LEFT		SPACE	...
accou...										
acctNum	...		Manda...	String	12	38	LEFT		SPACE	...
acctT...	...		Manda...	String	12	50	LEFT		SPACE	...
acctBal	...		Manda...	Double	10	62	LEFT		SPACE	...
trans...										
transID	...		Manda...	String	9	72	LEFT		SPACE	...
trans...	...		Manda...	Date	8	81	LEFT		SPACE	...
trans...	...		Manda...	Double	10	89	LEFT		SPACE	...
transT...	...		Manda...	Double	3	99	LEFT		SPACE	...

Databases

Contents

Creating a Database Resource

Using the Database Connection

Rational Integration Tester provides the ability to interact with any database that supports connectivity through the JDBC specification. These interactions can be used to extract or update data where necessary, and test data sets may be taken directly from database table content.

This chapter describes how to create and test database resources in Rational Integration Tester.

13.1 Creating a Database Resource

To provide access to a database, logical and physical resources need to be added to the Rational Integration Tester project. These resources are added in Rational Integration Tester's Architecture School. For more information, see [Architecture School](#).

See the following sections for details about creating a database resource.

- [Adding the Required Libraries](#)
- [Adding a Logical Resource](#)
- [Adding a Physical Resource](#)
- [Binding Logical to Physical](#)
- [Configure Connection Details](#)
- [Schemas and Stored Procedure Filter](#)

13.1.1 Adding the Required Libraries

Before adding a new database to your project, you should ensure that the JDBC drivers used to connect to the database have been configured in Rational Integration Tester's Library Manager.

Multiple databases are supported by Rational Integration Tester. If you need to add support for another database, however, you can add a new provider under the Database (JDBC) plugin.

For more information, refer to *IBM Rational Integration Tester Installation Guide*.

13.1.2 Adding a Logical Resource

Follow the steps below to add a logical database resource:

1. Open the Logical View of Rational Integration Tester's Architecture School perspective.
2. Select the service component to which you want to add the database (or select the blank palette to add the database to the top level of the project).
3. Select **Database Server** from the **General** component menu, or right-click the component/palette and select **New > General > Database Server**. The **Create a Database Server** dialog is displayed.



4. Enter a unique name for the database and click OK.

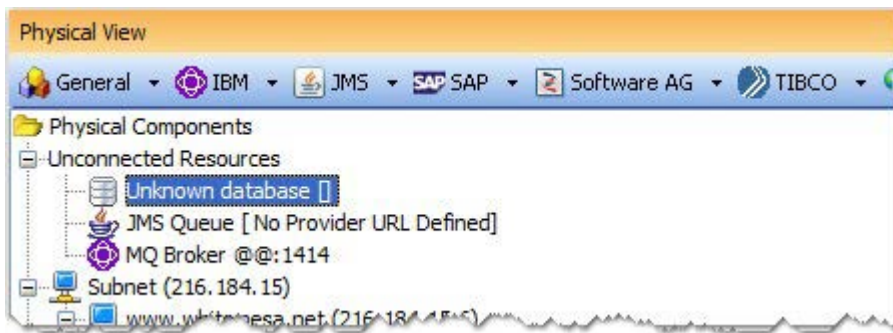
13.1.3 Adding a Physical Resource

The physical database represents an actual database that you want to test. Connection details and stored procedure settings will be configured on the physical resource.

Follow the steps below to add a physical database resource:

1. Open the Physical View of Rational Integration Tester's Architecture School perspective.
2. Select **Database** from the **General** component menu.

A new physical resource named *Unknown database* is created.

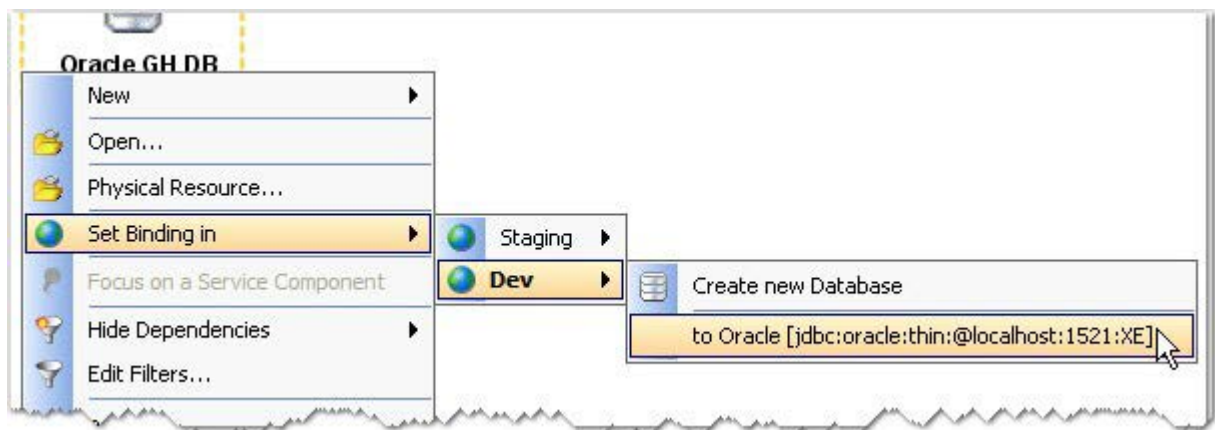


13.1.4 Binding Logical to Physical


The logical database resource is selected in Rational Integration Tester tests. Therefore, it must be bound to an actual (physical) database before any real testing can occur. This binding between logical and physical resources is configured in an environment, which can be done in one of two ways:

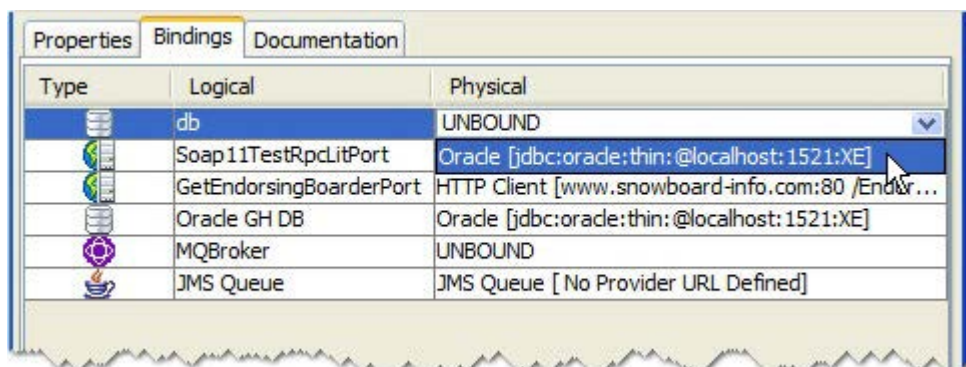
Set Binding in Architecture School

- Right-click the logical database in Architecture School's Logical View and select the **Set Binding in** option, choosing the desired environment and physical database resource.



Set Binding in Environment Editor

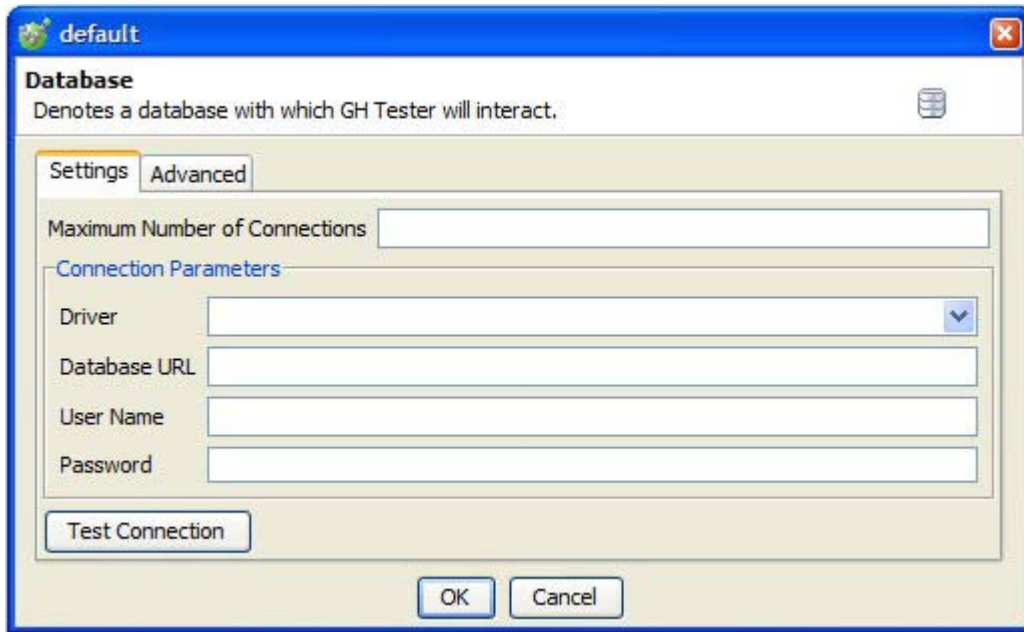
1. Click the environment icon  to open the environment editor.
2. From the list of environments on the left, select the environment in which you want to configure the binding.
3. In the row containing the logical database, select the physical resource.



13.1.5 Configure Connection Details

Once a physical database has been created, you can configure the connection details and test the connection. Follow the steps below to configure a physical database:

1. Open the Physical View of Rational Integration Tester's Architecture School perspective.
2. Double-click the database you want to configure. The **Database** dialog is displayed.



3. Configure the database connection details, as follows:

Maximum Number of Connections	Enter the maximum number of connections (from Rational Integration Tester) that should be allowed to the database.
Driver	Enter the driver details for the specified database type, or select one of the included drivers from the dropdown list.
Database URL	Enter the connection URL for the database, or modify the default URL after selecting one of the included drivers.
Username/ Password	Enter a valid user name and password combination to send when connecting to the database. Both fields support tags, but the context menu is not supported in the Password field.

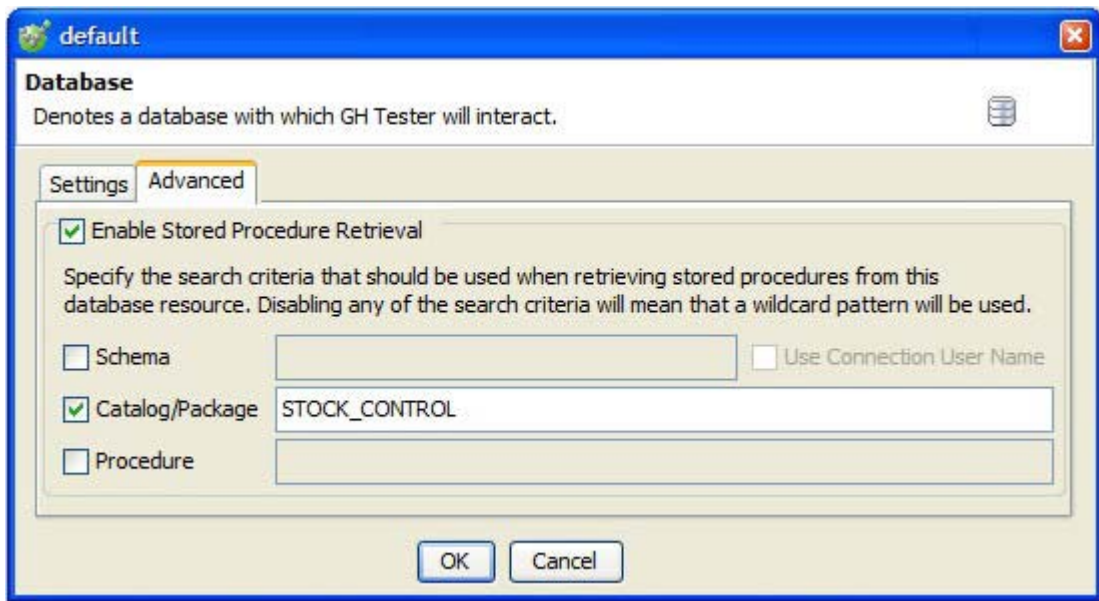
4. Click **Test Connection** to verify the connection parameters.

NOTE: See [Schemas and Stored Procedure Filter](#) for more information about the settings available under the **Advanced** tab.

13.1.6 Schemas and Stored Procedure Filter

If you are working with stored procedures (see [Stored Procedure](#)), you can configure Rational Integration Tester to discover procedures available to you, and also their input/output values. Follow the steps below to configure schema and stored procedure filters:

1. Open the Physical View of Rational Integration Tester's Architecture School perspective.
2. Double-click the database you want to configure. The **Database** dialog is displayed.
3. Click the **Advanced** tab to view the stored procedure retrieval settings.



4. Tick the box to enable the retrieval of stored procedures.

-
5. Configure the filtering options as follows:

Schema	Enter the name of a schema to limit retrieval to that schema (available only if the next option is disabled).
Use Connection Username	Enable this option to retrieve procedures only from the user's schema (based on the user name/password supplied under the Settings tab).
Catalog/Package	Enter a catalog/package name to limit retrieval to only the procedures contained by the catalog/package.
Procedure	Enter a procedure name to retrieve only that specific procedure.

6. Click **OK** to save your settings and rebuild the database schemas, which could take a long time depending on the size of the database and the number of schemas available.

13.2 Using the Database Connection

Once a database resource is properly configured in Architecture School, Rational Integration Tester provides a number of ways to interact with it. Three different actions can be used in tests to execute SQL queries or commands and call a stored procedure. Details about each test action are provided in the following sections:

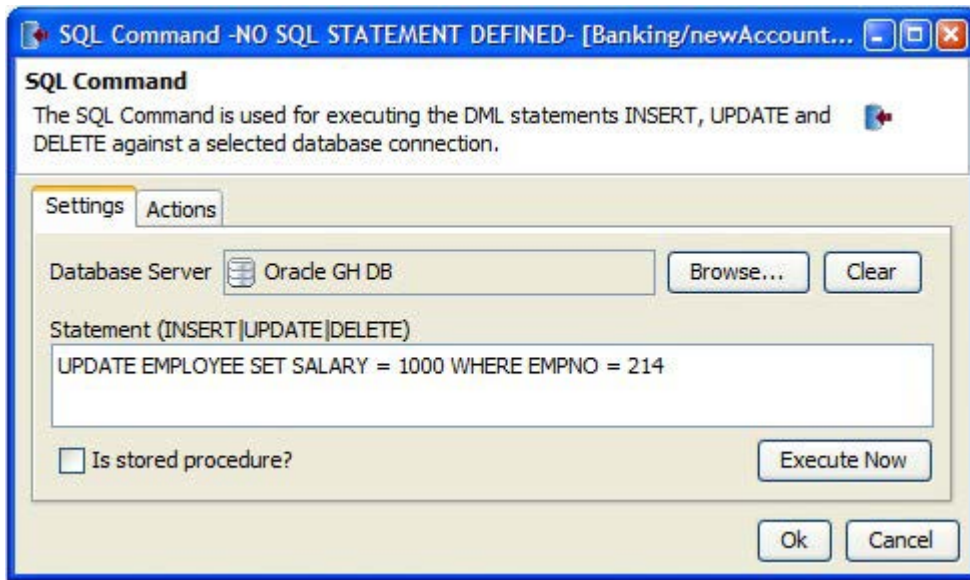
- [SQL Command](#)
- [SQL Query](#)
- [Stored Procedure](#)

NOTE: See [Test Factory](#) for more information about building tests and [Appendix A: Test Actions](#) for information about adding database test actions to them.

13.2.1 SQL Command

The SQL Command test action executes a SQL command (INSERT, UPDATE, or DELETE) on a configured database. Follow the steps below to configure the SQL Command action:

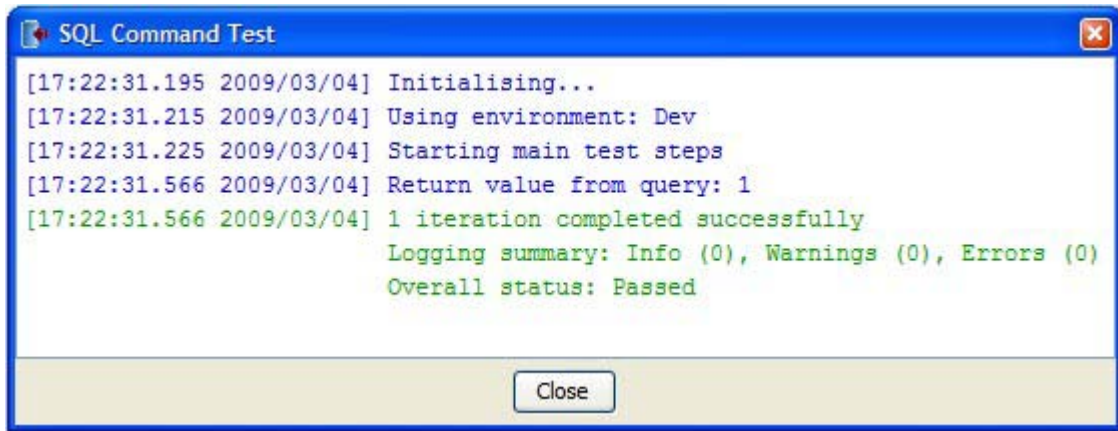
1. After adding the SQL Command action to a test, double-click it to edit it.



2. Click **Browse** to select a database resource from those available in the project.
3. Enter a command (INSERT, UPDATE, or DELETE) in the Statement field.
4. If entering a stored procedure, check the **Is stored procedure?** option and modify the syntax of the query appropriately: {call procedure (var, var)}.

NOTE: If issuing a stored procedure call, you may want to use the Stored Procedure action instead (see [Stored Procedure](#)). This action presents the available functions and input/output parameters more clearly.

-
5. To test the statement without running the test, click the **Execute Now** button. An execution panel is displayed to show the results of the statement (that is, success or failure, and the number of rows that were affected by the command).



NOTE: When the action is executed within a test, it returns the number of rows affected by the SQL command. The return value can then be validated and/or stored.

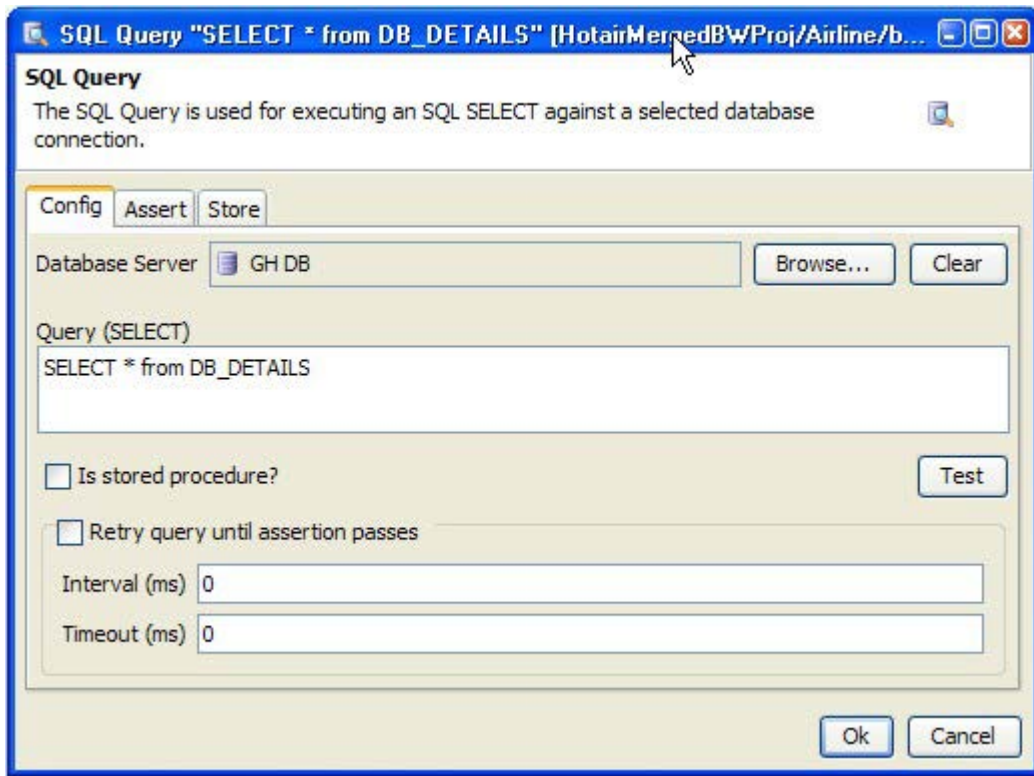
-
6. Select the **Actions** tab to configure validation and store options (that is, to store the return value in a tag, or to specify a condition relating to the return value, such as validating the number of rows affected by the command).

The screenshot shows a software interface with two main tabs: 'Settings' and 'Actions'. The 'Actions' tab is active. Inside the 'Actions' tab, there are two sub-tabs: 'Validate' and 'Store'. The 'Validate' sub-tab is selected. In the 'Validate' section, there is a list box containing one item: 'Equality', which is checked with a green checkmark. Below this list box are three buttons: 'New' (with a plus icon), 'Delete' (with a trash icon), and 'Clone' (with a copy icon). Below these buttons, there is a section with three fields: 'Action Type' is a dropdown menu currently showing 'Equality'; 'Description' is an empty text field; and a large text area containing the number '0'.

In the example above, the return value is validated by the condition that it equal zero. Hence, this test step will fail if any rows are updated by the specified command.

13.2.2 SQL Query

The SQL Query test action provides a method of verifying that the contents of the selected database match expected values.



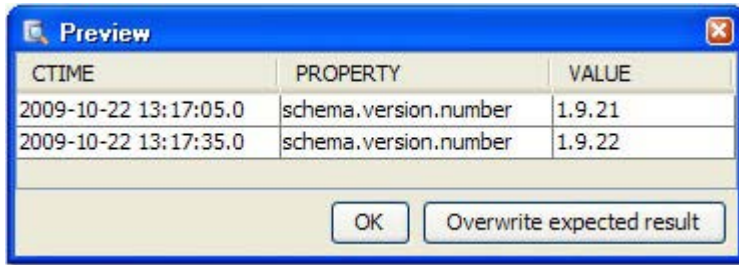
A sample query can be run to populate the data in the test action, or you can manually enter the data (or you can use a combination of both). When the action is executed within a test, the database query is run and the returned data is compared against what is contained in the test action. It is also possible to tag values from the query.

Follow the steps below to configure the SQL Query test action:

1. After adding the SQL Query action to a test, double-click it to edit it.
2. Click **Browse** to select a database resource from those available in the project.
3. Enter a query (that is, a SELECT statement or stored procedure call) in the **Query** field.
4. If entering a stored procedure, check the **Is stored procedure?** option and modify the syntax of the query appropriately: {call procedure (var, var)}.

NOTE: If issuing a stored procedure call, you may want to use the Stored Procedure action instead (see [Stored Procedure](#)). This action presents the available functions and input/output parameters more clearly.

5. Click **Test** to run the query and retrieve the data.
6. The results of the query are displayed and any data that would be returned is displayed in the **Preview** dialog.



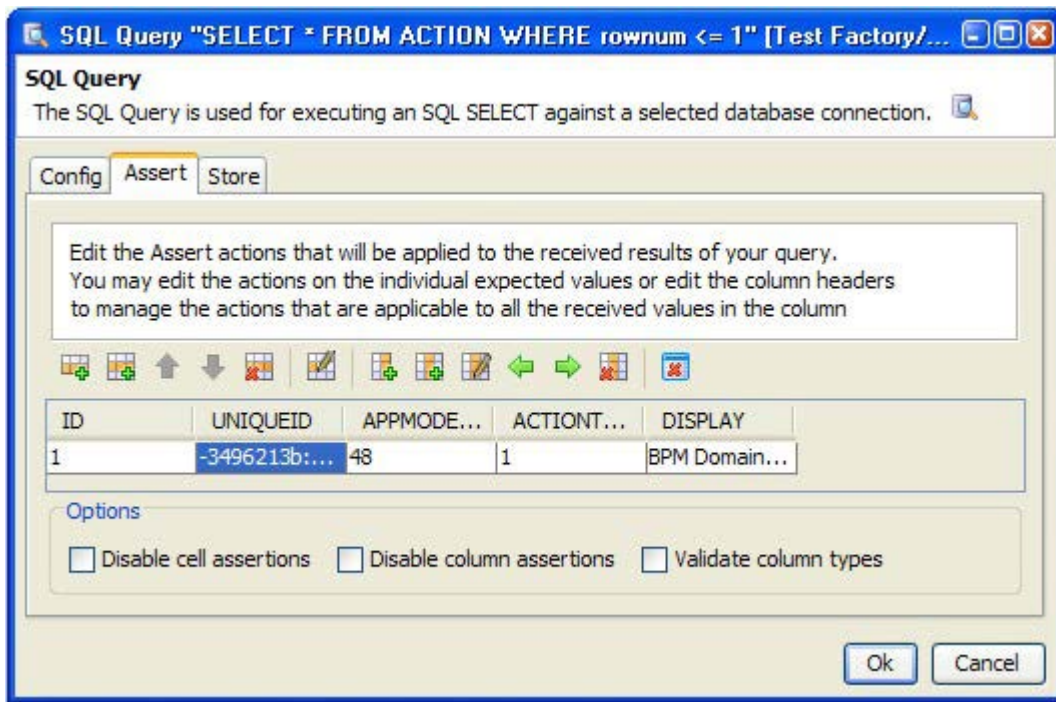
The screenshot shows a 'Preview' dialog box with a table of results. The table has three columns: CTIME, PROPERTY, and VALUE. There are two rows of data. Below the table are two buttons: 'OK' and 'Overwrite expected result'.

CTIME	PROPERTY	VALUE
2009-10-22 13:17:05.0	schema.version.number	1.9.21
2009-10-22 13:17:35.0	schema.version.number	1.9.22

NOTE: If the query was cancelled before finishing, any data that was built before cancelling will be populated.

7. Click **Overwrite expected result** to copy the results into the **Assert** tab of the SQL Query dialog. This is the data that will be compared with the query during test execution. To return to the SQL Query dialog without copying the results, click **OK**.

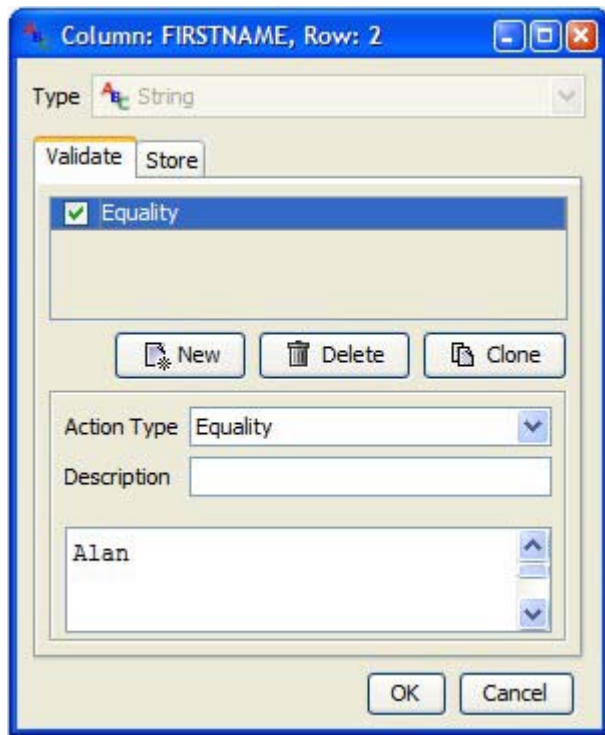
-
8. Once the column headings (and data) are populated in the **Assert** tab, the rows, columns, and cells can be edited and arranged using the buttons above the data.



By default, all of the data cells will be validated against the data shown in them. Therefore, when the test is executed, the SQL query must return this exact data.

You can customize the validation options using the **Disable cell assertions** (disable any assertions actions that exist in the cells), **Disable column assertions** (disable any assertions made on any received value, independent of the cell assertions), and **Validate column types** check boxes. If assertions are disabled, assertions can still be created and modified, but they will not be used – including the comparison between the number of rows expected and the number received.

Cells can be edited to change their value or validation setting. When editing a cell, the field editor is displayed, letting you change a cell's value or validation options.



You can also use the **Store** tab (in the field editor or in the main test action editor) to perform store actions in the same way as with other test actions. Multiple values can be stored as a list using the **Append to list of values** option.

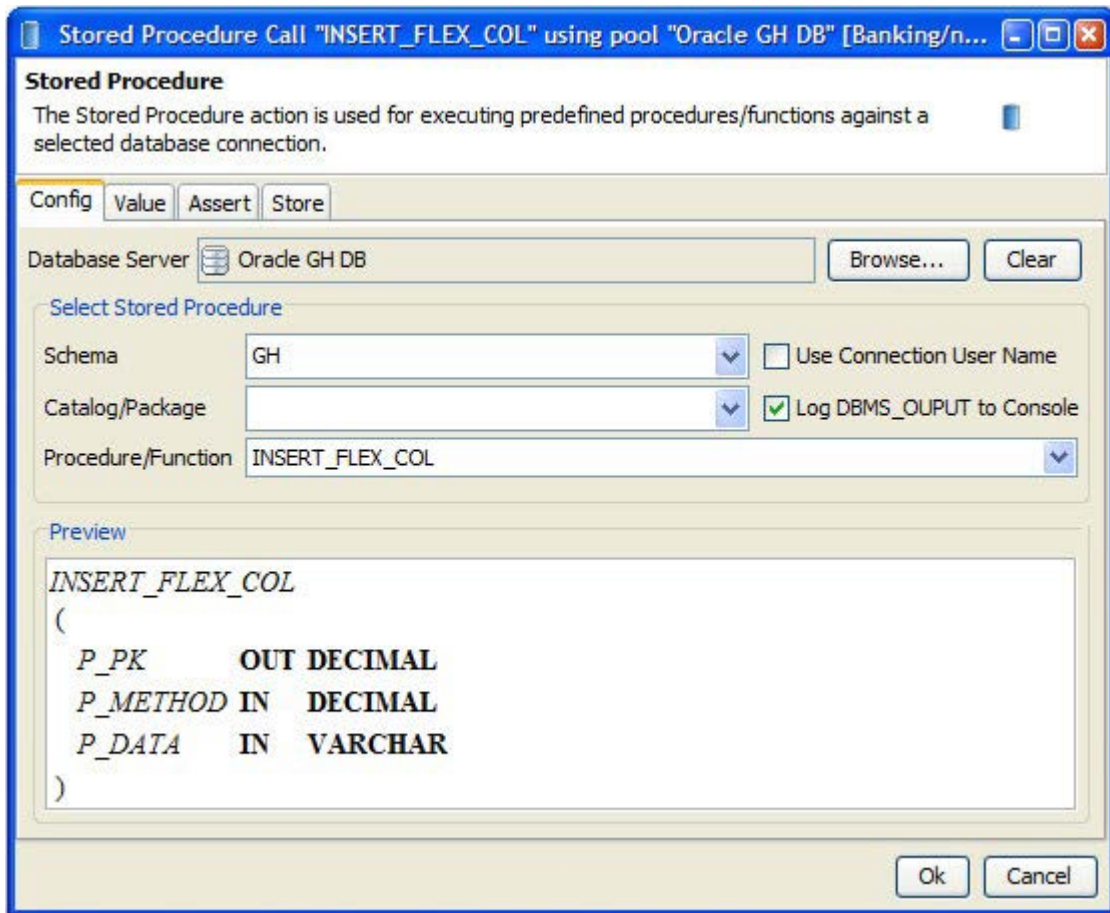
9. Use the **Retry query until assertion passes** option and set the **Interval** and **Timeout** options to have the test action repeat the validation attempt (that is, in the case where database writes are performed by another process running asynchronously to Rational Integration Tester, and it's possible that your query runs before the data is committed). The query will be retried every [Interval (ms)] until [Timeout (ms)] is reached.
10. When you are finished configuring the query settings, click **OK**.

13.2.3 Stored Procedure

Use the Stored Procedure action to call a database stored procedure. As you configure this action, you can view the code of the stored procedure, specify input values, tag return values, or specify expected return values.

Follow the steps below to configure the Stored Procedure action:

1. After adding the Stored Procedure action to a test, double-click it to edit it.



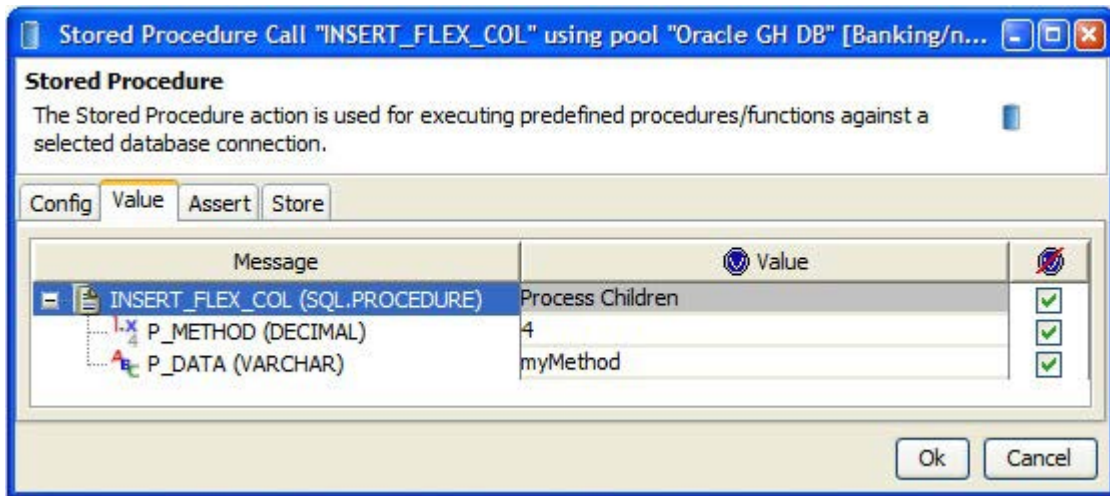
NOTE: The retrieval of stored procedures must be enabled in the physical database component before you can configure these options in the Stored Procedure test action (see [Schemas and Stored Procedure Filter](#) for more information).

2. Click **Browse** to select a database resource from those available in the project.

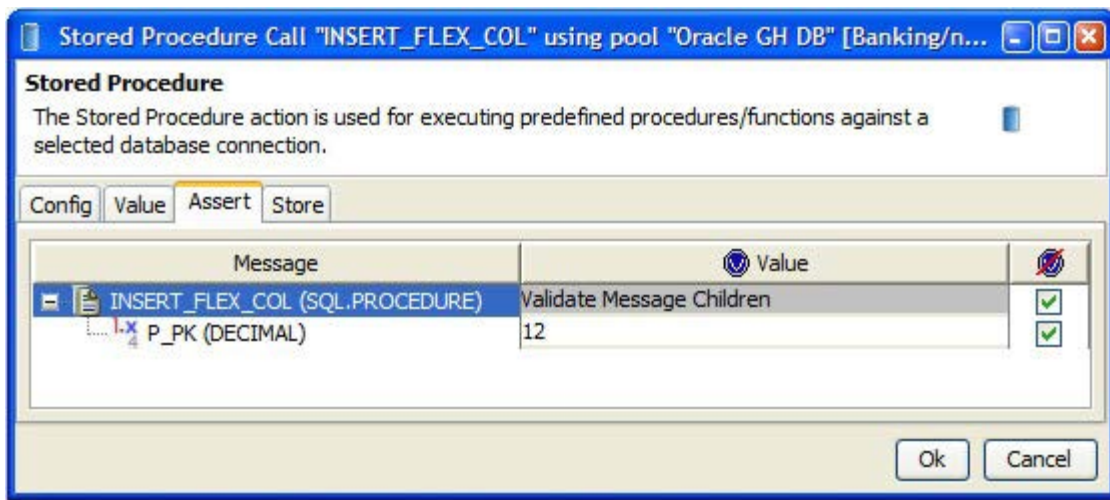
-
3. Narrow the procedures available by selecting a schema and catalog/package (if available).
 4. For Oracle database connections, if you use DBMS_OUTPUT.PUT_LINE() in your stored procedure code, you can have its output sent to the console when executing tests by enabling the **Log DBMS_OUTPUT to Console** option.
 5. Enable the **Use Connection User Name** option to retrieve procedures only from the user's schema (based on the user name/password supplied under the **Settings** tab of the physical database resource). This can be affected if tags are used for the database connection settings.


NOTE: If you do not see the procedures you are expecting, check the connection settings and the Stored Procedure filters that have been configured on the physical database resource.

6. Select the **Value** tab to specify values that you want to supply as inputs to your stored procedure (tags are supported).

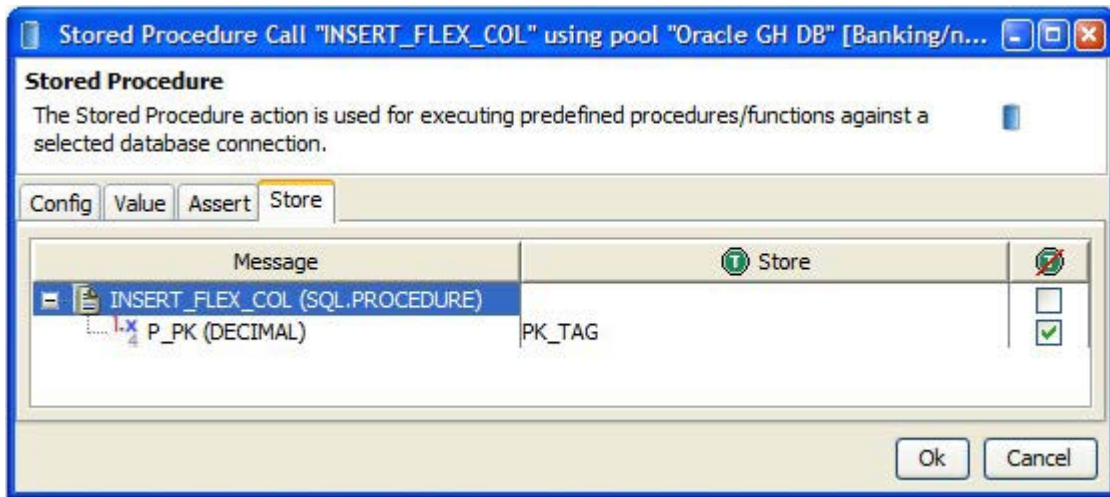


-
7. Select the **Assert** tab to specify expected return values.



NOTE: The test will fail if the stored procedure does not return the expected values. If you want to skip validation for any row, you must disable the validation check box  directly to its right.

8. Select the **Store** tab if you wish to tag any of the return values from your stored procedure.



9. Type in the name of a tag (new or existing), or right-click on the field name and select **Contents > Quick Tag** to create a tag of the same name.

NOTE: Ensure that you have disabled validation on the Assert tab if you are expecting different values each time.

Identity Stores and SSL

Contents

Overview

Creating an Identity Store

Configuring an Identity Store

This chapter provides information about identity stores in Rational Integration Tester and how to create them. Details about using an identity store to enable SSL communications in various messaging transports can be found in the appropriate reference guide.

14.1 Overview

An identity store provides the ability to group together a collection of one or more certificates that can be used in Rational Integration Tester to perform validation and authentication of connections using SSL.

The underlying component of a Rational Integration Tester identity store is a Java KeyStore. You can use an existing keystore that has been created using the JDK tools, or you can create a new keystore when you create an identity store.

Once created, the following certificate types can be imported into a Rational Integration Tester identity store:

- Personal Information Exchange (PKCS#12)
- Personal Information Exchange (PKCS#8)
- X.509 Certificates

Most commonly you would export certificates from your browser and then import the certificate file into a Rational Integration Tester identity store. These certificates can then be used to verify the chain of trust from any certificate that a server sends to the Rational Integration Tester client connection.

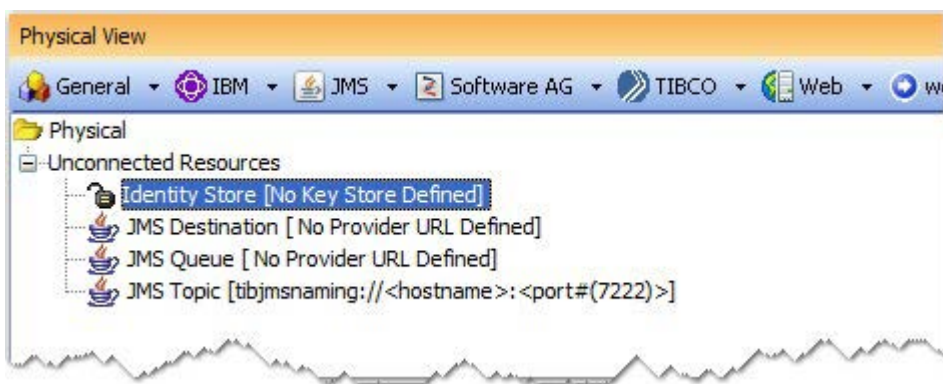
NOTE: Currently, a client's identity can only be created using the JDK tools, which is done against a keystore that can subsequently be used to identify one end of an SSL connection.

14.2 Creating an Identity Store

Identity stores are created in the Physical View of Rational Integration Tester's Architecture School perspective. You can create an identity store in one of two ways:

- Select **Identity Store** from the **General** menu in the Physical View's component toolbar.
- Right-click the root of the physical resource tree and select **New > General > Identity Store** from the context menu.

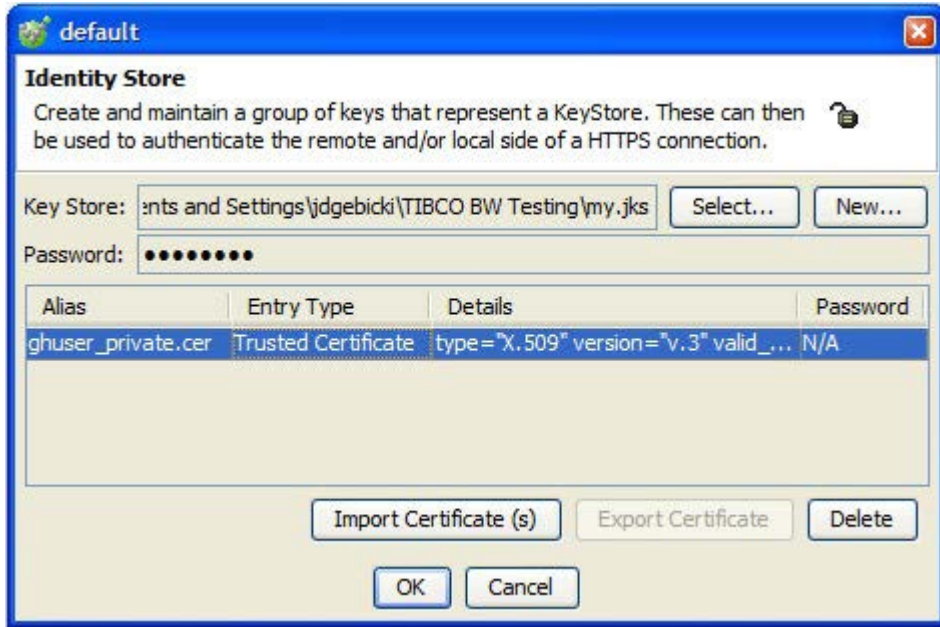
The new identity store is created under the Unconnected Resources in the Physical View.



14.3 Configuring an Identity Store

Follow the steps below to configure a new or existing identity store in Rational Integration Tester:

1. Double-click the desired identity store in the Physical View of Architecture School.
The **Identity Store** editor is displayed.



2. Click **Select** to locate and open an existing Java Keystore (.jks) file. When prompted, enter the keystore password.
3. To create a new keystore with Rational Integration Tester, click **New** and select the location and name of the new keystore.
4. Click **Import Certificate(s)** to import a certificate into the selected keystore, then locate and open the desired certificate file.
5. To export an existing certificate (if it can be exported), select it and click **Export Certificate**.
6. To delete a certificate from the keystore, select it and click **Delete**.

Functions

Contents

[Availability and Usage](#)

[Assertion Functions](#)

[Mathematical Functions](#)

[String Functions](#)

[Other Functions](#)

As described earlier, functions represent a single or composite expression from the registered set of functions. These can be used in the Function or Decision action types, and within individual message fields to provide dynamic content.

This chapter describes functions in Rational Integration Tester.

15.1 Availability and Usage

A Function action within a test allows you to enter an expression that refers to a specific function by name (built-in or custom). More complex expressions can refer to several functions using a nested bracket syntax, which should be familiar to programmers.

For example: **eq(5,abs(5.2))**, which is the same as **eq(5,5.2)** and evaluates to “false.”

Functions are available to all users of a project. If a project is cloned, the same files and functions from the original project will be available in the cloned project.

NOTE: Functions are case sensitive.

NOTE: It is possible to use the result of one function as the input to another. This way, the result of an XPath query can be used as a comparison against the result of a database query.

NOTE: When using functions, any string objects should be enclosed in straight quotes (for example, "My text").

15.2 Assertion Functions

The following sections describe the assertion functions that are available in Rational Integration Tester. The ordering of comparison is natural or lexicographical, as appropriate.

NOTE: Assertion functions are normally used within the context of a decision step rather than as an element of the Function action.

15.2.1 And

The *And* function performs a logical AND on the results of the one or more expressions, returning true or false.

Usage: `and(expr , expr ,)`

Example: `and(eq("yes", "%answer%", validateXSD("%doc%", "example.xsd"))`

15.2.2 Equals

The *Equals* function returns true if the expressions are logically equal.

Usage: `eq(expr , expr)`

NOTE: Both expressions are evaluated as strings, regardless of whether or not one or both expressions are quoted (including tags).

15.2.3 Greater than

The *Greater than* function returns true if the first expression is greater than the second.

Usage: `gt(expr , expr)`

15.2.4 Greater or equal

The *Greater or equal* function returns true if the first expression is greater than or equal to the second.

Usage: `ge(expr , expr)`

15.2.5 Less than

The *Less than* function returns true if the first expression is less than the second.

Usage: `lt(expr, expr)`

15.2.6 Less or equal

The *Less or equal* function returns true if the first expression is less than or equal to the second.

Usage: `le(expr, expr)`

15.2.7 Not equal

The *Not equal* function returns true if the first expression is unequal to the second.

Usage: `ne(expr, expr)`

15.2.8 Or

The *Or* function performs a logical OR on the results of the one or more expressions, returning true or false.

Usage: `or(expr, expr,)`

Example: `or(eq("yes", "%%answer%%"), validateXSD(%%doc%%, "example.xsd"))`

15.3 Mathematical Functions

The following sections describe the mathematical functions that are available in Rational Integration Tester.

15.3.1 Absolute Value

The *Absolute Value* function returns the absolute value of the expression.

Usage: `abs (expr)`

15.3.2 Add

The *Add* function returns the sum of the expressions.

Usage: `add (expr , expr , ...)`

15.3.3 Divide

The *Divide* function returns the quotient of the expressions (the first divided by the second).

Usage: `divide (expr , expr)`

15.3.4 Floor

The *Floor* function returns the highest integer that is less than the specified expression.

Usage: `floor (expr)`

15.3.5 Modulo

The *Modulo* function returns the remainder of dividing the first expression by the second.

Usage: `mod (expr , expr)`

15.3.6 Multiply

The *Multiply* function returns the product of two expressions.

Usage: `multiply (expr , expr)`

15.3.7 Round

The *Round* function returns the value of the expression, rounded to the (optional) number of decimal places or to the nearest whole number.

Usage: `round(expr[, decimal places])`

If no `decimal places` argument is provided, or if the value provided is 0, the function returns an integer value.

Example: `round(add(-0.22,300),2)` returns 299.78

`round(add(-0.22,300))` returns 300

15.3.8 Subtract

The *Subtract* function returns the difference of two expressions (the second subtracted from the first).

Usage: `subtract(expr,expr)`

15.4 String Functions

The following sections describe the string functions that are available in Rational Integration Tester.

15.4.1 Create Text

The *Create Text* function returns a string of text.

Usage: `createText(data size in bytes, [repeated character sequence])`

The character sequence will be repeated until the returned string reaches the size specified.

15.4.2 Format Date UTC

The *Format date UTC* function returns the number of milliseconds that have passed between January 1, 1970 and the supplied date and time, which could be used for mathematical operations or functions. For example, if a test receives a message that was published with a timestamp in it, you would have two dates expressed as strings (the one in the message and the system tag that contains the current). You could run `formatDateUTC` against the two numbers and subtract the results from one another to determine the latency of the message.

Usage: `formatDateUTC(date, inputFormat)`

date - string representation of a specific date and time (for example, "Jan 01, 2009 08:15:30")

inputFormat - a string that denotes how date is encoded (for example, "MMM dd, yyyy HH:mm:ss")

For example, `formatDateUTC("Jan 01, 2009 08:15:30", "MMM dd, yyyy HH:mm:ss")` returns 1230822930000.

NOTE: If the `inputDate` parameter contains spaces, it must be enclosed in quotes.

15.4.3 Get Token

The *Get Token* function extracts a token from a tokenized string (that is, it takes a portion of a larger string as determined by the delimiters in use and a specified index value.

Usage: `getToken ("Delimiter", Index [>=0] , Data,
[includeEmptyStrings true|false])`

Delimiter - enclosed in quotes, one or more characters used to separate values in the data string

Index - starting from zero, the delimited position within the string from which the token should be extracted (for example, “0” indicates the position before the first delimiter, “1” indicates the position before the second delimiter, and so on)

Data - a string containing a set of values separated by the specified delimiter

includeEmptyStrings - set to true or false, indicates whether or not empty strings (that is, two successive delimiters) should be included when calculating the index position

Note the following examples:

`getToken (",", 2 , "aaa,bbb,,ccc,ddd", true)` returns null since empty strings are included and the third position (index 2) is null.

`getToken (",", 2 , "aaa,bbb,,ccc,ddd", false)` returns “ccc” since empty strings are ignored, meaning that the third position (index 2) is “ccc”.

15.4.4 Regular Expression

The *Regular Expression* function executes a regular expression extraction against a source string. By default, this will be the first group that is returned, however an optional fourth parameter can be used to extract other groups.

Usage: `regex(string, expr, [instance], [extractionGroup])`

- **string** is the source string against which the regular expression will be run.
- **expr** is the regular expression.
- **instance** is the optional instance to be extracted – without this parameter, **true** or **false** will be returned based on whether or not the match succeeds.
- **extractionGroup** is the optional extraction group that controls how content is returned – 0 (default) returns all text, 1 returns only the first match, 2 returns only the second match, and so on

Examples

The following examples illustrate the use of the `regex` function:

The simplest case is to see if a pattern exists within the string:

```
regex ( "hello world", "h") returns 'true'.
```

```
regex ( "hello world", "xxx") returns 'false'.
```

```
regex ( "hello world", "l.", 1) returns 'll'.
```

Example: If a time field (format hh:mm:ss) has been extracted from a message and is held in a tag named "timeField", the following function will return the value for the hours as instance 1:

```
regex ( "%%timeField%%", "(\\d\\d)", 1)
```

Increasing the instance argument to “2” or “3” provides access to the minute and second values.

Example: It is possible to specify multiple sections of text within () brackets. In this case, the function should extract some words from the following tag in an XML document specified by %%%XML%%:

```
<tag_name>Here is some writing to be transformed.</tag_name>
```

`regex("%%XML%%", "<tag_name>.* (t\\w*).*(t\\w*).*</tag_name>", 1, 1)`
finds the first () match – the word beginning with 't' – and returns 'to'.

`regex("%%XML%%", "<tag_name>.* (t\\w*).*(t\\w*).*</tag_name>", 1, 2)`
finds the second () match and returns 'transformed'.

NOTE: If **extractionGroup** is set to zero, the whole string match is returned.

Using Modifiers

Modifiers can be used in regular expression (for example, those used to process strings that span multiple lines). Modifiers are placed in front of the regular expression in round brackets, as follows:

`(?s)\\<\\?xml.*` extracts an XML string over multiple lines.

Modifiers may be combined in the same brackets using a single '?', as follows:

`(?si)\\<\\?xml.*` extracts an XML string over multiple lines while ignoring case.

The following is a list of commonly-used modifiers:

- `?i` case insensitive
- `?x` comment mode
- `?d` unix line mode
- `?m` multiple line mode
- `?s` single line mode

Modifiers may be used anywhere that regular expressions are used in Rational Integration Tester, for example in tagging operations. Refer to the Perl regular expression documentation for more information.

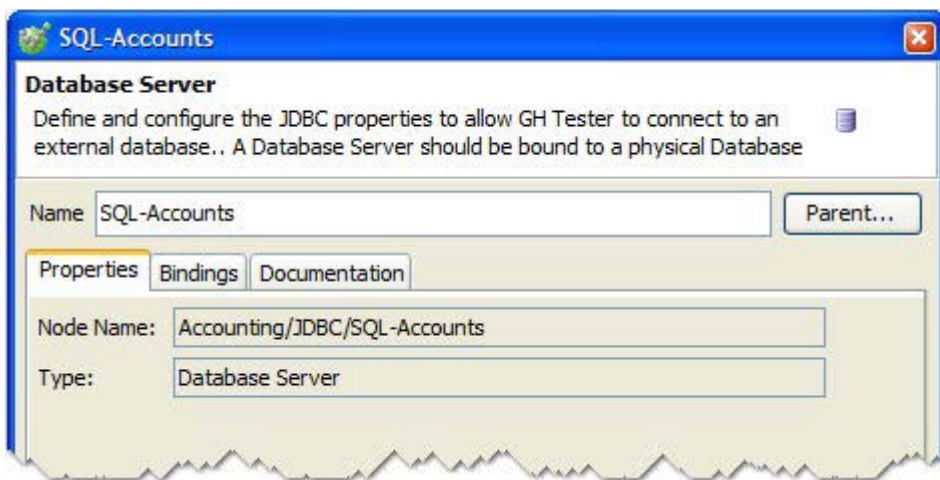
15.5 Other Functions

The following sections describe the remaining functions that are available in Rational Integration Tester.

NOTE: All database functions require an ID (`connectionId`) that represents the logical connection to be used. The ID is the fully qualified name of a previously configured database in the following format:

Logical/<Node Name>

The **Node Name** can be discovered by viewing the properties of the database in the Logical view of Rational Integration Tester's Architecture School.



See [Databases](#) for more information about creating database resources.

15.5.1 Current Time

The *Current Time* function returns the current time using the specified date time pattern (see [Appendix B: Date & Time Patterns](#) for more information). The function's output can be used in other actions (for example, in a log action to be sent to the console) or stored for use by other actions within the test.

Usage: `now(simpleDateFormat)`

15.5.2 Database Query

The *Database Query* function executes a SQL query against a configured database, returning the value from the first column of the first row of the result set (that is, it is best used to request singular elements of data).

Usage: `dbQuery(connectionId, query)`

Parameter	Description
<code>connectionId</code>	The project path of the logical database resource you want to query (for example, "Acct/JDBC/SQL"). Note that the "Logical/" prefix is not used.
<code>query</code>	The SQL statement to run against the selected database.

Example: `dbQuery("Acct/JDBC/SQL", "SELECT * FROM credits")`

15.5.3 Database Update

The *Database Update* function executes an update-based SQL statement against a configured database, returning the number of rows affected by the statement.

Usage: `dbUpdate(connectionId, updateSQL)`

Parameter	Description
<code>connectionId</code>	The full path of the logical database resource you want to update (for example, Logical/MyProject/Accounting/JDBC/SQL-Accounts).
<code>updateSQL</code>	The SQL statement to run against the selected database.

For example: `dbUpdate("Logical/Accounting/JDBC/SQL-Accounts", "UPDATE Persons SET Acct='Savings' WHERE LastName='Taylor' AND FirstName='Robert'")`

15.5.4 Join

The *Join* function will combine the listed arguments into a single string using the specified delimiter.

Usage: `join(Delimiter, expr, expr, ...)`

For example, `join(; , 2, 3, 4)` returns `2 ; 3 ; 4`. As an example, the function can be used to store multiple tag values into a single tag.

15.5.5 Lookup Test Data

The *Lookup Test Data* function provides keyed access to a data set.

Usage: `lookupTD(datasetPath,[keyColumn,matchValue,]*
resultColumn,<defaultValue>)`

datasetPath – the project path (string) to the test data set to search

keyColumn, matchValue – the column name to match against and the value in that column to find (can be repeated to create “and” logic)

resultColumn – the column from which to take the result (in the row where the value has been matched)

defaultValue – the value to return if no match is found

NOTE: This function works the same way as the [Lookup Test Data](#) test action, except that matching results are returned to the function rather than mapped to a tag.

Example: `lookupTD("SC2/OP/iterations/excel.ext",arg0,a,arg1,b)`

Test data sets should be specified using the proper extension, according to the data set type, as follows:

- File = .sit
- Directory = .fst
- Excel = .ext
- Database = .dbt

NOTE: The “Logical/” prefix is not used when indicating the data set path.

15.5.6 Null

The *Null* function returns a null value, which can be used either in the setting of tag values or for comparison against a database query result.

Usage: `null()`

15.5.7 Replace Tag(s)

The *replaceTags* function replaces tags used in other tags with their value in the current tag store. The contents of the tag (using the replaced values) can then be stored in a new tag.

Usage: `replaceTags(value)`

For example, consider the following tags and their default values:

`%%tag1%% = A`

`%%tag2%% = B`

`%%tag3%% = %%tag1%% comes before %%tag2%%`

If `replaceTags(%%tag3%%)` is executed and the returned value is stored in a new tag (named “replaced”), the value of `%%replaced%%` will be “A comes before B”.

15.5.8 Reset Tag(s)

The *resetTags* function resets all tags matching a stated pattern to their default values.

Usage: `resetTags([tagNamePattern])`

15.5.9 Settlement Date

The *settlementDate* function determines a working date that is a defined number of days from a given start date. The function counts only working days, ignoring weekends and holidays according to the local calendar, and holidays are configurable using a simple text file containing a list of dates.

Usage: `settlementDate ("startDate", days <Int +/->, "holidayFilename" [, "dateFormat" <default="yyyyMMdd">])`

Parameter	Description
startDate	The starting date, a string value containing a formatted date.
days <Int +/->	A positive or negative value (indicated with “+” or “-”) that defines the number of days after or before startDate .
holidayFilename	The full path to a simple text file that contains a list of dates to ignore, separated by the new line character.
dateFormat	An optional argument that defines the format to be used for startDate , dates in holidayFilename , and the date returned by the function (yyyyMMdd is used by default).

15.5.10 Set a Tag's Value

The *Set Tag* function replaces the contents of an existing tag with a function, expression, or constant specified by “value” – this can also be achieved by using the store operation functionality from within the Function action.

NOTE: The specified tag must have been previously created.

Usage: `setTag(tagName, value)`

15.5.11 Text File Content

The *Text File Content* function reads and returns the contents of a text file in a specified location. The file contents can be stored in a tag for use in other phases of the test.

Usage: `textFileContent(path)`

NOTE: The full location of the file can be specified as a local or UNC path.

15.5.12 Validate Against DTD / Validate Against XSD

The Validate DTD and XSD functions determine whether an XML document (xml) is valid with respect to the provided schema(s) (dtdURL or schemaURL).

Usage: `validateDTD(xml, [dtdURL_1, ..., dtdURL_n])`

`validateXSD(xml, [schemaURL_1, ..., schemaURL_n])`

Example: `validateXSD(%%doc%%, file:///c:\Schemas\example.xsd)`

NOTE: These functions behave differently depending upon whether they are invoked from within a function action (where they return the textual output from the validation process) or in a decision action (where the output will be merged into a boolean result so that an appropriate path can be chosen).

15.5.13 XML Database Query

The *XML Database Query* function executes a query-based SQL statement against a configured database. The result set of this query is then encoded as XML and provided as the return value from the function.

Usage:XMLdbQuery(connectionId, query)

Parameter	Description
connectionId	The full path of the logical database resource you want to query (for example, Logical/MyProject/Accounting/JDBC/SQL-Accounts).
query	The SQL statement to run against the selected database.

The XML fragment that is returned conforms to the following schema:

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="resultSet">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="columns">
          <xs:complexType>
            <xs:sequence>
              <xs:element maxOccurs="unbounded" name="column">
                <xs:complexType>
                  <xs:attribute name="name" type="xs:string" use="required" />
                  <xs:attribute name="type" type="xs:string" use="required" />
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element maxOccurs="unbounded" name="row">
          <xs:complexType>
            <xs:sequence>
              <xs:any minOccurs="0" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="rowCount" type="xs:decimal" use="required" />
    </xs:complexType>
  </xs:element>
</xs:schema>
```

An example of the function and its output are shown below:

Function:

```
XMLdbQuery("Logical/Users/JDBC/SQL-People","SELECT person, age  
FROM users")
```

Output:

```
<?xml version="1.0" encoding="UTF-8"?>  
<resultSet rowcount="2">  
  <columns>  
    <column name="person" type="string">  
      <column name="age" type="number">  
    </columns>  
    <row>  
      <person>John</person>  
      <age>33</age>  
    </row>  
    <row>  
      <person>Helen</person>  
      <age>31</age>  
    </row>  
  </resultSet>
```

15.5.14 XPath Query

The *XPath Query* function returns the result of performing an XPath query (xpath) on some XML source that is to be validated (xml). This can also be achieved by using the store operation functionality from within the Function action.

Usage: `xpath(xml, xpath)`

For example, `xpath("%%srcData%%", "/resultSet/@rowCount")` will return the number of rows that are provided in the output from the execution of an *xmlDbQuery* operation (detailed previously).

NOTE: Rational Integration Tester uses the XPATH 2.0 draft standard. Therefore, a common issue when matching nodes is that the fully qualified node name (with the namespace) needs to be used.

NOTE: If attributes are included in the XML, the XML must be quoted and the quotes surrounding the attribute must be escaped. For example:

```
xpath("<a><b action=\"copy\"/></a>", "a/b")
```

Custom Functions

Contents

Overview

The `formatDate` Example

Create a Plugin in Eclipse

Develop the Function Class

Create and Configure an Extension Point

Configure and Use the Function in Rational Integration Tester

Create a Function without Eclipse

Converting Existing Function Classes with Eclipse

Converting Existing Function Classes without Eclipse

A custom function is a Java class that extends `com.ghc.ghTester.expressions.Function`. Using a custom function, you can add calculations or operations to tests in Rational Integration Tester.

This chapter describes how to develop, package, and configure a custom function in Rational Integration Tester.

16.1 Overview

The following steps provide an overview of how to develop a function for use in Rational Integration Tester:

1. Create a plugin, which is a Java code wrapper that allows the plugin to be loaded into Rational Integration Tester. Using the Eclipse IDE is the easiest way to create a plugin (see [Create a Plugin in Eclipse](#)), but you can also create a plugin from another IDE, if desired (see [Create a Function without Eclipse](#)).
2. Develop your function class, which involves creating a Java class and implementing the function behavior (see [Develop the Function Class](#)). If you have an existing function and want to modify it to work with Rational Integration Tester, see [Converting Existing Function Classes with Eclipse](#).
3. Create a function extension point to provide Rational Integration Tester with details about your function, such as its name and the class that will implement the function (see [Create and Configure an Extension Point](#)).
4. Configure and use the function in Rational Integration Tester, which lets Rational Integration Tester load your function to be used inside your Rational Integration Tester project (see [Configure and Use the Function in Rational Integration Tester](#)).

16.2 The formatDate Example

To help explain the concepts of creating a custom function, this chapter will demonstrate how to build a function, called **formatDate**, that parses a string-based date representation and returns it in an alternate format.

The function's syntax is as follows:

```
formatDate( date, inputFormat [, outputFormat] )
```

NOTE: The **outputFormat** parameter is optional. If it is not entered, the function returns the date in the default format, which is yyyy-MM-dd.

Note the following examples of the function:

```
formatDate( "05102004", "ddMMyyyy" ) returns the date in the format "2004-10-05."
```

```
formatDate( "05/10/04 13:25", "dd/MM/yy HH:mm", "dd MMM, yyyy  
hh:mm:ss a" ) returns the date in the format "05 Oct, 2004 01:25:00 PM."
```

NOTE: The complete source code and deployable **FormatDate** function can be found in the `examples\FunctionsSamplePlugin` folder under your Rational Integration Tester installation.

16.3 Create a Plugin in Eclipse

Rational Integration Tester recognizes functions that are packaged as Eclipse plugins. This section demonstrates how to create a plugin and register it as a function in Rational Integration Tester from an Eclipse IDE. If you don't have an Eclipse IDE, then see [Create a Function without Eclipse](#).

NOTE: You can create multiple functions in a single plugin. If you do, simply add an extension point for each function implementation.

16.3.1 Prerequisites

The Eclipse Plugin Development Environment (PDE) simplifies the creation of plugins. If it is not already loaded, you can download the PDE from <http://www.eclipse.org/downloads>, under the **Eclipse for RCP/Plug-in Developers** link.

16.3.2 Create Eclipse Plugin Project

A plugin project is needed to hold the Java class file and other resources. To create an Eclipse plugin project, use the New Project wizard:

1. Create a new Eclipse workspace.

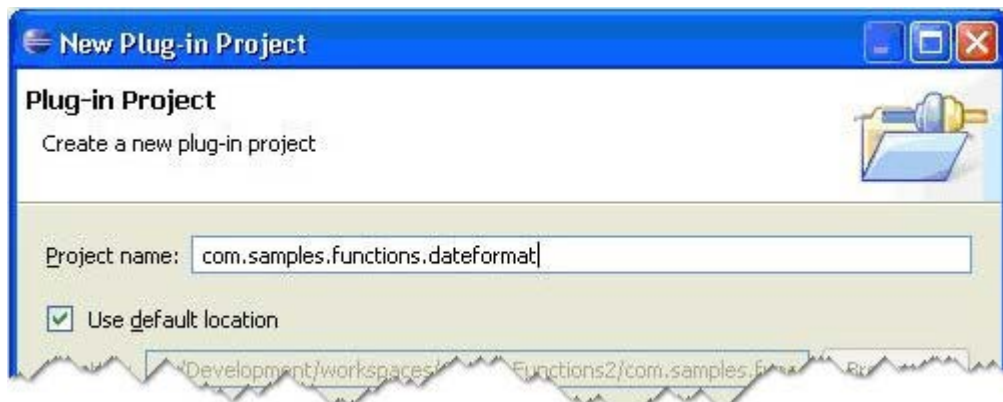
NOTE: You can work in an existing workspace, but you will be changing some settings that might affect other plugin development.

2. Select **New > Project** from the **File** menu.

The **New Project** wizard is displayed.

3. Select **Plug-in Development > Plug-in Project**, then click **Next**.

The **Plug-in Project** screen should be displayed.

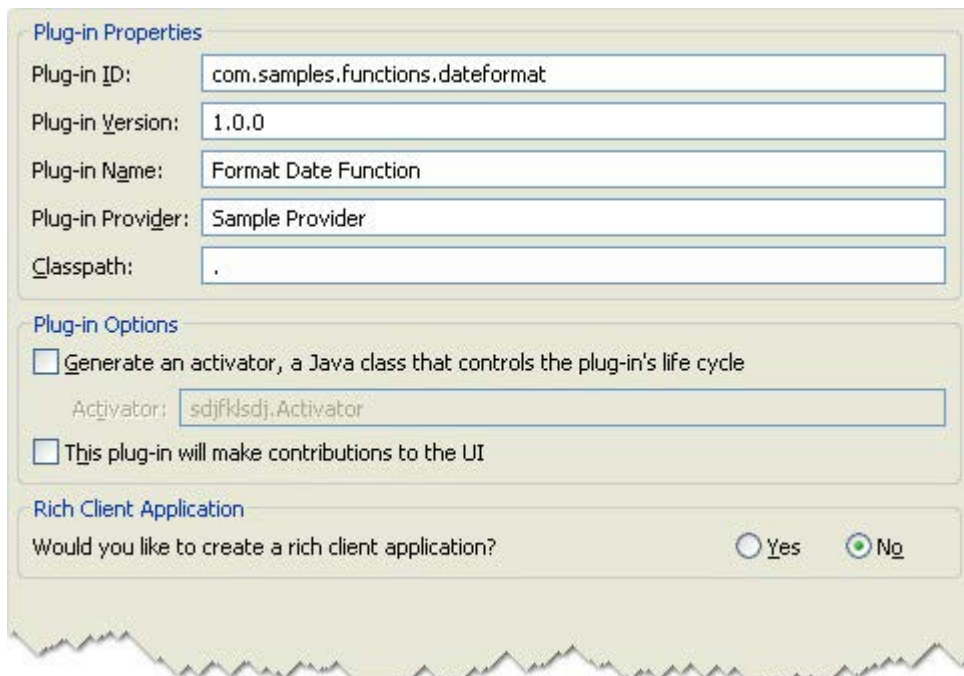


-
4. Enter a value in the **Project name** field.

This is the project's name as well as the plugin's ID. For our example, enter `com.samples.functions.dateformat`.

5. Click **Next** and enter the following values in the **Plug-in Content** screen.

Option	Value/Setting
Plug-in ID	This must be a unique ID for your plugin. If you will be developing more than one plugin, make sure that they have different IDs.
Plug-in Version	The version of the plugin (for example, 1.0.0).
Plug-in Name	A descriptive name for the plugin (for example, Format Date Function).
Plug-in Provider	The name of the company/individual who provided the plugin, normally the name of your company (for example, Sample Provider).
Classpath	Enter a "." (dot) to add the root of your plugin to the classpath.
Plug-in Options	Disable both options.
Rich Client Application?	Select "No" for this option.



Plug-in Properties

Plug-in ID:

Plug-in Version:

Plug-in Name:

Plug-in Provider:

Classpath:

Plug-in Options

☐ Generate an activator, a Java class that controls the plug-in's life cycle

Activator:

☐ This plug-in will make contributions to the UI

Rich Client Application

Would you like to create a rich client application? ☐ Yes ☒ No

6. Click **Finish**.

-
7. If you are prompted to switch to the Plug-in Development perspective, click **No**.

A plugin project should be created in your workspace. Additionally, a plugin project should exist in your Package Explorer and Eclipse should have opened the plugin's manifest (MANIFEST.MF) in the Manifest Editor.

16.3.3 Set Dependencies to Rational Integration Tester Functions

A dependency on Rational Integration Tester is needed to extend and implement the Function class. This is similar to adding a Java build path entry to Rational Integration Tester.

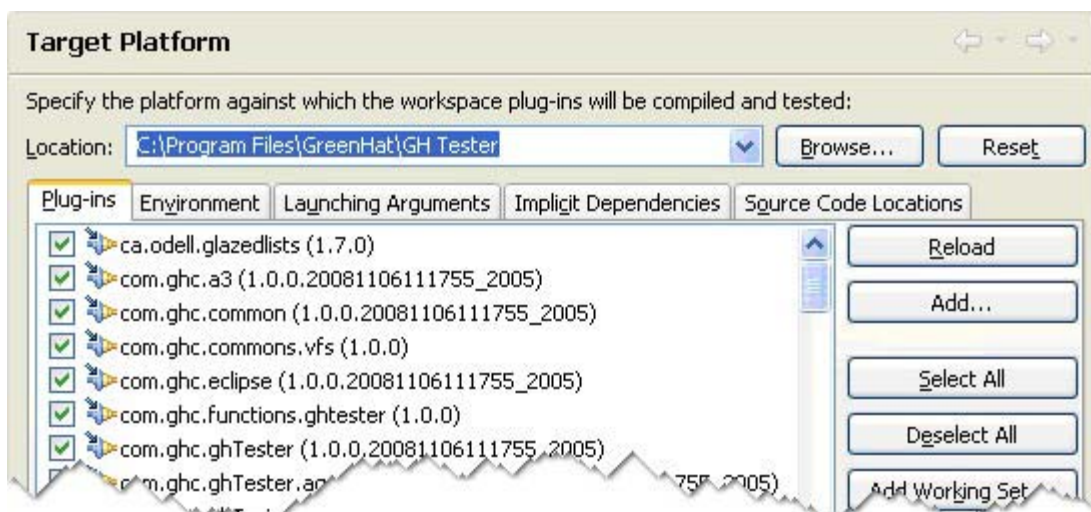
In Eclipse, the dependency is created by pointing the Target Platform to the Rational Integration Tester installation. Next, access to the function package in Rational Integration Tester must be declared.

Setup the Target Platform

Follow the steps below to set the target platform in Eclipse:

1. Go to **Window > Preferences > Plug-in Development > Target Platform**.
2. Click **Browse** and navigate to the Rational Integration Tester installation folder (C:\Program Files\IBM\Rational Integration Tester, by default).
3. Click **OK** to select the Rational Integration Tester installation folder and close the browse dialog.

All of the plugins in the Rational Integration Tester folder will be read and listed in the **Plug-ins** tab of the **Target Platform** preferences.



-
4. Click **OK** to close the **Preferences** dialog.

Set the Dependency in the Manifest

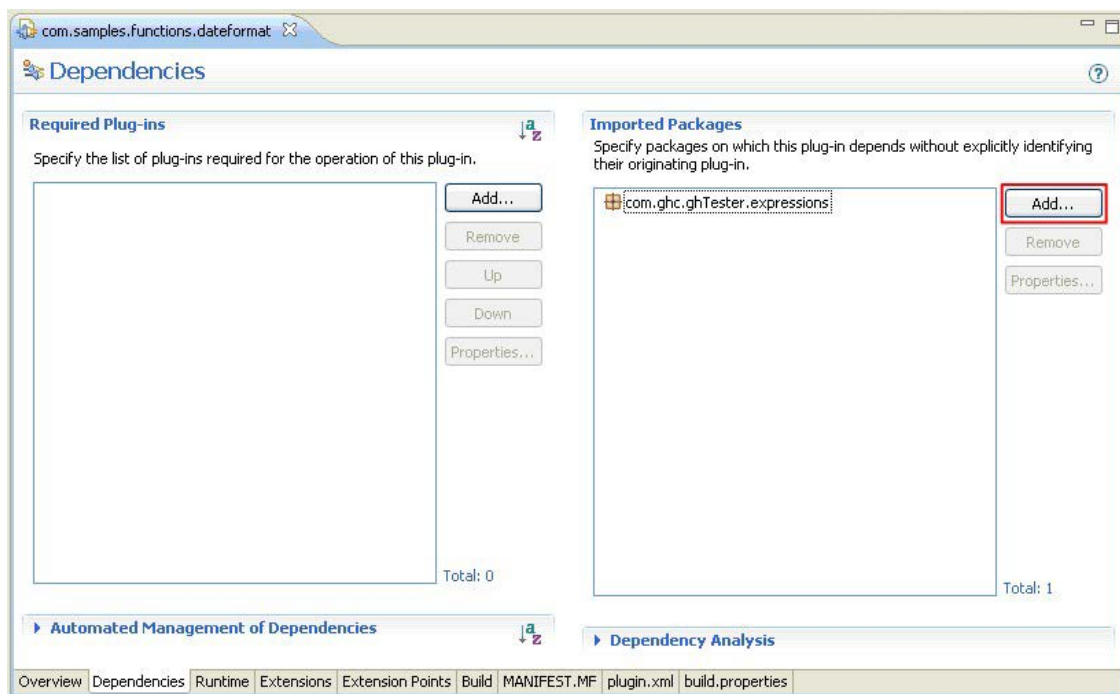
Follow the steps below to edit the plugin manifest:

1. Open the plugin's manifest by double-clicking **com.samples.functions.dateformat > META-INF > MANIFEST.MF**.
2. The file will be opened in an editor with a number of tabs running along the bottom.
3. Select the **Dependencies** tab and click **Add** next to **Imported Packages**.

The **Package Selection** dialog displays all available packages that the plugin can access.

4. Select **com.ghc.ghTester.expressions** and click **OK** to add the package to the list.

NOTE: You can use wildcards to find the package more easily. For example, ***expressions** should filter the required package to the top of the list.



5. Save the manifest when finished.

WARNING: Failing to save MANIFEST.MF at this point will cause the next steps in the process to fail.

16.4 Develop the Function Class

This section demonstrates how to implement the Function class.

NOTE: If you have a function written for Rational Integration Tester v4 and want to convert it to work in v5, see [Converting Existing Function Classes with Eclipse](#).

NOTE: Source code for the example can be found in the **examples\FunctionsSamplePlugin\src** folder of your Rational Integration Tester installation.

16.4.1 Java Class Extending from Function

Under the **src** folder of the plugin project, create a Java class that extends from `com.ghc.ghTester.expressions.Function` (for example, `com.samples.function.FormatDate`).

NOTE: Ensure the class is placed in a package other than the default package (that is, make a valid package declaration at the top of the Java class).

Now the following methods need to be implemented:

- The default, public constructor
- `create(int, Vector)`
- `evaluate(Object)`

16.4.2 Default Constructor

Create a default, public constructor with an empty body. You can add other initialisation as required, but it does not have to do anything, by default. The default constructor has to be public because of the way that functions are created.

```
public FormatDate() {  
}
```

16.4.3 Override `create(int size, Vector)` Parameters

The `create(int, Vector)` method is a factory method that creates an instance of the particular function with the parameters provided. The vector of parameters need to be treated as functions themselves and are likely to need processing at runtime.

```
public Function create(int size, Vector params) {
```

```
Function outputFormat = null;
if (size == 3) {
    outputFormat = (Function) params.get(2);
}
return new FormatDate((Function) params.get(0), (Function) params
    .get(1), outputFormat);
}
```

The three-argument constructor used in the example can be found in the source provided in the examples folder of the Rational Integration Tester installation folder.

16.4.4 Override evaluate(Object) Data

The evaluate(Object) method performs the actual work of the function. Rational Integration Tester passes the context of the current evaluation to your function. You can use this to obtain the result from embedded functions and then return your function's own result.

```
public Object evaluate(Object data) {
    String date = m_fDate.evaluateAsString(data);
    String inputFormat = m_fInputFormat.evaluateAsString(data);
    String outputFormat = "yyyy-MM-dd"; // Default format
    if (m_fOutputFormat != null) {
        outputFormat = m_fOutputFormat.evaluateAsString(data);
    }

    //...
    if (EvalUtils.isString(date)) {
        date = EvalUtils.getString(date);
    }
    if (EvalUtils.isString(inputFormat)) {
        inputFormat = EvalUtils.getString(inputFormat);
    }
    if (EvalUtils.isString(outputFormat)) {
        outputFormat = EvalUtils.getString(outputFormat);
    }

    SimpleDateFormat inputFormatter = new
        SimpleDateFormat(inputFormat);
```

```
String formattedDate = "";
try {
    Date d = inputFormatter.parse(date);
    SimpleDateFormat outputFormatter =new
    SimpleDateFormat(outputFormat);
    formattedDate = outputFormatter.format(d);
} catch (ParseException ex) {
    // ...
}

return "\"" + formattedDate + "\"";
}
```

NOTE: When implementing your evaluate method, the vector of parameters that are passed to you are functions that should be evaluated to discover their value. Where a string literal is expected, the evaluated function should return a value surrounded by quotes. There is a helper function available to assist with the removal of these:
`EvalUtils.getString(Object string).`

16.5 Create and Configure an Extension Point

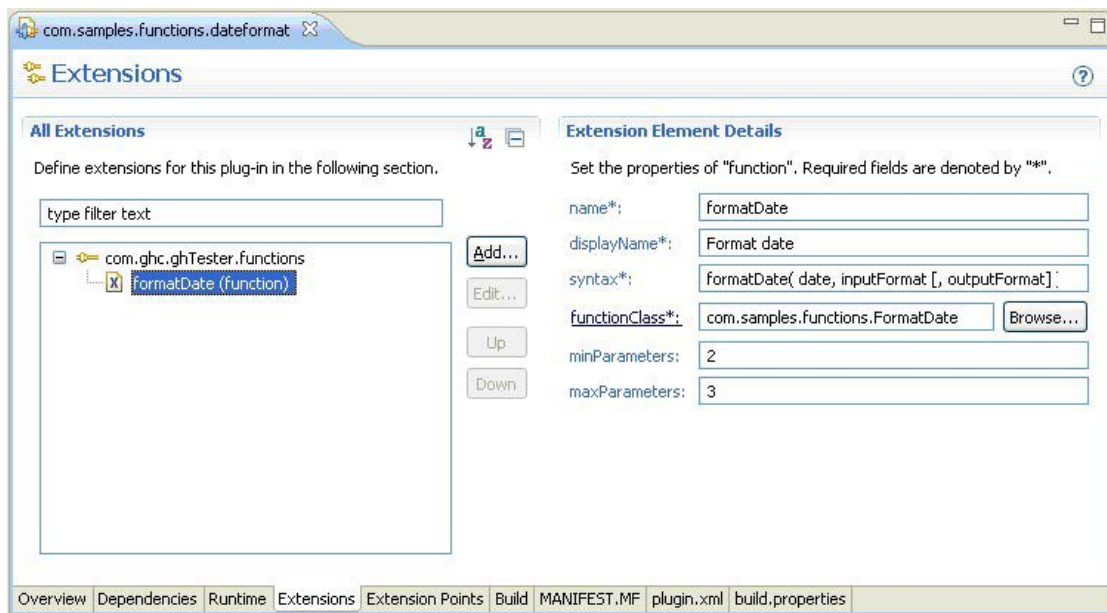
The function's extension point tells Rational Integration Tester that you want to contribute your own function and tells what it looks like. The extension point consists of a number of attributes that describe the function (for example, name) and the class that will be executed to do the function's work.

This section demonstrates how to create and configure the extension point:

16.5.1 Create the Extension Point

1. Open the plugin's manifest by double-clicking **com.samples.functions.dateformat > META-INF > MANIFEST.MF**.
2. The file will be opened in an editor with a number of tabs running along the bottom.
3. Select the **Extensions** tab and click **Add**.
4. Select **com.ghc.ghTester.functions** in the **New Extension** dialog.

NOTE: If the extension doesn't appear in the list, the dependency to Rational Integration Tester Functions has not been set up correctly.



5. Click **Finish**.
6. An extension point and an element will be added to the list of extensions.

16.5.2 Configure the Extension Point Element

1. Fill in the following fields in the Extension Element Details as described below. In all cases, exclude quotes from the example values:

Field	Value
name	The function's name (for example, "formatDate").
displayName	The name under which the function should appear in the Functions menu (for example, "Format Date"). To place the function under a specific sub-menu, prefix the display name with the name of the sub-menu and a forward slash (/) character. For example, "Custom/Format Date" would place the function under a "Custom" sub-menu. Any number of sub-menus is permitted using the same naming format.
syntax	The syntax of the function [for example, "formatDate(date, inputFormat [, outputFormat])"].
functionClass	The class that performs the function's work. This should be a fully-qualified class name that extends from com.ghc.ghTester.expressions.Function. You can enter the name manually or search for it using the Browse button. In the example, this is the com.samples.function.FormatDate class.
minParameters	The minimum number of parameters to pass to the function (for example, "2").
maxParameters	The maximum number of parameters to pass to the function, (for example, "3"). Entering "-1" indicates that there is no maximum and any number of arguments that is greater than or equal to minParameters will be accepted

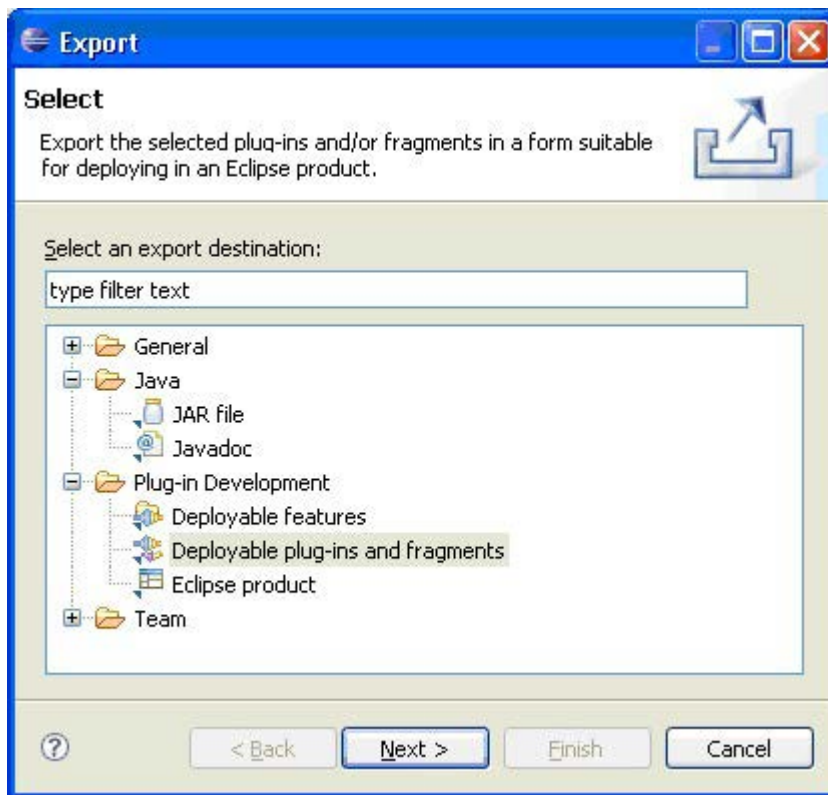
2. Save the manifest when finished.

WARNING:Failing to save MANIFEST.MF at this point will cause the next steps in the process to fail.

16.5.3 Generate the Plugin

Once the plugin and class have been created, they must be packaged. The plugin must be generated to enable Rational Integration Tester to pick it up (this is similar to generating a JAR).

1. Select **File > Export**.
2. In the **Export** dialog, select **Plugin-in Development > Deployable plug-ins and fragments** from the list of exports.



3. Click **Next** to display the **Deployable plug-ins and fragment** dialog.
4. In the **Available Plug-ins and Fragments** list, select the plugin you want to generate and export (for example, com.samples.functions.dateformat).

-
5. Select the **Directory** option and enter the directory where the plugin will be generated (that is, enter the path to the Functions folder under the root of your Rational Integration Tester project). You can also generate it in a temporary location and copy the JAR over into the Rational Integration Tester project.

NOTE: Eclipse will generate the plugin in a folder called “plugins”. Rational Integration Tester supports plugins that are located directly in the “Functions” folder or under a “plugins” subfolder.

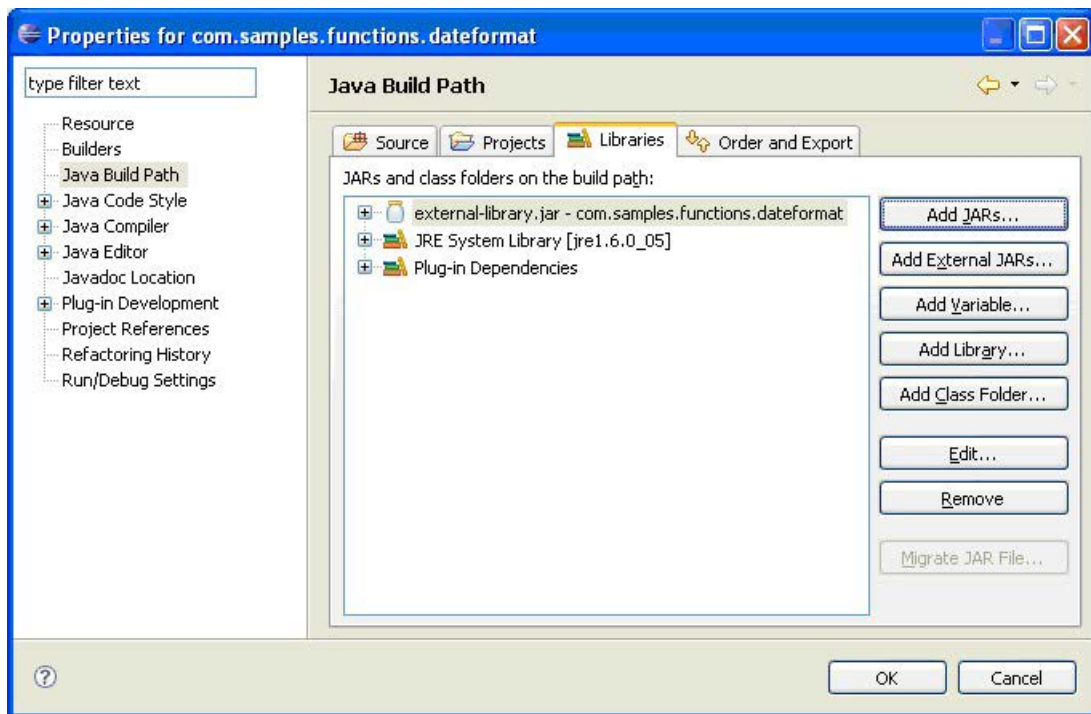
6. Click **Finish**.

The plugin is generated and placed in the folder that you specified earlier. The destination directory should contain a “plugins” folder and a JAR file (for example, com.samples.functions.dateFormat_1.0.0.jar).

16.5.4 Adding External Libraries

This section explains how to express a dependency on an external JAR and use it in your function. It assumes that you have a plugin defined in Eclipse, as explained in [Create a Plugin in Eclipse](#).

1. Copy the external JAR to the root of the plugin project.
2. Add the JAR to the plugin project's build path.
 - a. Right-click on the project and select **Properties > Java Build Path > Libraries**.
 - b. Click **Add JARs** and browse for the JAR you just added under the plugin project.
 - c. Close the **Properties** dialog.



3. Open the plugin's manifest file for editing and select the **Runtime** tab.
4. In the **Classpath** section, click **Add** and select the JAR that was copied into the project.
5. Save the manifest when finished.

Now you can start using the JAR's functionality inside your function.

16.6 Configure and Use the Function in Rational Integration Tester

After the function has been developed ([Create a Plugin in Eclipse](#)) and a plugin has been generated, ([Develop the Function Class](#)), you need to tell Rational Integration Tester about it.

Follow the steps below to configure the function for use in Rational Integration Tester:

1. If you have not yet generated the plugin in the “Functions” folder of the Rational Integration Tester project, then copy it there now (for example, if your project is stored under C:\RationalIntegrationTester\Project1, copy the plugin JAR or “plugins” folder to C:\RationalIntegrationTester\Project1\Functions).
2. In Rational Integration Tester, click **Tools > Reload Custom Functions** to load all functions in the “Functions” folder and the “plugins” subfolder.
3. If desired, click **Tools > View All Functions** to verify that your custom function has been loaded.

The Functions dialog displays a list of all the plugins known to Rational Integration Tester, sorted alphabetically by function name.

NOTE: If the new function is not listed, review the steps that have been carried out to ensure that nothing was missed.

You can now use the custom function like any other function in Rational Integration Tester.

16.7 Create a Function without Eclipse

Eclipse should be used to create the plugin. If this is not possible, however, this section demonstrates how to build a function using the Sun Java 6 SDK.

NOTE: The following steps assume that Rational Integration Tester is installed in C:\Program Files\IBM\Rational Integration Tester.

1. Create a folder in which to develop your custom function (for example, c:\customFunction).
2. Create folders for the source and the build output (for example, c:\customFunction\src and c:\customFunction\build).
3. Copy **plugin.xml** and **MANIFEST.MF** (in “META-INF”) from C:\Program Files\IBM\Rational Integration Tester\examples\FunctionsSamplePlugin into the build directory – both files should be copied to the same target directory (for example, C:\custom\build).
4. Create the necessary directory structure under “src” to contain your custom function code. If you have existing code, copy the root package and all subpackages under the “src” folder (for example, C:\custom\src\com\samples\functions\FormatDate.java).
5. From the “src” folder, compile the custom function into the build directory.

```
c:\customFunction\src> javac -d ../build -classpath .;"C:/Program Files/IBM/Rational Integration Tester/plugins/*" com/samples/functions/FormatDate.java
```

6. Edit **MANIFEST.MF** in your build directory and change the following values:

Value	Change to...
Bundle-Name	Enter a descriptive name for the plugin (for example, Format Date Function).
Bundle-SymbolicName	Enter a unique ID that will describe your plugin. This must be different from any other plugin, as two plugins with the same ID can not be loaded at one time. Leave the “singleton:=true” part untouched.
Bundle-Vendor	Enter your company's name or some other provider description.

7. Open **plugin.xml** (in your build directory) in a text editor and update the values as described in [Configure the Extension Point Element](#).

-
8. If you have a dependency on an external library (JAR), follow the steps in the next section, [Adding External Libraries](#).

9. From the build directory, create a JAR containing your custom function classes, **MANIFEST.MF**, and **plugin.xml**.

```
c:\customFunction\build> jar cvfm custom-function-plugin_1.0.0.jar  
META-INF\MANIFEST.MF com plugin.xml
```

NOTE: You can choose any name for your JAR, but you should follow a convention that prevents conflicting names.

10. Copy the new JAR file into the “Functions” folder of your Rational Integration Tester project.

When finished, follow the instructions to load the function into Rational Integration Tester (see [Configure and Use the Function in Rational Integration Tester](#)).

NOTE: An example of how the plugin should look can be found under
C:\Program Files\IBM\Rational Integration
Tester\FunctionsSamplePlugin.

16.7.1 Adding External Libraries

This section explains how to add a JAR as an external library to your plugin, and it assumes that you have worked from the template plugin, as described above.

1. Copy the external JAR to the root of the “build” folder.
2. Edit **MANIFEST.MF** in the “build” folder and locate the line starting with **Bundle-ClassPath**.
3. Add the JAR to the classpath after the dot.

It should be comma-separated and can be on the same line or on a separate line. If the JAR is on a separate line, ensure that you add a space at the beginning of the line.

```
Bundle-ClassPath: .,  
    external-lib.jar
```

4. Run the **jar** command from the “build” folder (as described in the previous section) to create a JAR that includes everything in the build folder, making sure to include your external JAR.

```
c:\customFunction\build> jar cvfm custom-function-plugin_1.0.0.jar  
MANIFEST.MF com plugin.xml external-library.jar
```

16.8 Converting Existing Function Classes with Eclipse

If you have an existing function written for Rational Integration Tester v4, you can use Eclipse to convert the function for use with Rational Integration Tester v5.

NOTE: It is unlikely that functions compiled for Rational Integration Tester v4 will work with Rational Integration Tester v5.

1. Remove the `register()` method as it is not required anymore.
2. Change the visibility of the default constructor to be public. No other visibility (for example, protected) can be used, otherwise an exception will be generated when trying to execute the function.
3. Remove the `super(String, String, int, int)` call in the default constructor.
This super constructor does not exist anymore, so any calls to it will produce an error. This call is replaced by the information contained in the extension point definition.
4. Remove the method `getSyntax()`. This information is now contained in the extension point definition.
5. Create an extension point to provide Rational Integration Tester with details about your function (see [Create and Configure an Extension Point](#)).

16.9 Converting Existing Function Classes without Eclipse

Eclipse should be used to convert an existing function. If this is not possible, however, this section demonstrates how to modify and build a function using the Sun Java 6 SDK.

NOTE: The following steps assume that Rational Integration Tester is installed in `C:\Program Files\IBM\Rational Integration Tester`.

1. Create a folder in which to develop your custom function (for example, `c:\customFunction`).
2. Create folders for the source and the build output (for example, `c:\customFunction\src` and `c:\customFunction\build`).
3. Copy **plugin.xml** and **MANIFEST.MF** (in “META-INF”) from `C:\Program Files\IBM\Rational Integration Tester\examples\FunctionsSamplePlugin` into the build directory – both files should be copied to the same target directory (for example, `c:\custom\build`).
4. Create the necessary directory structure under “src” to contain the code from your existing custom.
5. From your existing function, copy the root package and all subpackages under the “src” folder (for example, `c:\custom\src\com\samples\functions\FormatDate.java`).
6. Remove the `register()` method as it is not required anymore.
7. Change the visibility of the default constructor to be public. No other visibility (for example, protected) can be used, otherwise an exception will be generated when trying to execute the function.
8. Remove the `super(String, String, int, int)` call in the default constructor.
This super constructor does not exist anymore, so any calls to it will produce an error. This call is replaced by the information contained in the extension point definition.
9. Remove the method `getSyntax()`. This information is now contained in the extension point definition.
10. From the “src” folder, compile the custom function into the build directory.

```
c:\customFunction\src> javac -d ../build -classpath .;"C:/Program Files/IBM/Rational Integration Tester/plugins/*" com/samples/functions/FormatDate.java
```

-
11. Edit **MANIFEST.MF** in your build directory and change the following values:

Value	Change to...
Bundle-Name	Enter a descriptive name for the plugin (for example, Format Date Function).
Bundle-SymbolicName	Enter a unique ID that will describe your plugin. This must be different from any other plugin, as two plugins with the same ID can not be loaded at one time. Leave the “singleton:=true” part untouched.
Bundle-Vendor	Enter your company’s name or some other provider description.

12. Open **plugin.xml** (in your build directory) in a text editor and update the values as described in [Configure the Extension Point Element](#).

13. If you have a dependency on an external library (JAR), follow the steps in the next section, [Adding External Libraries](#).

14. From the build directory, create a JAR containing your custom function classes, **MANIFEST.MF**, and **plugin.xml**.

```
c:\customFunction\build> jar cvfm custom-function-plugin_1.0.0.jar  
META-INF\MANIFEST.MF com plugin.xml
```

NOTE: You can choose any name for your JAR, but you should follow a convention that prevents conflicting names.

15. Copy the new JAR file into the “Functions” folder of your Rational Integration Tester project.

When finished, follow the instructions to load the function into Rational Integration Tester (see [Configure and Use the Function in Rational Integration Tester](#)).

NOTE: An example of how the plugin should look can be found under
C:\Program Files\IBM\Rational Integration
Tester\FunctionsSamplePlugin.

External Tools

Contents

Overview

Command Line Execution

ANT

HP Quality Center

HP QuickTest Professional

This chapter describes how to execute Rational Integration Tester tests, suites, and stubs from a command line, using ANT, and in HP Quality Center.

17.1 Overview

Rational Integration Tester is primarily GUI-based, but it supports the execution of tests, suites, and stubs outside of the Rational Integration Tester GUI. Several methods of externally executing Rational Integration Tester test artifacts are available, as follows:

- [Command Line Execution](#) - after being created in the GUI, test, stubs, and suites can be executed manually or in an automated fashion from the command line.

NOTE: Command line execution is supported on Windows and UNIX platforms.

- [ANT](#) - Rational Integration Tester can generate a custom ANT script that will execute selected tests, suites, or stubs as part of your overall build process.
- [HP Quality Center](#) - Rational Integration Tester can generate a VAPI-XP Java script that can be pasted into HP Quality Center to execute the selected test, suite, or stub as part of your Quality Center test plan.
- [HP QuickTest Professional](#) - Resources from Rational Integration Tester can be executed from QuickTest Professional (and vice versa) using scripting and command line options in both tools.

NOTE: If command line or ANT execution will be used in projects that use permissions, single sign on must be configured. This can be achieved by adding a properly configured `krb5.ini` file (one that includes your realm and key distribution center) to your Windows directory (typically `C:\WINDOWS`). Alternatively, these details can be provided as arguments in Library Manager (for command line) or within the ANT script (manually added). For more information about this, refer to *IBM Rational Integration Tester Installation Guide*.

17.2 Command Line Execution

To facilitate automatic execution of Rational Integration Tester resources (for example, as part of a daily build process), Rational Integration Tester can be operated from the command-line using the RunTests executable or script.

NOTE: If you are using the runtests script, which uses a network licence file, the license needs to be accessible every time the script runs (unlike the GUI which only needs to access the license once).

NOTE: Rational Test Control Panel 8.0.1 (or later) has a command-line client application that is part of the GHTesterCmd.exe executable. (For information about this, refer to *IBM Rational Test Virtualization Server Reference Guide*.)

The command line executable and script have the following usage:

```
runtests [-licence <key>] [-noHTTP] [-noResultsPublishers]
[-useResultsPublishers <pub_1>,<pub_2>] [-resultsServerLogging
absolute|relative|ignore] -environment <environment name> -project
<file> -run "<res_1>;<res_2>"
```

or

```
runtests [-licence <key>] [-noHTTP] [-noResultsPublishers]
[-useResultsPublishers <pub_1>,<pub_2>] [-resultsServerLogging
absolute|relative|ignore] -parameterFile <file>
```

NOTE: If specifying more than one test resource, they should be separated by a semicolon and enclosed in quotes.

For example:

```
C:\> "C:\Program Files\IBM\RationalIntegrationTester\runtests" -
noHTTP
-useResultsPublishers MyPub,TIB_Pub -resultsServerLogging ignore
-environment Hotair -project
C:\RationalIntegrationTesterProjects\HotAir\HotAir.ghp -run
"Hotair/Airline/booking/MakeBooking/CardType = Visa"
```

NOTE: If any of the command line parameters include spaces, they must be enclosed in quotes.

The command line parameters are described in the following table:

<code>-license <key></code>	The license key <key>, if required, to use for running Rational Integration Tester.
<code>-noHTTP</code>	An optional switch that is used to disable the internal web server. This switch should not be used when running performance tests.
<code>-noResultsPublishers</code>	An optional switch that is used to disable any results publishers that may be configured in the project.
<code>-useResultsPublishers <pub_1>,<pub_2></code>	When executing a test suite, specifies one or more results publishers (previously configured in the project) that should be enabled for publishing. Each publisher is designated by the name that it was given when created in Rational Integration Tester, and multiple publishers should be separated with a comma.
<code>-resultsServerLogging absolute relative ignore</code>	Specifies if and how the Results Server URL for executed items should be written to the console. The argument has the following options: <ul style="list-style-type: none">• ignore: The report URL will not be written to the console.• absolute: The full URL will be written to the console (default value if none are specified).• relative: A URL relative to the project's current Results Server location will be written to the console.
<code>-environment <environment name></code>	The name of the Rational Integration Tester environment <environment name> to use when executing the test item(s).
<code>-project <file></code>	The full path to the Rational Integration Tester project file <file> that contains the specified environment and test resources.
<code>-run <res_1>;<res_2></code>	The full path (within the project) of the test resources <res_1>...<res_n> to be executed.
<code>-parameterFile <file></code>	The full path to a parameter file that contains run options for one or more resources (see below).

If desired, the **project**, **environment**, and **run** options can be set in a parameter file, which can be specified using the **parameterFile** option.

Run options can be specified for one or more test resources, grouped by project and separated by a semicolon. As an example, consider a file named **myparams.txt** that has the following contents:

```
-project C:\Projects\AETesting\myproj.ghp -environment Env1
-run "creditTest1";"creditTest2";
-project C:\Projects\BWTesting\myproj.ghp -environment HotAir
-run "bookVisa";"bookMaestro";"regSuite1";"regSuite2";
```

The parameter file can be utilized in the `runtests` command, as follows:

```
runtests -noHTTP -resultsServerLogging ignore -parameterFile  
C:\RationalIntegrationTesterTests\myparams.txt
```

In the case of the example file, the command will execute the following resources:

In the **AETesting** project, **creditTest1** will be executed followed by **creditTest2**, and both will be executed in the **Env1** environment. Next, in the **BWTesting** project and using the **HotAir** environment, **bookVisa**, **bookMaestro**, **regSuite1**, and **regSuite2** will be executed (in that order).

NOTE: A final trailing semicolon is used in the parameter file to close the list of artefacts to execute.

17.3 ANT

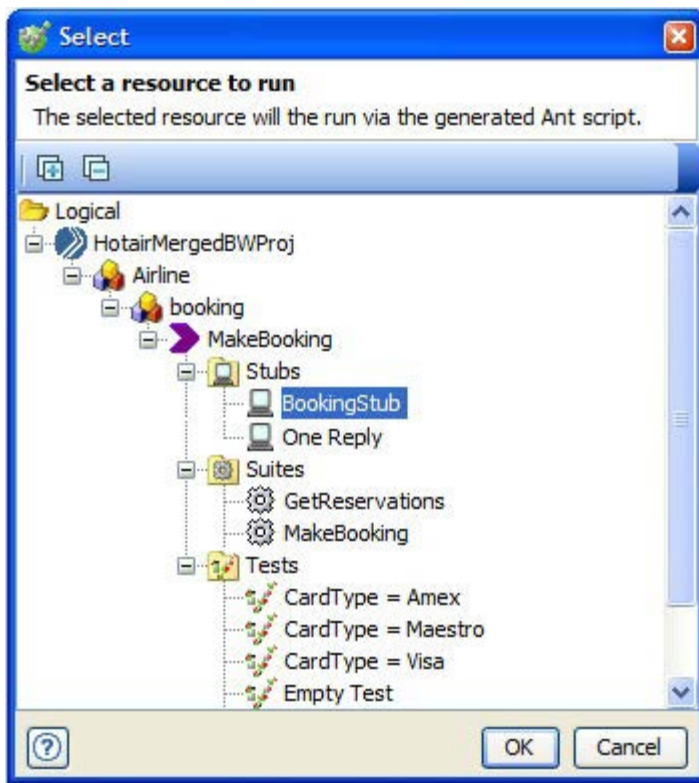
Rational Integration Tester can generate scripts that can be used to run test resources automatically from within ANT. For more information about ANT, go to <http://ant.apache.org>.

NOTE: Rational Integration Tester Platform Pack 8.0.1 (or later) includes an optional Rational Test Control Panel Ant Client application. (For information about this application, refer to *IBM Rational Test Virtualization Server Reference Guide*.)

Follow the steps below to generate an ANT script from Rational Integration Tester:

1. Select **Generate ANT Script** from the **Tools** menu.

The project resource tree is displayed.



2. Select the tests, suites, or stubs that should be included in the script (use **Ctrl** or **Shift** to select multiple items) and click **OK** to proceed.
3. When prompted, select a location and name for the build script and click **Save** to save the file.

NOTE: Rational Integration Tester will not overwrite an existing file if the filename you select is already in use. It will let you use a modified filename, or you can cancel and enter a new name.

An example ANT script, generated by Rational Integration Tester, is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<project default="run-ghtester" basedir="C:\Projects\HotAir Project">
  <property name="install.dir" value="C:\Program
Files\IBM\RationalIntegrationTester"/>
  <taskdef name="ghtester" classname="com.ghc.ghTester.ant.GHTester"
classpath="${install.dir}\plugins\com.ghc.ghTester.ant.jar"/>
  <target name="run-ghtester" depends="version-check">
    <ghtester environment="Hotair" licence="1212...3434"
project="${basedir}\HotAir Project.ghp" resultsServerLogging="absolute">
      <tests>
        <filelist dir="${basedir}">
          <file name="Hotair/MakeBooking/BookingStub"/>
          <file name="Hotair/MakeBooking/MakeBooking"/>
          <file name="Hotair/MakeBooking/Maestro"/>
          <file name="Hotair/MakeBooking/Empty Test"/>
        </filelist>
      </tests>
    </ghtester>
  </target>
  <target name="version-check">
    <condition property="${correct.version}">
      <available classname="org.apache.tools.ant.taskdefs.condition.AntVersion"/>
    </condition>
    <antcall target="supported-version"/>
  </target>
  <target name="supported-version" unless="${correct.version}">
    <fail message="Please install ant version 1.7.0 or higher"/>
  </target>
</project>
```

The script would be run in ANT like any other build script (for example, ant -buildfile HotAir-build.xml). When using JMS- or JDBC-based items in Rational Integration Tester, the JNDI provider classes need to be made available on the main classpath using the **-lib** argument (for example, ant -lib <JAR location> -buildfile HotAir-build.xml).

If your project is configured to use permissions, single sign on must be configured (see [Overview](#)). If krb5.ini is not available in your Windows directory, the following properties can be manually added to the script:

```
<AppProperties>
```

```
<property name="java.security.krb5.realm" value="<REALM>" />
<property name="java.security.krb5.kdc" value="<KEY DIST CENTER>" />
...
</AppProperties>
```

17.4 HP Quality Center

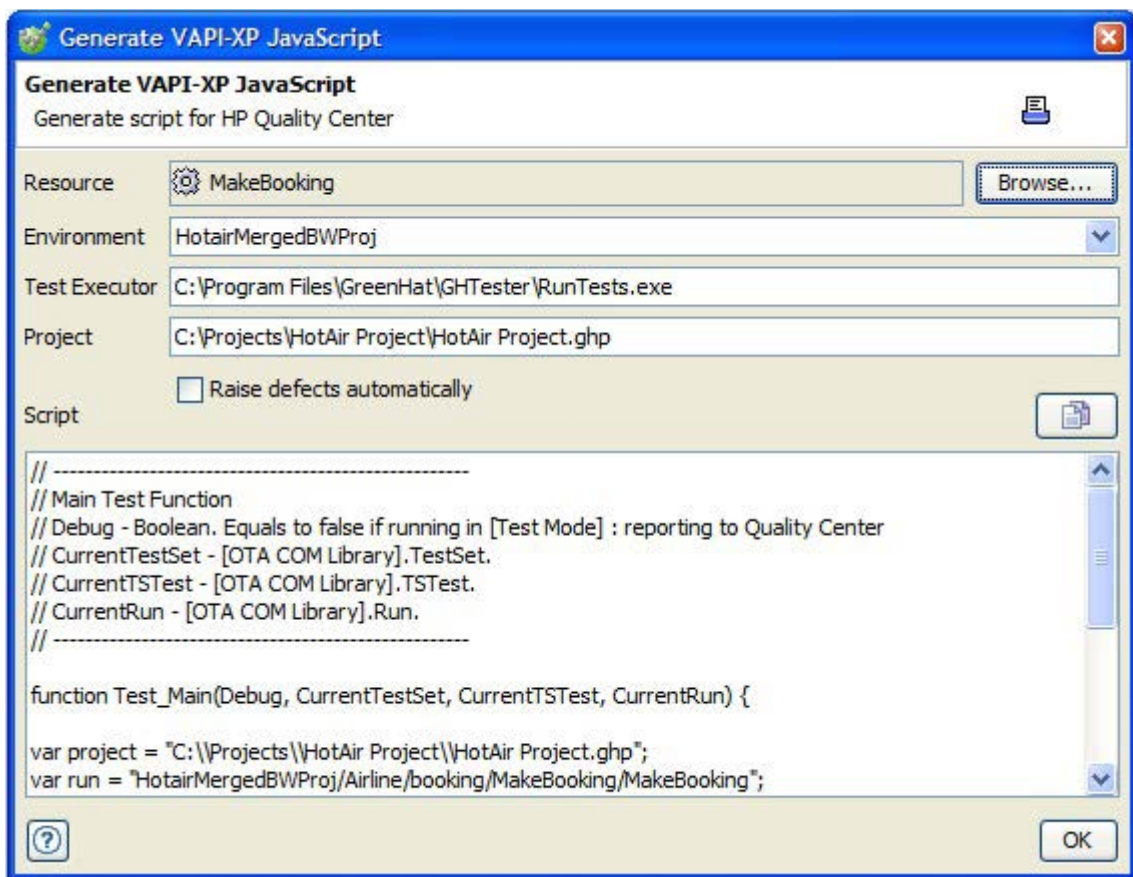
If you are running VAPI-XP tests in HP Quality Center, you can generate VAPI-XP script from Rational Integration Tester that you can use to create tests in Quality Center.

NOTE: Rational Integration Tester must be installed on the Quality Center client machine to enable the execution of Rational Integration Tester test resources.

Follow the steps below to generate a VAPI-XP script from Rational Integration Tester:

1. Select **Generate VAPI-XP JavaScript** from the **Tools** menu.


The **Generate VAPI-XP JavaScript** dialog is displayed.



-
2. Modify the script properties using the fields at the top of the dialog, as follows:

Resource	The Rational Integration Tester test, suite, or stub to be executed by the script. If a resource had been selected in the Test Factory perspective, it will be automatically selected. Click Browse to locate and select a different test resource from the project tree.
Environment	Select the existing Rational Integration Tester environment in which the selected resource should be executed.
Test Executor	The path to the RunTests executable, which will be used by Quality Center to execute the selected resource. This field is populated according to the Rational Integration Tester installation location, and it should not be modified.
Project	The path to the project file that is currently open in Rational Integration Tester. This field is editable, but it should not be changed.
Raise defects automatically	Enable this option if you want to have Quality Center raise defects automatically in the case of a failed execution. Otherwise, this field should be left empty (disabled).

NOTE: The contents of the script, in the lower portion of the dialog, will be modified according to any changes made to the fields above.

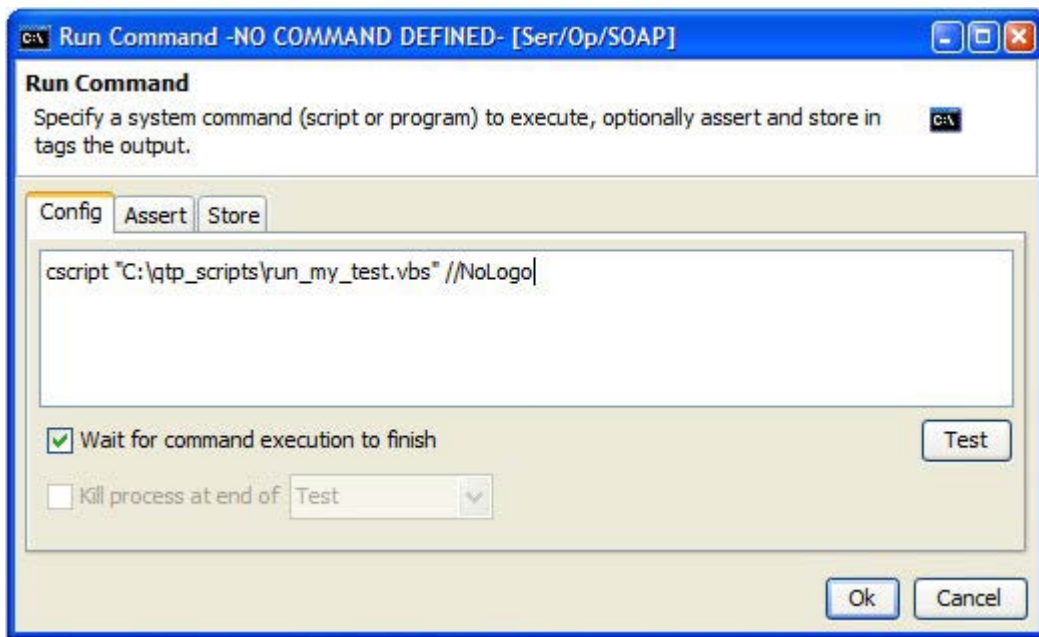
3. When you have set the script properties as desired, click the copy icon  to copy the contents of the script to the clipboard.
4. In Quality Center, paste the contents of the clipboard into your test and save it.

The selected Rational Integration Tester resource can now be run from Quality Center using the new VAPI-XP test.

17.5 HP QuickTest Professional

A manual integration between Rational Integration Tester and HP QuickTest Professional can be achieved by running one tool from the other by means of supported command line options.

To execute a QuickTest Professional test, you can use the **Run Command** action within a Rational Integration Tester test. You can execute a VB script that defines the QTP test and run parameters using the built-in **cscript.exe** command on the Windows platform.



Additional information can be found in the QuickTest Professional documentation.

Similarly, Rational Integration Tester resources can be invoked from QuickTest Professional by utilizing Rational Integration Tester's command line execution options. See [Command Line Execution](#) for more information.

Troubleshooting

Contents

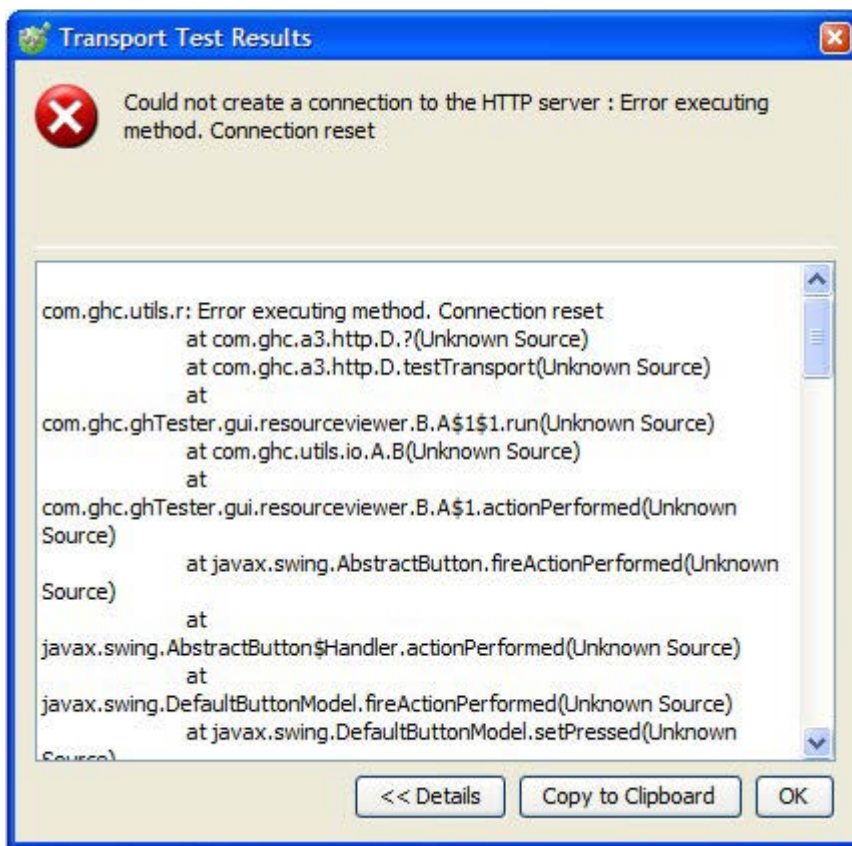
Transport Diagnostics

This chapter describes how to use the Transport Diagnostic troubleshooting utility that is included with Rational Integration Tester.

18.1 Transport Diagnostics

Transport Diagnostics, which are built into the configuration of each transport (by means of the **Test Transport** button), displays a summary and optional details of any problems you might have when configuring transports in Rational Integration Tester.

When configuring transports in Rational Integration Tester, it is useful to test the configuration details before saving the transport (that is, click the **Test Transport** button in the transport editor). If the test is successful you can be confident that the transport is configured properly. If the test is unsuccessful, however, the Transport Test Results dialog is displayed. This dialog provides additional details that may be useful in diagnosing the issue with the transport configuration.



You can show or hide the problem details by clicking the **Details** button. To copy the exception or problem details that have been captured – regardless of whether or not the details are displayed – click the **Copy to Clipboard** button. You can then save the details by pasting them into a file, or paste them into an email that can be sent to IBM for help in troubleshooting the issue. After you have completed viewing/using the transport diagnostics, click **OK** to close the dialog.

Appendix A: Test Actions

Contents

Messaging Actions

BPM Actions

Flow Actions

General Actions

Performance Actions

This chapter provides information about the various test actions that can be configured in Rational Integration Tester tests and stubs.

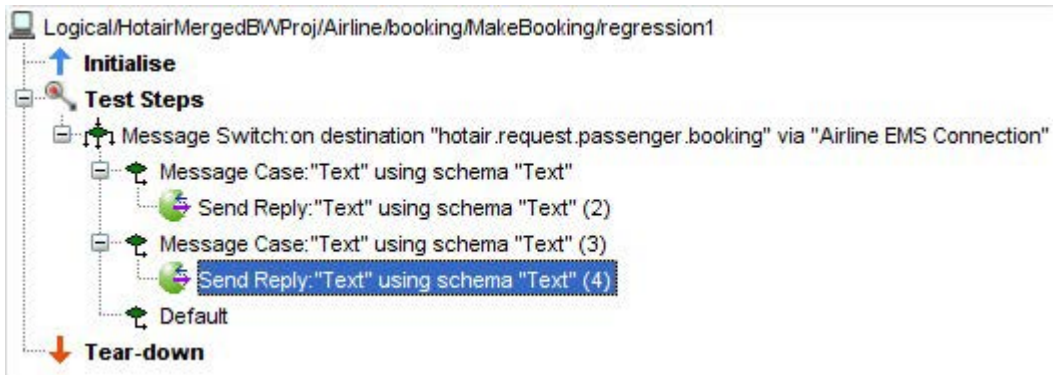
19.1 Messaging Actions

The following message-based actions are available for Rational Integration Tester tests:

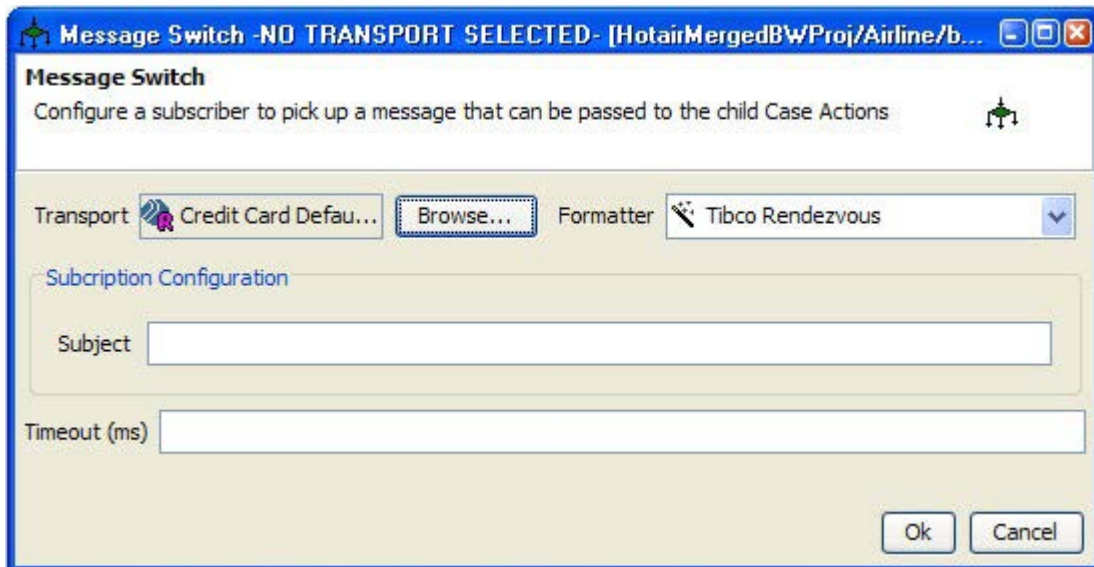
- [Message Switch](#)
- [Publish](#)
- [Receive Reply](#)
- [Receive Request](#)
- [Send Reply](#)
- [Send Request](#)
- [Subscribe](#)
- [Unsubscribe](#)

19.1.1 Message Switch

The Message Switch action, which is the basis for most stubs, is a subscriber that passes on received messages to the message cases it contains.



In the above example, all messages sent to “hotair.request.passenger.booking” will be picked up on the Airline EMS Connection transport and passed on to the subsequent case actions. Configuring a message switch is the same as configuring a Subscribe action.



Within the message switch you can add message cases (see [Add a Message Case](#)). Each case is intended to handle a different message. The final case (created with the message switch) is the **Default** case that will catch any message in case none of the other cases are matched. Matching a received message occurs sequentially, from top to bottom, until a match is found. The matching is manipulated with filters in the message case.

Within each message case, users can insert additional actions to be carried out when the case is matched (for example, send a reply to the request, insert a value in the database, and so on).

19.1.2 Publish

The Publish action is used to send messages to a subject, topic or queue. To publish messages, at least one transport must be configured in the project.

Publish
Publish a message on a transport. Select the transport, formatter, and define the message content to be sent.

Config Value Store

Transport OpsServiceAg... Browse... Formatter Tibco Rendezvous

Message Header

Subject

Reply Subject

☐ Generate Inbox

Message	Value	
(RV Message)	Process Children	<input checked="" type="checkbox"/>
ID (String)	01256894	<input checked="" type="checkbox"/>
Live (Boolean)	True	<input type="checkbox"/>

Actions

E	Action	Value
<input type="checkbox"/>	Validate Message Children	Any Field Order:false Ignore Extra Fields:false
<input checked="" type="checkbox"/>	Process Children	
<input type="checkbox"/>	Name	
<input type="checkbox"/>	Type	

Ok Cancel

Before considering the body of the message to send, you must first configure the context of the message:

- Click **Browse** to select a transport from the resource selection dialog. See [Transports and Formatters](#) for more information if you need to create a transport.
- Select a formatter for the specified transport, as detailed in [Transports and Formatters](#).
- Configure the message header, as explained in [Messages](#). When using a TIBCO transport, it is usually a matter of setting the subject. When using a JMS transport, the queue/topic name must be configured.

The message configuration varies according to the transport in use (see [Messages](#) for more information). Note, however, that the header and the body of the message must both be defined.

The header is the part of the message that controls how the transport treats it. It includes, for instance, the queue name in JMS or the subject name in Rendezvous, as well as other fields.

The body is the part of the message that has meaning for applications consuming/receiving/subscribing to the message. Normally, the body structure is either arbitrary or defined at design time and stored in a schema.

19.1.3 Receive Reply

The Receive Reply action is essentially the same as a Subscribe action, except that the transport is defined in the Send Request action, so there is no need (or ability) to set it. See [Send Request](#) for more information.

In the Receive Reply action, a timeout period can be set so that if no messages have been received after the timeout has elapsed, the action will fail. The timeout period is specified in milliseconds.

Used in conjunction with the Timeout is the Tolerance, which is also set in milliseconds. The timeout tolerance specifies the number of additional milliseconds (beyond the existing timeout period) to wait for a received message.

If a message is received after the timeout but during the additional tolerance period, the test still fails and no validation is performed, but the console reports that the message was received late. The default timeout tolerance can be set under the General settings of Rational Integration Tester's preferences.

19.1.4 Receive Request

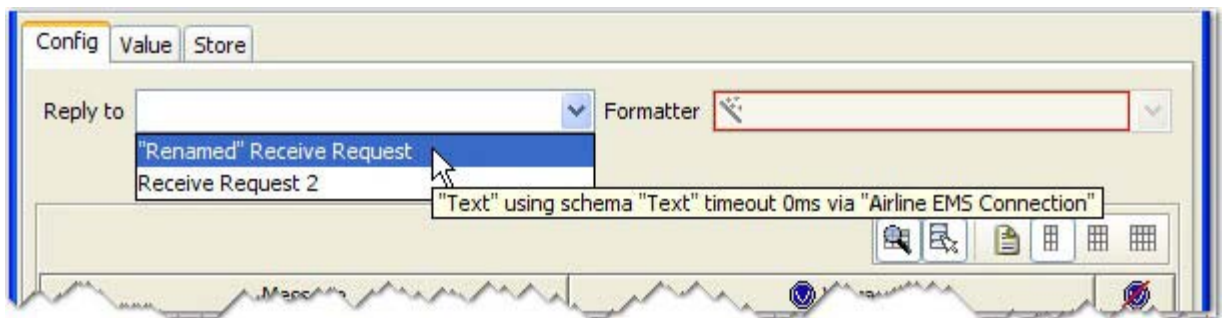
A Receive Request action is identical to a subscriber, except that it is used together in a test with a Send Reply action (and must come before the Send Reply in a test).

See [Send Reply](#) for more information.

19.1.5 Send Reply

A Send Reply action is the same as a Publish action. The only difference is that a Send Reply action must be used somewhere after a Receive Request action (since it is supposed to reply to it). Therefore, instead of selecting a transport, you select an existing Receive Request action. The remainder of the action and the message are edited in the normal way.

The name displayed in the “Reply to” combo box is a numbered step, unless the step has been renamed in the Business View. In this case, the assigned name will be shown instead, but a tool-tip on each entry in the combo box will detail the Technical View name for that step.

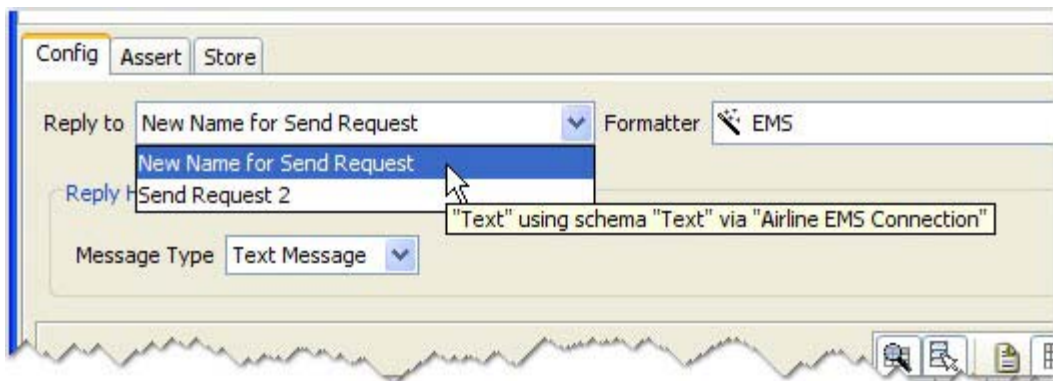


19.1.6 Send Request

The Send Request action is essentially the same as a Publish action, except that the request is intended for another specific action, the Receive Reply. When creating a Send Request, the corresponding Receive Reply is created at the same time.

You can insert different step actions between the Send Request and Receive Reply, so a list of mappings is offered for each response.

In the Receive Reply, the name displayed in the “Reply to” combo box is a numbered step, unless the step has been renamed in the Business View. In this case, the assigned name will be shown instead, but a tool-tip on each entry in the combo box will detail the Technical View name for that step.



If the mapping selected in a Receive Reply action becomes invalid (that is, the action is placed above the Send Request action or the Send Request action is deleted), then a red border will appear around the combo box to warn the user.

19.1.7 Subscribe

The Subscribe action is used to receive messages from a subject, topic or queue. For each Subscribe action, a transport and formatter must be selected, and you must specify the requirements for the messages that will be received. The success or failure of the action depends on whether the received messages conform to the requirements.

Subscribe
Choose a transport and formatter on which to receive and validate a message.

Config Filter Assert Store

Transport: OpsServiceAg... Browse... Formatter: Tibco Rendezvous

Subscriber Configuration

Subject:

Message	Value		Store	
(RV Message)	Validate Message Children	<input checked="" type="checkbox"/>		<input type="checkbox"/>
ID (String)		<input type="checkbox"/>	ID	<input checked="" type="checkbox"/>
Live (Boolean)	True	<input checked="" type="checkbox"/>		<input type="checkbox"/>

Actions

E	Action	Value
<input type="checkbox"/>	Equality	
<input checked="" type="checkbox"/>	Name	
<input checked="" type="checkbox"/>	Type	
<input checked="" type="checkbox"/>	Store copy of field in tag 'ID'	ID

Timeout (ms): 0 Tolerance (ms): 5000

Ok Cancel

In the above example, the field called “ID” is set so that the subscriber will not attempt to validate its content, but will stored the content in a tag also called “ID”. The content of the field called “Live” is set to be validated but not stored.

At the top of the Subscribe action window are four tabs: Config, Filter, Assert, and Store.

- **Config** – configure general subscriber settings (for example, transport, formatter, subject, and so on)
- **Filter** – restrict which messages to receive based on the content of the message's header and body (only those messages that conform to the specified filter(s) will be validated or stored; non-conforming messages will be ignored)
- **Assert** – contains the validation settings for the message header and body
- **Store** – contains the store settings for the message header and body

A timeout period can be set, so that if no messages have been received after the timeout has elapsed, the action will fail. The timeout period is specified in milliseconds.

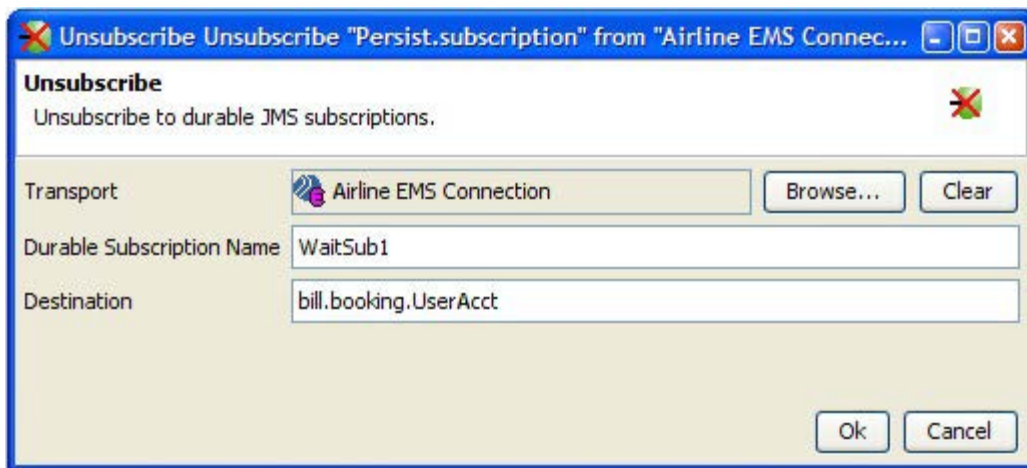
Used in conjunction with the Timeout is the Tolerance, which is also set in milliseconds. The timeout tolerance specifies the number of additional milliseconds (beyond the existing timeout period) to wait for a received message.

If a message is received after the timeout but during the additional tolerance period, the test still fails and no validation is performed, but the console reports that the message was received late. The default timeout tolerance can be set under the General settings of Rational Integration Tester's preferences.

19.1.8 Unsubscribe

When using JMS messaging, it is possible to subscribe to a topic-based destination using a durable subscription. This can be enabled by selecting the Durable option when creating or modifying a Subscribe action. A durable subscription instructs the applicable messaging broker to retain any messages that arrive while the subscription is inactive. These retained messages will be presented to the application the next time it connects.

If you want to stop receiving such messages without changing the subscription, you can use the Unsubscribe action.



To unsubscribe, specify the Transport, the Durable subscription name and the Destination that were configured previously in the applicable Subscribe action.

19.2 BPM Actions

Currently, Rational Integration Tester interfaces directly with TIBCO iProcess to simulate user interactions, such as starting a case or retrieving a work item from a work queue. For specific steps in a test you can filter the appropriate work queue, update data in an iProcess work item, and move on to different steps.

Using the test actions available in Rational Integration Tester, it is possible to move a case through its step definitions without any human intervention. For example, step form fields (Staffware form fields) can be updated with new data using the Process Work Item action. Rational Integration Tester can also simulate actions in 3rd party systems (for example, BusinessWorks or a web service) upon which iProcess might have dependencies, such as receiving triggers or creating cases.

The following iProcess actions are available for Rational Integration Tester tests:

- Start Case
- Retrieve Case
- Close Case
- Retrieve Task
- Modify Task
- Trigger Event

For additional details about using these test actions to interface with TIBCO iProcess, refer to *IBM Rational Integration Tester Reference Guide for TIBCO*.

NOTE: No action in Rational Integration Tester directly modifies case data, the data is only retrieved. Three actions, however, can alter the state of data in the iProcess engine: Start Case, Modify Task, and Trigger Event.

19.3 Flow Actions

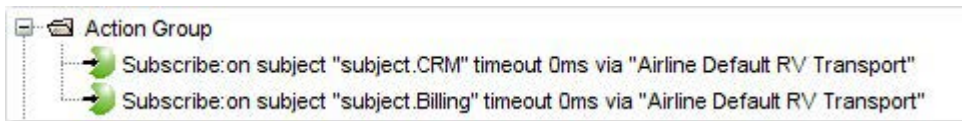
The following flow-based actions are available for Rational Integration Tester tests:

- [Action Group](#)
- [Assert](#)
- [Decision](#)
- [Fail](#)
- [Fetch Test Data](#)
- [Iterate Actions](#)
- [Iterate Test Data](#)
- [Iterate While](#)
- [Lookup Test Data](#)
- [Pass](#)
- [Run Test](#)
- [Sleep](#)

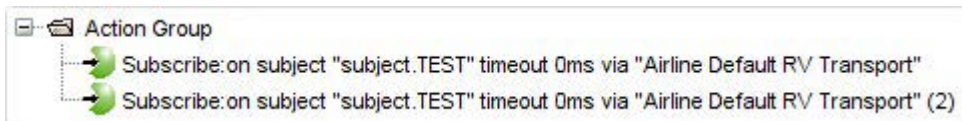
19.3.1 Action Group

An Action Group combines multiple actions and treats them as a single test step (for example, to control how subscription-based operations are started and how they process their received messages). Messages in an Action Group are processed in parallel, as opposed to one after the other when used as normal steps in a test sequence.

In the example shown below, an Action Group contains two subscribers on different subjects. When executed, the test will start both subscribers and validate their corresponding messages, regardless of the order in which they are received. The Action Group is considered complete after both messages have been received.



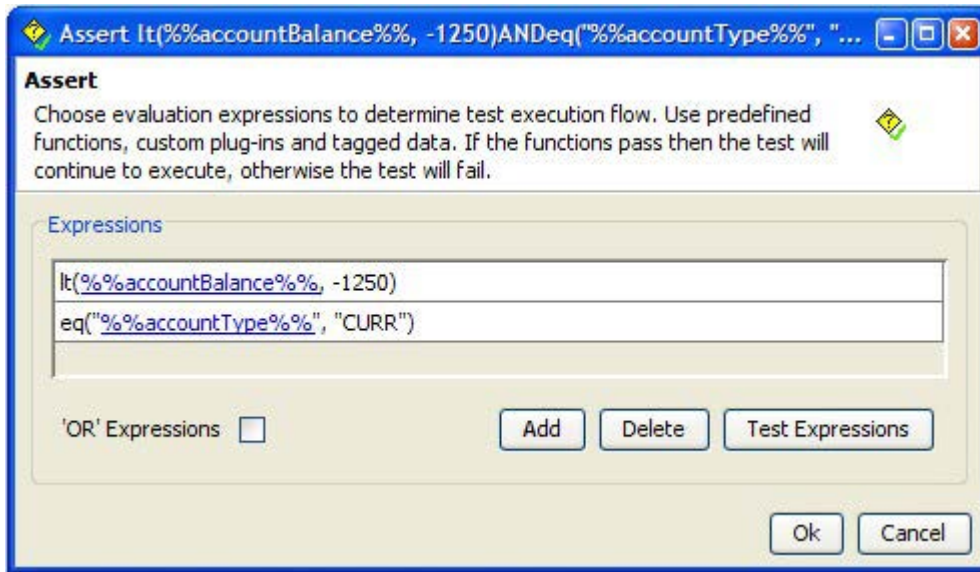
Below, two subscribers have the same subject (that is, they will both receive the first subject message simultaneously). This enables separate filters to be applied at the same time.



19.3.2 Assert

The Assert action is a conditional action that lets the user construct a list of expressions that determine whether or not a test should proceed. If the Assert action

passes, the test continues. If the Assert action fails, the test fails and no further test actions are executed.

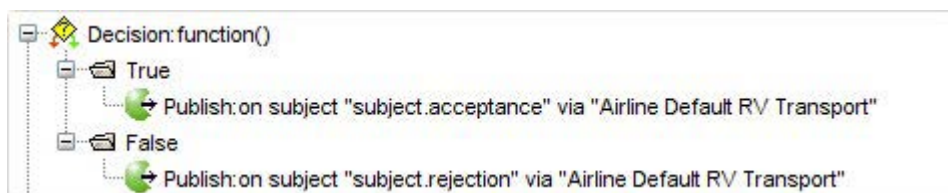


To enter an expression, click **Add** and type it into the empty input field. To remove an expression, select it and click **Delete**. To test the expression(s), click **Test Expressions**.

By default, multiple expressions are joined with a logical AND, meaning that all expressions must pass in order for the action to pass and for the test to proceed. To join expressions with a logical OR, enable the **'OR' Expressions** option, which means that only one of the expressions needs to pass in order for the action to pass and for the test to proceed.

19.3.3 Decision

A Decision is a conditional action, the body of which is essentially a function and whose result controls test flow. A Decision can have one of two outcomes: **True** or **False**.

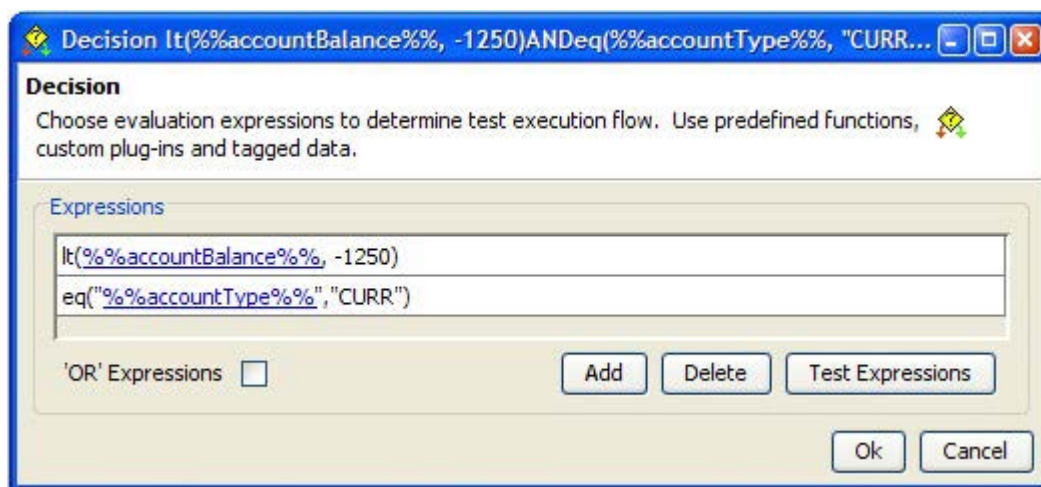


Any Rational Integration Tester function can be added as an expression from the Decision editor's context menu (for example, simple expressions using XPath value comparison operators, such as equals, less than, and so on). Expression arguments can be tags or literals.

Multiple expressions are processed as being joined together by a logical AND. To override this (that is, join them with a logical OR), enable the **'OR' Expressions** option.

NOTE: The Decision action can not test whether a tag has been populated.

To enter an expression, click **Add** and type it into the empty input field. To remove an expression, select it and click **Delete**. To test the expression(s), click **Test Expressions**.

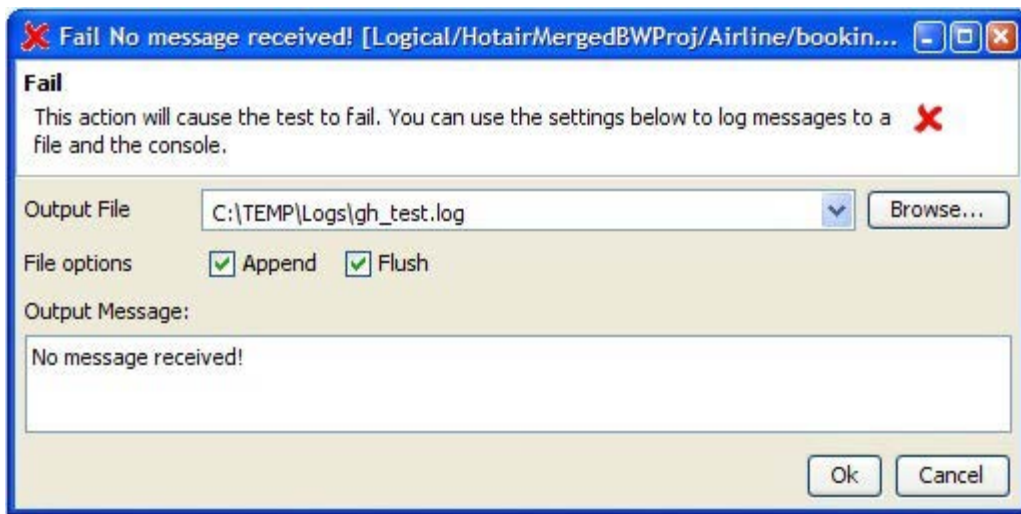


In the example shown, the Decision tests if **%%accountBalance%%** is less than -1250 and the **%%accountType%%** is equal to "CURR".

NOTE: String comparisons are performed if the parameters are enclosed in double-quotes, otherwise numeric comparisons are performed. When comparing Boolean values, they must be treated as strings (for example, `eq("%%myTag%%", "false")`).

19.3.4 Fail

When a Fail action is encountered during the execution of a test, the test is deemed to have failed and its execution stops. This action is commonly used as one of the paths under a Decision step to force a test to fail its current iteration.



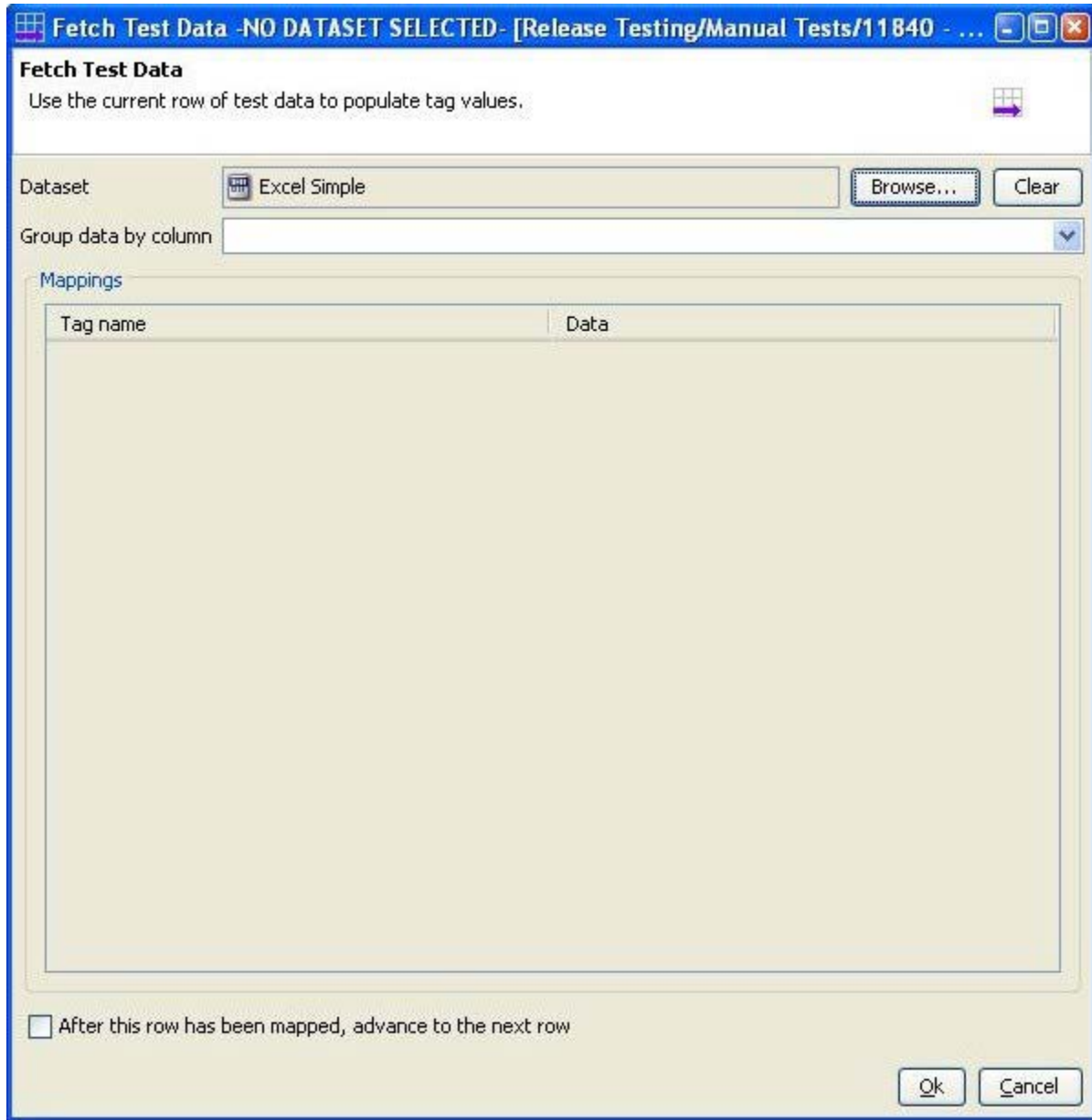
In the Fail action editor you can set the location of the logging output file by typing the file's full path (which can include tags), clicking **Browse** to locate the file, or by selecting an entry from a list of the most recently used files. If no output file is set, no logging will occur.

Once an output file has been designated, two additional file options can be enabled/disabled: **Append** and **Flush**.

- If Append is enabled, the test action will add new logging to the existing file contents. Otherwise, the existing contents will be overwritten.
- If Flush is enabled, the test action will write the message to the file immediately. Otherwise, messages are buffered and written in batches (which may improve performance).

19.3.5 Fetch Test Data

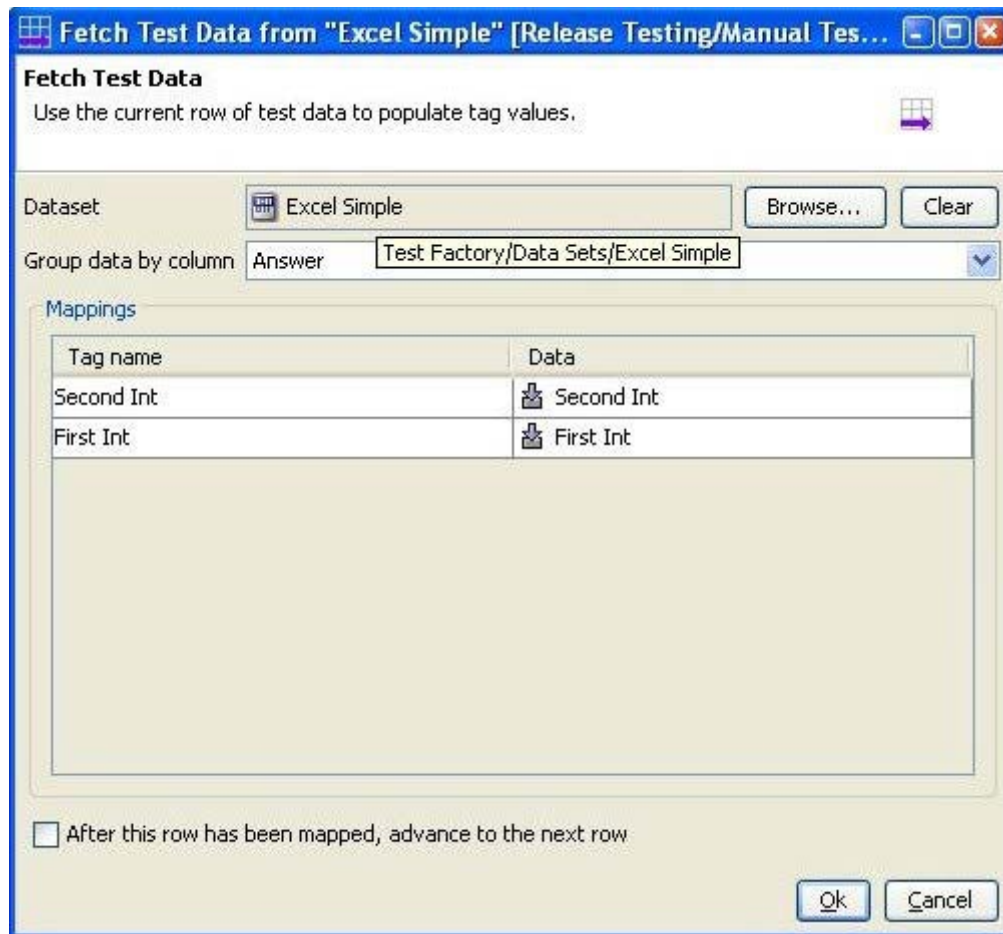
The Fetch Test Data action can be used to map data from a test data source to existing test tags.



Click **Browse** to locate and select the data source from the project resource dialog. The “Group data by column” list, which is sorted alphabetically, facilitates working with test data sets containing repeating elements. The column that uniquely defines the rows that belong to the group (usually a parent ID) must be selected.

In the **Mappings** section, select the data source field to map for each of the desired tags.

If the fetch action will be executed in multiple iterations, you can map the next row of data for each subsequent iteration by enabling the “advance” option at the bottom of the action editor.



19.3.6 Iterate Actions

The Iterate actions ([Iterate](#), [Iterate Test Data](#), and [Iterate While](#)) let you specify an action or group of actions that should be executed multiple times. The number of iterations is specified in different ways, according to the specific test action in use.

When using test iterations with the **Creates new test iteration** option enabled, the built in tag SYSTEM/ITERATION/NUMBER is set to 1 and will be incremented for each loop inside the iterator. While running the test, the **Progress** column in Test Lab will display the progress as a percentage of the current iteration against the total iterations.

NOTE: If this option is not used, the entire test will run in a single iteration, and the progress bar will move from 0% to 100% upon completion.

If two iterators are used in a test with this option enabled, the progress bar will run through to 100% in the first iterator, and then again for the second iterator.

Iterate

The Iterate action lets you specify an action or group of actions to be executed multiple times. Actions within the Iterate action will be executed the specified number of times.

Iterate (2) [Tests/Sequences/HTTP 1]

Iterate
Repeat test steps a number of times. Each iteration must be explicitly listed e.g. 1,5,7-12.

Config **Store**

Iterations: 1-4

Pacing

☒ Pacing pause between iterations for 2 seconds

Runtime Settings

☐ Creates new test iteration

☒ Continue on fail

Iteration Timing

☒ Fail the iteration if it takes longer than 10 seconds

☐ Cancel iterator if total time exceeds 0,0 seconds

Ok Cancel

The table below describes the options available for configuring the action:

Option	Description
Iterations	The number of times the contained actions should be executed.
Pacing	Control how frequently the iteration will run, as follows: pause between iterations for: pause for n seconds after executing one iteration before executing the next (it will not pause after the last iteration) run iterations no more frequently than once every: if an iteration completes in less than n seconds, pause for the remaining time (for example, if $n = 5$ seconds and an iteration completes in 3 seconds, pause for 2 seconds before executing the next iteration)
Creates new test iteration	Considers repeated iterations as real iterations in console output and reports.
Continue on fail	If enabled, a failure within the iteration group will not force the entire test to fail. Otherwise, if an action in the iteration group fails, the test fails and any remaining steps are cancelled.
Fail the iteration...	If enabled, the active iteration will fail if it takes longer than the specified period of time (in seconds).
Cancel iterator...	If enabled, the iterations will be canceled if they are not completed within the specified period of time (in seconds).

Iterate Test Data

The Iterate Test Data action is similar to the Iterate action, letting you specify an action or group of actions that should be executed multiple times. The number of iterations, however, is controlled by the number of filter matches found in the selected data set.

Iterate Test Data [Tests/Sequences/HTTP 1]

Iterate over a test data set

Config Filter Store

Test data set Iterate Data Browse... Clear

Group data by column ID

Iterations

Pacing

☒ Pacing pause between iterations for 3 seconds

Runtime Settings

☒ Creates new test iteration

☒ Continue on fail

Iteration Timing

☒ Fail the iteration if it takes longer than 10 seconds


☐ Cancel iterator if total time exceeds 0.0 seconds

Ok Cancel

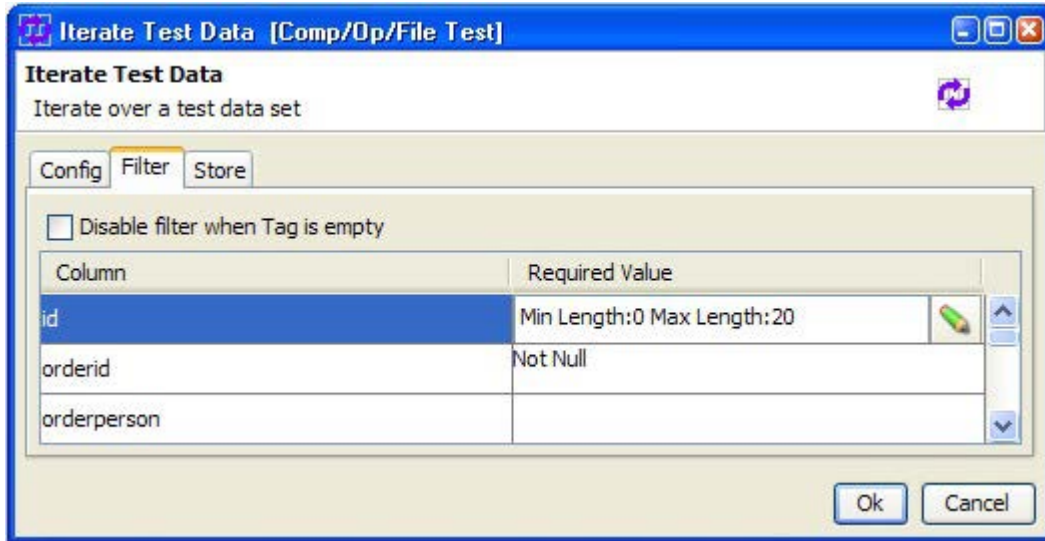
The table below describes the options available for configuring the action:

Option	Description
Test data set	The data set to search.
Group data by column	The (optional) column within the selected data set to use as the basis for repeating elements. When used, all rows within the data set that have the same value in the “grouping” column will be grouped together and sorted alphabetically as repeating elements.
Iterations	The number of times the contained actions should be executed.
Pacing	Control how frequently the iteration will run, as follows: pause between iterations for: pause for n seconds after executing one iteration before executing the next (it will not pause after the last iteration) run iterations no more frequently than once every: if an iteration completes in less than n seconds, pause for the remaining time (for example, if $n = 5$ seconds and an iteration completes in 3 seconds, pause for 2 seconds before executing the next iteration)
Creates new test iteration	Considers repeated iterations as real iterations in console output and reports.
Continue on fail	If enabled, a failure within the iteration group will not force the entire test to fail. Otherwise, if an action in the iteration group fails, the test fails and any remaining steps are cancelled.
Fail the iteration...	If enabled, the active iteration will fail if it takes longer than the specified period of time (in seconds).
Cancel iterator...	If enabled, the iterations will be canceled if they are not completed within the specified period of time (in seconds).

NOTE: If using the “Group tests by column” option, ensure that any repeating fields in the data set are marked as such in the message (if repeating elements are not marked, errors will be generated).

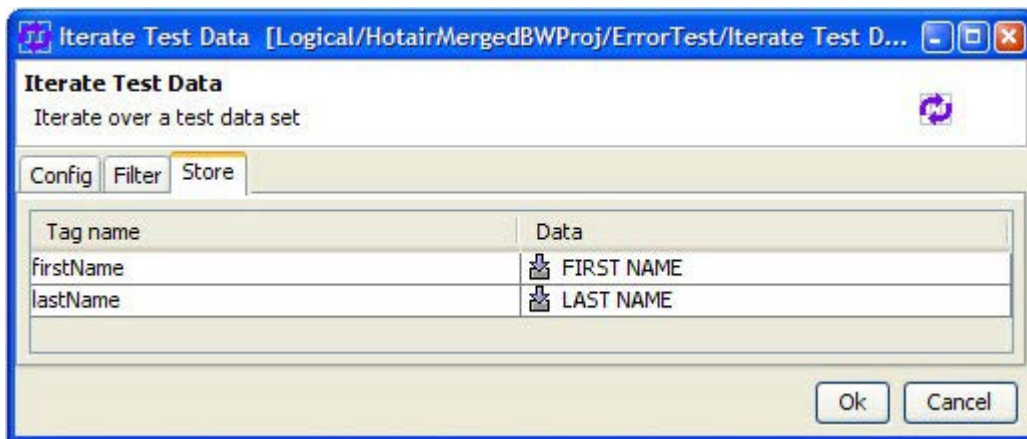
Under the **Filter** tab you can specify values to match in columns within the data set. Text entered under the **Required Value** field will be used as an “Equality” match by default. To modify the filter type, click the  icon, select the desired filter action, and enter the matching criteria as needed.

Only the rows containing data that matches all filters will be used. If the **Disable filter when Tag is empty** option is enabled (ticked), the filter will not be applied when an empty value ("" or NULL) is found in the Tag.



NOTE: The number of rows matching all filters determines the number of iterations that will be carried out.

Under the **Store** tab you can map the data in specific columns (within matching rows) to existing tags in the test.



For example, based on the **filter** and **store** actions shown above, the values for FIRST NAME and LAST NAME will be mapped to `%%firstName%%` and `%%lastName%%`, and only the rows where FIRST NAME equals "John" and CITY equals "London" will be used.

Iterate While

The Iterate While action is similar to the Iterate action, letting you specify an action or group of actions that should be executed as long the specified conditions are met. One or more conditions can be specified. If AND expressions are used (default), then all conditions must be true. If OR expressions are used, then at least one condition must be true.

Iterate While
Repeat test steps while a specified condition is true.

Config

Condition

It(%%ID%%, 20)

'OR' Expressions ☐

Add Delete Test Expressions

Pacing

☐ Pacing run iterations no more frequently than once every 0.0 seconds

Runtime Settings

☐ Creates new test iteration

☒ Continue on fail

Iteration Timing

☐ Fail the iteration if it takes longer than 0.0 seconds

☐ Cancel iterator if total time exceeds 0.0 seconds

Ok Cancel

The table below describes the options available for configuring the action:

Option	Description
Condition	One or more conditions to compare when running the action. For AND expressions, all listed expressions must return true for the action to continue. For OR expressions, at least one of the listed conditions must return true for the action to continue. Conditions can be added and removed using the Add and Delete buttons. To test the results of the condition, click Test Expressions .
Pacing	Control how frequently the iteration will run, as follows: pause between iterations for: pause for n seconds after executing one iteration before executing the next (it will not pause after the last iteration) run iterations no more frequently than once every: if an iteration completes in less than n seconds, pause for the remaining time (for example, if $n = 5$ seconds and an iteration completes in 3 seconds, pause for 2 seconds before executing the next iteration)
Creates new test iteration	Considers repeated iterations as real iterations in console output and reports.
Continue on fail	If enabled, a failure within the iteration group will not force the entire test to fail. Otherwise, if an action in the iteration group fails, the test fails and any remaining steps are cancelled.
Fail the iteration...	If enabled, the active iteration will fail if it takes longer than the specified period of time (in seconds). Tags can be used in this field.
Cancel iterator...	If enabled, the iterations will be canceled if they are not completed within the specified period of time (in seconds). Tags can be used in this field.

19.3.7 Lookup Test Data

The Lookup Test Data action provides keyed access to a dataset. For example, in a stub you might receive a request message and tag value for customerName. You can

search the dataset, keying off the customerName, and returning data you can use to send a reply.

Lookup Test Data
Use a value from the current run to extract information from a test data set.

Dataset: Iterate Data Browse... Clear

Lookup Values

Column Key	Lookup Value
ID	8
Last Name	Thomas

Add Lookup Remove Lookup

☐ Return all matching results

Mappings

Tag name	Data
ID	No mapping
Address	No mapping
Last Name	Last Name
First Name	First Name

Ok Cancel

The table below describes the options available for configuring the action:

Option	Description
Dataset	The dataset, previously configured in Rational Integration Tester (see Test Data Sources), to use for looking up data.
Lookup Values	The Column Key within the dataset to use as the lookup key, and the Lookup Value to search for within the dataset based on that key. The value is likely to be dynamic and come from a tag that is populated earlier in the test sequence. To use multiple columns, click Add Lookup . To remove a column, select it and click Remove Lookup .
Return all matching results	If enabled, the action will return multiple matching results as a list instead of a single tag value.

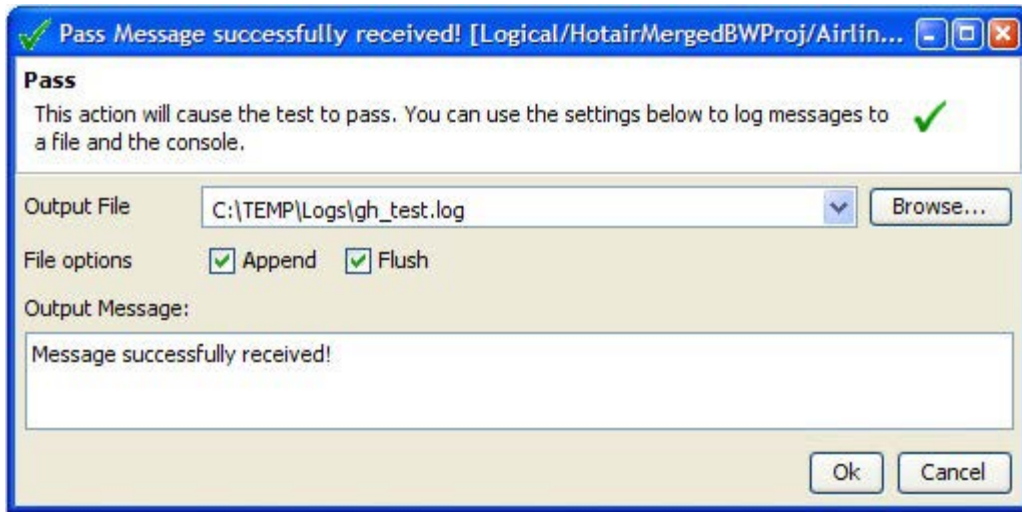
Mappings

Tag name	The name of an existing tag within the test.
Data	The column value from a returned row in the data set to which the tag should be mapped. For example, if %%customerName%% is found, then the values in the ADRESS and PHONE columns of the same row should be mapped to tags named %%Address%% and %%Phone%%.

Like the Decision action, Lookup Test Data provides two execution paths. If a row is found in the dataset, the **Found** branch is executed. Otherwise, the **Not Found** branch is executed. Additional actions can be added to the two paths to provide different success or failure results.

19.3.8 Pass

A Pass action is used to indicate that the current iteration of the test should be considered to have passed – no subsequent actions in the sequence will be processed. This action is commonly used as one of the paths under a decision step to make a test pass its current iteration.



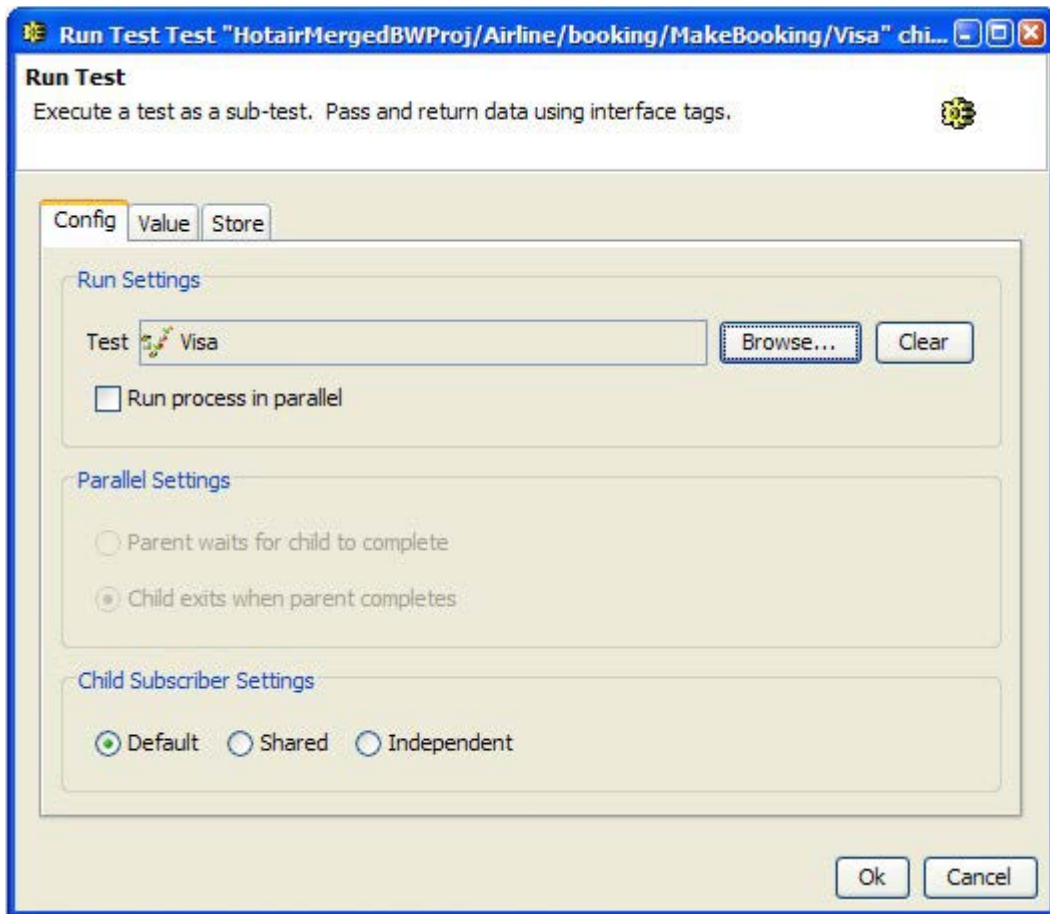
In the Pass action editor you can set the location of the logging output file by typing the file's full path (which can include tags), clicking **Browse** to locate the file, or by selecting an entry from a list of the most recently used files. If no output file is set, no logging will occur.

Once an output file has been designated, two additional file options can be enabled/disabled: **Append** and **Flush**.

- If Append is enabled, the test action will add new logging to the existing file contents. Otherwise, the existing contents will be overwritten.
- If Flush is enabled, the test action will write the message to the file immediately. Otherwise, messages are buffered and written in batches (which may improve performance).

19.3.9 Run Test

The Run Test action executes another test as a step (sub-test) of the current test.



The table below describes the general options available for the action under the **Config** tab:

Run Settings

Test	Click Browse to select the test to execute. Click Clear to reset the field.
Run process in parallel	Enable this option to execute the selected test in parallel with the parent test (that is, the test that contains this Run Test action). NOTE: If this option is enabled, the Store tab is unavailable.

Parallel Settings (used in conjunction with “Run process in parallel”)

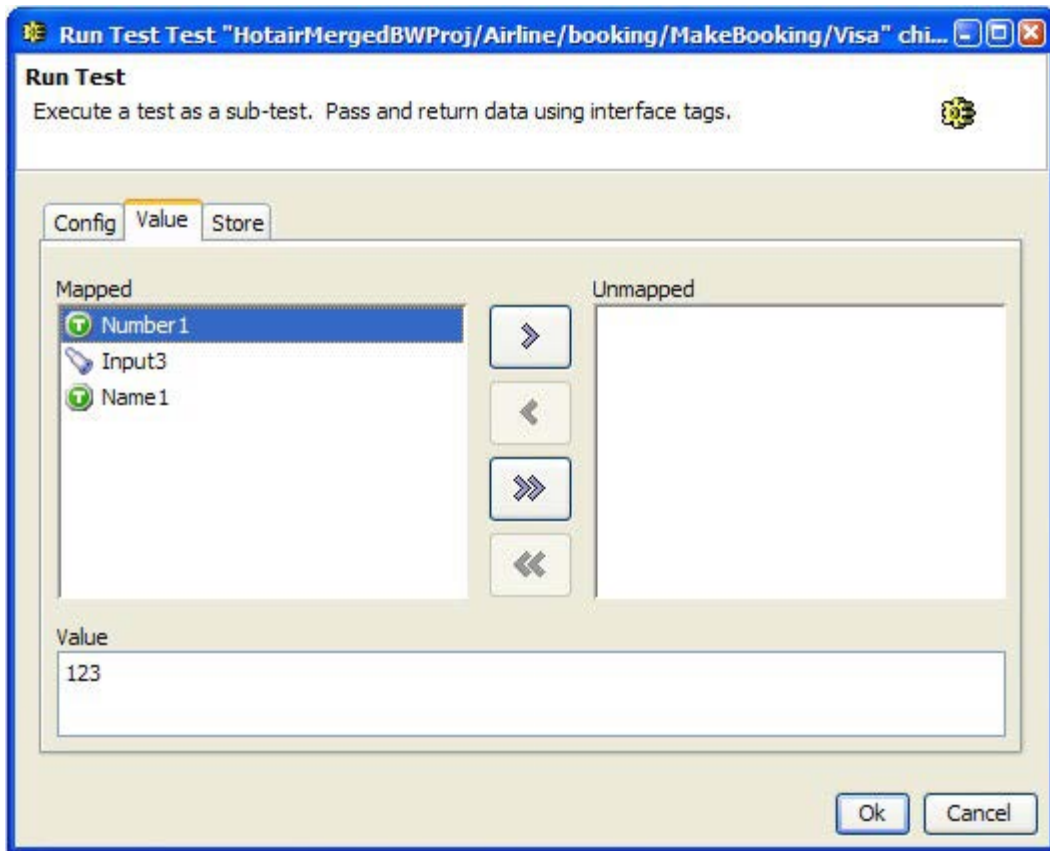
Parent waits for child to complete	The parent test will wait for the selected test to finish before terminating.
Child exits when parent completes	The selected test will be terminated as soon as the parent test is finished executing.

Child Subscriber Settings (controls message delivery when parent and child tests subscribe to the same message subjects)

Default	Use independent subscribers in parallel and shared subscribers in series.
Shared	Separate message queues deliver messages to both processes.
Independent	Subscribers retrieve messages in turn from a single message queue.

NOTE: Ensure that the selected test does not contain a Run Test action that calls the current test sequence (that is, a mutual recursion).

The **Value** and **Store** tabs specify whether and how mappings can be made between input and output tags in the current test sequence and those in the test sequence(s) of the resource to be executed. See [Test Properties](#) for more information about input and output tags.

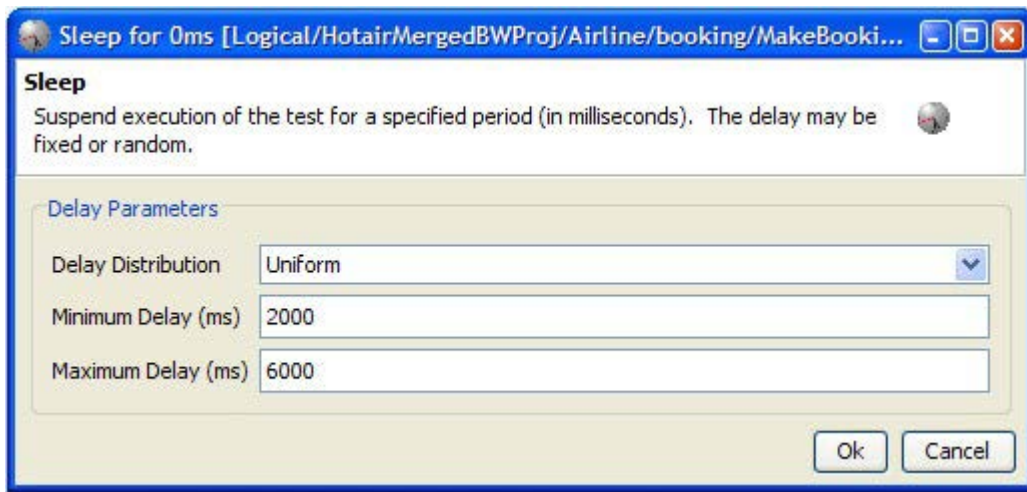


For instance, you might want to pass in the values for the name and number tags so that they can be used in the child resource. To do that, we must move the tags from the **Unmapped** list to the **Mapped** list, as shown in the image above.

NOTE: If the resource is executed in series, the values of the tags from the child resource can be extracted as well.

19.3.10 Sleep

A sleep action stops a test's execution for a specified period of time, in milliseconds. You can specify a fixed period of time, a uniformly distributed random period of time, or a random period of time with Gaussian distribution. The time range for the Uniform and Gaussian delay is defined by the values used in the Minimum Delay and Maximum Delay fields).



A sleep action may be used to make a test wait to ensure some previous processing is complete, for simulating delays in adapter stubs, and so on

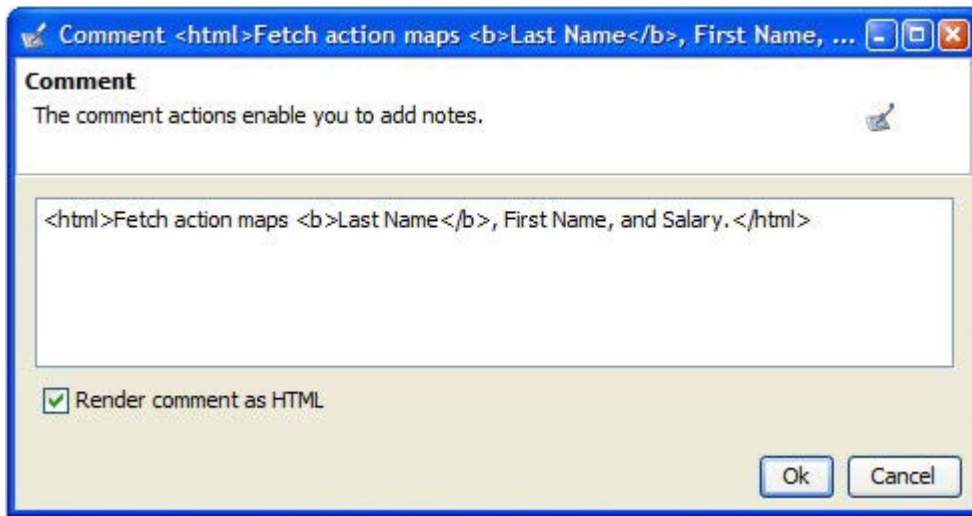
19.4 General Actions

The following general actions are available for Rational Integration Tester tests:

- [Comment](#)
- [Compare Files](#)
- [Function](#)
- [GUI Interaction](#)
- [Log](#)
- [Map](#)
- [Run Command](#)
- [SQL Command](#)
- [SQL Query](#)
- [Stored Procedure](#)
- [User Interaction](#)

19.4.1 Comment

The Comment action lets you add comments or notes within the actions of a test.



Enter the content of the comment in the text field. By default, the content is rendered as plain text. If you want to utilize HTML code and tags in your comment, enable the **Render comment as HTML** option.

NOTE: If you paste one or more lines of text when a test is opened for editing, each line will be added as a new Comment action within the selected test phase.

NOTE: The contents of the Comment action are logged in reports for the containing test.

19.4.2 Compare Files

The Compare Files action lets you compare individual records within delimited files. Two distinct records can be compared for equality, or multiple records can be summed and compared to the value of a single summary record. Before the action can be utilized in a test, however, certain resources must be configured in Rational Integration Tester as follows:

- Define [Record Layouts](#) (in the Schema Library) for each type of record to be compared. You must specify the appropriate field type for numeric fields, but all other fields can be strings. All columns in the layout must be defined and have unique names, and you must specify the delimiter in use under the **Options** tab.

NOTE: Every column in the record must be defined in the record layout. If there are more columns in a record than expected, the action will fail. If a row contains fewer columns than expected, however, the values for the “missing” columns defaults to null.

- Create [File Schemas](#) for each type of file and apply the correct record layout to it. In general, you will create one file schema per record layout. Under the **Settings** tab, the schema type must be set to **Delimited**. Additionally, you can tell Rational Integration Tester how many rows to skip at the start of the file (for example, for header rows).
- Create File Contents resources in Architecture School’s Logical View (for more information, refer to *IBM Rational Integration Tester Reference Guide for Files*). With each resource you can optionally define a specific file or a specific file directory if desired – the file or directory can be overridden in the Compare Files action. You must also apply the appropriate schema (created earlier) in the **File Schema** field.

NOTE: If you are using a directory of files, a pattern must be specified (for example, *.dat, or *.* for all files). The pattern is a directory listing wildcard, not a regular expression.

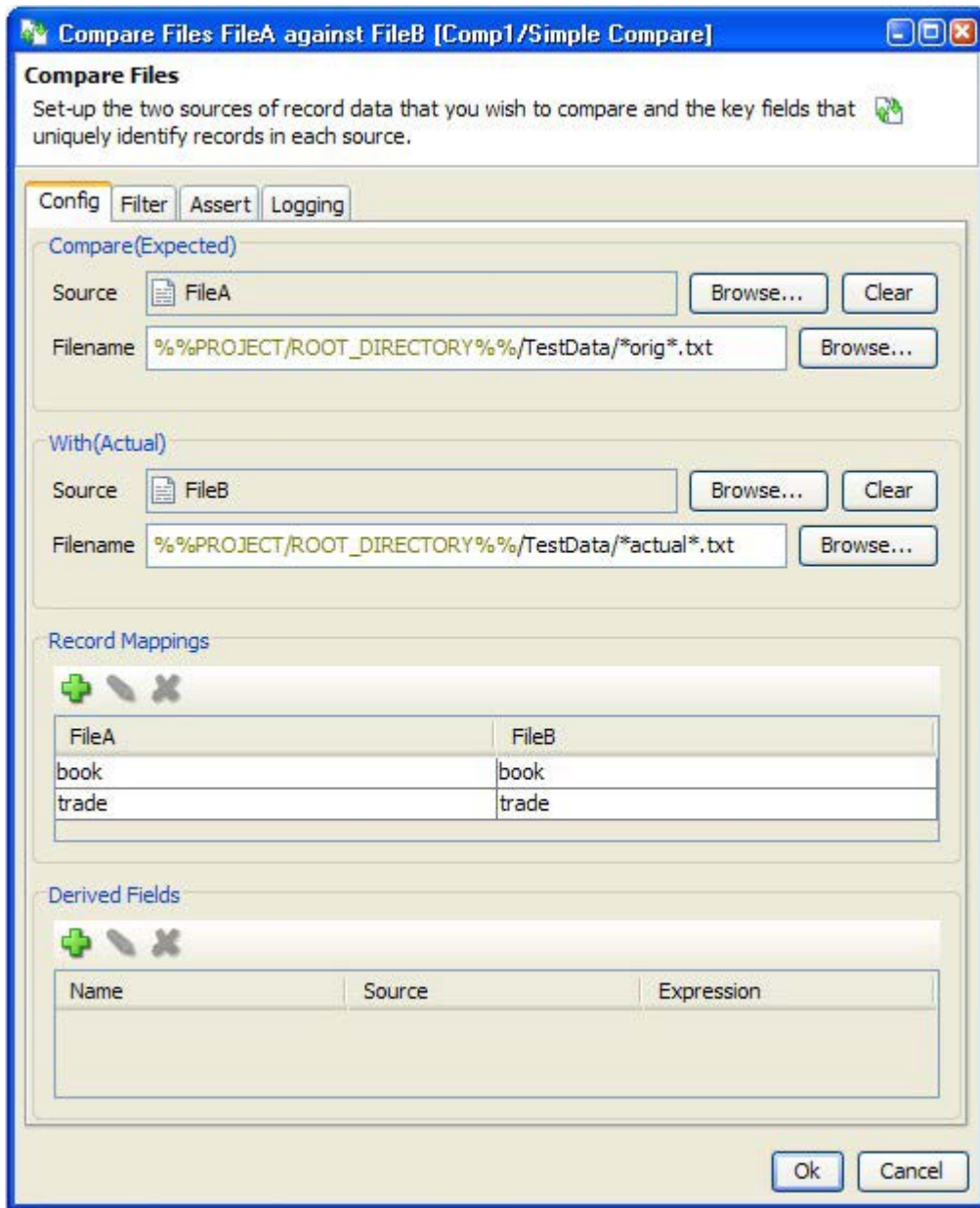
Once the required artefacts and resources have been configured in Rational Integration Tester, you can add the Compare Files action to a test and begin to configure it.

A record-based file comparison is made between two files or groups of files. Each source in the comparison (expected and actual) can consist of one or more files that will be concatenated together for the purposes of the comparison.

The basic configuration consists of telling Rational Integration Tester which columns within each source are used to identify a record (the record mappings) and how these

are matched up between the two sides of the comparison. Next, configure which columns in matching records are to be compared.

The Compare Files action is configured using four tabs: **Config**, **Filter**, **Assert**, and **Logging**.



The **Config** tab is used to [Select File Resources and Record Mappings](#), the **Filter** tab is used to [Configure File Resource Filters](#), the **Assert** tab is used to [Configure Comparison Options](#), and the **Logging** tab is used to [Configure Logging Options](#).

For each resource, record mappings are specified to identify which rows to compare. For the sake of performance, the fewest number of mappings possible should be used to uniquely identify a row, and the contents of a record mapping column must not be null. Once a matching row or set of matching rows has been found, the comparison occurs as defined under the **Assert** tab.

NOTE: Each source in the comparison can be one file or a directory of files. A collection of files can be expressed using normal filepath notation (for example, c:\files*.dat, or c:\files*.*)). All matching files will be concatenated before the comparison starts.

Columns of any type (defined in the record layout) can be compared for equality, and numeric columns can be compared against a tolerance. When using a tolerance, columns that do not differ by more than the specified amount (+ or -) will be considered to match. For example, a tolerance of '0.1' means that 1.5 will be “equal” to 1.4 and 1.6.

NOTE: Only columns that are specified as numeric types (in the record layout) can use a tolerance for the comparison.

If you are comparing summary data, you can specify a sum comparison that will add up values in the target resource (right side) and compare the total to a single row in the source resource (left side).

NOTE: Rows in the expected resource can not be summed and compared to a single row in the actual resource, and you can not add rows in both resources to compare.

NOTE: Only one type of comparison can be defined in the test action, regardless of the actual number of comparisons (that is, they will all be **Equality** or all **Sum of** comparisons).

Finally, you can configure the action to report any record mapping columns that are found in one resource and not the other.

NOTE: If any key columns are null, they will be reported as rows found in the expected resource but not in the actual resource.



The **gh.filecompare.maxlinelength** argument can be applied as a JVM argument in Library Manager to force the comparison to skip blank lines or lines that are too long, as these may cause Rational Integration Tester to run out of memory. The default value is 10,000. For example,
gh.filecompare.maxlinelength=6000.

Select File Resources and Record Mappings

Select the **Config** tab to define the file resources to be compared and the columns to match in each resource for determining which records to compare (record mappings).

The file resources to be compared (expected and actual) are configured in the top half of the **Config** tab, and the details of each resource are configured in the same way.

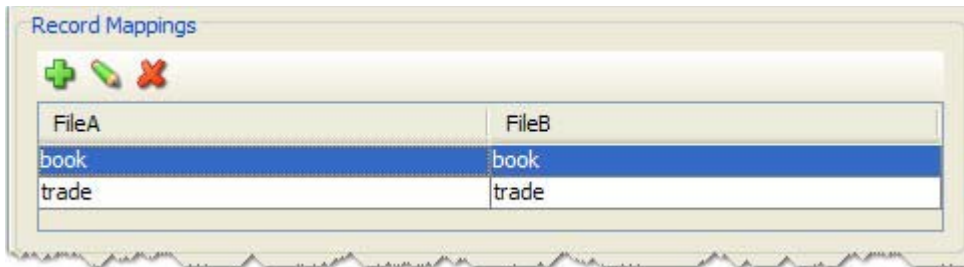
The screenshot shows a software window with four tabs: Config, Filter, Assert, and Logging. The Config tab is active. It is divided into two main sections: 'Compare(Expected)' and 'With(Actual)'. Each section contains a 'Source' field with a file icon, a 'Browse...' button, a 'Clear' button, and a 'Filename' field with a 'Browse...' button. In the 'Compare(Expected)' section, the Source is 'FileA' and the Filename is '%%PROJECT/ROOT_DIRECTORY%%/TestData/*orig*.txt'. In the 'With(Actual)' section, the Source is 'FileB' and the Filename is '%%PROJECT/ROOT_DIRECTORY%%/TestData/*actual*.txt'.


1. Click **Browse** next to the **Source** field to select a File Contents resource from the project.
2. In the **Filename** field, enter the path to a specific file or a file pattern (for example, `c:/files/*.dat`), or click **Browse** to locate and select a file (tags are supported). This field is optional if a file or directory pattern has been specified in the File Contents resource, but it can also be used to override any configured file or pattern at runtime.

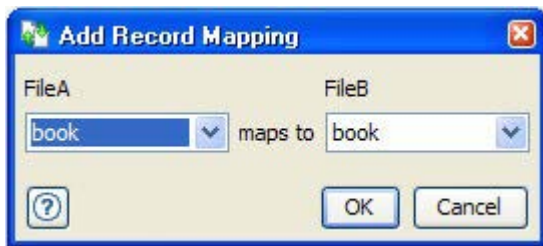
NOTE: Each source in the comparison can be one file or a directory of files. A collection of files can be expressed using normal filepath notation (for example, `c:\files*.dat`, or `c:\files*.*`). All matching files will be concatenated before the comparison starts.



Use the **Record Mappings** panel to define the columns to match in the record layout associated with each file resource (that is, the one applied to the file schema in use for the selected resource).

NOTE: Matching the content of record mapping fields in the source files is case-sensitive.

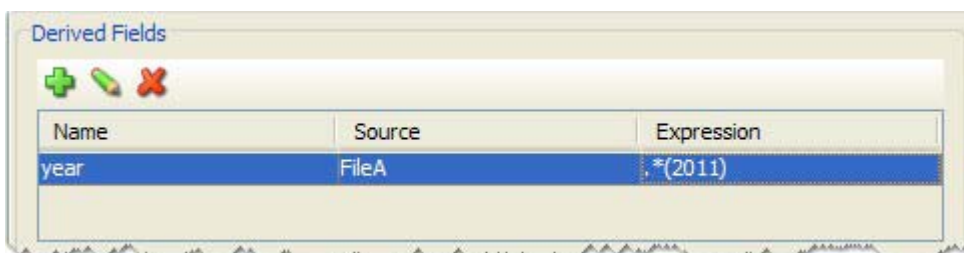



3. Click  to add a mapping to the table, and the **Add Record Mapping** dialog is displayed.

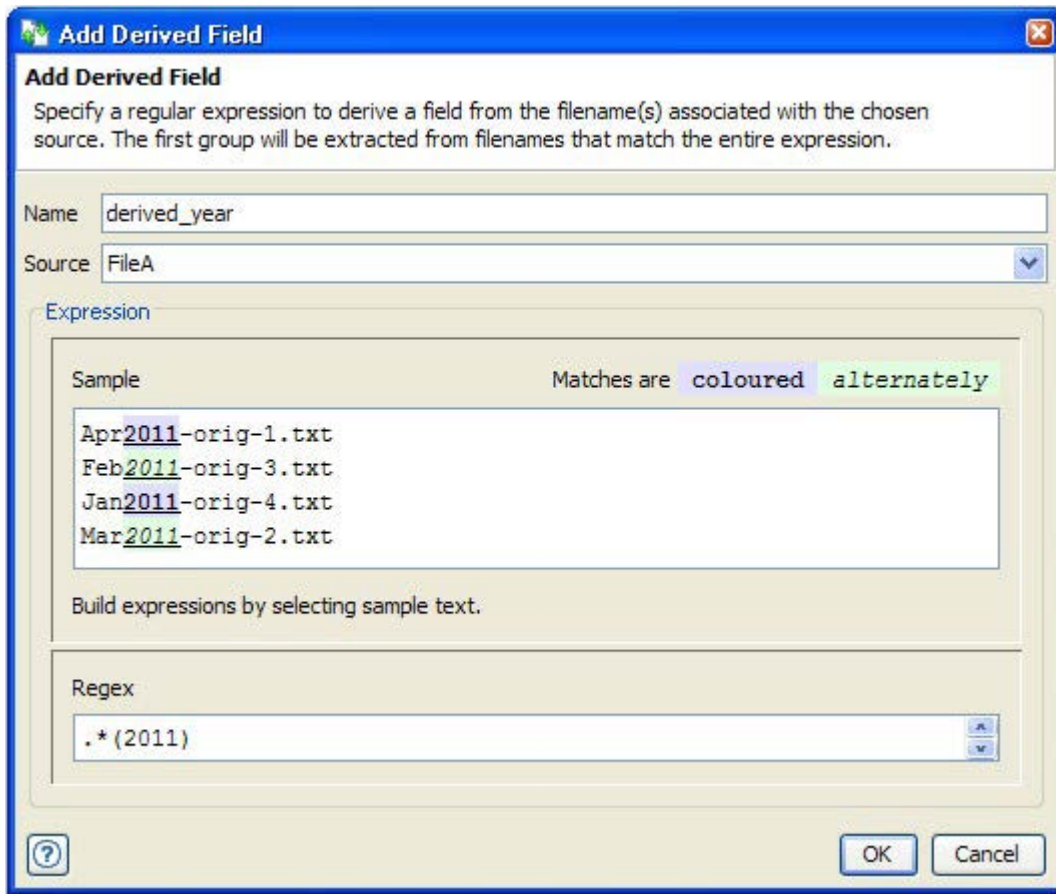


4. Select the column to be matched from each file resource, then click **OK** to continue.
5. If you want to edit an existing mapping, select it and click the  icon. If you want to delete an existing mapping, select it and click the  icon.



Use the **Derived Fields** panel to derive one or more record mapping fields from the filename associated with the selected source (for example, if a number in one of the resource filenames must be matched to a column in the other file).



-
- Click  to add a field to the table, and the **Add Derived Field** dialog is displayed.



- Provide a name for the field, which can be selected as a record mapping field, in the **Name** field, and select the file resource from which the field name will be derived from the **Source** combo box.

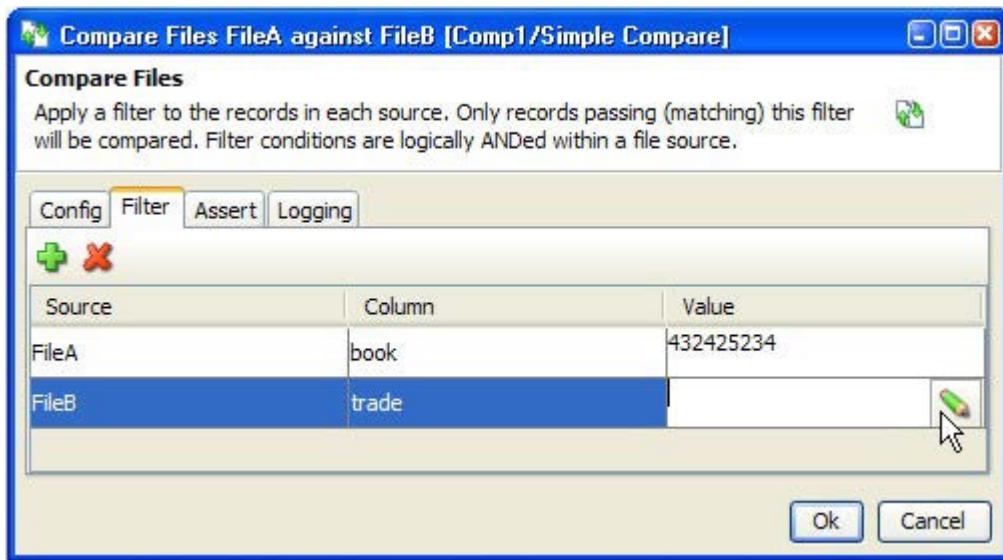
In the **Sample** field, the filenames associated with the selected resource are displayed.
- Select text in the **Sample** field to display a context menu that can help build a regular expression, or simply enter the expression in the **Regex** field. The expressions use groupings, so the content to be extracted is what is found in the first grouping. If a valid expression is entered, matching text in the **Sample** field is highlighted.
- When you are satisfied with the expression and its results, click **OK** to continue.
- If you want to edit an existing derived field, select it and click the  icon. If you want to delete an existing derived field, select it and click the  icon.

Once a derived field entry has been created, it can be selected as a mapping field in the **Record Mappings** panel.

Configure File Resource Filters

Select the **Filter** tab to create filters that can limit the potential number of records that may match and be compared within a file resource.

NOTE: Filter matching is case-sensitive.



1. Click to add a filter to the table or click to remove an existing filter.
2. For any existing filter, select the desired file resource in the **Source** field and the desired column from that source in the **Column** field.
3. To define the filter value and type, click the **Value** field next to the selected source, then click the icon.

The **Filter Editor** is displayed.



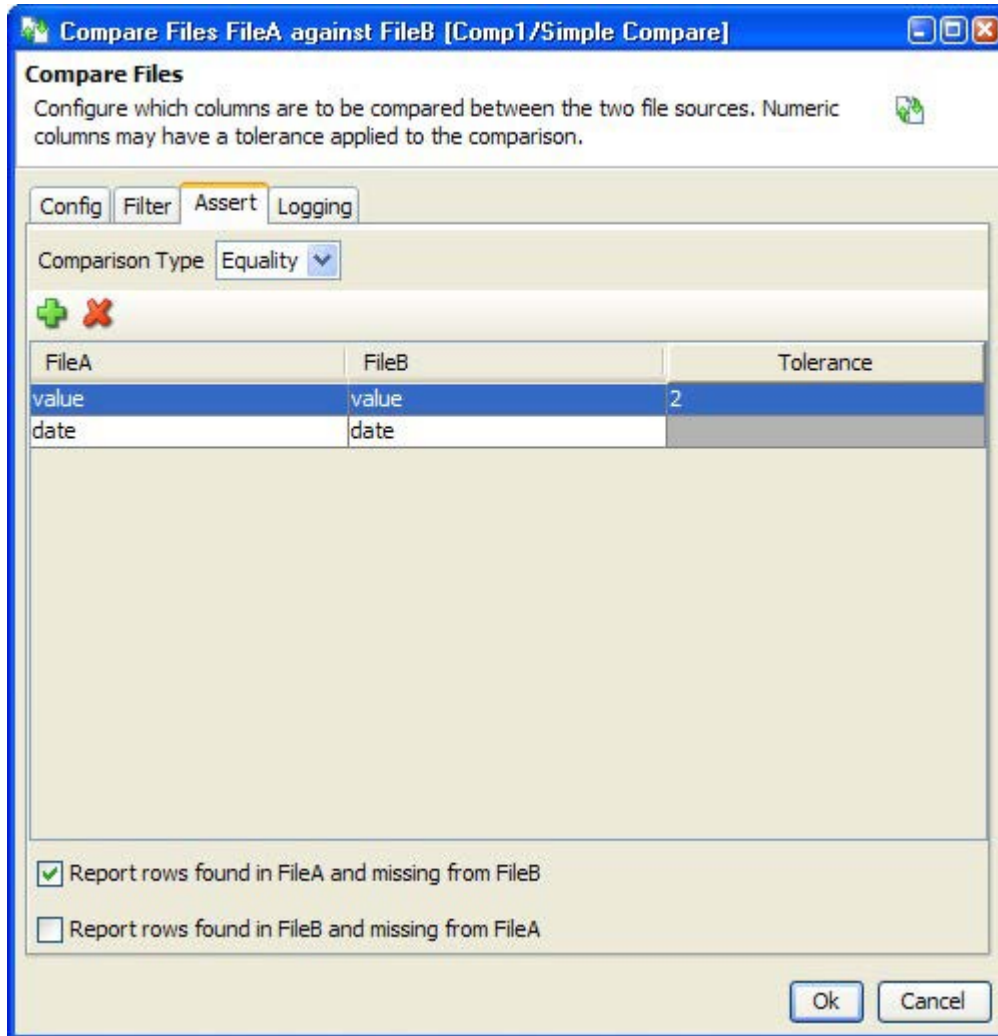
4. Leave the **Enabled** check box checked to enable the filter, or uncheck it to disable the filter. Disabling a filter lets you leave it configured in the table but not apply it at runtime.
5. Select the filter type (**Equality** or **Inequality**) from the **Action Type** combo box. If the type is **Equality**, all rows containing the matched filter value can be included in the comparison. If the type is **Inequality**, only rows that do not match the filter value can be included in the comparison.
6. In the field below the **Action Type**, enter the desired filter value, then click **OK** to return to the **Filter** tab.



NOTE: When filters are in use, they are applied first before any other matching is done.

Configure Comparison Options

Once the file resources, record mappings, and (optional) filters have been defined, use the **Assert** tab to configure the comparisons that should be carried out within matching records.

NOTE: The comparison of fields between configured sources is case-sensitive.



1. Select the type of comparison to apply at runtime (**Equality** or **Sum of**) from the **Comparison Type** combo box.
2. Click  to add a comparison or click  to remove an existing comparison.

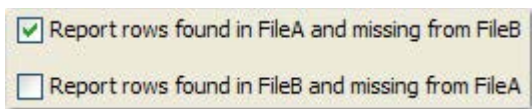
-
3. In a new or existing comparison row, select the columns to compare in matching records under the **<Expected Source>** and **<Actual Source>** columns.

NOTE: Columns used as record mappings are not available for comparison.

4. If the selected columns are numeric types (as defined in the associated record layout), you can enter an optional value under **Tolerance**.

NOTE: If a tolerance value is used, the values or sum comparison must be within + or - the tolerance value to be considered a match. If no tolerance is used, the values or sum comparison must be equal to be considered a match.

Below the comparison table, you can configure the reporting of rows that are found in one resource and not the other by enabling the desired options.



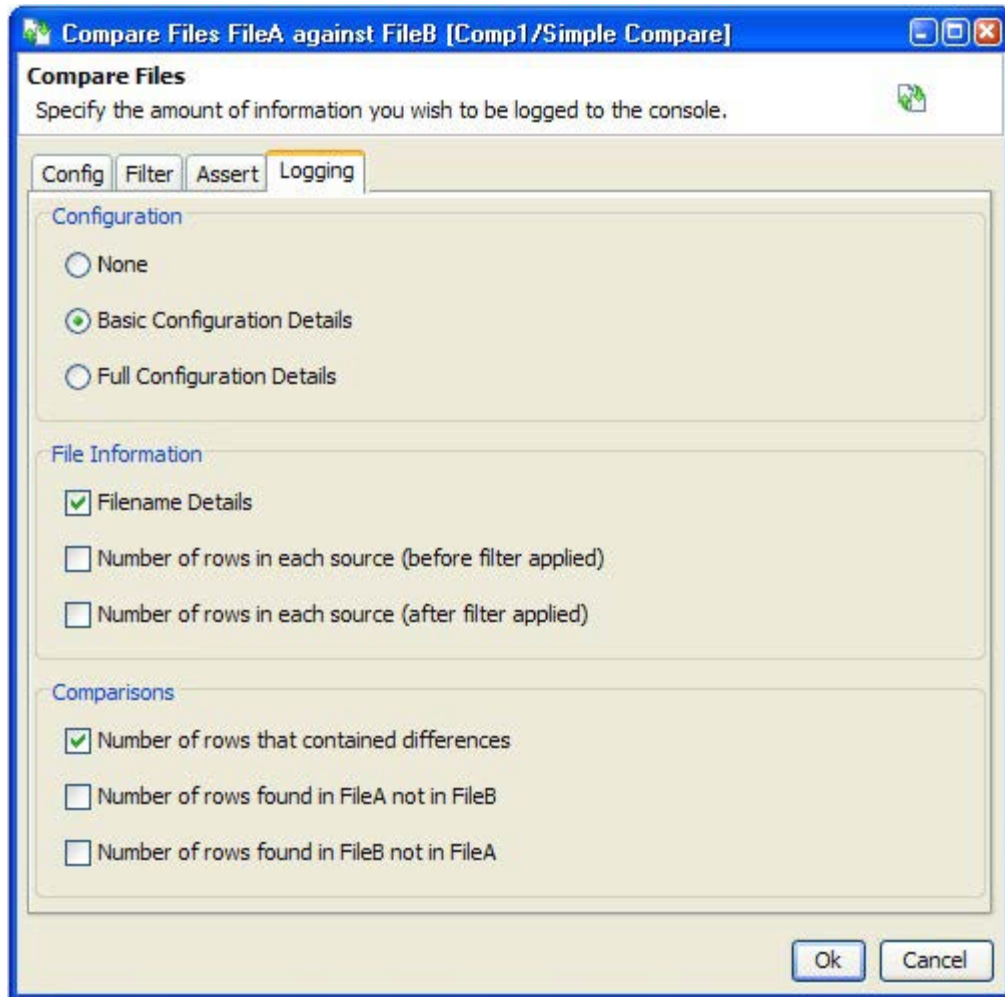
☒ Report rows found in FileA and missing from FileB

☐ Report rows found in FileB and missing from FileA

Missing rows will be reported in the test console when the action is executed.

Configure Logging Options

Select the **Logging** tab to define the type of logging information and level of logging detail that should be displayed in the console when the action is executed.



The options in the **Configuration** panel determine whether or not the record mapping columns (**Basic**) and comparison columns (**Full**) are logged. If **None** is selected, neither the record mapping nor the comparison column names will be logged.

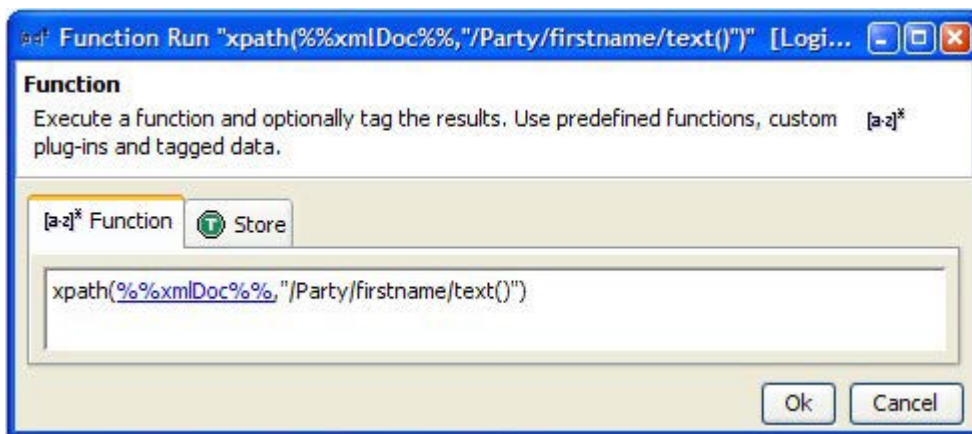
Under **File Information**, you can enable or disable the logging of the filenames used by the action and the logging of the number of rows counted in each source before filters are applied.

Under **Comparisons**, you can enable or disable the logging of the number of rows containing conflicts and the logging of the number of rows found in one source but not the other.

19.4.3 Function

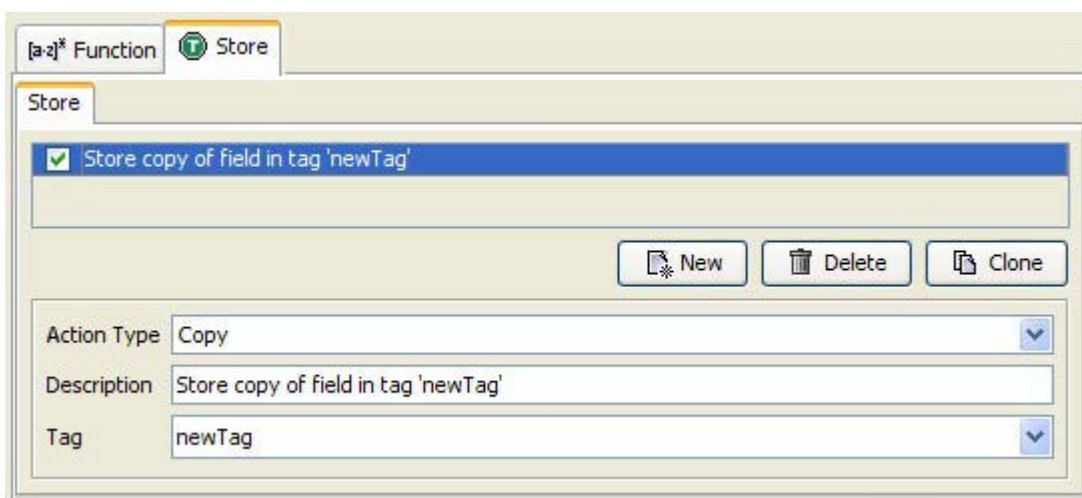
A Function action executes a single or composite set of functions from the registered set. Functions and expressions are loaded from a folder contained within the project Classes folder and will be available to all users of the project. A full list of the available functions, along with details on how they can be utilized, is provided in [Functions](#).

Expressions used as parameters will usually be strings, numerical values, or tags. Tags can be entered manually or added from [The Tag Data Store](#).



The example shown above executes the *xpath* function, which retrieves the first name of the party from the *xmlDoc* tag.

By clicking the **Store** tab, you can store the value returned by the function into a tag:



19.4.4 GUI Interaction

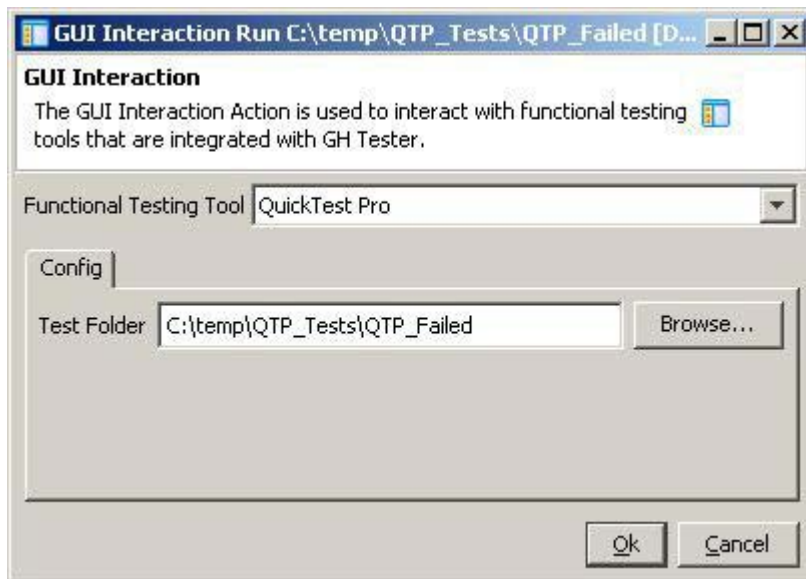
Rational Integration Tester provides market leading GUI testing capabilities through the use of separately licensed test modules (TestDrive and QuickTest Professional). With the GUI Interaction test action, users can specify an existing GUI test script to be executed from Rational Integration Tester.

NOTE: Before you can edit the GUI Interaction test action, at least one of the GUI test clients (TestDrive or QuickTest Professional) must be installed and configured on the local machine (that is, the same machine as Rational Integration Tester), and the path to the client installation must be specified in the Library Manager (refer to *IBM Rational Integration Tester Installation Guide*).

Running QuickTest Professional Scripts

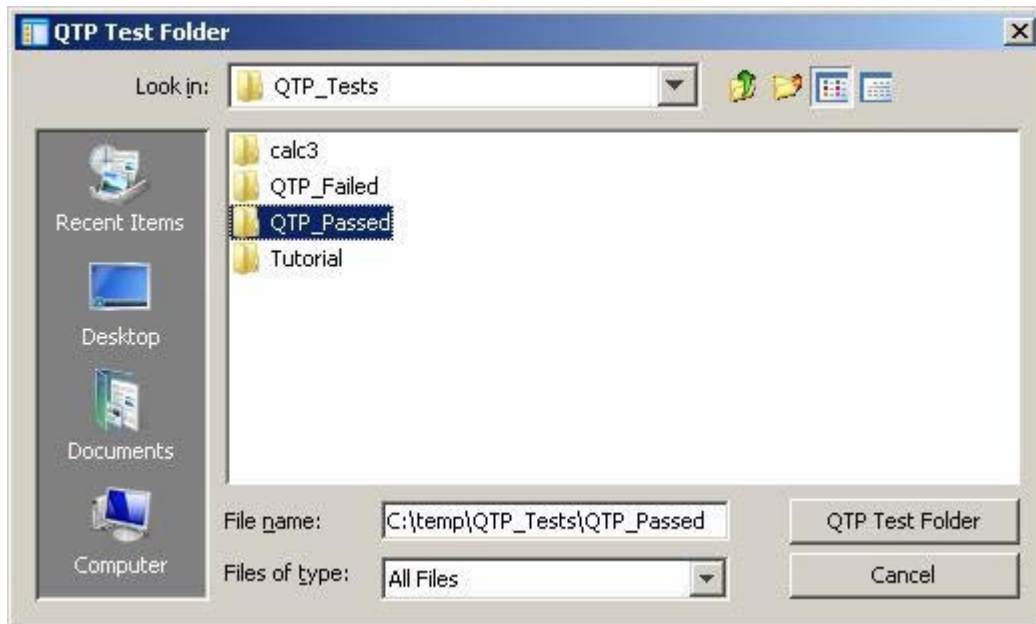
Follow the steps below to configure the GUI Interaction for a QuickTest Professional script.

1. Open the test action and select **QuickTest Pro** from the **Functional Testing Tool** menu.



NOTE: If QuickTest Professional is the only GUI testing tool installed, the drop-down menu next to **Functional Testing Tool** will not be available.

-
2. Next to the **Test Folder** field, click **Browse** to locate and select an existing QuickTest Professional test folder.



3. Select the desired folder and click **QTP Test Folder** to close the dialog.
4. When finished editing the test action, click **OK**.

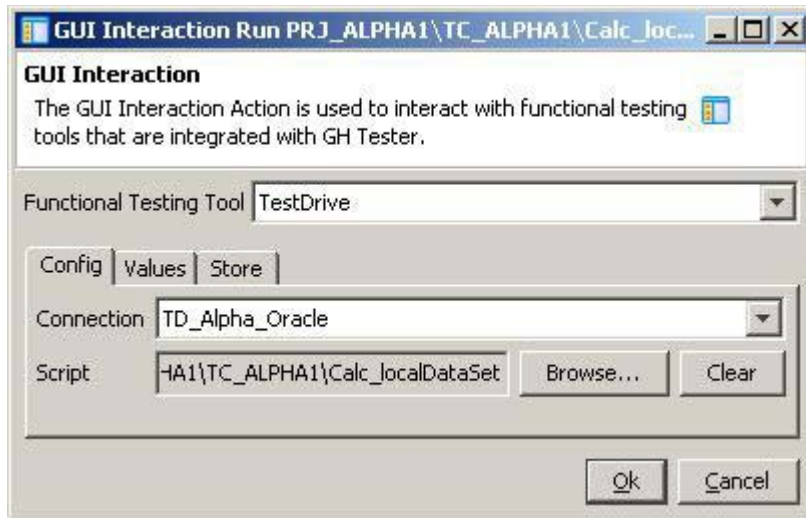
When the GUI Interaction test action is executed from the test, Rational Integration Tester will launch QuickTest Professional and execute the selected script. The results of the script will be displayed in the console, the same as with any other Rational Integration Tester test.

When viewing detailed results by clicking the results link in the console, the QuickTest Professional results viewer will be displayed.

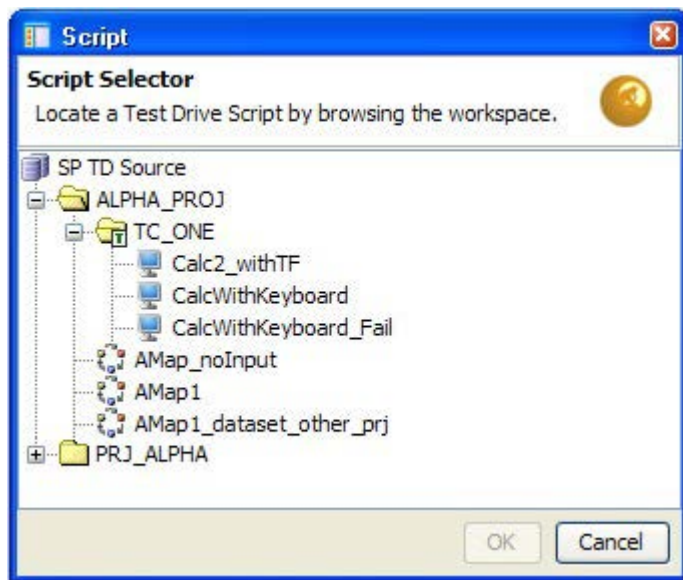
Running TestDrive Scripts or Action Maps

Follow the steps below to configure the GUI Interaction for a TestDrive script or Action Map.

1. Open the test action and select **TestDrive** from the **Functional Testing Tool** menu.



2. Under the **Config** tab, specify an existing connection to the TestDrive server under the **Connection** field – this is an ODBC Connection that must be configured in TestDrive before it can be selected in the test action.
3. Next to the **Script** field, click **Browse** to locate and select an existing script or action map from the TestDrive database.

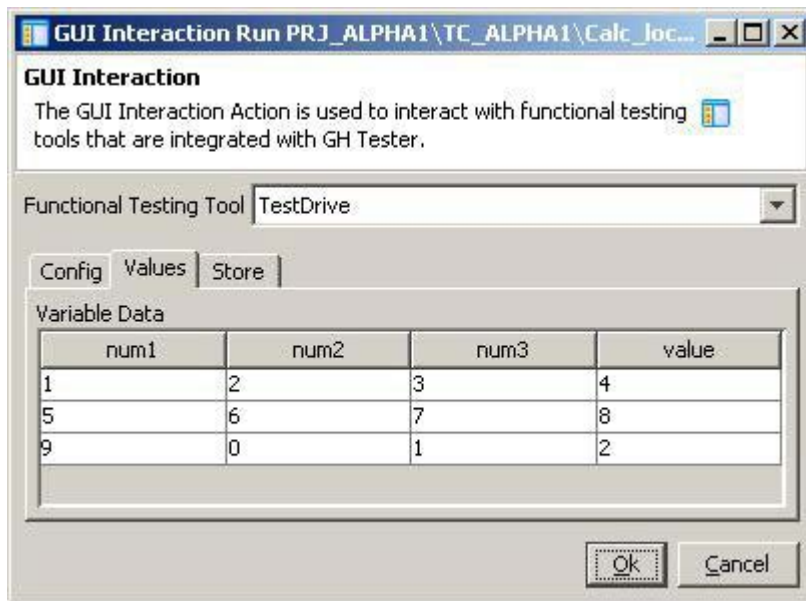


The items available will depend upon the ODBC Connection that was specified.

4. Select the desired script or action map and click **OK** to close the dialog.

NOTE: When working with action maps, tracked fields are not supported, but input data sets are.

If the selected script or action map includes variables, they will be displayed in a grid under the **Values** tab.

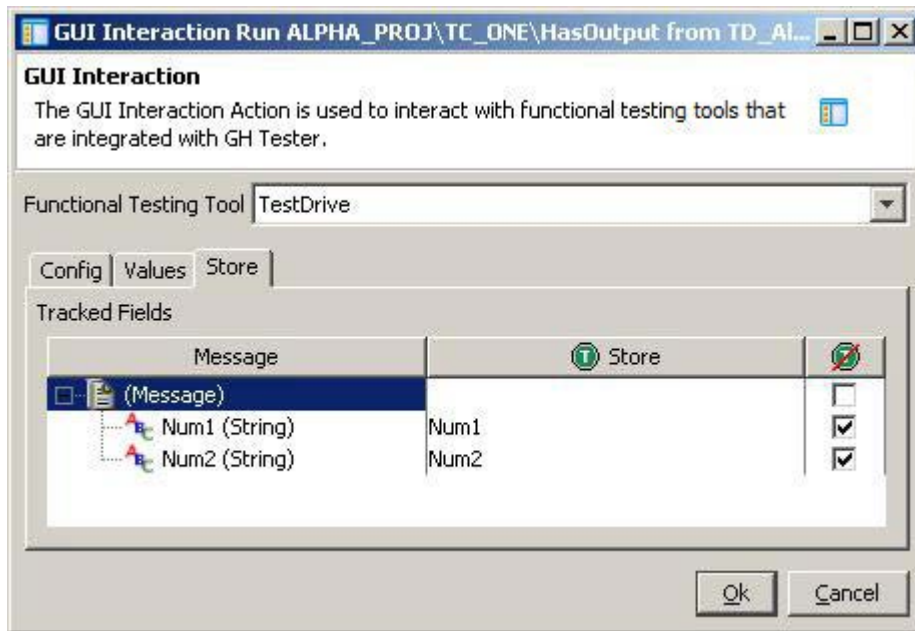


These variables can be edited within the test action, and tags can be inserted from the context menu for any of the fields.

5. Double-click a field to edit it, then right-click in the active field to bring up the context menu.

NOTE: Any variables that are modified in the test action will be modified in the TestDrive database.

-
6. Under the **Store** tab you can specify if any of the variable data from the script should be stored into tags.

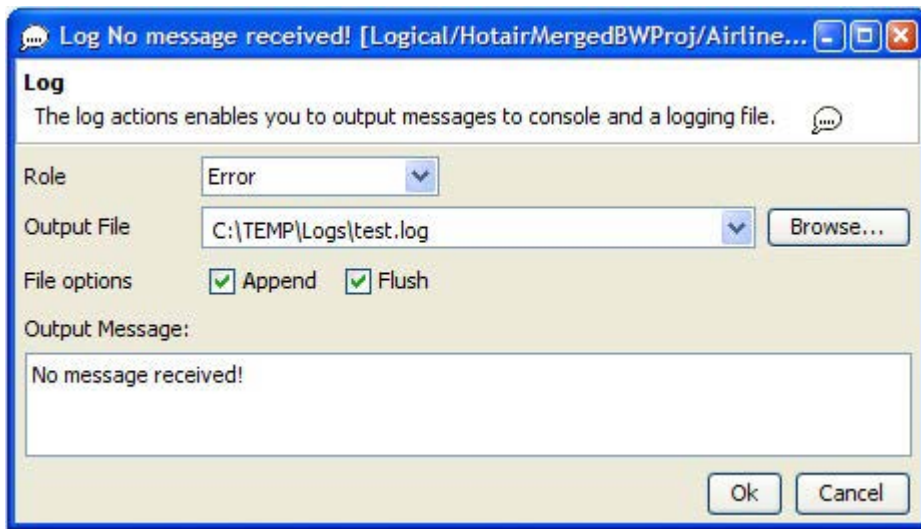


7. When finished editing the test action, click **OK**.

When the GUI Interaction test action is executed from the test, Rational Integration Tester will launch TestDrive and execute the selected script or action map. The results of the run will be displayed in the console, the same as with any other Rational Integration Tester test.

19.4.5 Log

The Log action lets you write the specified **Output Message** (the use of tags is supported) to the console and to an optional logging file. The file can be stored locally or remotely (by means of a UNC path).



In the Log action editor you can denote the log messages as either “Info,” “Warning,” or “Error.” You can set the location of the logging output file by typing the file’s full path (which can include tags), clicking **Browse** to locate and select the file, or by selecting one of the most recently used files.

NOTE: Logging an “Error” message will give the overall test a fail status, but subsequent test steps will be executed as the logic of the test denotes.

NOTE: If you want the output from different test sessions to appear on its own line, you must include a carriage return at the end of the output message.

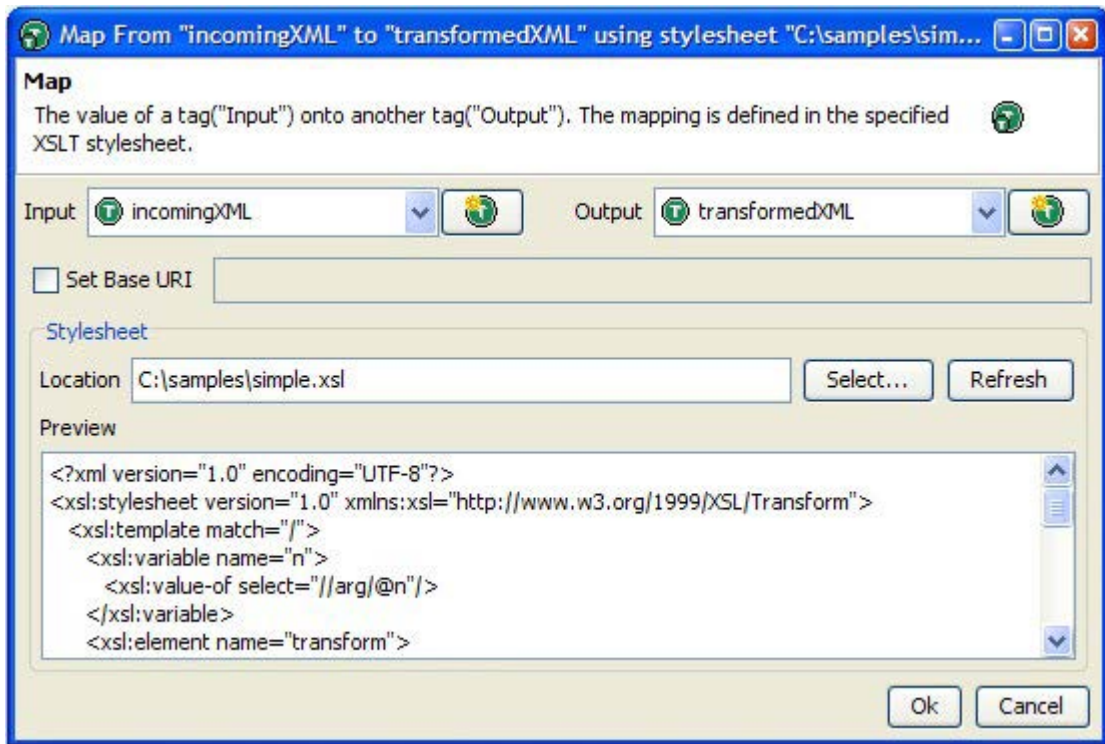
If an output file is designated, the **Append** and **Flush** options can be used.

- If **Append** is enabled, the test action will add new logging to the existing file contents. Otherwise, the existing contents will be overwritten.
- If **Flush** is enabled, the test action will write the message to the file immediately. Otherwise, messages are buffered and written in batches (which may improve performance).

Tip: The Log action can be used as a file-writer process.

19.4.6 Map

The Map action lets you transform XML-based data using XSL-based stylesheets (for example, to transform a document prior to publishing).



The table below describes the options available for configuring this action:

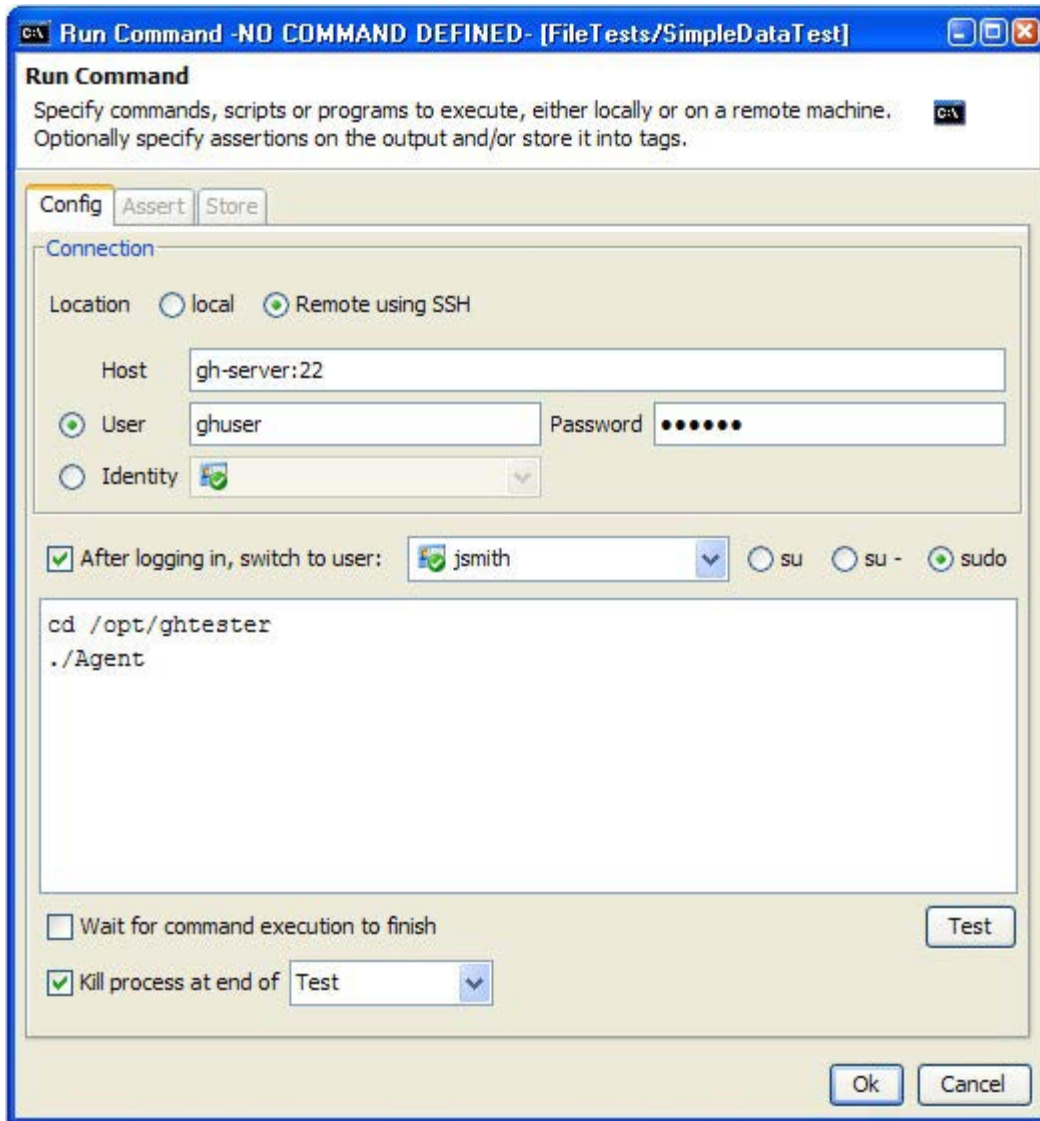
Option	Description
Input	Select or create the tag containing the source XML (to be transformed).
Output	Select or create the tag to which the output should be copied. If the destination tag is the same as the input tag, the value of the input tag will be overwritten.
Set Base URI	Enable this option to set the base directory for any XSL documents that are referenced in the main transform.
Stylesheet location	The full path to the XSLT that will perform the transformation. Click Select to enter a path or browse to one, or to enter a URL. Click Refresh to refresh the contents of the preview window (if the XSLT has changed outside of the action dialog).

If you add a failure path to the action (see [Setting Failure Paths](#)), four paths are added:

- **Invalid Input:** The data is not in XML or message format.
- **Invalid Stylesheet:** An error in compiling the contents of the stylesheet.
- **Processing Error:** An error during processing (for example, an invalid or erroneous function).
- **Conversion Error:** When the input data is message-based, this error occurs if the XML contents can not be converted back into message format.

19.4.7 Run Command

The Run Command action executes one or more commands or programs that are normally executed from the command line or shell. This action can validate or store the command's output, or stop and start system resources that might be used within a test or scenario.



The image shows a 'Run Command' dialog box with a title bar that reads 'C:\ Run Command -NO COMMAND DEFINED- [FileTests/SimpleDataTest]'. The dialog has three tabs: 'Config' (selected), 'Assert', and 'Store'. Below the tabs, there is a description: 'Specify commands, scripts or programs to execute, either locally or on a remote machine. Optionally specify assertions on the output and/or store it into tags.' The 'Connection' section has two radio buttons: 'local' and 'Remote using SSH' (selected). The 'Host' field contains 'gh-server:22'. There are radio buttons for 'User' (selected) and 'Identity'. The 'User' field contains 'ghuser' and the 'Password' field is masked with dots. Below this, there is a checkbox 'After logging in, switch to user:' which is checked, followed by a dropdown menu showing 'jsmith' and three radio buttons: 'su', 'su -', and 'sudo' (selected). A text area contains the commands 'cd /opt/ghtester' and './Agent'. At the bottom, there is a checkbox 'Wait for command execution to finish' (unchecked) and a checkbox 'Kill process at end of' (checked) with a dropdown menu showing 'Test'. There are 'Test', 'Ok', and 'Cancel' buttons.

Run Command
Specify commands, scripts or programs to execute, either locally or on a remote machine. Optionally specify assertions on the output and/or store it into tags.

Config | Assert | Store

Connection

Location: ☐ local ☒ Remote using SSH

Host: gh-server:22

☒ User: ghuser Password:

☐ Identity: [icon]

☒ After logging in, switch to user: [jsmith] ☐ su ☐ su - ☒ sudo

cd /opt/ghtester
./Agent

☐ Wait for command execution to finish

☒ Kill process at end of: Test

Test Ok Cancel

Enter one or more scripts, programs, or executables to run under the **Config** tab – multiple commands can be entered on separate lines. The **Test** button can be used to see the results of the command at runtime (see [Test a Command](#)). The options for running the command are specified in the table below:

Option	Description
Location	Select how to run the command, locally or remotely. If local , the command will be executed on the machine where Rational Integration Tester is installed. If Remote using SSH , the command will be executed on the remote host specified.
Working Directory	For local commands, specifies the local directory from which the command should be executed.
Host	The host on which the remote command should be executed, and the port on which the host is listening for SSH connections. The host and port should be entered in the form <code>host:port</code> .
User	For remote commands, specifies that a user/password combination will be used to connect, and specifies the user ID to send for connecting to the remote host.
Password	For remote commands, if User is selected, the password associated with the specified user ID.
Identity	Enable this option to specify a Rational Integration Tester Identity instead of a user/password combination.
After logging in, switch to user	Enable this option to switch to a different user, selecting the Rational Integration Tester identity for the user to switch to, and specify the command to use (su , su - , or sudo) for switching.
(command)	Enter the command or commands in the field below the Connection parameters.
Wait for command execution to finish	Enable this option (default) to have Rational Integration Tester wait for the command to execute and finish. If disabled, the command will run in the background and Rational Integration Tester will continue to the next test step. If this option is not enabled, the Assert and Store tabs are unavailable.
Kill process at end of ...	If the command is running in the background, you can enable this option to kill the command once the containing test resource finishes (Test , Scenario , Suite , or Execution). If you select Suite or Scenario and the test is not run inside a suite or a scenario, then the process ends at the end of the test. This option is unavailable if Wait for command execution to finish is enabled.

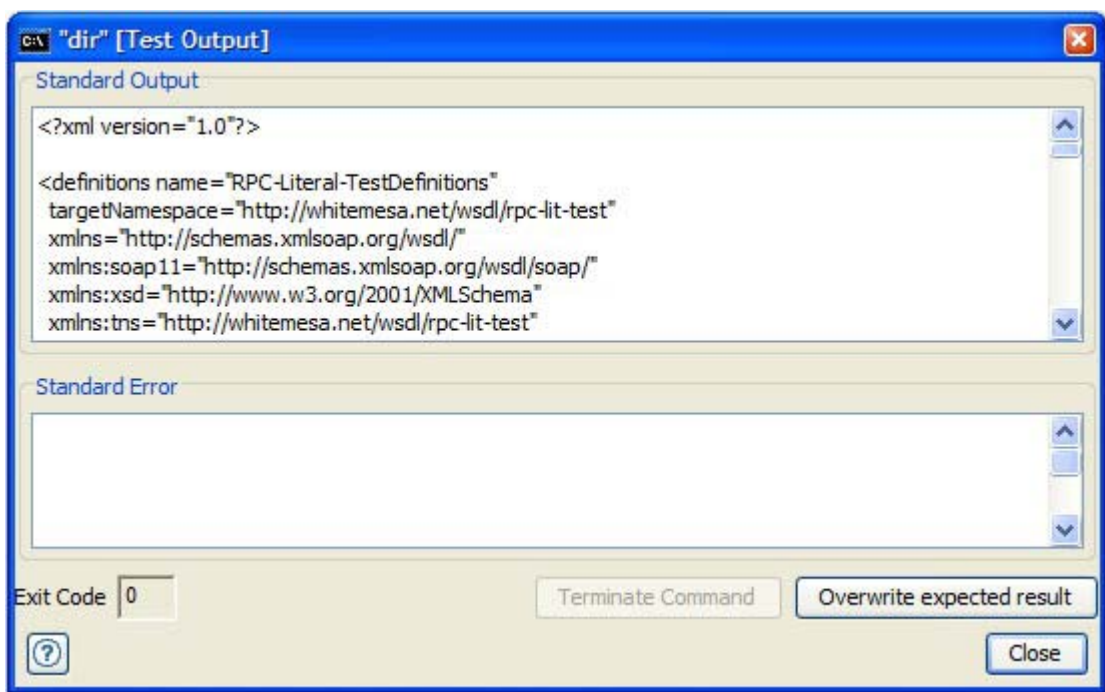
NOTE: When running commands remotely by means of SSH, you may experience different behaviour than if you were logged into the actual system (for example, “command not found” error). Remote SSH connections run a non-interactive shell, which may result in different settings in some of the system environment variables. Consult your system or application administrators for assistance, but you should be able to run the relevant environment scripts before the command, as follows:

```
. ~/.profile ; <my_command>
```

Test a Command

You can test the command(s) under the **Config** tab at any time by clicking the **Test** button. In the Test Output window, the standard output and error output of the command are displayed, as well as the program exit code.

NOTE: If any of the commands throw an exception, it will be displayed in a separate popup window.



If the command does not terminate by itself, click **Terminate Command** to kill the process or program. To copy the contents of each field to the **Assert** tab, click **Overwrite expected result** (then click **Close** to view or modify the contents of the **Assert** tab).

Validate Command Output

Any of the three types of output generated by the command – standard output, standard error, or the program exit code – can be validated when the command is executed as part of a test or suite. To configure validation, select the **Assert** tab.

The screenshot shows a Windows-style dialog box titled "Run Command 'dir' [Svc1/Op1/Commands]". Inside, there's a section "Run Command" with the instruction "Specify a system command (script or program) to execute, optionally assert and store in tags". Below this are three tabs: "Config", "Assert" (which is selected), and "Store". Under the "Assert" tab, there are three checkboxes: "Standard Output" (checked), "Standard Error" (unchecked), and "Exit Code" (unchecked). The "Standard Output" checkbox is followed by a text area containing XML schema definitions:

```
<?xml version="1.0"?>

<definitions name="RPC-Literal-TestDefinitions"
  targetNamespace="http://whitemesa.net/wsdl/rpc-lit-test"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:soap11="http://schemas.xmlsoap.org/wsdl/soap/"
```

 Below the "Exit Code" checkbox is a text field containing the number "0". At the bottom right are "Ok" and "Cancel" buttons.

To enable validation for a field, simply tick the check box next to the field name. If the check box is left empty for a field, validation will not be performed for that field.

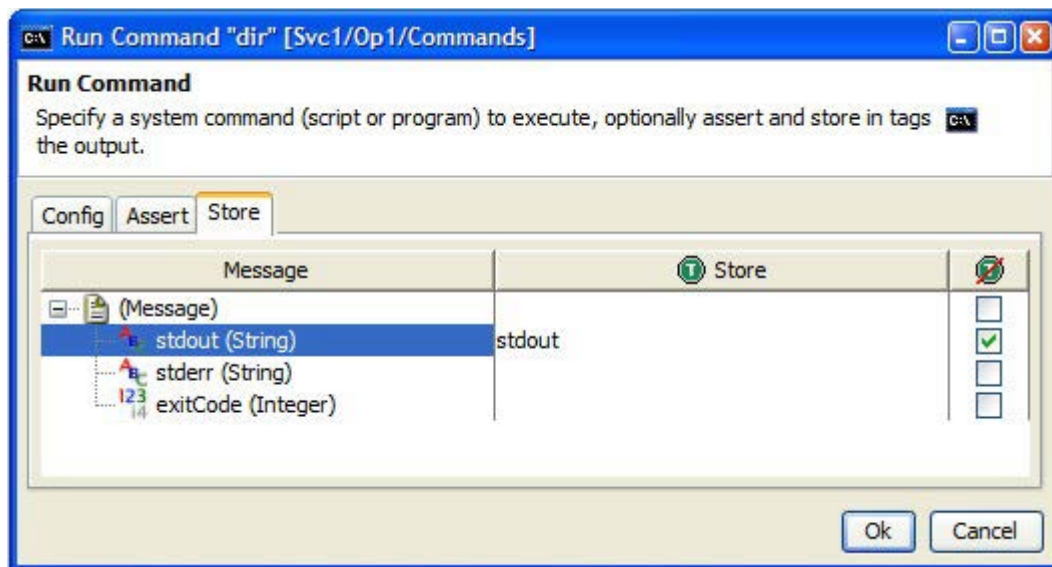
NOTE: By default, fields will be validated using the “Equality” action, but this can be modified in the field editor.


You can enter content manually into any of the fields, or you can right-click within a field and select **Edit Cell** to open the field editor (you may also double-click the field to open the field editor). Alternatively, you can populate the contents of each field by testing the command and clicking the **Overwrite expected result** button (see [Test a Command](#)).

See [The Field Editor](#) for more information about validation options and using the field editor.

Store Command Output in Tags

Command output is treated like message fields, meaning that any of the field content generated by the command can be stored in new or existing tags. To configure any of the tag storage options, select the **Store** tab.



To store any of the output fields in a tag, tick the corresponding check box in the Enable Store Action  column. The default name for the tag will equal the name of the message field (that is, stdout, stderr, or exitCode). To modify the default name, double-click it and enter a new value. See [Messages](#) for more information.

19.4.8 SQL Command

This action executes a SQL command on a database resource (that is, to insert, update, or delete rows from the database). See [Databases](#) for more information.

19.4.9 SQL Query

This action queries the selected database to verify that its contents match selected values. See [Databases](#) for more information.

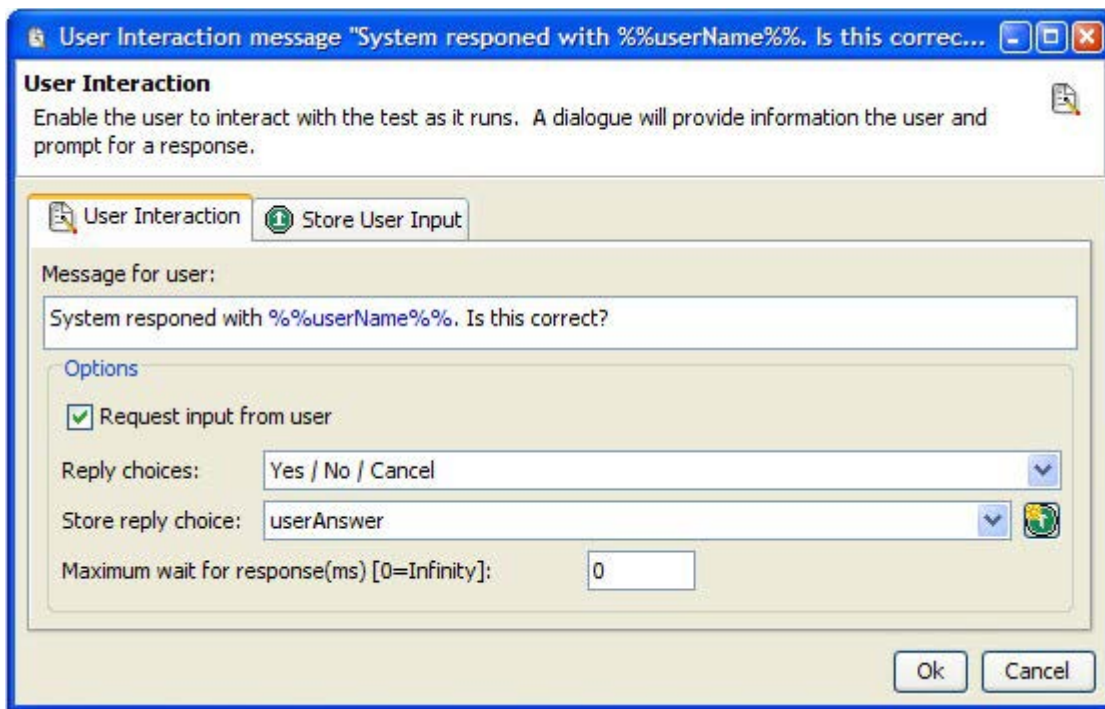
19.4.10 Stored Procedure

This action calls a stored procedure from the selected database. As you configure this action, you can view the code of the stored procedure, specify input values, tag return values, or specify expected return values. See [Databases](#) for more information.

19.4.11 User Interaction

The User Interaction action can present the user with information or request user input during the execution of a test (for example, to confirm an event outside of Rational Integration Tester). The user's reply or any data entered by the user can be stored in a tag.

NOTE: This action is not available when a test is launched from the command line, from ANT, or within a performance test.



Enter the message to present to the user, which can include tags, under **Message for user**. Additional options are specified in the table below:

Option	Description
Request input from user	Includes a free text input field in the message dialog. If enabled, the Store User Input tab is enabled, letting you store the user response in a tag.
Reply choices	Select which set of reply buttons to include in the message (OK/Cancel or Yes/No/Cancel). If the user clicks Cancel , the test run will be cancelled.
Store reply choice	If desired, you can store the code that corresponds to the user's reply in an existing tag (or click the Tag button to create a new tag).
Maximum wait for response	Enter the amount of time (in milliseconds) that Rational Integration Tester should wait for a user response. Leaving the default value of "0" will wait indefinitely.

19.5 Performance Actions

The following performance actions are available for Rational Integration Tester tests:

- [Begin Timed Section](#)
- [End Timed Section](#)
- [Log Measurement](#)

NOTE: For more information about performance actions and performance tests, refer to *IBM Rational Performance Test Server Reference Guide*.

19.5.1 Begin Timed Section

For use in performance testing, the Begin Timed Section action is a marker that indicates the beginning of a section of test steps. This action is used in conjunction with the End Timed Section step.

19.5.2 End Timed Section

For use in performance testing, the End Timed Section action is a marker that indicates the end of a section of test steps. This action is used in conjunction with the Begin Timed Section step.

19.5.3 Log Measurement

The Log Measurement action writes custom data to the project database during a performance test.

Appendix B: Date & Time Patterns

Contents

Overview

Formatting and Parsing Date/ Time Patterns

Examples

Some fields in Rational Integration Tester allow you to define patterns for date-time formatting. This chapter provides information about how to define such patterns and also provides some examples.

20.1 Overview

Date and time formats are specified by date and time pattern strings. Within date and time pattern strings, unquoted letters from 'A' to 'Z' and from 'a' to 'z' are interpreted as pattern letters representing the components of a date or time string. Text can be quoted using single quotes (') to avoid interpretation. "'" represents a single quote. All other characters are not interpreted; they're simply copied into the output string during formatting or matched against the input string during parsing. The following pattern letters are defined (all other characters from 'A' to 'Z' and from 'a' to 'z' are reserved):

Letter	Date or Time Component	Presentation	Examples
G	Era designator	Text	AD
y	Year	Year	1996; 96
M	Month in year	Month	July; Jul; 07
w	Week in year	Number	27
W	Week in month	Number	2
D	Day in year	Number	189
d	Day in month	Number	10
F	Day of week in month	Number	2
E	Day in week	Text	Tuesday; Tue
a	Am/pm marker	Text	PM
H	Hour in day (0-23)	Number	0
k	Hour in day (1-24)	Number	24
K	Hour in am/pm (0-11)	Number	0
h	Hour in am/pm (1-12)	Number	12
m	Minute in hour	Number	30
s	Second in minute	Number	55
S	Millisecond	Number	978
z	Time zone	General time zone	Pacific Standard Time; PST; GMT-08:00
Z	Time zone	RFC 822 time zone	-0800

20.2 Formatting and Parsing Date/Time Patterns

Pattern letters are usually repeated, as their number determines the exact presentation of the date/time information that they represent. This section provides additional information about how the various pattern types are formatted and parsed in Rational Integration Tester.

Text

For formatting, if the number of pattern letters is four or more, the full form is used. Otherwise a short or abbreviated form is used if available. For parsing, both forms are accepted, independent of the number of pattern letters.

Number

For formatting, the number of pattern letters is the minimum number of digits, and shorter numbers are zero-padded to this amount. For parsing, the number of pattern letters is ignored unless it's needed to separate two adjacent fields.

Year

For formatting, if two pattern letters are used, the year is truncated to two digits. Otherwise, the year is interpreted as a number.

For parsing, if the number of pattern letters is more than two, the year is interpreted literally, regardless of the number of digits. For example, the pattern "MM/dd/yyyy", "01/11/12" parses to Jan 11, 12 A.D.

For parsing with the abbreviated year pattern ("y" or "yy"), Rational Integration Tester interprets the abbreviated year relative to the century by adjusting dates to be within 80 years before and 20 years after the current date/time.

Take for example that the current date/time is January 1, 1997 and you use a pattern of "MM/dd/yy". The string "01/11/12" would be interpreted as Jan 11, 2012, while the string "05/04/64" would be interpreted as May 4, 1964.

During parsing, only strings consisting of exactly two digits will be parsed into the default century. Any other numeric string, such as a one digit string, a three or more digit string, or a two digit string that isn't all digits (for example, "-1"), is interpreted literally. So "01/02/3" or "01/02/003" are parsed, using the same pattern, as Jan 2, 3 AD. Likewise, "01/02/-3" is parsed as Jan 2, 4 BC.

Month

If the number of pattern letters is three or more, the month is interpreted as text. Otherwise, the month is interpreted as a number.

General Time Zone

Time zones are interpreted as text if they have names. For time zones representing a GMT offset value, the following syntax is used: **GMT +/- Hours:Minutes**, which represents GMT plus or minus some offset time in hours and minutes.

Hours must be between 0 and 23, and they can be expressed as a single digit or as two digits (for example, 3:00 or 11:00). Minutes must be between 00 and 59, and they will always be expressed as two digits.

For parsing, RFC 822 time zones are also accepted.

RFC 822 Time Zone

For formatting, the RFC 822 4-digit time zone format is used: +/-HHmm, which represents the current offset from GMT in hours and minutes.

Hours must be between 00 and 23, always expressed as two digits, and minutes must be between 00 and 59.

For parsing, general time zones are also accepted.

NOTE: Rational Integration Tester also supports localized date and time pattern strings. In these strings, the pattern letters described above may be replaced with other, locale dependent, pattern letters. Rational Integration Tester does not deal with the localization of text other than the pattern letters.

Examples

The following examples show how date and time patterns are interpreted in the U.S. locale. The given date and time are 2001-07-04 12:08:56 local time in the U.S. Pacific Time time zone.

Date and Time Pattern	Result
"yyyy.MM.dd G 'at' HH:mm:ss z"	2001.07.04 AD at 12:08:56 PDT
"EEE, MMM d, 'yy"	Wed, Jul 4, '01
"h:mm a"	12:08 PM
"hh 'o'clock' a, zzzz"	12 o'clock PM, Pacific Daylight Time
"K:mm a, z"	0:08 PM, PDT
"yyyyy.MMMMM.dd GGG hh:mm aaa"	02001.July.04 AD 12:08 PM
"EEE, d MMM yyyy HH:mm:ss Z"	Wed, 4 Jul 2001 12:08:56 -0700
"yyMMddHHmmssZ"	010704120856-0700

Glossary

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

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

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

Notices

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

COPYRIGHT LICENSE:

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

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

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

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

Trademarks and service marks

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

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.

