

Rational Performance Test Server



# Reference Guide

*Version 8.0.0*



**Note**

Before using this information and the product it supports, read the information in “Notices” on page 96.

This edition applies to version 8.0.0 of Rational Performance Test Server and to all subsequent releases and modifications until otherwise indicated in new editions.

© **Copyright IBM Corporation 2001, 2012.**

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

# Contents

<b>About this Publication</b> .....	<b>v</b>
Intended Audience .....	vi
Scope .....	vi
Typographical Conventions .....	vi
Contacting IBM Support .....	vi
<b>Overview</b> .....	<b>1</b>
Features in Brief .....	2
Solving Real World Problems .....	3
Architecture .....	4
Typical Deployment .....	7
<b>Creating and Running Performance Tests</b> .....	<b>9</b>
Before Creating Performance Tests .....	10
Creating Performance Tests .....	16
Running Performance Tests .....	39
<b>Viewing Test Results</b> .....	<b>43</b>
Working With Data Sets .....	44
Managing Charts and Chart Templates .....	56
Modifying Chart Counters .....	58
Modifying Chart Styles .....	61
<b>Appendix: Probes Reference</b> .....	<b>66</b>
Introduction .....	67
System Statistics Probe .....	68
Windows Performance Monitor Probe .....	70

---

TIBCO BusinessWorks Probe.....	71
TIBCO Rendezvous Probe .....	74
TIBCO Rendezvous Distributed Queues Probe .....	77
TIBCO Rendezvous Trace Probe .....	78
TIBCO EMS Probe.....	81
SonicMQ Probe.....	84
webMethods Broker Probe .....	86
webMethods Integration Server Probe.....	89
Probe Error Handling.....	91
<b>Glossary .....</b>	<b>94</b>
<b>Notices .....</b>	<b>96</b>
Trademarks and service marks .....	99

# About this Publication

## **Contents**

### **Intended Audience**

### **Scope**

### **Typographical Conventions**

### **Contacting IBM Support**

This guide provides details of how to use the performance testing features of IBM® Rational® Performance Test Server to execute performance tests and examine the results, information covering the deployment and configuration of the product can be found in *IBM Rational Integration Tester Installation Guide* and *IBM Rational Performance Test Server Installation Guide*.

Performance testing relies on a number of probes to gather information from infrastructure components, such as messaging servers. Configuration of these probes, and details of the information they provide is contained in this guide.

---

## Intended Audience

Readers should already be familiar with using Rational Integration Tester for functional testing, in particular the concepts of tests, environments, test data sets and general product usage are assumed. Further details on these concepts can be found in *IBM Rational Integration Tester Reference Guide*.

## Scope

This document is concerned only with Rational Performance Test Server, its environment and configuration. It does not discuss messaging technologies themselves, such as TIBCO Rendezvous, JMS implementations, and so on. If you wish to familiarize yourself with such technologies please refer to documents provided by the relevant companies or individuals.

For an introductory tour of the product with worked examples using a sample project, refer to *IBM Rational Performance Test Server Getting Started Guide*.

## Typographical Conventions

The following typographical conventions are observed throughout this document:

---

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

---

## Contacting IBM Support

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

# Overview

## **Contents**

### **Features in Brief**

### **Solving Real World Problems**

### **Architecture**

### **Typical Deployment**

This chapter provides an overview of Rational Performance Test Server and its architecture.

---

## 1.1 Features in Brief

Rational Integration Tester is a rich, scripting-free environment for developing automatic tests for SOA and all messaging technology projects.

Rational Performance Test Server extends this functionality to enable users to:

- Orchestrate a large number of events across multiple computers.
- Change the load generated by those multiple computers automatically.
- Capture key performance indicators from the execution of these tests.
- Capture key performance indicators from other items of infrastructure impacted by the tests.
- Examine the results using powerful flexible charting.
- Compare current results with past results from the same test.
- Compare any results with any other result from other tests.
- Embed these charts in reports.
- Save the charts for sharing with colleagues.

---

## 1.2 Solving Real World Problems

A variety of real problems can be solved with the performance testing features included in Rational Performance Test Server, including:

---

Soak Tests	Running a repetitive test over and over, monitoring CPU and memory utilisation of the processes exercised by the test.
Load Tests	Changing the load on the system to establish how it behaves under different load conditions.
Stress Tests	Taking the load on the system to extreme levels, and ensuring that it does not develop any untoward side effects.

---

Deploy system statistics probes to monitor key components of the infrastructure, in particular the CPU and memory utilization of individual processes, and of any computers upon which the system under test depends.

### 1.2.1 Soak Tests

Soak tests can generate a lot of information because they run for a long time. Longer summary intervals will avoid filling the database with an unnecessary level of detail. Increased intervals on probe configurations can be used for the same reason.

The Constant Growth load profile can be used with a zero increment to provide a constant load. If a varying load is required, use the Externally Defined load profile together with a test data set.

### 1.2.2 Load Tests

Load tests subject the system to realistic loads (current and anticipated). It is important to subject the system to realistic loads (for example, a pension system might have 10,000 users around the world, but all users are unlikely to access the system at the same time). Conversely, a concert ticket system might be idle most of the time, but for a few hours (when tickets first go on sale) the number of users could far exceed even the total number of tickets available. This last example may be considered a stress test.

An Externally Defined load profile offers the best opportunity for controlling the load presented to the system over time.

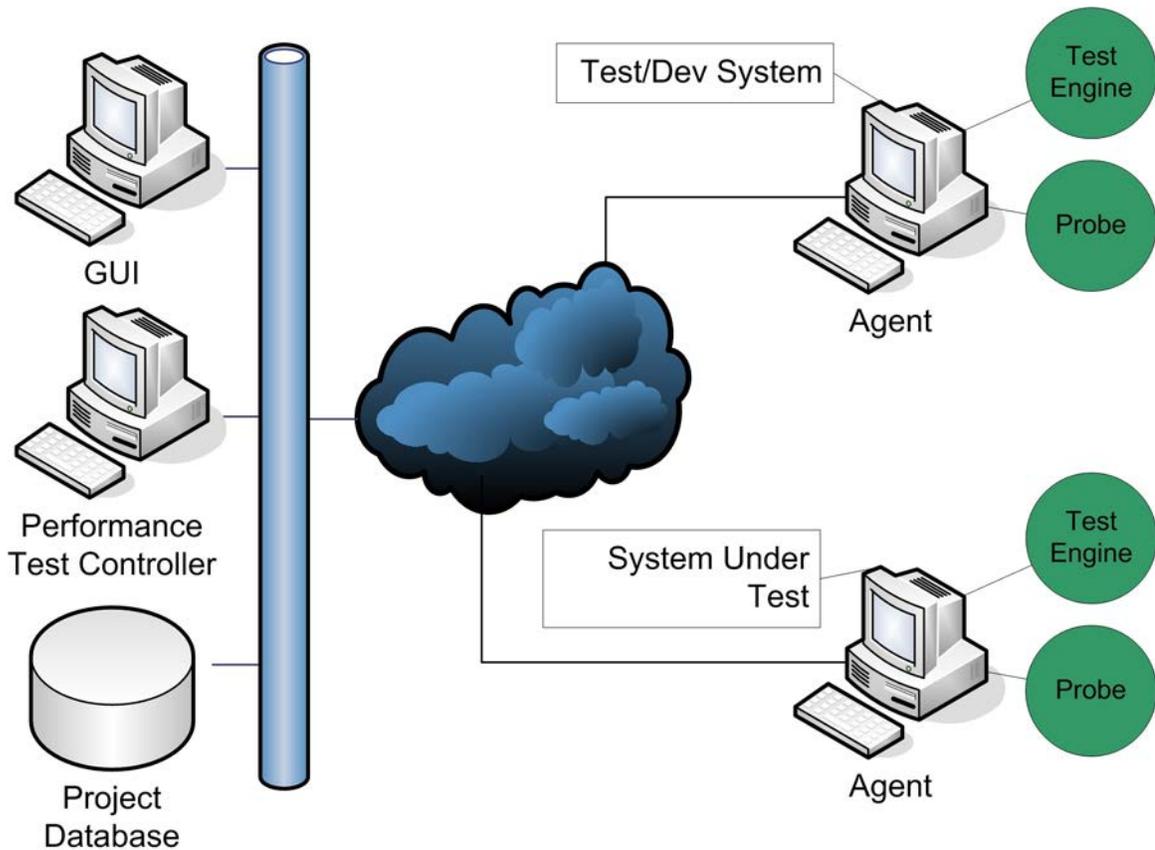
### 1.2.3 Stress Tests

Stress tests help ensure that the system fails gracefully under extreme loads and does not develop any undesired side effects (for example, memory leaks, excessive time-outs, and so on).

---

## 1.3 Architecture

The following diagram gives an overview of the Rational Performance Test Server architecture.



Note the following about network connectivity and deployment:

- An agent must be installed on every computer that runs a probe or a test engine instance to manage these processes (that is, launching them when required and shutting them down when they are no longer needed).
- Every computer that runs an agent must be able to connect to the database.
- The performance test controller computer must be able to contact the database and the agents.
- Computer used within a test will need to have their clocks synchronized, this can be achieved using NTP.

---

### 1.3.1 Project Database

The project database is essential because it stores all of the historical data for Rational Performance Test Server. The connection is JDBC-based, requiring a single user ID with its own schema, and all results are permanently stored for later analysis. For more information about supported databases and how they are created, refer to *IBM Rational Integration Tester Installation Guide*.

### 1.3.2 Performance Test Controller

This is Rational Performance Test Server running to control the execution of performance tests including local and remote agents and probes. A special licence is required in order for Rational Performance Test Server to act in this way, otherwise it will function as Rational Integration Tester with no performance functionality.

The controller may be run by means of the Rational Performance Test Server GUI, or it may be run on the command-line.

### 1.3.3 Rational Integration Tester

Rational Performance Test Server includes all of the same functionality as Rational Integration Tester, with the addition of performance testing. This means that Rational Performance Test Server can be used to create functional tests that are executed as part of a performance test. Additionally, an external Rational Integration Tester installation may be used to create test projects that are used in performance tests using Rational Performance Test Server.

### 1.3.4 Agent

An agent is a Java process that is used by the Performance Test Controller to start test engines and probes. An agent may be installed on the local computer or on a remote server. Communication between the agent and other components utilizes the HTTP protocol.

### 1.3.5 Test Engine

A test engine is an instance of the Rational Integration Tester engine, started by an agent, that is executing a test. Test results are stored directly into the project database by means of a JDBC connection. Multiple test engines can be configured to run from a single agent, which allows for multiple distributed tests to be executed concurrently using a single agent.

---

### 1.3.6 Probe

A probe is a Java program that captures performance data that is external to the Rational Performance Test Server environment. Probes run during performance tests to measure the impact of functional and load tests on the IT infrastructure. For example, a Rational Integration Tester test sends high volumes of message traffic to a messaging or web server. A probe would be used to collect statistics on the receiving servers and processes. This data can then be analysed along with messaging data to assess how well the infrastructure can cope with the load. More than one probe may be run from a single agent process.

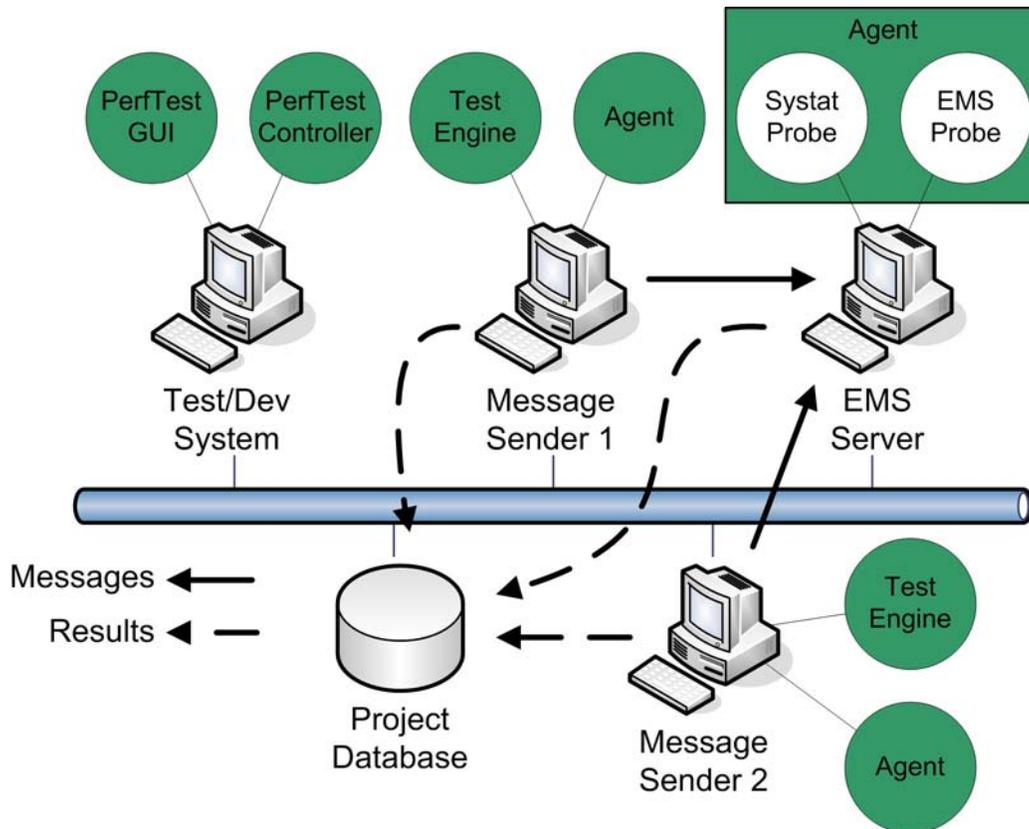
Probes write their results, as a set of counters, directly to the project database by means of a JDBC connection.

For more information about probes, refer to [Appendix: Probes Reference](#).

---

## 1.4 Typical Deployment

The following diagram illustrates a typical deployment, showing the various Rational Performance Test Server components and their locations.



The test sends a known quantity and rate of TIBCO EMS messages to an EMS server, records the reply times, and measures the impact on the receiving computer.

The messaging load requires two computers to send the messages, and agents are installed on each computer. A separate test computer is used to create, edit, and run the tests. An agent monitors the EMS server and probes capture server statistics.

The flow of EMS messages and test results are shown on the diagram. For the sake of clarity, some additional connections between the controller and agents have not been drawn. Each agent is started before any tests, and they each listen on an HTTP port for connections from the Performance Test Controller. These connections are used to send test configurations and to start/stop the tests.

---

The following sequence of events takes place from start to finish in the EMS messaging test that has been illustrated:

1. Agents are started on all computers that will run tests or probes.
2. The performance test is run by means of the GUI or a command prompt.
3. The controller calculates the individual message rate to be used by the test engines, and the number of messages to send is split evenly between the available test engines.
4. The test project data and configuration, including message rate, is sent to the remote agents by means of HTTP.
5. Agents start their probes and test engines and report status back to the controller.
6. When everything is ready the controller issues the start test command.
7. The test engines and probes perform their tasks and store their results into the project database as they go along.
8. The controller signals the agents to terminate the test, and the agents stop their respective test engines and probe processes. The agents remain running so that they are available for the next test.
9. The controller quits the test and reports its status.
10. The user can now extract the results from the database and plot some charts.

# Creating and Running Performance Tests

## **Contents**

**Before Creating Performance Tests**

**Creating Performance Tests**

**Running Performance Tests**

This chapter describes how to create, configure, and run performance tests.

---

## 2.1 Before Creating Performance Tests

Agents are required on the computers that will run Rational Integration Tester tests and the probes that monitor the systems. Therefore, before creating any performance tests, you should create and configure the necessary agents.

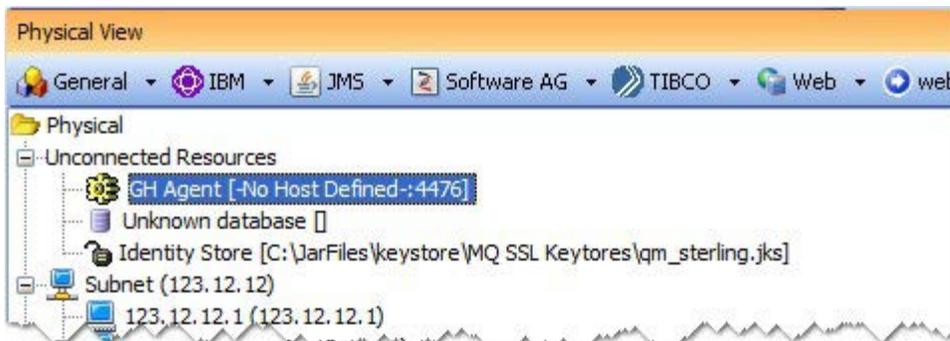
After an agent has been installed, it is managed in Architecture School's Physical View, where it can be created and configured as needed.

### 2.1.1 Creating Agents

To create an agent:

1. In Physical View, click **General > Agent** on the main toolbar, or right-click the **Physical** folder and click **New > General > Agent** on the shortcut menu.

The new agent will be created under the **Unconnected Resources** node because no host for the agent has yet been defined.



- 
2. Double-click the new agent (or an existing one) to edit it.



3. In the **Host** field, enter the host name or IP address of the computer where the agent is installed.
4. In the **Port** field, enter the port number where the agent is listening.

**NOTE:** Host names are preferred because an IP address is more likely to change. You should also avoid using `localhost` because problems could arise if the agent's test is run from a different computer.

Each agent includes a default test engine (named `default`) that cannot be renamed or deleted. To add another test engine, click **Add**. To rename an existing engine, select it and click **Rename**. To delete an existing engine, select it and click **Delete**.

**NOTE:** For information about configuring multiple test engines, refer to [Creating Multiple Test Engines \(Optional\)](#).

5. To view or modify additional notes or information about the agent, click the **Documentation** tab.
6. When you are finished configuring the agent, click **OK**.

---

The agent is displayed under an appropriate subnet and host in the Physical View. If the host cannot be resolved, the agent will remain under **Unconnected Resources**.

### 2.1.1.1 Creating Multiple Test Engines (Optional)

The `Agent.config` file (which is located in `<Rational Integration Tester Installation Directory>\config`) defines internal agent parameters and test engine names.

The contents of the `Agent.config` file are shown below. Two engine instances, `Engine1` and `Engine2`, are highlighted in **blue**.

```
<?xml version="1.0" encoding="UTF-8"?>
<applications>
  <application type="TestEngine" workingDirectory=".">
    <args>
      <arg value=".\\TestEngine.exe"/>
      <arg value="-processId"/>
      <arg id="ProcessId"/>
      <arg value="-eventURL"/>
      <arg id="EventURL"/>
      <arg value="-httpPort"/>
      <arg value="0"/>
    </args>
    <instance name="Engine1"/>
    <instance name="Engine2"/>
  </application>
  <application type="Probe" workingDirectory=".">
    <args>
      <arg value=".\\Probe.exe"/>
      <arg value="-p"/>
      <arg id="ProcessId"/>
      <arg value="-m"/>
      <arg id="EventURL"/>
    </args>
  </application>
</applications>
```

After modifying the file, the agent must be restarted to enable any changes that you have made to take effect. When running from a command prompt, the engine names will be displayed, as shown below:

---

```
C:\Program
Files\IBM\RationalIntegrationTester\bin>ghtesteragent.bat

Agent listening for requests on port: 4475

Available application: TestEngine/Engine2

Available application: Probe

Available application: TestEngine/Engine1
```

## Logging

The logging entry in the `Agent.config` file should be uncommented (that is, remove “`<!--`” and “`-->`” on either side) to enable logging for the agent process.

```
<application type="TestEngine" workingDirectory=". ">
  <logging enabled="true" directory="./log" />
```

**NOTE:** The specified log directory must exist for logging operations to succeed. In this case, it is `<Rational Integration Tester Installation Directory>\log`.

## Process Threads

By default, the number of processing threads (the pool) available to the agent program is 750. However, in some cases, you may want to use more (for example, when sending large numbers of request-reply messages). If the replies are received at a slower rate than the requests, the number of active test instances and the number of active threads will increase.

**NOTE:** Thread usage data is stored in the results database but thread pool usage as a percentage and as an absolute value is available separately. This data is found within the **Test Engine** section of results.

To set the value for the size of the thread pool, add the following lines within the `<args>` section of the agent configuration file, replacing `<# of threads>` with the number of threads to use.

---

```
<args>
...
<arg value="-maxThreadPoolThreads"/>
<arg value="<# of threads>"/>
...
</args>
```

## 2.1.2 Running Agents

The Rational Integration Tester agent is normally configured to run continuously, listening for HTTP connections from the Performance Test Controller.

The following sections describe how to start agents on Windows systems and Linux/Unix systems.

### 2.1.2.1 Microsoft Windows Agents

On computers running Windows, the agent is typically installed and run as a Windows Service, as shown in the following graphic.

To verify that the agent service is installed, open the Services control panel **Start > Settings > Control Panel > Administrative Tools > Services**. The Windows service name is **IBM RIT Agent**.

To install or remove the Windows service manually, use the following batch files (which are located in *<Rational Performance Test Server Installation Directory>\bin*):

- InstallAgentService.bat
- RemoveAgentService.bat

There are two ways to start an agent on a computer running Windows:

- Use a command prompt.
- Use the `GHTesterAgent.bat` file.

The following sections describe both of these methods.

---

### Using a Command Prompt to Start/Stop the Agent

The agent service can be started and stopped from a command prompt by using the following net commands:

- > net start "Rational Integration Tester Agent"
- > net stop "Rational Integration Tester Agent"

### Using the GHTesterAgent.bat File to Start the Agent

The agent can be started manually by running the GHTesterAgent.bat file (which is located in <Rational Performance Test Server Installation Directory>\bin).

The batch file runs the Agent.exe file using the default port (4476) and points to the default agent configuration file (config/Agent.config). If desired, you can modify the file to specify a different port using the -p <port> argument, or override the location of the Agent.config file.

For example:

```
.\Agent.exe -p 4492 -c config\newdir\Agent.config
```

## 2.1.3 Linux/Unix Agents

On Linux and Unix systems, the agent can be launched by running the Agent script in the root of the Rational Performance Test Server installation directory (for example, /opt/ghtester/).

The script can be incorporated into a system startup script to ensure that the agent is always launched when the server starts.

**NOTE:** Temporary files are stored in /var/tmp. Ensure that the user running Rational Performance Test Server has write permissions on this directory.

The Agent script runs the agent, using the /bin/run\_ini.sh script, using the default port (4476) and points to the default agent config file (config/Agent.config). If desired, you can modify the script to specify a different port using the -p <port> argument, or override the location of the config file.

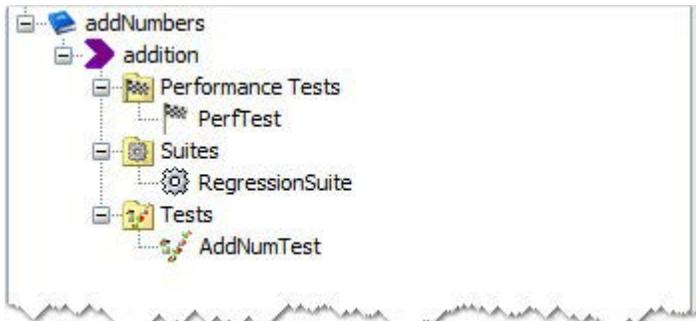
For example:

```
./bin/run_ini.sh ./Agent.ini -p 4492 -c ./config/Agent.config  
"$@"
```

---

## 2.2 Creating Performance Tests

A performance test is created within a Rational Performance Test Server project.



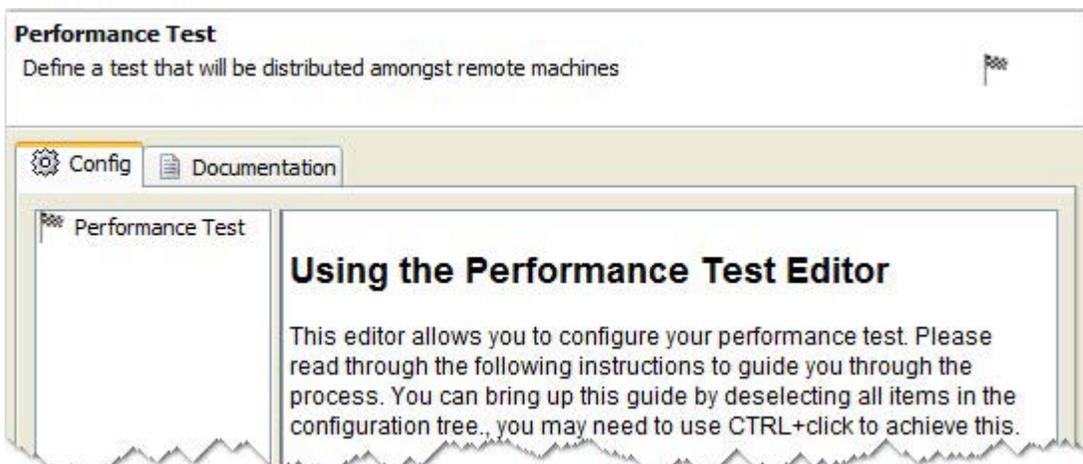
To create a performance test:

1. Right-click the desired operation or folder and click **New > Tests > Performance Test**.

Alternatively, click the arrow next to the test icon at the top of the component tree and click **Performance Test**.

2. In the Create new Performance Test dialog box, enter a suitable name for the performance test (the test's results will be identified by this name).
3. Click **OK**.

The empty performance test is displayed in the design view, to the right.



The new test is displayed at the top of an empty tree under the **Config** tab. The tree is created as items are added to the test. When an item in the tree is selected, its details

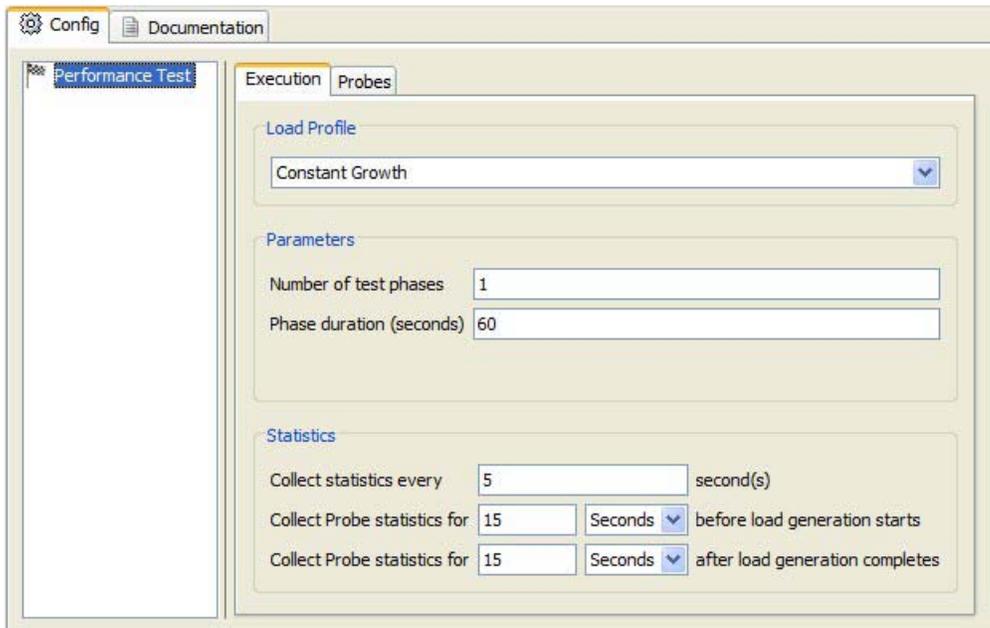
---

are displayed on the right panel. With nothing selected (right-click the empty space beneath the tree), some documentation about using the test editor is displayed.

**NOTE:** On the **Documentation** tab, you can enter a description of the test and enter the name or user ID of the test's owner.

### 2.2.1 Configuring Execution

Select the root of the performance test in the configuration tree to display its configuration details, as shown below.



The **Execution** tab (on the **Config** tab) contains settings that control the way in which the performance test's contents will be run, and the **Probes** tab is used to manage the probes that are part of the test.

---

### 2.2.1.1 Configuring Load Profiles

The load profile defines the way in which the tests in the performance test will run (that is, the rate at which new test iterations will be started).

The following table describes the two available load profile options.

---

<b>Load Profile Option</b>	<b>Description</b>
Constant Growth	This profile starts tests at a given rate and increases the rate by a fixed amount for each new test phase. The rates are defined within the details of the individual tests. The parameters available for constant growth are the <b>Number of test phases</b> (minimum of 1) and the <b>Phase duration</b> , or the number of seconds each test phase should last.
Externally Defined	When using the <b>Externally Defined</b> profile, the load is extracted from a test data set that has been previously configured within the project. This enables greater flexibility because the duration of each test phase can be different, as can the test rate.

---

**NOTE:** Remote test data can be pulled from the controller to the agent, but only for test data stored within the project, and only for file and Microsoft Excel data sources.

### 2.2.1.2 Configuring Parameters

The data set must contain two columns that will be used by the profile. The first column defines the phase duration and the second defines the target number of iterations.

For example:

---

<b>myTime</b>	<b>myIterations</b>
30	50
60	75
120	100

---

**NOTE:** If necessary, these columns could be added to an existing data set since the test configuration allows you to select the first column to use (for the phase duration).

---

After choosing the test data set, select the column to use for the phase duration and the units of time to apply (seconds, minutes, or hours).

The screenshot shows the 'Parameters' section of a configuration window. Under 'Data set for load settings', there is a text box containing 'Perf Profile' and buttons for 'Browse...' and 'Clear'. Below this is the 'Execute test phases' field, which is currently empty. The 'Phase duration read from column' dropdown menu is open, displaying two options: 'myTime' (which is highlighted) and 'myIterations'. To the right of this dropdown is the 'Phase duration units' dropdown, which is set to 'Seconds'. In the 'Statistics' section below, the 'Collect statistics every' field is set to '5' with the unit 'second(s)'.

Each row of data in the data set represents a test phase, and all phases are executed by default. However, a range of phases to execute can be specified by entering a comma-separated list of phases in the **Execute test phases** field.

For example:

- 1 executes only phase one.
- 1, 2, 5, 6 executes phases one, two, five, and six.
- 1, 10-20 executes phase one and phases 10 to 20 inclusive.

### 2.2.1.3 Configuring Statistics

The **Statistics** options define the time interval during which probe statistics should be collected.

The screenshot shows the 'Statistics' section of a configuration window. It contains three rows of settings. The first row is 'Collect statistics every' with a text box containing '5' and the unit 'second(s)'. The second row is 'Collect Probe statistics for' with a text box containing '15', a dropdown menu set to 'Seconds', and the text 'before load generation starts'. The third row is 'Collect Probe statistics for' with a text box containing '15', a dropdown menu set to 'Seconds', and the text 'after load generation completes'.

The interval at which counters are returned (**Collect statistics every...**) can be set for each period, and the minimum interval is one second. Additionally, you can configure probe statistics to be collected for some period of time before tests are executed and after they have completed. The pre-load statistics can provide insight into the system in its quiescent state, and the after period can show how the system is impacted after the tests have completed (for example, there may be residual issues with CPU or

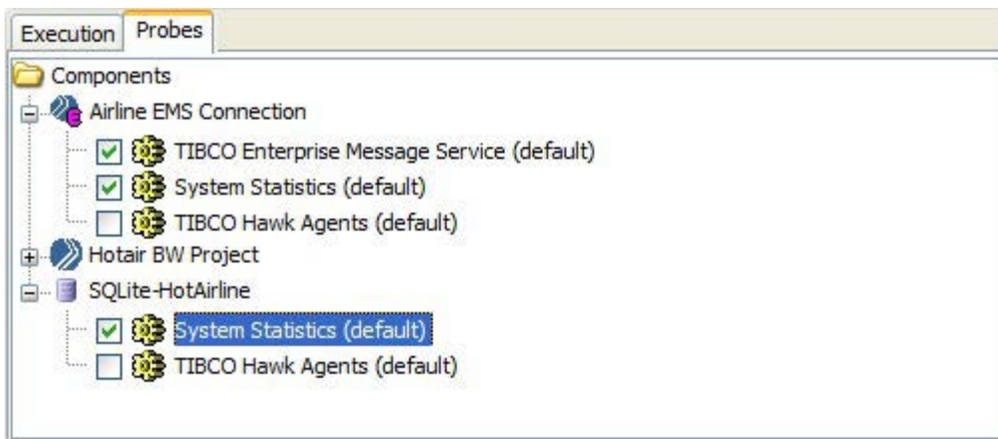
---

memory while programs process data after the Rational Integration Tester tests have completed).

**NOTE:** Using a shorter interval may not improve results. Instead, it may simply increase the amount of data in the database.

## 2.2.2 Configuring Probes

The **Probes** tab (on the **Config** tab) is used to configure which probes to run during the test, and where to run them. Probes are started by agents, so the local and/or remote agents must be configured in the performance test before the probes (see [Before Creating Performance Tests](#)).



The available probes are determined by the components that are utilized in the Rational Integration Tester tests that have been added to the performance test. For example, if the distributed test references a TIBCO EMS transport, the TIBCO Enterprise Message Service probe will be available.

**NOTE:** It is not possible to create a performance test with only probes (that is, no actual tests). Always ensure that you configure at least one distributed test (background or load-generating) that executes an existing Rational Integration Tester test. For information about this, refer to [Adding Distributed Tests \(Optional\)](#).

By default, probes are not enabled initially after adding Rational Integration Tester tests to the background or load-generating tests. To enable one or more probes, select the check box next to the desired probe.

**NOTE:** For more information about available probes and their functionality, including error handling, refer to [Appendix: Probes Reference](#).

---

### 2.2.3 Adding Distributed Tests (Optional)

The following types of distributed tests may be added to an overall performance test:

- A load-generating test, which can have a fixed or varied load profile, such as a linear increase in published messages.
- A background test, which defines a constant load and is used to produce a steady stimulus.

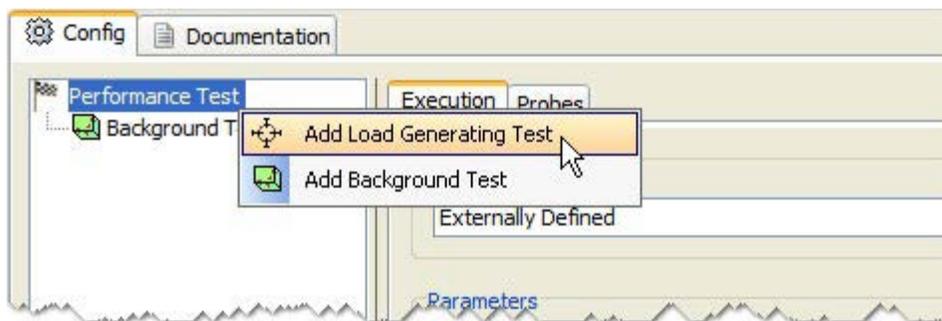
**NOTE:** Background tests are not used to collect timing information (any timed sections are ignored). Instead, they are either used to generate fixed load or to simulate (stub) missing systems.

Both test-types run an existing Rational Integration Tester test and data may be mapped in using an existing test data set.

One or more tests can be added to the root of the performance test.

**NOTE:** If a performance test has more than one distributed test added to it, they are executed concurrently.

To add a distributed test to the performance test, right-click the root of the test and click the desired distributed test to add.



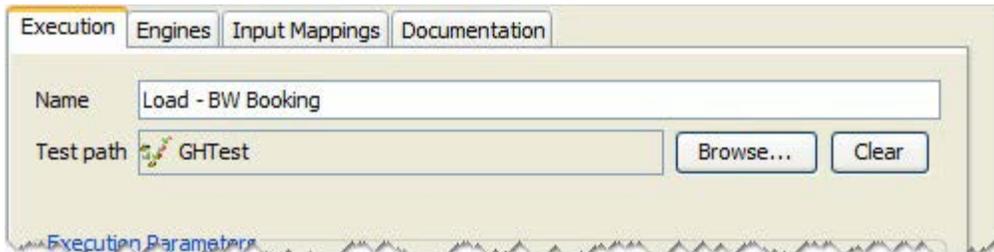
**NOTE:** Rational Integration Tester enables you to add test actions that provide timing information for specific parts of the tests. For information about this, refer to [Configuring Timed Sections](#).

After a distributed test has been added to the performance test, it must be configured. Select the desired test from the performance test tree and use the following tabs to modify its properties:

---

### 2.2.3.1 Configuring Execution Options

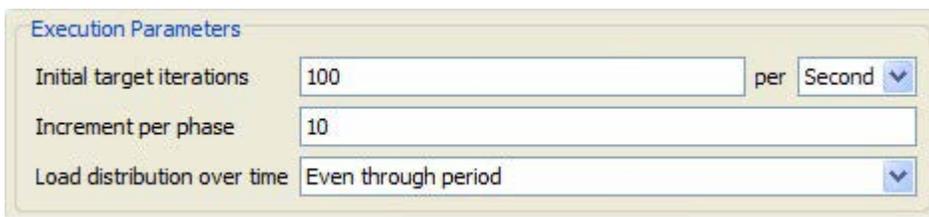
On the **Execution** tab, you can modify the name of the selected test as well as the Rational Integration Tester test that it should run. Additionally, and depending on which type of distributed test is selected, you can configure how the Rational Integration Tester test should be executed.



1. Enter the name for the test in the **Name** field.
2. Click **Browse** to locate the Rational Integration Tester test that should be run by the distributed test.
3. Configure additional execution options according to the type of distributed test (that is, [Load-Generating Tests](#) or [Background Tests](#)).
4. When finished, click **Save** (or press CTRL+S) to save your changes.

#### Load-Generating Tests

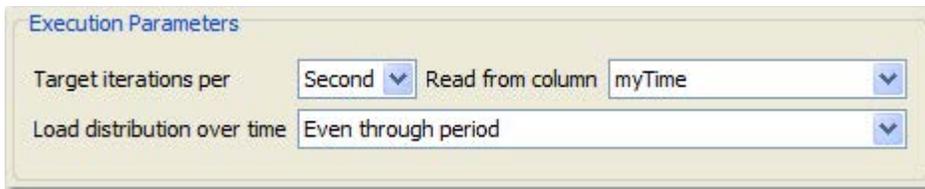
If the load profile for the selected test is **Constant Growth**, the execution parameters will be defined in this section and the number of test phases and their duration are fixed.



By default, load generation is set to **Even through period**, which will spread iterations evenly through the period. In the example shown above, 100 iterations per second will result in 100 messages per second (that is, one every 10 milliseconds). If the **Front-loaded burst** option is selected, the 100 iterations would be run as fast as possible at the beginning of the time period.

---

If the load profile for the selected test is **Externally Defined**, the execution parameters are taken from a test data set.



The unit of time for target iterations (seconds, minutes, or hours) must be selected, and the number of iterations is read from the selected column of the test data set. The load distribution settings are the same as those for [Load-Generating Tests](#).

**NOTE:** If the load profile is changed while configuring the test, you must save the change before setting the execution parameters.

### Background Tests

A background test simply runs the selected test repeatedly while the performance test is active. The only execution options to configure are whether or not the performance test should be stopped if the Rational Integration Tester test fails (the **Terminate on failure** check box) and the maximum number of times the Rational Integration Tester test should be executed (the **Instances** field).



### 2.2.3.2 Configuring Engines

Test engines are instances of Rational Integration Tester that execute performance tests. When a performance test is started, the performance test controller sends a copy of the project to the test engine for it to run.

For Windows users, the engine process name is `TestEngine.exe`.

For Linux\Unix users, the engine process is a Java process that may be identified by using long `ps` output.

Each new agent includes a single test engine instance called `default`. This test engine can run one distributed test at a time. If desired, you can configure additional test engines on a single agent to run multiple tests simultaneously.

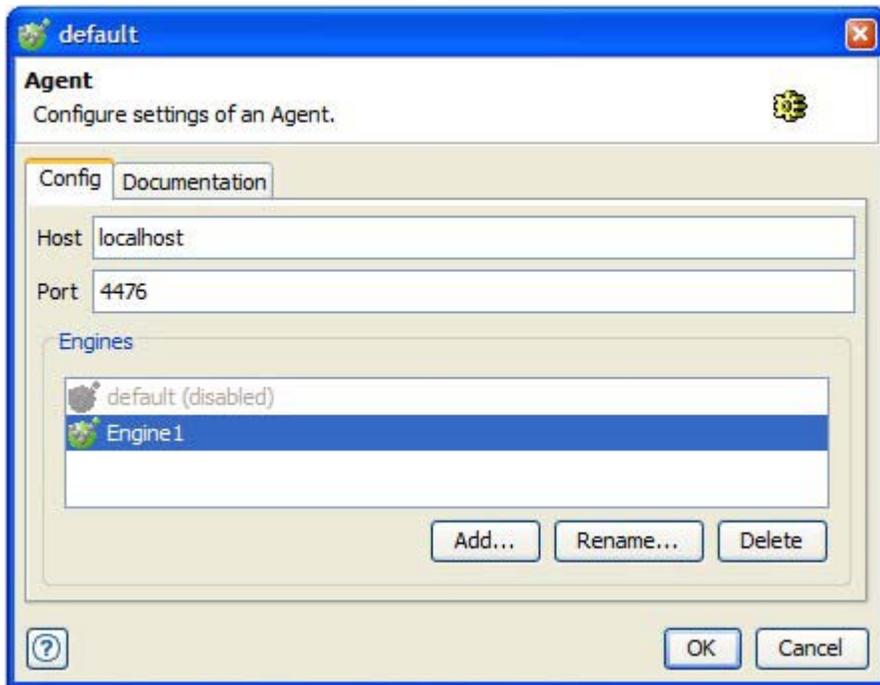
The following sections describe how to create additional test engines and how to link test engines to performance tests.

---

## Creating Test Engines

To create a test engine:

1. Double-click an existing agent in Architecture School's Physical View.



2. Click **Add** (to add another test engine).

Alternatively, click **Rename** to rename the selected engine or click **Delete** to delete the selected engine.

3. Enter a name for the new test engine when prompted.

**NOTE:** When you create a new test engine, the default instance is disabled.

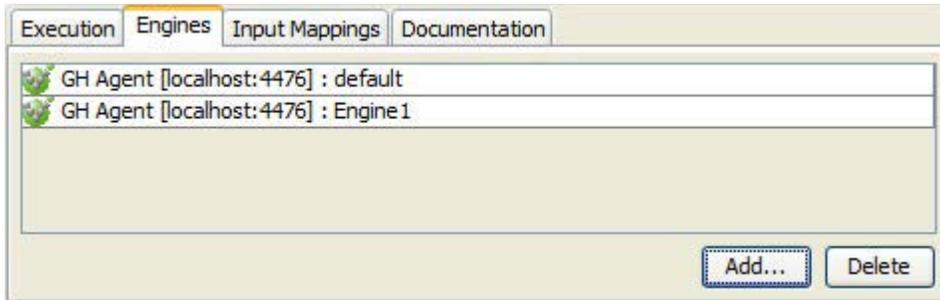
4. Click **OK** to close the dialog box and save your changes.

**NOTE:** The `Agent.config` file (which is located in `<Rational Integration Tester Installation Directory>\config`) must be modified to include the engine names listed in the GUI.

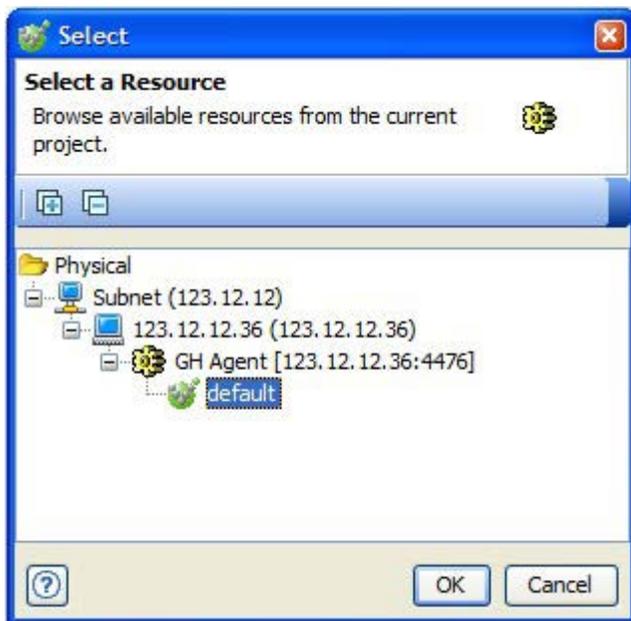
---

## Linking Test Engines to Tests

On the **Engines** tab, you can add one or more test engines that are required to run the selected distributed test.



To link an engine to test, click **Add** and select the desired engine from the project resource tree. The engines are listed by subnet, then by host, then by agent.

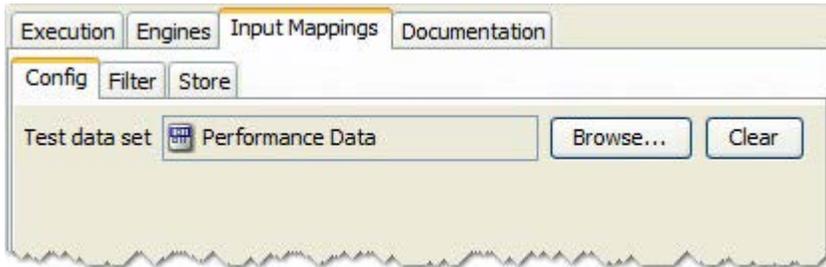


---

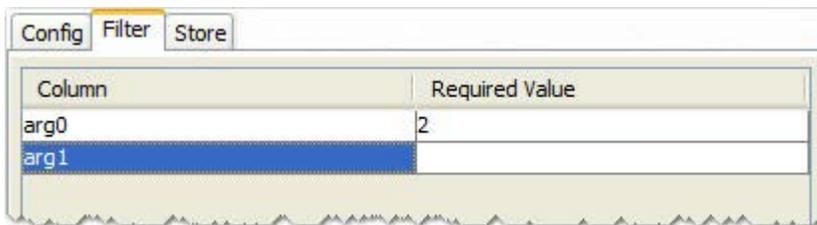
### 2.2.3.3 Configuring Input Mappings

On the **Input Mappings** tab, you can map values from a selected test data set to the tags in your test. You can also apply filters to the data within the data set.

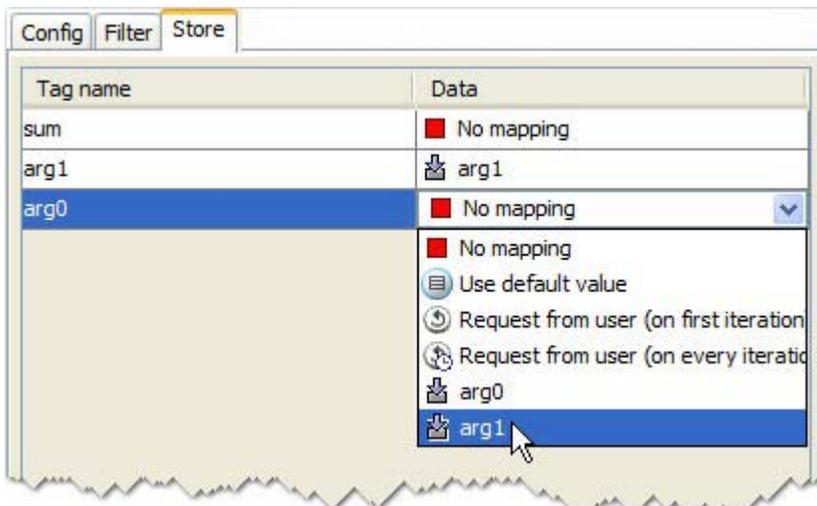
1. Click the **Config** tab and click **Browse** to select an existing test data set.



2. If you want to filter the rows to be used in the data set, click the **Filter** tab and enter the values to match for each of the columns in the data set.



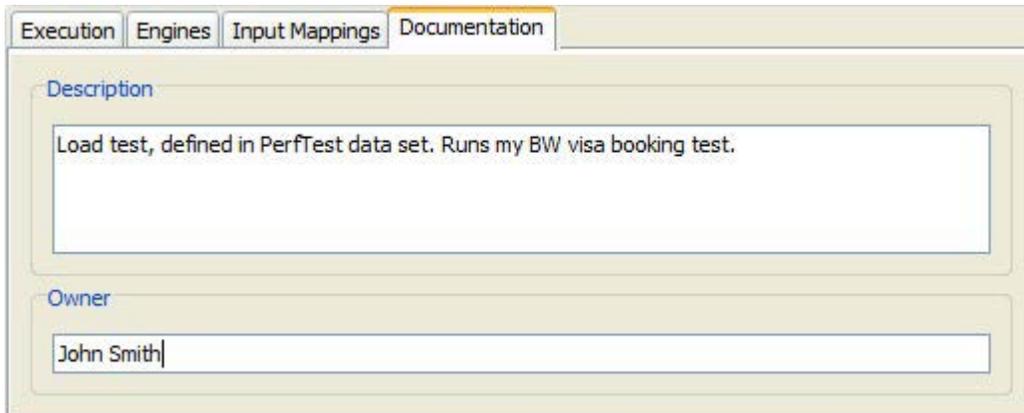
3. Click the **Store** tab to select the data that should be mapped to each of the tags in the test.



---

### 2.2.3.4 Adding Descriptions/Notes

On the **Documentation** tab, you can enter a description or notes about the selected test, and specify the test's owner.

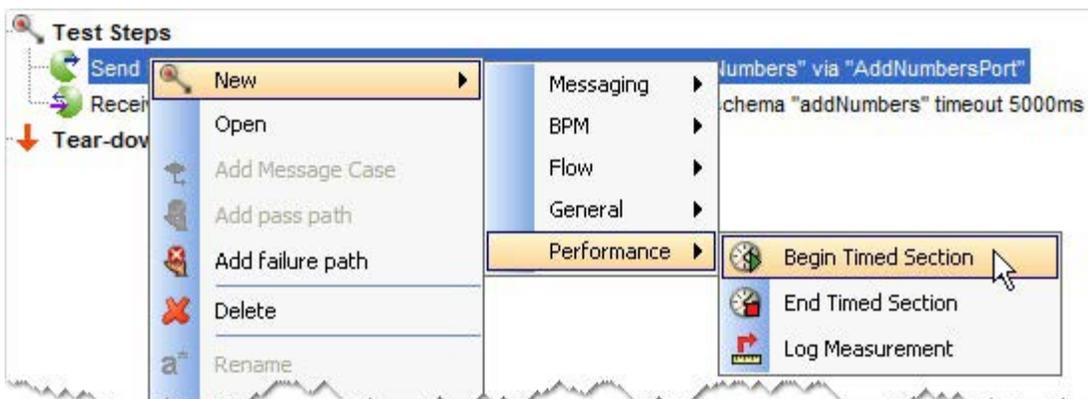


**NOTE:** All of the information on the **Documentation** tab is optional and does not affect how a test is executed.

### 2.2.3.5 Configuring Timed Sections

When tests are run within a distributed test in Rational Performance Test Server, the overall execution times (that is, start to finish) are recorded.

In addition to this overall time, it is possible to measure individual sections within the steps of the Rational Integration Tester test. This can be done by adding markers (that is, the [Configuring Begin Timed Section Actions](#) and [Configuring End Timed Section Actions](#) actions) to the test steps to create a timed section.



**NOTE:** Timing markers are only allowed within the Test Steps section of the test, and more than one timed section is allowed within a single test.

---

**NOTE:** Timed sections are ignored in background tests.

**NOTE:** Timed sections have no effect when the test is executed outside of the performance test.

The following sections describe which test actions support timed sections and how to configure Begin Timed Section and End Timed Section actions.

### Test Action Support for Timed Sections

The following table lists the various test steps and the support offered when using timed sections around them.

**NOTE:** Support relates to placing the begin/end steps directly before or after them. An unsupported step may still reside within a timed section if it complies with the limitations described in the following table.

---

<b>Test Action</b>	<b>Begin Timed Section</b>	<b>End Timed Section</b>
Assert Function	(Cannot be used within timed sections.)	
Action Group	As the action group is entered.	When the last child has finished.
Decision	Immediately before command execution	Immediately after the true or false path is followed.
Execute Resource	Immediately before the resource is called.	Immediately after execution returns back from the resource.
Fail	Not supported.	When the action is reached.
Function	Immediately before the function is executed	Immediately after the function returns.
Log	Immediately before the message is logged.	Immediately after the message is logged.
Log Measurement	(Cannot be used within timed sections. For information about this action, refer to <a href="#">Configuring Log Measurement Actions (Optional)</a> .)	
Lookup Test Data	Immediately before the test data is looked up.	Immediately after the true or false path is followed.
Pass	Not supported.	When the action is reached.
Publish	Immediately before the message is published.	Immediately after the message is published.

---

---

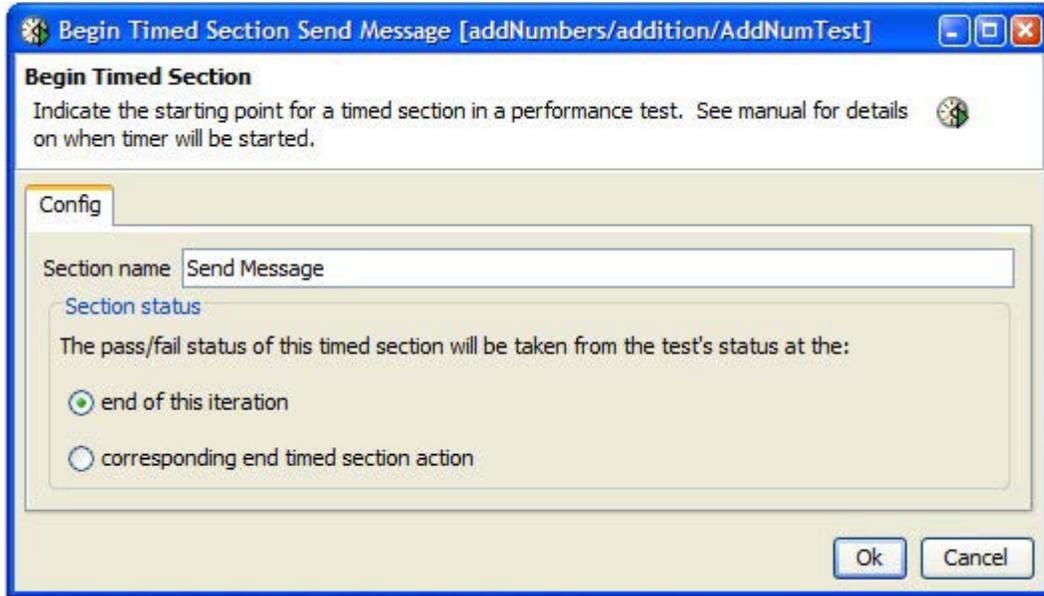
<b>Test Action</b>	<b>Begin Timed Section</b>	<b>End Timed Section</b>
Receive Reply	(Not supported.)	Immediately after a response is received.
Receive Request	Immediately before the system starts waiting for a request.	Immediately after a request is received.
Run Command	Immediately before command execution	Immediately after command execution.
Send Reply	Immediately before responding.	Immediately after responding.
Send Request	Immediately before sending the request.	Immediately after sending the request.
Sleep	Immediately before the sleep begins.	Immediately after the sleep finishes.
SQL Command	Immediately before statement execution	Immediately after statement execution
SQL Query	Immediately before statement execution	Immediately after statement execution
Subscribe	Immediately before the system starts waiting for an event.	Immediately after an event is received.
	Validation is ignored for the timestamp but the pass/fail status may be affected by it.	
Subscribe Until	(Not supported. Action is deprecated.)	
User Interaction	(Cannot be used in performance tests.)	

---

---

## Configuring Begin Timed Section Actions

The Begin Timed Section action marks the start of an individually timed section within the test.



You can assign a name to the test action to help identify it or to identify the series of steps that it spans. The name of the action will appear in the charting configuration tree, so you should choose a name that is meaningful.

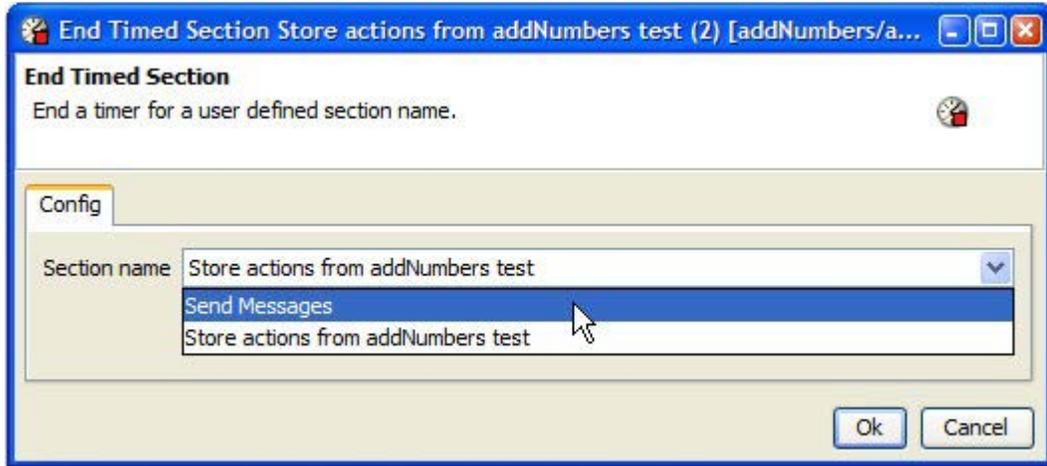
The action will store a pass or fail status based on the status of the test that contains it, which allows you to plot charts based on the outcome of the test iterations. You can store the status of the overall test iteration (end of this iteration) or the current status when the timed section ends (corresponding end timed section action).

**NOTE:** The status can be different for each of these selections (for example, the test could be passing when the timed section ends, but later actions in the test could cause it to fail).

---

## Configuring End Timed Section Actions

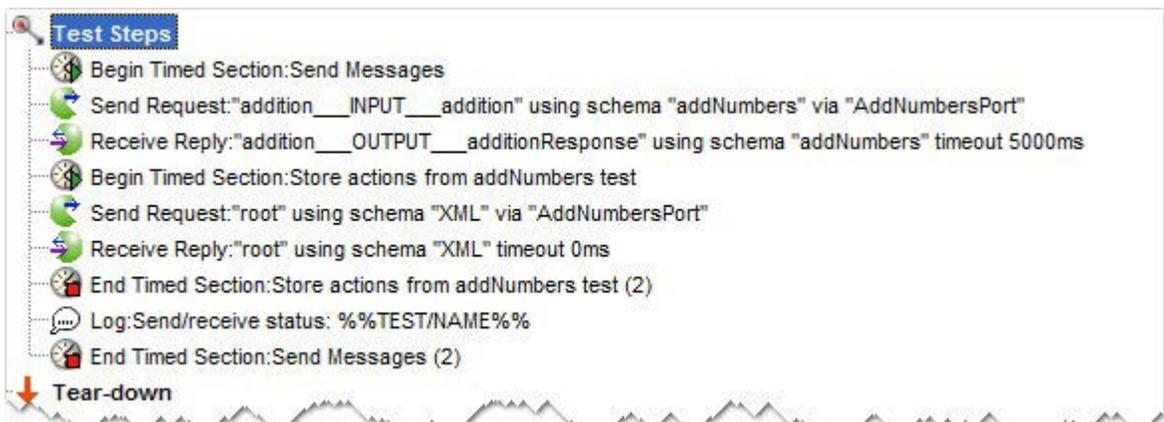
The End Timed Section action marks the end of an individually timed section within the test.



Except for selecting the appropriate action that marks the start of the desired timed section in the test, there is nothing to configure within the test action.

**NOTE:** If the test contains more than one timed section, it is important that each Begin Timed Section action is terminated with an appropriate End Timed Section action.

Using two timed sections, a simple timed test might look the test shown in the following graphic.



---

**NOTE:** Certain test actions are not supported within timed sections. For information about this refer to [Test Action Support for Timed Sections](#).

## 2.2.4 Configuring Log Measurement Actions (Optional)

The Log Measurement action is used to store arbitrary numeric data into the database to be displayed later as part of the performance charts. Each item of data is called a counter, and a number of text attributes can be configured and used to identify the counters. In addition, a UTC timestamp can be stored with each set of counters.

**Log Measurement**  
Please enter the name value pairs for the counters and attributes. The input in the value column will be evaluated as a function and will be written into the project database.

**Counters**  
Add the numeric data values that you want made available for later graphing.

Name	Type	Value
------	------	-------

Add Delete

**Attributes**  
Define the set of attributes which identify the series under which the counters will appear.

Name	Value
------	-------

Add Delete

**UTC Timestamp (ms)**  
 Use current timestamp  
Define custom timestamp

Ok Cancel

**NOTE:** Log Measurement actions should not be used within timed sections.

---

### 2.2.4.1 Configuring Counters

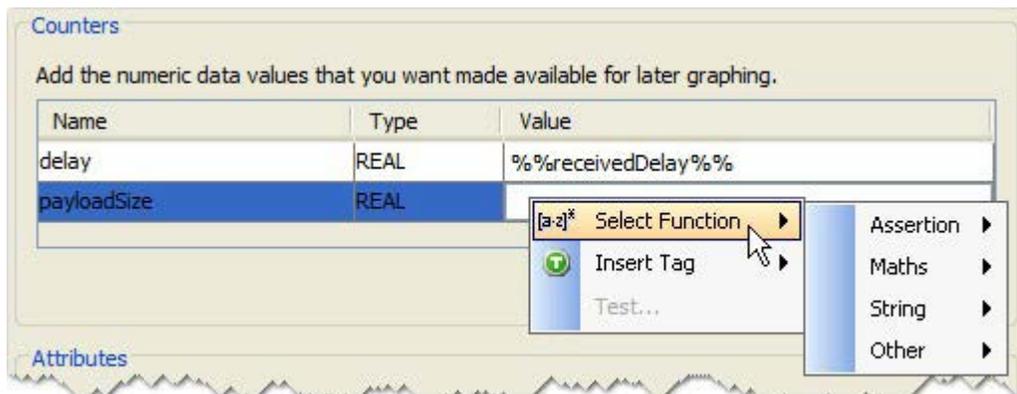
There are two types of counters:

- Numeric (REAL)
- String (STRING)

By default, the number of counters is limited to eight REAL and four STRING. (For information about increasing the number of counters, refer to [Modifying Counter Limits](#).)

To add a counter to a Log Measurement action, click **Add** under **Counters** on the Log Measurement dialog box.

To delete a counter from a Log Measurement action, select it on the Log Measurement dialog box and click **Delete**.



After a counter is added, you can configure it with a name, type, and value.

The name can contain spaces and may be up to 255 characters in length, although very long names will be slower to use and harder to display.

Select the type of each counter (**REAL** or **STRING**) on the **Type** column.

Finally, enter a value for the counter or right-click within the fields to insert a function or select a tag. When using a function, multiple tags can be referenced, if required.

---

## Modifying Counter Limits

If your project database connection is valid, Rational Integration Tester will silently query the counter limits. If the limits are exceeded, invalid rows will be shown in red and no rows can be edited until the extra ones are removed.

You can increase the number of counters by modifying a table definition in the database creation script.

The example below is from the Oracle create script. (Some lines of SQL have been omitted for brevity.)

The **blue bold** entries show the lines that need to be added to increase the number of REAL and STRING counters by one (CTR\_ID\_9 and STRCTR\_ID\_5).

```
CREATE TABLE "LOGMEASUREMENT_TS" (  
    "IDX"                NUMBER(19, 0)    NOT NULL,  
    "LOGMEASUREMENT_MRV_ID" NUMBER(19, 0)    NOT NULL,  
    "EXECUTION_ID"       NUMBER(19, 0),  
    "TIME"               NUMBER(20, 0)    NOT NULL,  
    ...  
    "CTR_ID_8"           NUMBER(19, 0),  
    "CTR_VAL_8"          NUMBER(20, 3),  
    "CTR_ID_9"           NUMBER(19, 0),  
    "CTR_VAL_9"          NUMBER(20, 3),  
    ...  
    "STRCTR_ID_4"        NUMBER(19, 0),  
    "STRCTR_VAL_4"       VARCHAR2(255),  
    "STRCTR_ID_5"        NUMBER(19, 0),  
    "STRCTR_VAL_5"       VARCHAR2(255),  
    FOREIGN KEY ("EXECUTION_ID")  
    REFERENCES "EXECUTION" ("ID") ON DELETE CASCADE,  
    FOREIGN KEY ("LOGMEASUREMENT_MRV_ID")  
    REFERENCES "LOGMEASUREMENT_MRV" ("ID") ON DELETE CASCADE,  
    FOREIGN KEY ("CTR_ID_1")  
    REFERENCES "COUNTER_DETAILS" ("ID") ON DELETE CASCADE,  
    ...  
    FOREIGN KEY ("STRCTR_ID_4")  
    REFERENCES "COUNTER_DETAILS" ("ID") ON DELETE CASCADE,  
    FOREIGN KEY ("CTR_ID_9")
```

---

```

REFERENCES "COUNTER_DETAILS"("ID") ON DELETE CASCADE,
FOREIGN KEY ("STRCTR_ID_5")
REFERENCES "COUNTER_DETAILS"("ID") ON DELETE CASCADE,
)
;

```

**NOTE:** The table sizes can be increased even if the project database contains data. If you need help, consult your local database administrator (DBA).

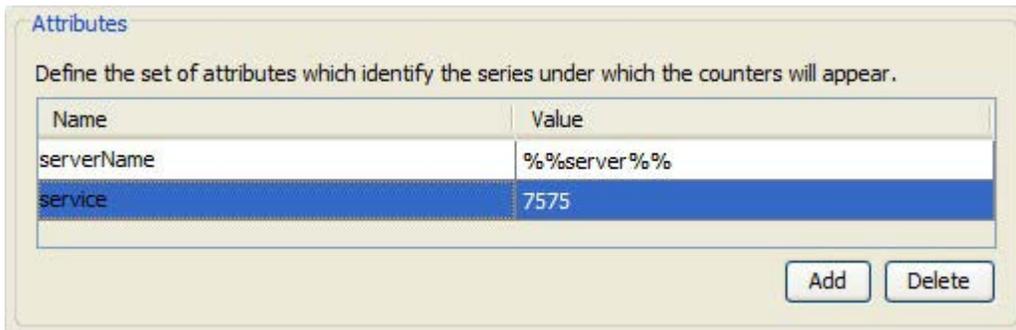
#### 2.2.4.2 Configuring Attributes

Attributes can be used to identify the data that is stored by the counters. The combination of attributes can identify a series of data items because they are grouped together and plotted as a single data series (line) in charts.

By default, the number of attributes is limited to five. (For information about increasing the number of attributes, refer to [Modifying Attribute Limits](#).)

To add an attribute to a Log Measurement action, click **Add** under **Attributes** on the Log Measurement dialog box.

To delete an attribute from a Log Measurement action, select it on the Log Measurement dialog box and click **Delete**.



Attributes are configured with a name and a value, each can be up to 255 characters, including spaces. The value may be text or numeric. Tag values may be used, but remember that counters are grouped by the combined values of the attributes.

**NOTE:** If you use a tagged value which is different for each iteration of the test, none of the counters will be in the same data series in charts (that is, they will all be plotted on separate lines).

---

A typical use for attributes would be to record the source of received messages, (for example, the server or topic/queue name). For example, you could plot results by message source to see how the various parts of the system are performing.

### Modifying Attribute Limits

By default, the number of attributes is limited to five.

If your project database connection is valid, Rational Integration Tester will silently query the attribute limits. If the limits are exceeded, invalid rows will be shown in red and no rows can be edited until the extra ones are removed.

You can increase the number of attributes by modifying a table definition in the database creation script.

The example below is from the Oracle create script. (Some lines of SQL have been omitted for brevity.) The **blue bold** entries show the lines that need to be added to increase the number of attributes by one (ATTR\_ID\_6).

```
CREATE TABLE "LOGMEASUREMENT_MRV" (  
    "ID"          NUMBER(19, 0)    NOT NULL,  
    "ATTR_ID_1"   NUMBER(19, 0),  
    "ATTR_VAL_1"  VARCHAR2(255),  
    ...  
    "ATTR_ID_5"   NUMBER(19, 0),  
    "ATTR_VAL_5"  VARCHAR2(255),  
    "ATTR_ID_6"   NUMBER(19, 0),  
    "ATTR_VAL_6"  VARCHAR2(255),  
    CONSTRAINT "LOGMEASUREMENT_MRV_PK" PRIMARY KEY ("ID"),  
    FOREIGN KEY ("ATTR_ID_1")  
    REFERENCES "COUNTER_DETAILS" ("ID") ON DELETE CASCADE,  
    ...  
    FOREIGN KEY ("ATTR_ID_5")  
    REFERENCES "COUNTER_DETAILS" ("ID") ON DELETE CASCADE,  
    FOREIGN KEY ("ATTR_ID_6")  
    REFERENCES "COUNTER_DETAILS" ("ID") ON DELETE CASCADE  
)  
;
```

---

**NOTE:** The table sizes can be increased even if the project database contains data. If you need help, consult your local database administrator (DBA).

### 2.2.4.3 Configuring UTC Timestamp

A UTC timestamp, which is generated by Rational Integration Tester when the test is executed, is stored with each set of counters.



By default, the timestamp generated by Rational Integration Tester is used (**Use current timestamp** check box). However, you can define a custom timestamp using an existing tag, or create a new tag to use by clicking the new tag  button.

**NOTE:** Performance charts display all data that was saved from the start of the test to the end of the test. If you use a tag whose timestamp values are outside of the test time (that is, before or after), the counter data will not be displayed on the charts.

### 2.2.4.4 Example Log Measurement Action

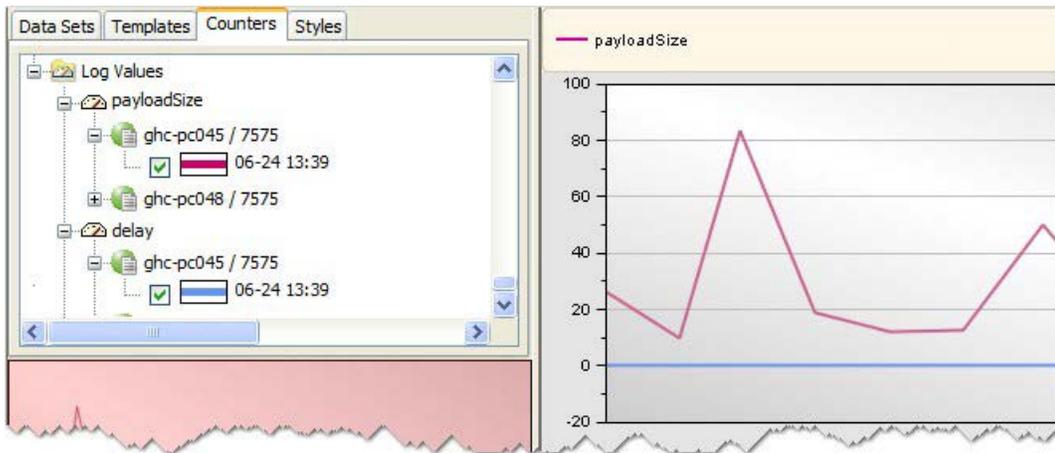
In the following example, two counters have been defined:

- payloadSize
- delay

In addition, two attributes have been used to identify the data:

- host name
- service number

Both counters are shown in the chart configuration tree against each combination of attributes.



Although **service number** remained constant (7575) during the entire test, **host name** had two different values. Therefore, two plots are available for each counter, one for each host.

---

## 2.3 Running Performance Tests

Performance tests are executed in Rational Integration Tester and Rational Performance Test Server in the same way and from the same perspectives as any other test resource.

For more information about executing tests, refer to *IBM Rational Integration Tester Reference Guide*.

### 2.3.1 Initialise/Tear-Down Phase Differences

When running Rational Integration Tester tests outside of a performance test, the Initialise and Tear-down phases are each run only once. However, when a test is run within a performance test, the behavior is slightly different:

- In load-generating tests, the Initialise and Tear-down phases are run once for each test phase on each configured agent.
- In background tests, the Initialise and Tear-down phases run once for each agent because there is no concept of test phases in background tests.

### 2.3.2 Execution Options

There are two options for executing a performance test:

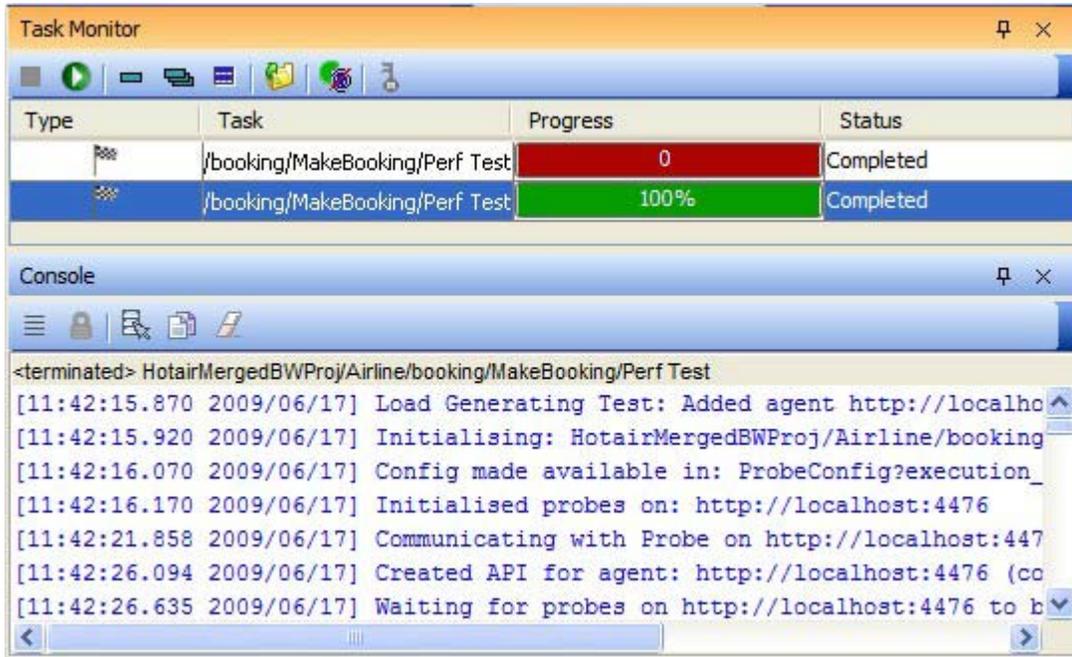
- GUI execution
- Command prompt execution

The following sections describe these options.

---

### 2.3.2.1 GUI Execution

In the Rational Performance Test Server application, tests are executed primarily from the Test Lab perspective and the progress of the test is displayed in the Console view.



More console messages are displayed for performance tests than for standard tests because additional tasks are required, for example, starting agents, starting probes, and so on.

#### Monitoring Test Progress

The following example test console output was generated while running a performance test of 10 transactions per second using two agents: gh-debian1 and ghc-pc048. (Log line timestamps have been omitted to save space.)

```
Load Generating Test: Added agent http://ghc-pc048:4476
Load Generating Test: Added agent http://gh-debian1:4476
Initialising: Tests/ Message Responses
Created API for agent: http://gh-debian1:4476 (communicating on:
http://gh-debian1:53455)
Created API for agent: http://ghc-pc048:4476 (communicating on:
http://ghc-pc048:4663)
```

The transaction rating is listed:

```
Preparing Load Generating Test with target of 10
```

Transactions are divided equally between the agents and the agents are started:

---

Preparing Load Generating Test for execution on http://ghc-pc048:4476 target of 5

Preparing Load Generating Test for execution on http://gh-debian1:4476 target of 5

Task 1 prepared on agent http://ghc-pc048:4476

Task 2 prepared on agent http://gh-debian1:4476

Successfully prepared Load Generating Test

Load Generating Test: starting agent task: 1

Load Generating Test: starting agent task: 2

The test run starts with regular progress updates (details provided in next section):

```
http://ghc-pc048:4476: started (25), passed (135), timed out (0),
failed (0), pending db writes (0)
```

```
http://gh-debian1:4476: started (25), passed (114), timed out (0),
failed (0), pending db writes (0)
```

The test comes to a close:

```
http://ghc-pc048:4476: started (1), passed (1797), timed out (0),
failed (0), pending db writes (0)
```

```
http://gh-debian1:4476: started (0), passed (1773), timed out (0),
failed (0), pending db writes (6)
```

Load Generating Test: all agent tasks completed

Summarising performance test data...

Summarising performance test data completed

[Passed] Tests/Message Responses: completed.

The test has completed and may now be viewed in the Results Gallery.

The following table describes test progress counters.

---

Counter	Description
Started	Iterations started in the report interval, controlled by the <b>Collect statistics every...</b> field on the Performance Test <b>Statistics</b> tab.  The default is five seconds, so if the test is set for 10 transactions per second, this would show a total of 50 each time.
Passed	Iterations passed so far during the test.
Timed Out	Iterations in which message receivers did not get a response within the configured time-out period.
Failed	Iterations failed so far.
Pending DB Writes	Database writes queued on the results database. Large numbers indicate that database access is slower than required and may be a result of a slow network connection. The writes are buffered, however, and do not slow down the test rate.

---

---

**NOTE:** A performance test may run on longer than the specified time while remaining test instances complete and database writes are flushed. In this case, the “started” figure will be at zero for those intervals since the given number of iterations has already been started.

### 2.3.2.2 Command Prompt Execution

Performance tests can also be run from a command prompt by using the `runtests` command (which is located in *<Rational Integration Tester Installation Directory>\bin*).

For more information about command prompt execution, refer to *IBM Rational Integration Tester Reference Guide*.

# Viewing Test Results

## **Contents**

**Working With Data Sets**

**Managing Charts and Chart Templates**

**Modifying Chart Counters**

**Modifying Chart Styles**

This chapter describes how to work with Rational Performance Test Server charts in the Results Gallery perspective.

---

## 3.1 Working With Data Sets

After a performance test has completed, its results can be viewed in the Results Gallery perspective.

Start Time	End Time	Iterations	Iteration Avg Time(ms)	Iteration Max Time(ms)	Transactions	Trans Avg Time(ms)	Trans Max Time(ms)
13:09 19/06/2...	Running						
08:35 18/06/2...	08:36 18/06/2...	150	20	51	0		
08:33 18/06/2...	08:34 18/06/2...	351	5013	5167	0		
15:00 17/06/2...	15:01 17/06/2...	345	3936	5759	0		
11:42 17/06/2...	11:43 17/06/2...	346	3224	5427	0		

You can view results from a single test and from multiple tests on a single chart.

The following sections describe both of these options and how to use the chart window.

---

### 3.1.1 Viewing Results of a Single Test

To view available results for a specific performance test on chart:

1. Click **Browse** and select the test on the Project Resource dialog box.

To clear the selected entry, click **Clear**.

All of the available results for the selected performance test are displayed in the results table, including summary information about each result set (for example, start time, end time, and so on).

**NOTE:** The **End Time** is displayed as “Running” if a test has not yet completed.

If an expected entry does not appear in the table, click **Refresh** to obtain the latest results from the project database.

Each row in the table represents a data set that can be used to plot a chart. To remove one or more data sets from the project database, select them and click **Delete**.

2. To view charts for a data set (in a separate window), double-click the desired data set.

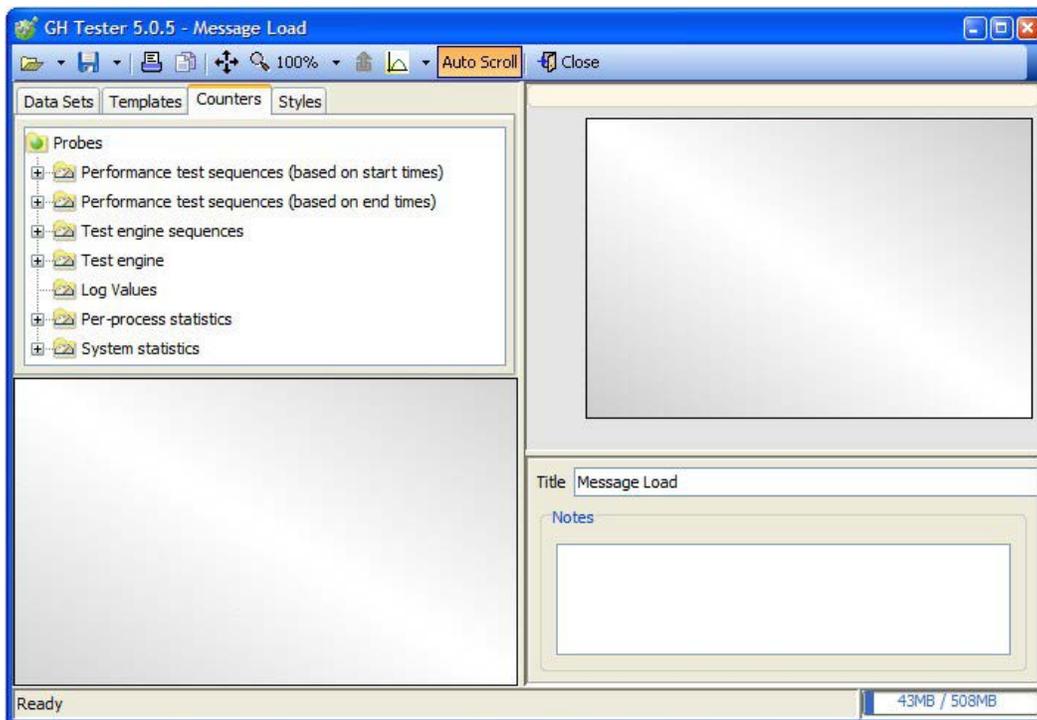
Alternatively, select it and click **Analyse Results**.

Multiple chart windows can be displayed at the same time.

**NOTE:** You can view a chart in real-time if you select it while its status is “Running.”

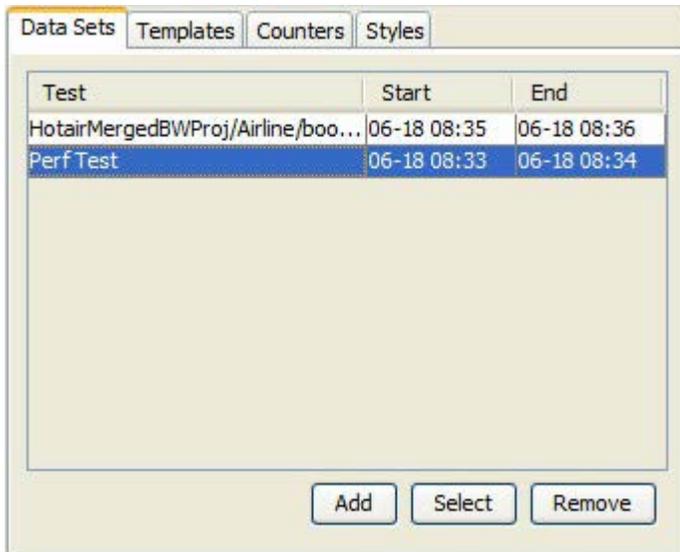
---

Results will be displayed on an empty chart window with the **Counters** tab selected.



- 
3. In the **Title** field, enter a title for the chart.

The selected data sets will appear on the **Data Sets** tab of the chart window.



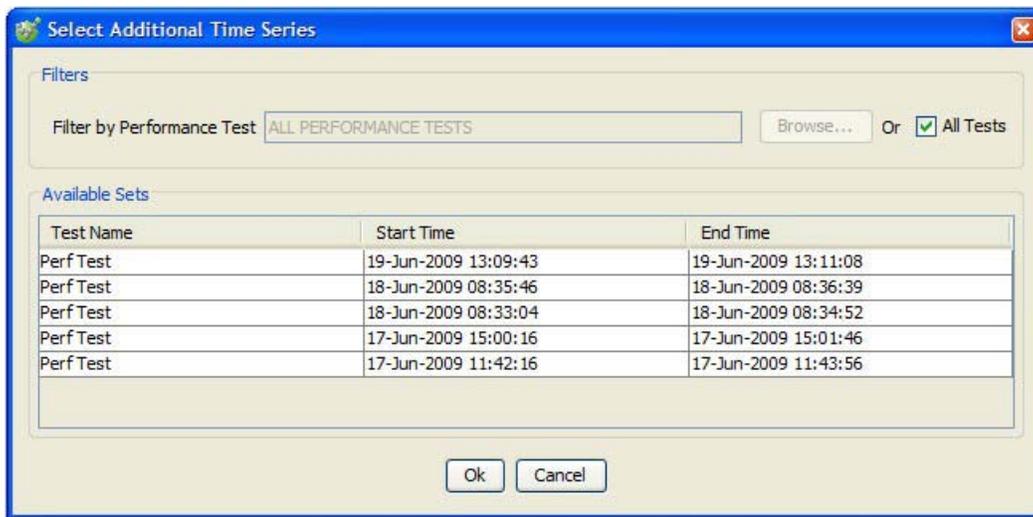
---

### 3.1.2 Viewing Results of Multiple Tests

Additional data sets can be added to a chart, including those from other performance tests. This enables you to plot data from multiple tests together on the same chart.

To add data sets to a chart:

1. On the **Data Sets** tab of the chart window, click **Add**.



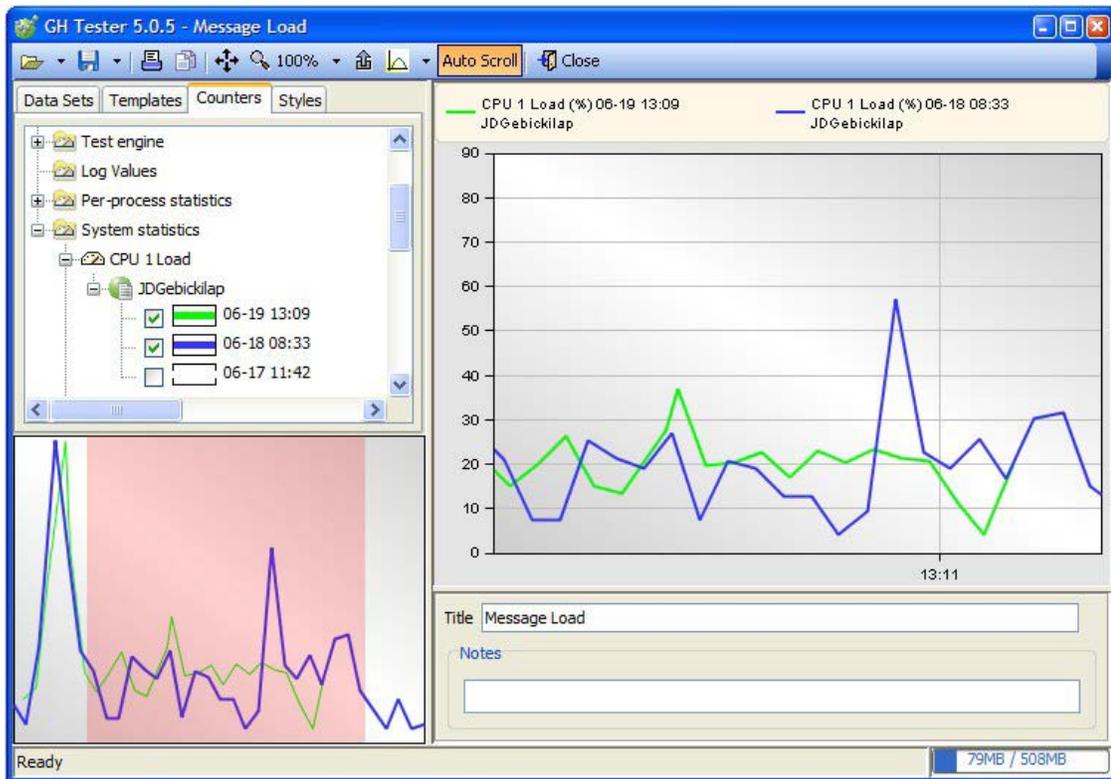
2. Click **Browse** to locate a specific set of results, or select the **All Tests** check box to display all available results.

**NOTE:** Click the column headings to sort the displayed results by that column.

3. Select one or more of the available results sets and click **OK**.

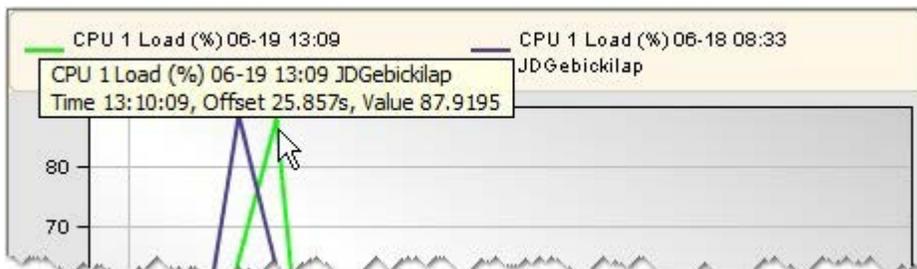
### 3.1.3 Using Chart Windows

Counter data is plotted on the upper right portion of the chart window. When counters are added to the chart, a random color is chosen for each newly plotted line.



The lower left portion of the chart window shows a summary of the complete data set. The shaded area indicates the portion of the data that is visible in the main chart. By dragging the shaded area left or right, the displayed portion of the main chart is panned left or right, accordingly.

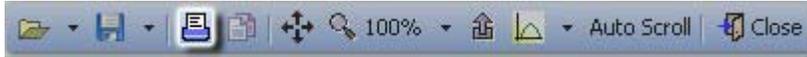
Moving the mouse over a single data point on the chart displays a tooltip that contains details about the underlying data item.



---

### 3.1.3.1 Printing Charts

You can print the current chart by clicking the **Print** icon on the chart window's toolbar.

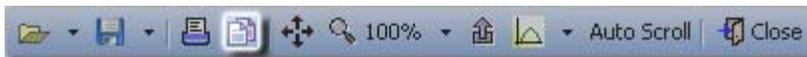


After clicking the icon, the Page Setup dialog box is displayed where you can configure the page layout and header/footer options for the printed chart.

**NOTE:** Only the chart and its legend are printed, along with the current chart title. If the chart contains no data sets it will not print, and if any axis in the chart is empty, that axis will not print.

### 3.1.3.2 Copying Charts to the Clipboard

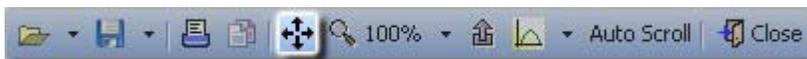
The current chart can be copied to the Clipboard by clicking the **Copy** icon on the chart window's toolbar.



The chart will be copied to the Clipboard as an image. You can then paste the copied image into another application (for example, email, presentation, document, and so on).

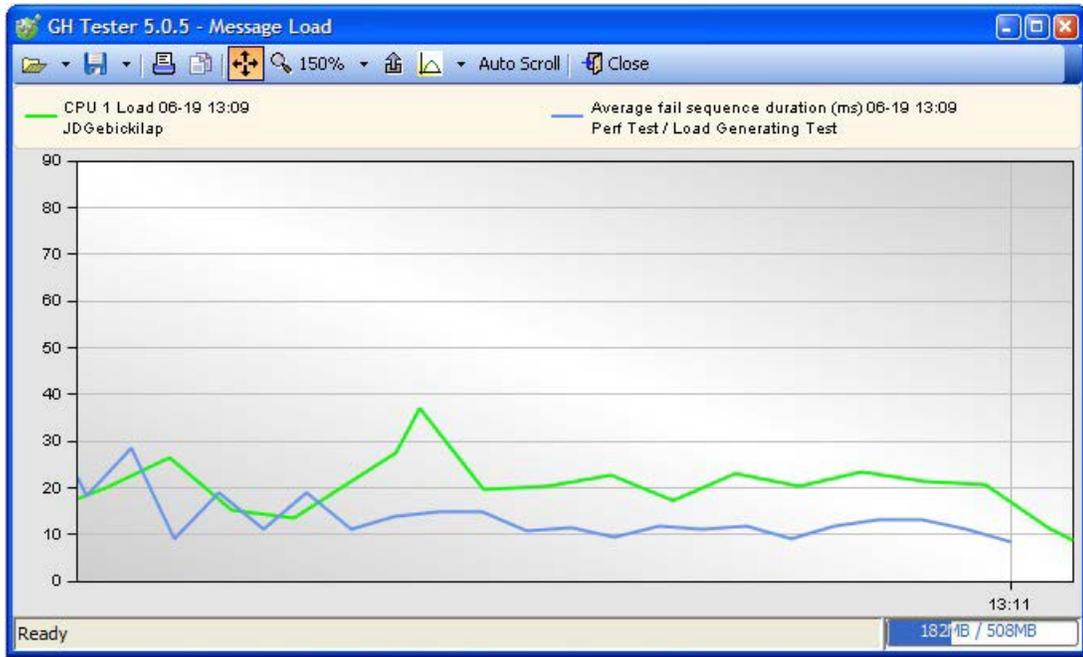
### 3.1.3.3 Maximizing Charts

If desired, you can fill the entire chart window with the current chart, maximizing its view within the currently available space. To maximize the view of the chart, click the **Full screen** icon on the chart window's toolbar.



---

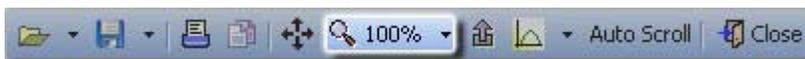
When the chart is maximized, the configuration tabs, overview chart, and chart details will be hidden (as shown below).



To restore the chart to its normal view, click the **Full screen** icon again.

### 3.1.3.4 Zooming In/Out

You can increase or decrease the zoom level of the current chart by using the **Zoom** icon on the chart window's toolbar.

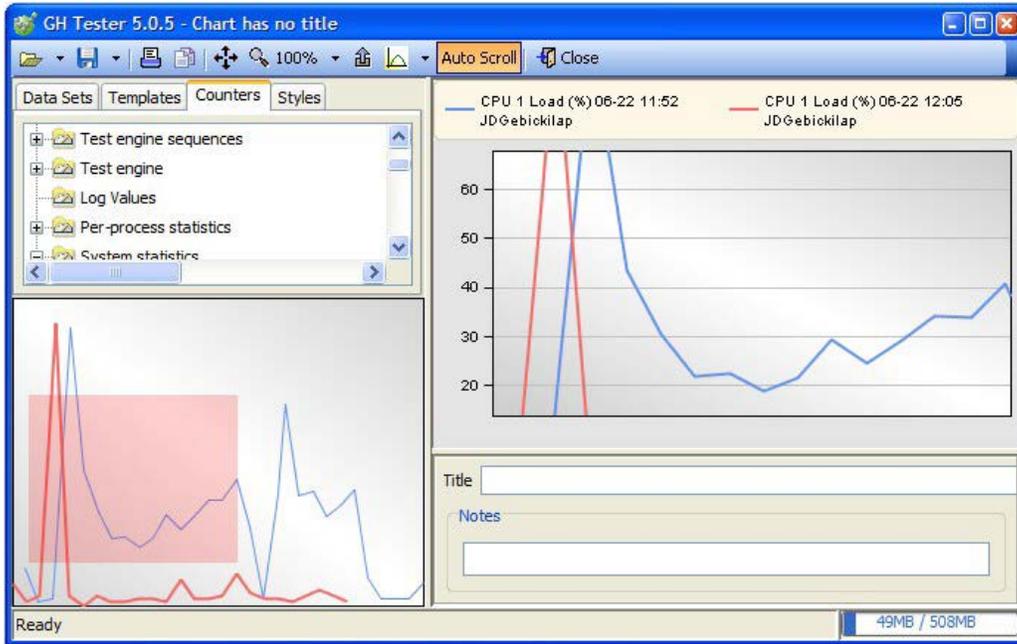


Click the current zoom level to toggle between three preset percentages (that is, 100%, 125%, and 150%), or click the small arrow next to the icon to zoom in or out, or to select one of the specific levels available.

**NOTE:** A zoom level of 100% will display the entire data set for the current chart.

Zooming in beyond 100% will display only a portion of the data set, reflected by the shaded area of the summary chart (below the configuration tabs). Slide the shaded area left or right to display a different portion of the chart.

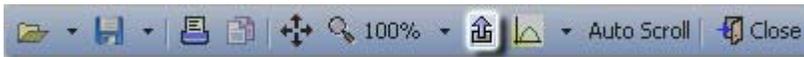
To manually zoom into a specific area of the chart, press and hold the SHIFT key and click and drag over the desired area using the left mouse button. The selected area will fill the chart when the mouse button is released.



You can also right-click the chart and select from the same options that are available when clicking the small arrow next to the **Zoom** icon.

### 3.1.3.5 Exporting Chart Data

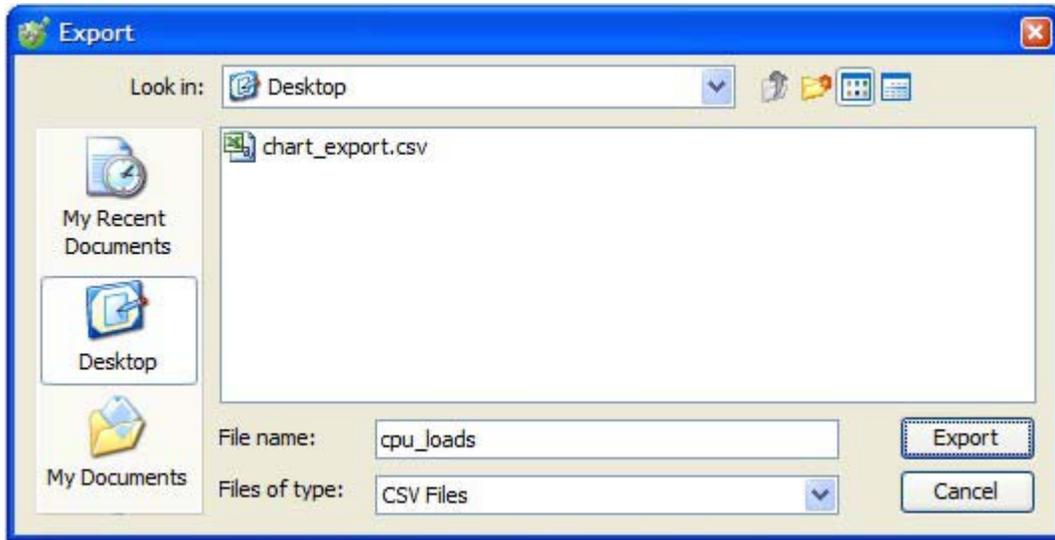
The data in the current chart may be exported to a comma-separated value (CSV) file by clicking the **Export** icon on the chart window's toolbar.



**NOTE:** Only the data in the counters that are currently displayed in the chart will be exported.

---

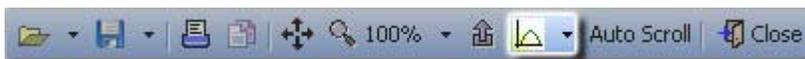
After clicking the icon to export the chart data, select the desired name and location of the file and click **Export**.



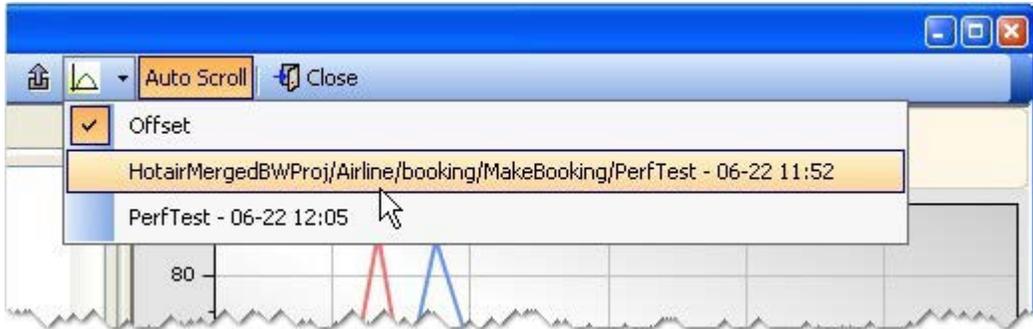
The exported file can be opened in other applications, such as Microsoft Excel, and the data can be manipulated or extracted as desired.

### 3.1.3.6 Configuring the Display of Different Test Times

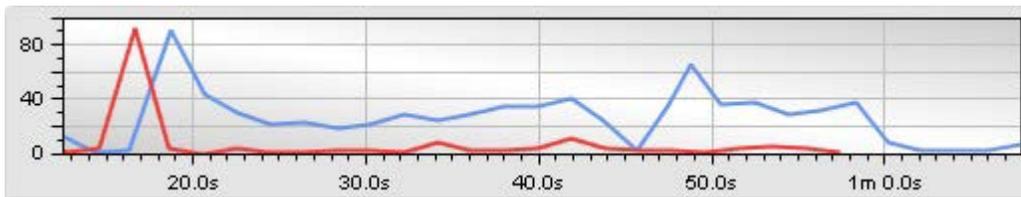
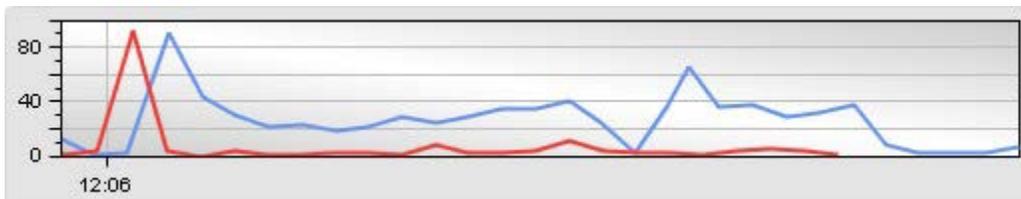
When you add more than one data set to a chart (see [Working With Data Sets](#)), the start of each data set is aligned and plotted together. You can, however, adjust how to display the different test times along the X-axis by using the **Offset** icon on the chart window's toolbar.



If you click the small arrow next to the **Offset** icon, you can select one of the individual tests to display the execution times for that test, or the **Offset** option to display the overall elapsed time starting from zero.

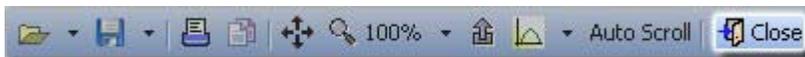


The images below illustrate how the time is displayed when a test is selected (first image) or when the Offset option is selected (second image).



### 3.1.3.7 Closing Charts

To close the current chart window, click the **Close** icon on the chart window's toolbar.

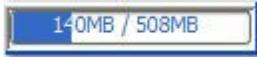


---

### 3.1.4 Managing Memory Usage

When using its charting features, Rational Performance Test Server has to load the data sets from the database into memory. Depending on the duration of the test, the statistics interval and the probes configured, there may be a lot of data.

The current memory usage is indicated in the lower right corner of the chart display.



The amount of memory available to Rational Performance Test Server is fixed, meaning that no more can be allocated if the limit is reached. If necessary, you can quit the application and use the Library Manager application to allocate a larger amount of memory to Rational Performance Test Server (for information about using Library Manger, refer to *IBM Rational Integration Tester Installation Guide*).

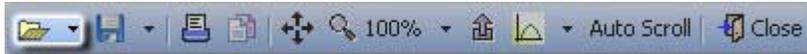
---

## 3.2 Managing Charts and Chart Templates

The following sections describe how to work with charts and chart templates.

### 3.2.1 Opening Charts/Chart Templates

Previously saved charts and chart templates can be opened with the **Open** icon on the chart window's toolbar.



#### 3.2.1.1 Charts

You can open a saved chart by clicking the **Open** icon or by clicking the small arrow next to the icon and selecting **Open chart**.

Charts that are available within the current data set are displayed with a summary image and title. If any description was added to the chart, it will be displayed when the chart is selected.

To select a chart from a different data set, click **Browse** to select another data set or select the **All Sets** check box to show charts in other data sets.

#### 3.2.1.2 Templates

You can open a saved template by clicking the small arrow next to the **Open** icon and selecting **Open template**. Select the desired template from the project resource tree.

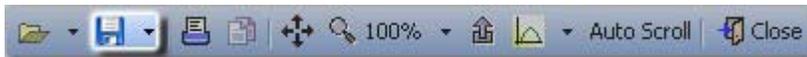
Alternatively, you can view available templates by clicking the **Templates** tab. Templates are listed with a summary image and title. Double-click the desired template to open it.

**NOTE:** When opening a template, the chart styles that it contains will be applied to the current data set.

---

## 3.2.2 Saving Charts/Chart Templates

The current chart can be saved (as a chart or a template) with the **Save** icon on the chart window's toolbar.



### 3.2.2.1 Charts

You can save the current chart by clicking the **Save** icon or by clicking the small arrow next to the icon and selecting **Save chart**.

**NOTE:** The current chart is saved using the title that has been assigned to it. If no title has been assigned, you must create one before saving. If the chart has already been saved, you will be prompted about overwriting the existing chart.

### 3.2.2.2 Templates

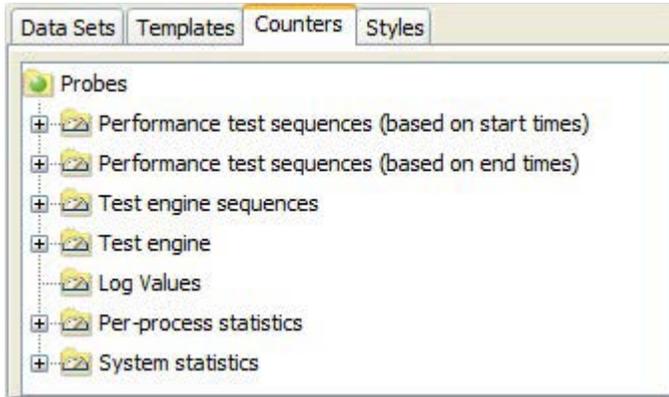
When saving a template, the current chart style (for example, counters used, colors, axis configuration, and so on) is saved and will be applied to a different data set when the template is opened later.

You can save the current chart as a template by clicking the small arrow next to the icon and selecting **Save template**.

---

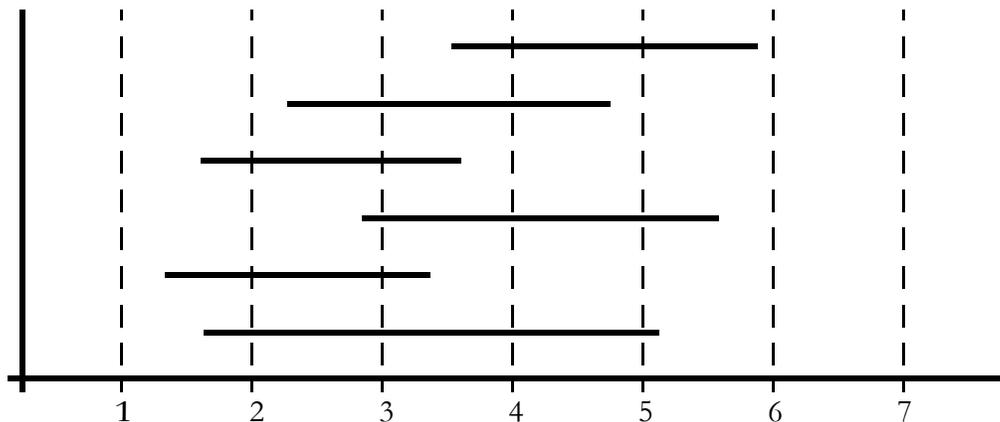
### 3.3 Modifying Chart Counters

The **Counters** tab on the chart windows displays the available counters, grouped by source. Each counter may be plotted individually on the chart.



Performance data will always be available and, depending on the probes that were used in the test, there may also be additional data.

Performance counters are provided for the entire test (start to finish) and for any timed sections within the test. Each of these is available based on start times and end times (that is, start times for iterations started during this period and end times for those ending during this period). The following diagram may help to explain this.



The diagram above shows the execution of several test iterations by time. No iterations start or end in the first time period, so all statistics will be zero. In the second time period, three iterations start, so if you are using counters based on start times, figures will be reported for the three periods. Counters based on end times will yield no data because no iterations end in this period. In time period six, only figures based on end times will be reported because three iterations end here.

The following chart is of a number of test iterations that all take 1000ms to complete. The plots are Sequence started rate and Sequence ended rate, and you can see that the ended rate lags the started rate by exactly 1000ms.

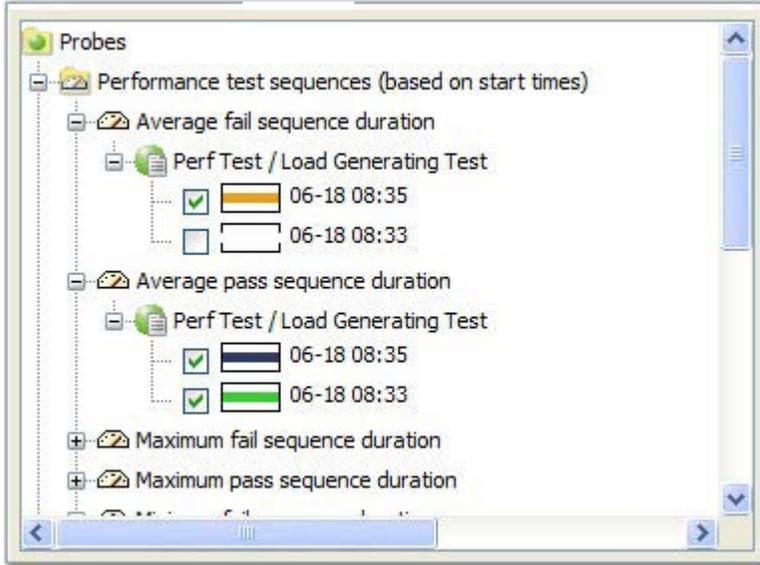


Probe counters are described in [Appendix: Probes Reference](#), but the built-in test counters are listed below.

Counter Name	Description
Average duration	The total time of all iterations (started or ended mode) this period, divided by iterations (in milliseconds).
Maximum duration	The longest iteration time (in milliseconds).
Minimum duration	The smallest iteration time (in milliseconds).
Started rate	The number of iterations started this period (in started mode).
Ended rate	The number of ended iterations (in end times mode).
Passed rate	The number of passed iterations this period.
Failed rate	The number of failed iterations this period.
Time-out rate	The number of tests that failed due to timeouts on received messages.

---

To add counters to a chart, expand the counters tree and select the desired entries. If multiple data sets are selected, they will be displayed in the list, grouped by performance test and test run.



---

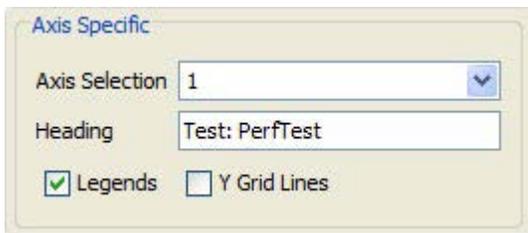
## 3.4 Modifying Chart Styles

The **Styles** tab on the chart window enables you to modify the general appearance of the chart, including general styles and styles that are specific to each axis (when more than one axis is displayed).

The following sections describe these options.

### 3.4.1 Configuring Axis-Specific Styles

For each chart that is displayed, you can add a heading to the chart, toggle the display of the plot legends, and toggle the display of the grid lines that extend from the Y-axis.



The heading, legend, and grid line options will be set for the axis that is selected in the **Axis Selection** list.

### 3.4.2 Configuring General Styles

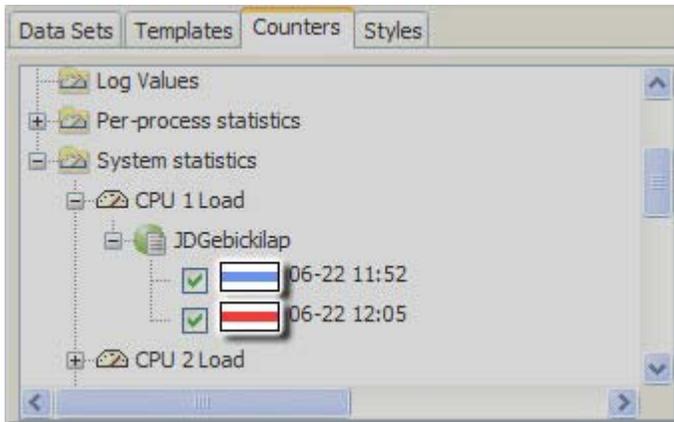
The options under **General Styles** will be applied to **all** displayed charts. You can toggle the display of the grid lines that extend from the X-axis, change the background color of the displayed charts, or toggle between a two- or three-dimensional view of the charts.



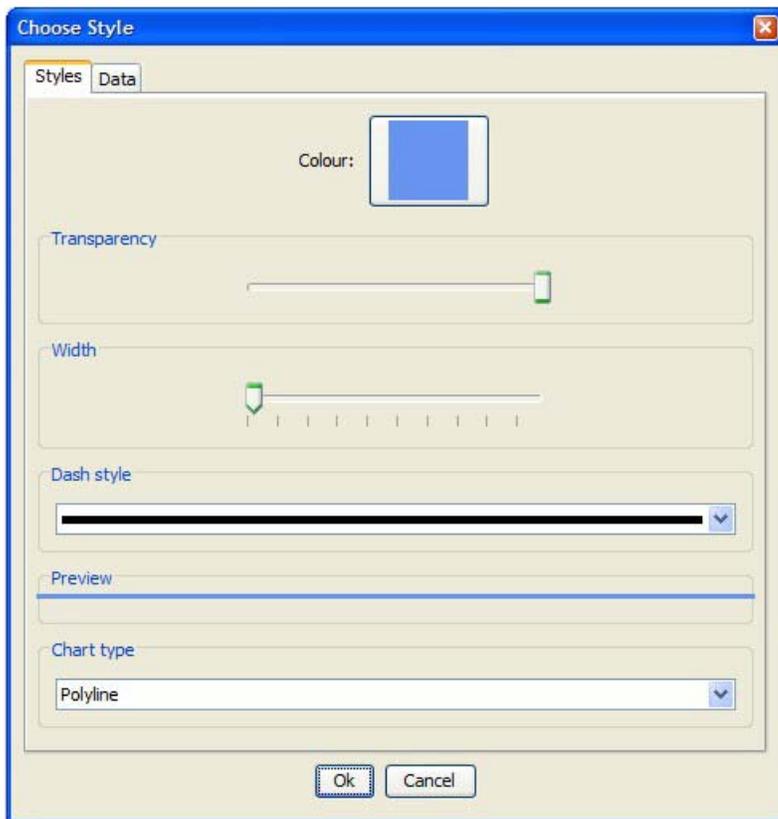
---

### 3.4.3 Configuring Styles Used for Plotting Counters

The styles used to plot each counter can be configured by double-clicking the counter's colored bar in the tree on the **Counters** tab.



After double-clicking the counter's plot line, the Choose Style dialog box is displayed.



---

On the **Styles** tab of the Choose Style dialog box, all aspects of the line and chart style for the selected counter can be modified, as follows:

- **Color** changes the current line color using the Choose stroke color dialog box. After selecting the new color, click **OK** to apply it.
- **Transparency** affects the strength of the plot lines color and whether or not other plot lines will show through the current line. As you vary the transparency, the color and line preview will reflect the change.
- **Width** changes the width or thickness of the selected plot line.
- **Dash style** enables you to select a solid or dashed line for the selected plot.
- **Chart type** enables you to select from five different chart styles, as follows:

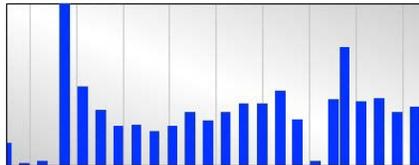
**Polyline**



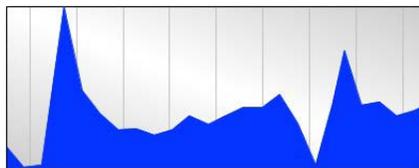
**Stacked Polyline**



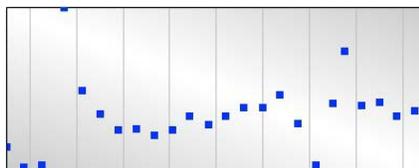
**Bar**



**Area**



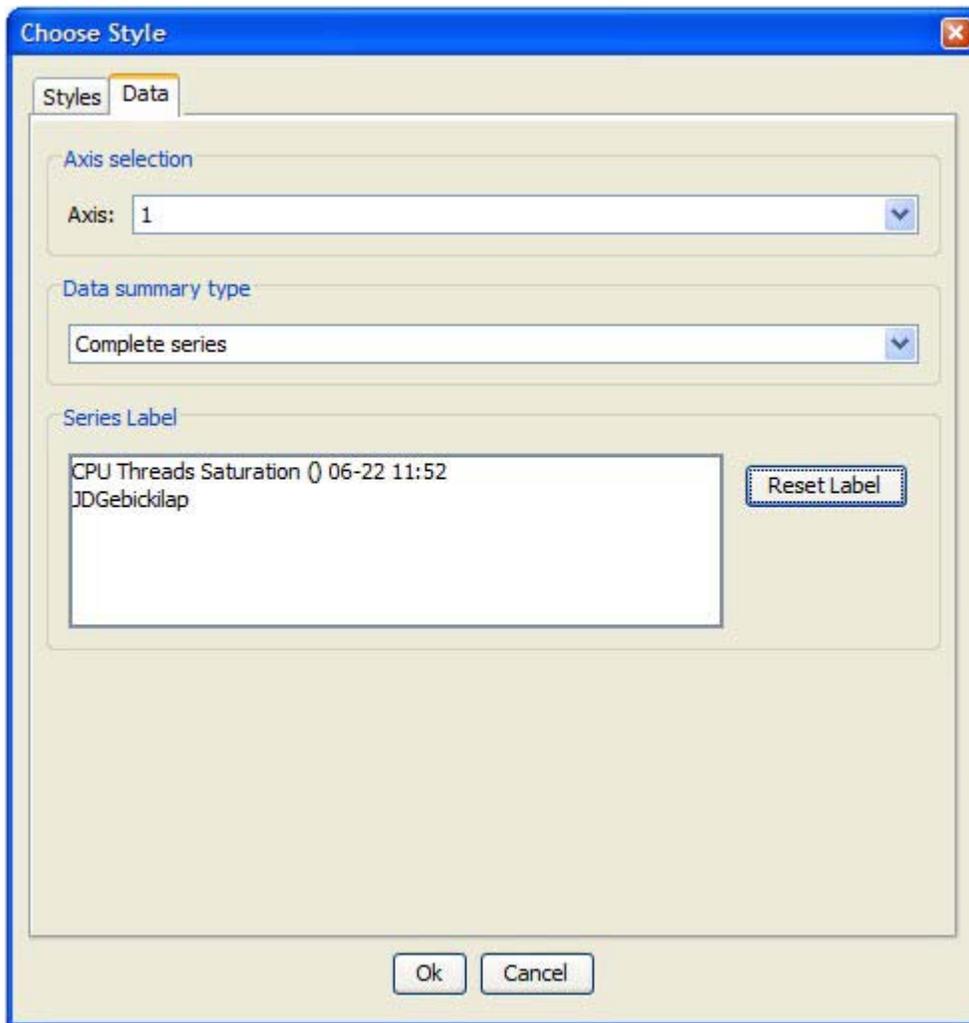
**Scatter**



---

### 3.4.4 Configuring Data Used for Plotting Counters

On the **Data** tab of the Choose Style dialog box, you can configure how the data for the selected counter should be plotted, including plotting the selected line on a new axis or changing the label displayed in the legend for the current line.



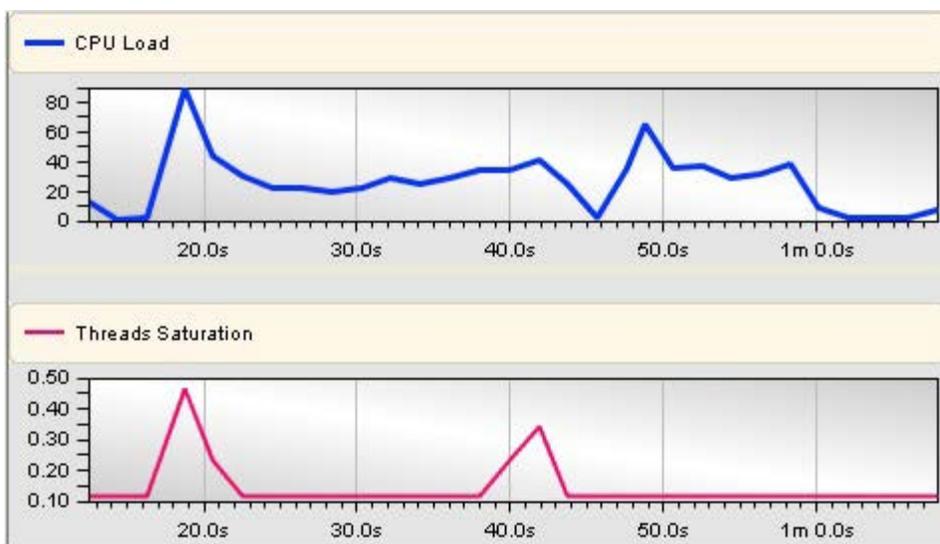
---

### 3.4.4.1 Selecting Axes

If some of the plotted counters have widely varying values (for example, the pass rate is in the tens range, but the duration is in the thousands), you may wish to display one or more counters on its own axis. Otherwise, the large range of numbers on the Y-axis will cause the smaller counters to lose their detail as they are plotted along the origin.

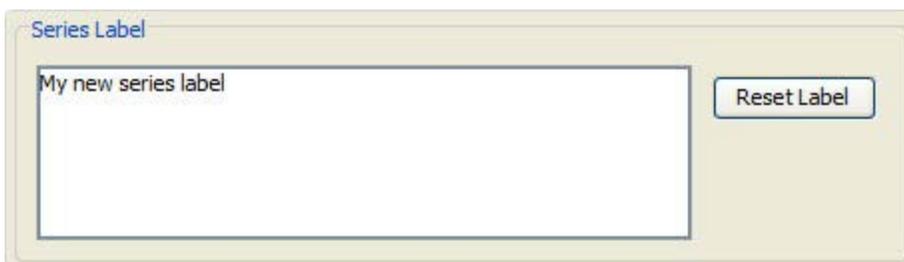
To plot the selected counter on an additional axis, use the **Axis** list to select the axis. You can select one more axis than the number that are currently displayed. That is, if one axis is displayed, you can select **1** or **2**. If two axes are displayed, you can select **1**, **2**, or **3**.

The following example shows two counters displayed on two axes:



### 3.4.4.2 Modifying Series Labels

You can change the label used in the legend for the selected plot line in the **Series Label** field.



To restore the label to its default value, click **Reset Label**.

# Appendix: Probes Reference

## **Contents**

[Introduction](#)

[System Statistics Probe](#)

[Windows Performance Monitor Probe](#)

[TIBCO BusinessWorks Probe](#)

[TIBCO Rendezvous Probe](#)

[TIBCO Rendezvous Distributed Queues Probe](#)

[TIBCO Rendezvous Trace Probe](#)

[TIBCO EMS Probe](#)

[SonicMQ Probe](#)

[webMethods Broker Probe](#)

[webMethods Integration Server Probe](#)

[Probe Error Handling](#)

This appendix describes the probes that can be used with Rational Performance Test Server.

---

## 4.1 Introduction

A Rational Integration Tester probe measures variables from a particular component in the system and reports back its findings for processing and storage.

Probes are started by the agent on the local computer. Therefore, the agent software must be installed on that computer. More than one probe may be run from the same agent, but only **one** performance test controller at a time can use the probes of an individual agent.

The probe process will terminate if it cannot contact the performance test controller that started it by means of the agent. For Windows users, the probe process name is `Probe.exe`. For Linux/Unix users, the probe process is a Java process that may be identified by using long `ps` output.

The probe process configuration file is `Probe.config`, which is located in `<Rational Integration Tester Installation Directory>\config`. This file can be edited to configure the level and location of the logging.

The following sections describe how to use particular Rational Integration Tester probes with Rational Performance Test Server. Probe error handling is also discussed.

**NOTE:** For additional background information about Rational Integration Tester probes, refer to *IBM Rational Integration Tester Reference Guide*. For information about deploying Rational Integration Tester probes, refer to *IBM Rational Integration Tester Installation Guide*.

---

## 4.2 System Statistics Probe

The System Statistics (Systat) probe captures key system statistics for the target host. The following table lists those statistics.

---

<b>Counter</b>	<b>Description</b>
Total CPU load	Percentage average load over all CPUs.
CPU 1 load	Percentage load on a single CPU.
CPU 2 load	Percentage load on a single CPU.
CPU 3 load	Percentage load on a single CPU.
CPU 4 load	Percentage load on a single CPU.
Used memory	Percentage used RAM.
Disk busy %	Percentage of time the hard disk(s) is (are) being read or written to.
Network usage	Percentage of available bandwidth used.
CPU threads saturation	Number of threads waiting for the CPU against total threads.
Memory saturation	Number of pages against number of page faults.
Disk saturation	Disk I/O operations waiting to execute against executed operations.
Network saturation	Rate of network errors for last time period.
Used swap space	Percentage of used swap space.

---

The Systat probe also captures key process statistics for the target host. The following table lists those statistics.

---

<b>Process</b>	<b>Description</b>
Per-process CPU load	Percentage CPU load exerted by a specific process.
Pre-process memory consumption	Memory consumption (in kilobytes) exerted by a specific process.

---

---

**NOTE:** If you want to run the Systat probe on Linux, you must first install Libstatgrab, which is a library that provides cross-platform access to statistics about the system where it is run. For more information about this, refer to *IBM Rational Integration Tester Agent Installation Guide*.

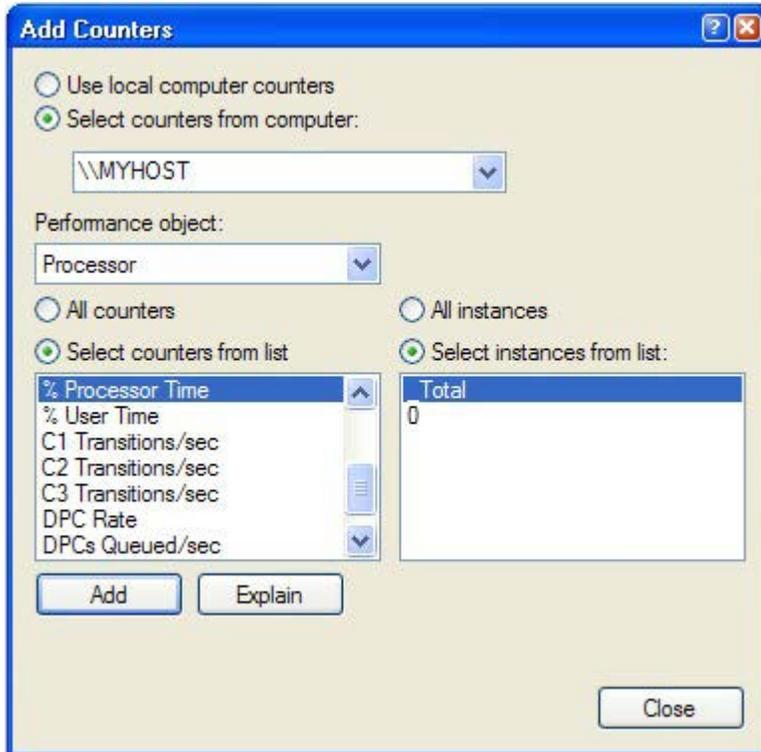
**NOTE:** The Systat probe cannot be used on IBM AIX® systems.

---

## 4.3 Windows Performance Monitor Probe

The Windows Performance Monitor probe captures the same statistics for the target host as those available through the Performance Monitor (`perfmon.msc`) on Windows systems.

The performance objects and counters available in the probe are the same as those available in the Windows Performance Monitor (an example is shown below).



For a description of any of the counters available in the probe, you can click the **Explain** button (shown above) in the Windows Performance Monitor when adding counters.

**NOTE:** The Windows Performance Monitor probe requires the installation of the Microsoft Visual C++ 2010 Redistributable package on each computer that is to be monitored. The package can be downloaded from the Microsoft website.

---

## 4.4 TIBCO BusinessWorks Probe

The TIBCO BusinessWorks (BW) probe accesses the TIBCO Hawk microagents within BusinessWorks to collect execution statistics from running BusinessWorks processes.

The probe creates a Hawk transport to communicate (by means of Rendezvous or EMS) with the systems to be monitored.

The following counters are recorded with the probe:

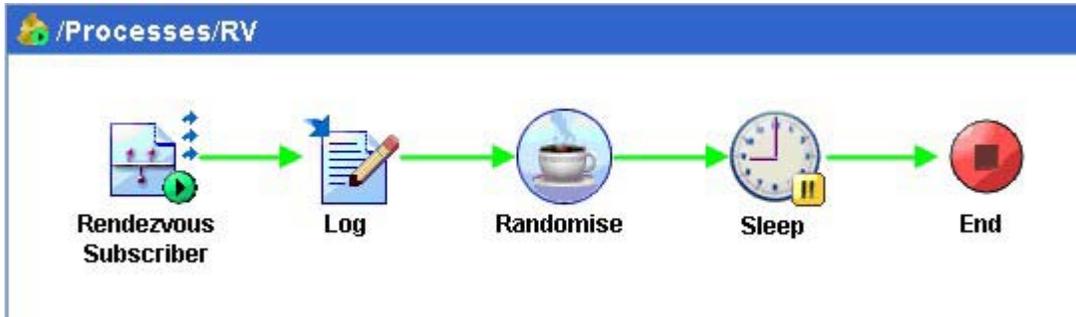
- Average elapsed time (ms)
- Minimum elapsed time (ms)
- Maximum elapsed time (ms)
- Average execution time (ms)
- Minimum execution time (ms)
- Maximum execution time (ms)
- Processes running
- Number aborted
- Number completed
- Number created
- Number queued
- Number suspended
- Number swapped
- Process creation rate

**NOTE:** The TIBCO Hawk JAR files will need to be present on the host computer used to configure and run this probe. This can be To Click Using the Probe

After the TIBCO BusinessWorks probe is configured, it will collect data during a performance test. The counters listed above are split among three different sections in the chart configuration tree.

---

The following is a very simple BusinessWorks process with five activities (including the End) which will be used to illustrate how the counters work.



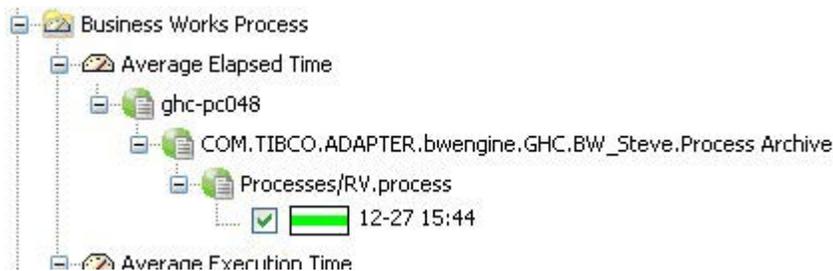
The counters are in the following branches of the chart configuration tree:

- BusinessWorks Process
- Business Works Activities
- Business Works Host

The following sections describe these counters.

#### 4.4.1 Process Counters

Timings are available for the BusinessWorks processes executed during the test time span.

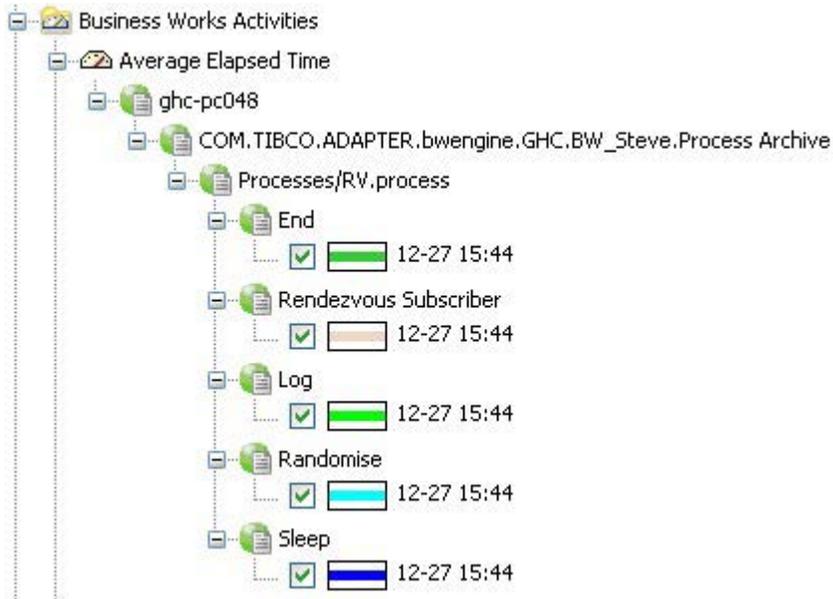


The graphic above shows the Average Elapsed Time counter being selected. This group is ordered by host name then process archive name.

---

## 4.4.2 Activities Counters

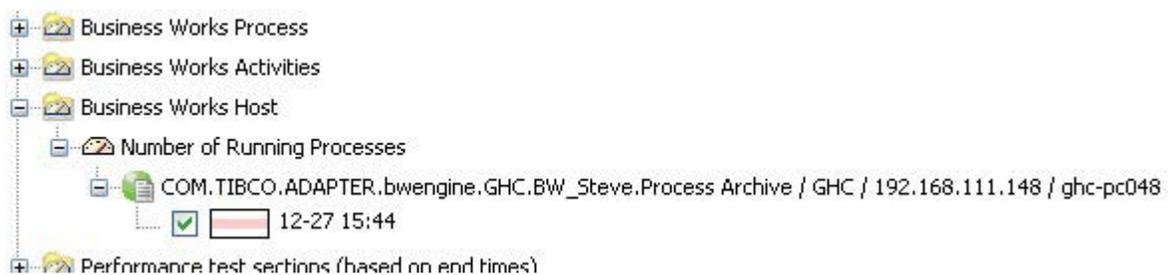
The following graphic shows that timing information is available for each activity in the BusinessWorks process.



**NOTE:** The counters match with the activities shown in the earlier graphic of the process design (in TIBCO Designer).

### 4.4.2.1 Host Counters

The number of running processes is shown for each host computer and the organization in the tree is ordered by BusinessWorks domain name, process archive name, domain name, Hawk domain, and host name.



---

## 4.5 TIBCO Rendezvous Probe

The TIBCO Rendezvous (RV) probe captures Rendezvous daemon (RVD) status messages that contain messaging and error rates. These status messages are published by RVDs every 90 seconds, and they consist of the following values:

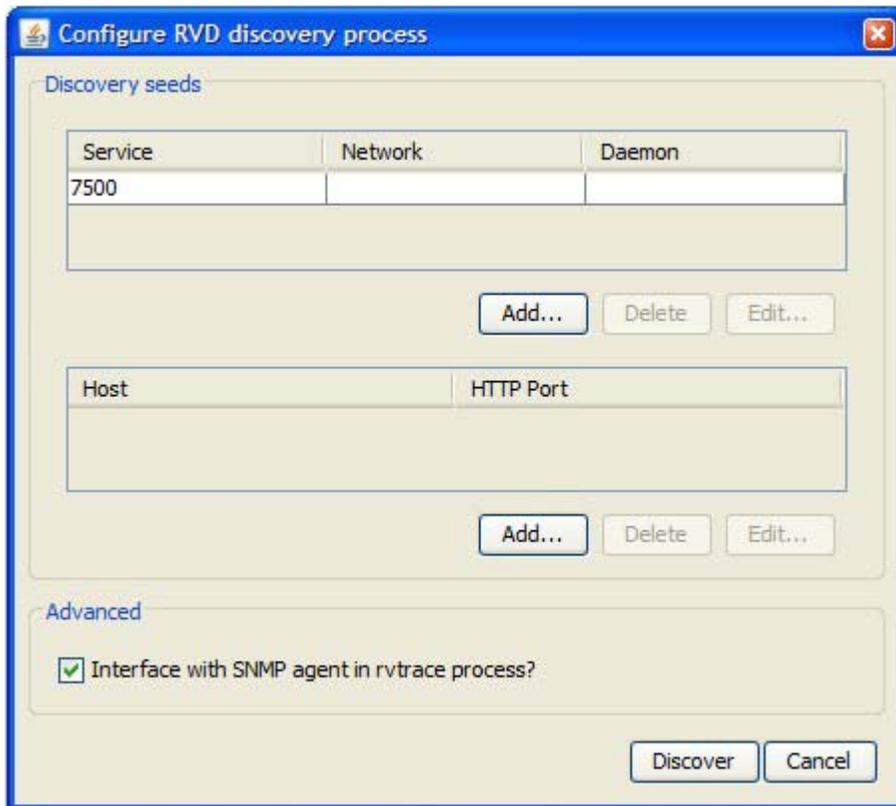
- Messages sent (ms)
- Packets sent (ps)
- Bytes sent (bs)
- Retransmission requests (rx)
- Serial Number
- Version
- HTTP Port
- Process user ID (owner)
- Messages received (mr)
- Packets received (pr)
- Bytes received (br)
- Packets missed (pm)
- Operating System
- HTTP Address
- Uptime (seconds)
- Process ID

**NOTE:** Status messages are published for each transport. To capture a complete picture of your network, you must configure RVDs and services that will capture all relevant data. It is possible to connect to a remote RVD to monitor a remote network but you must ensure that you comply with the licensing agreement. Your licensing may be based on geography, thus preventing you from monitoring multiple locations on a single server.

Using the RVD discovery process, the TIBCO Rendezvous probe can intelligently search for RV services on network RVDs to help you to build a configuration.

---

**NOTE:** The probe is able to start and use `rvtrace` to better discover hosts on the network by selecting the **Interface with SNMP agent in `rvtrace` process** check box. If `rvtrace` is enabled but not configured, a warning message will be displayed. (For more information about `rvtrace`, refer to [Using the `rvtrace` Program](#).)



If `rvtrace` is not used, the probe requires at least one RVD seed. You can listen on RVD transports (top) or RVD administration web pages (bottom) to search for services. Click **Add** in either section to add a transport or RVD administration URL (for example, `localhost:7580`).

To enable `rvtrace`, select the **Interface with SNMP agent in `rvtrace` process** check box.

When ready, click **Discover** to start the discovery process.

The process may take several minutes because the probe must allow enough time to detect all of the RVDs on the network. Progress is updated as the process continues.

---

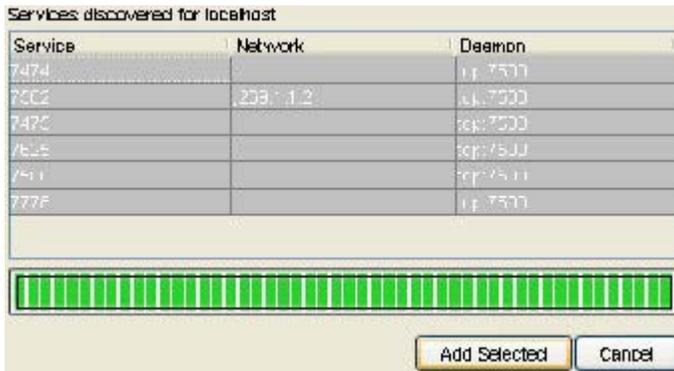
As new services are discovered, subscriptions are created to detect hosts on them, so the completion time may increase to allow enough time (90 seconds) for those advisory messages to be recorded.

The following settings are available to prevent the probe from creating too many simultaneous transports and possibly causing network problems:

- **Maximum active listeners** limits the number of transports that are created.
- **Maximum active listeners per host** limits the number of transports on a given host.

Both settings may be modified while the discovery process is active.

After the process is complete, the list of discovered RVDs is listed. Highlight the ones that you want to use and click **Add Selected**.



The new groups will be added to the configuration.

**NOTE:** The added groups will have been given default names, which you may want to change.

**NOTE:** It is acceptable to have more than one RVD listening on the same service on the same network.

---

## 4.6 TIBCO Rendezvous Distributed Queues Probe

The TIBCO Rendezvous Distributed Queues (RVDQ) probe uses Rendezvous daemon (RVD) transports to collect RV Distributed Queue statistics. It subscribes to the distributed queues (DQ) advisories and stores the values in the database.

You do not need to configure the DQs because they are discovered automatically as they appear on the network. However, as with the TIBCO Rendezvous probe (refer to [TIBCO Rendezvous Probe](#)), you will need to configure the Rendezvous (RV) transports.

The RVDQ probe collects the following data for overall queues and individual members:

- Tasks accepted
- Tasks rejected
- Tasks completed

To delete a probe, select the probe and click **Remove**.

To configure an existing probe, select it and modify its settings in the configuration panel on the right.

For probes other than the default one, you can enter a name in the **Name** field.

---

## 4.7 TIBCO Rendezvous Trace Probe

The TIBCO Rendezvous Trace (RVTRACE) probe uses the TIBCO program `rvtrace` to monitor networks and capture Rendezvous daemon (RVD) subject statistics.

**NOTE:** `rvtrace` is a tool that network administrators can use to monitor and categorize network packets in addition to reporting statistics at regular intervals. Licenses for `rvtrace` are purchased separately from Rendezvous component licenses. For more information about `rvtrace` licensing, contact your local TIBCO representative.

The TIBCO Rendezvous Trace probe connects to one or more running instances of `rvtrace` by means of an SNMP interface and collects data in the following categories:

- Subject
- Point-to-Point
- Multicast
- Network
- Retransmission

For example, the Subject data category contains messaging rates broken down by individual subjects, either as the number of messages or the number of bytes. (For a full description of all data items, refer to TIBCO Rendezvous documentation or visit the TIBCO website.)

By running multiple instances of `rvtrace`, it is possible to monitor more than one network.

**NOTE:** You should check your probe licensing agreement to verify that you are allowed to monitor all of the networks planned because there may be geographic or environment restrictions.

The following section describes how to use the `rvtrace` program.

---

## 4.7.1 Using the rvtrace Program

To force the `rvtrace` program to listen for SNMP connections from the TIBCO Rendezvous Trace probe, run program with the `-snmp` command-line option.

To limit console output, you can run the `rvtrace` program with the `-no-display` command-line option.

By default, `rvtrace` uses UDP port 161 (TIBCO Rendezvous documentation describes how to change this). You need to specify the update interval to match that of the configured probe. The default interval is 10 seconds, but you may want to use a longer interval to limit the amount of data that is stored. The interval is set with the `-u` command-line option.

Therefore, an example command to run `rvtrace` is as follows:

```
rvtrace -u 60 -snmp -no-display
```

The following sections provides some information about running `rvtrace` on Windows and Unix system. (For complete and comprehensive information about using `rvtrace`, refer to the TIBCO Rendezvous documentation or visit the TIBCO website.)

### 4.7.1.1 Windows Systems

When running `rvtrace` on Windows, you first need to install WinPcap, which is included in the Rational Integration Tester/Rational Performance Test Server installation program (for information about this, refer to *IBM Rational Integration Tester Installation Guide*).

The WinPcap software installs DLL files that `rvtrace` requires to capture RVD messages. You also need to ensure that the network interface used is the correct one (specified with the `-i` command-line option). If you do not use the correct interface, `rvtrace` will not collect any data. You can find network device names in the Windows registry or you can run WinDump, which is the Windows version of `tcpdump`, the command-line network analyzer for Unix. (WinDump can be downloaded from the WinPcap website.)

For example, if you run a command of the following format:

```
windump -D
```

The output may look similar to the following:

- 
1. \Device\NPF\_GenericDialupAdapter (Generic dialup adapter)
  2. \Device\NPF\_{AC533C74-32A4-44AD-8675-5A16D50D0170} (3Com EtherLink PCI)

You can use this information when running the `rvtrace` command.

For example:

```
rvtrace -i \Device\NPF_{AC533C74-32A4-44AD-8675-5A16D50D0170} -  
u 60 -snmp -no-display
```

#### 4.7.1.2 Unix Systems

When running `rvtrace` on Unix, no other software is required. However, `rvtrace` must be able to open the network port in promiscuous mode, which requires super user permissions.

Typically, the `rvtrace` program has “set user to root” enabled. If this is true, it may be run as a normal user. (Contact your system administrator if you need assistance or to verify this.)

You must configure `rvtrace` to monitor the correct network port. A list of network interfaces can be displayed by running `ifconfig`.

For example:

```
host% /sbin/ifconfig -a  
  
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232  
index 1 inet 127.0.0.1 netmask ff000000  
  
hme0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu  
1500 index 2 inet 192.168.111.8 netmask ffffffff broadcast  
192.168.111.255
```

In this case, the `hme0` interface will be used when running the `rvtrace` command:

```
rvtrace -i hme0 -u 60 -snmp -no-display
```

---

## 4.8 TIBCO EMS Probe

The TIBCO EMS probe makes a TCP connection to one or more TIBCO EMS servers and periodically extracts messaging statistics for the following categories:

- Topic
- Queue
- Route
- Server

For a full description of all of these data items and the settings for the EMS server reporting intervals, refer to the TIBCO EMS documentation or the `tibemsd.conf` configuration file or visit the TIBCO website.

The TIBCO EMS probe queries each EMS server to determine the statistics interval and uses it when polling for updates.

The following is an excerpt from the `tibemsd.conf` file:

```
server_rate_interval= 60

statistics= enabled

rate_interval= 60

detailed_statistics= NONE

statistics_cleanup_interval= 120

max_stat_memory= 64MB
```

The values for `server_rate_interval` and `rate_interval` are the same. The `server_rate_interval` setting determines how often server statistics are refreshed, and `rate_interval` is used for topics, queues, and routes.

The TIBCO EMS probe uses the value of `server_rate_interval`, and if the value for `rate_interval` is different, data will be lost. If you change these values, you must restart the EMS server.

**NOTE:** The default interval for EMS servers is one second, but this may create a heavy load on the collection process if several EMS servers are configured on the probe.

The TIBCO EMS probe will reconnect to EMS servers automatically if they restart.

---

The following table describes the topic and queue counters provided by the TIBCO EMS probe.

---

<b>Counter</b>	<b>Description</b>
pmc	Pending message count.
pms	Pending messages size (bytes).
imr	Inbound message rate (per second).
omr	Outbound message rate (per second).
ibr	Inbound bytes rate (per second).
obr	Outbound messages size (bytes).
imc	Inbound message count.
obc	Outbound message count.
ibc	Inbound bytes count.
obc	Outbound bytes count.
dc	Count of durable subscribers.
adc	Count of active durable subscribers.
sc	Subscriber count.

---

The following table describes the route counters provided by the TIBCO EMS probe.

---

<b>Counter</b>	<b>Description</b>
imr	Inbound message rate (per second).
omr	Outbound message rate (per second).
ibr	Inbound bytes rate (per second).
obr	Outbound messages size (bytes).
imc	Inbound message count.
obc	Outbound message count.
ibc	Inbound bytes count.
obc	Outbound bytes count.

---

---

The following table describes the route counters provided by the TIBCO EMS probe.

---

<b>Counter</b>	<b>Description</b>
totalConn	Total connections.
durConn	Durable connections.
msgMem	Message memory in use (bytes).
msgMemPooled	Message memory in use by pooling (bytes).
obr	Outbound bytes rate.
ibr	Inbound bytes rate.
omr	Outbound message rate.
imr	Inbound message rate.
pmc	Pending message count.
pms	Pending message size (bytes).
qc	Queue count.
tc	Topic count.

---

**NOTE:** Message rates are calculated over the interval period. If a topic receives a burst of 200 messages and the interval is set to 60 seconds, you would expect to see a message rate of 3 (200/60).

---

## 4.9 SonicMQ Probe

The SonicMQ probe collects statistics for Sonic broker, queue and agent components.

The following table describes the agent/container data that can be monitored from a Domain Manager.

---

<b>Counter</b>	<b>Description</b>
Pool waits	Number of times that transient management tasks had to wait because a pooled thread was not immediately available to service such tasks. Evaluated over the last 30 minutes.
Maximum threads used	Maximum size of thread pool used to service transient management tasks since last metrics reset.
Thread pool size	Size of thread pool used to service transient management tasks.
Total threads used	Total number of threads used by the container and its hosted components.
Maximum memory usage	Maximum heap space used by the container and its hosted components since last metrics reset.
Memory usage	Heap space used by the container and its hosted components.

---

The following table describes the broker data that can be monitored from a Domain Manager.

---

<b>Counter</b>	<b>Description</b>
Application messages received	Application messages received per second (excludes internal/management messages).
Application messages delivered	Application messages delivered per second (excludes internal/management messages).
Bytes received	Broker wide bytes received per second (includes internal/management messages).
Bytes delivered	Broker wide bytes delivered per second (includes internal/management messages).
Connections count	Inbound connection count to broker.
Connections rejected	Rejected connection attempts per minute.
Topic store size	Total size in bytes of topic message store.

---

---

The following table describes the broker connections data that can be monitored from a Domain Manager.

---

<b>Counter</b>	<b>Description</b>
Messages received	Messages received by a connection per second.
Messages delivered	Messages delivered to a connection per second.

---

The following table describes the broker queue data that can be monitored from a Domain Manager.

---

<b>Counter</b>	<b>Description</b>
Messages delivered	Messages delivered to a queue per second, including rejected messages.
Messages received	Messages received by a queue per second.
Maximum messages in queue	Maximum number of messages in a queue during a collection interval.
Message count	Number of messages in a queue.
Message size	Size of messages in a queue.

---

---

## 4.10 webMethods Broker Probe

The webMethods Broker probe collects statistics for webMethods brokers, event types, clients, and client groups.

The following table describes the broker data that can be monitored by the webMethods Broker probe.

---

<b>Counters</b>	<b>Description</b>
Clients	The number of connected and non-connected clients on the broker.
Event Types	The number of event-types installed.
Events Published	The number of events published by all clients.
Events Delivered	The number of events delivered by all clients.
Events Queued	The number of events queued by all clients.
Traces Published	The number of trace events published by the broker.
Max Publishes	The maximum number of publishes.
Max Events	The maximum number of events.
Next Sequence Number	Next operation sequence number.
Retry Attempts	The number of retry attempts.
Event Log Length	The length of the event log.
Reserved Publishes	Total number of reserved publishes.
Guaranteed Publishes	Total number of guaranteed publishes.
Volatile Publishes	Total number of volatile publishes.
Guaranteed Events	Number of guaranteed events.
Volatile Events	Number of volatile events.
Current Publishes	Number of current publishes.
Current Events	Number of current events.
Events In Queue	Number of events placed in the queue.
Last Event Enqueued	Time when the last event was enqueued.
Queue Length	Number of events in the queue.
Queue Size	Number of bytes worth of events in the queue.
Queue Highest Length	Highest value of Queue Length.

---

---

<b>Counters</b>	<b>Description</b>
Queue Highest Length Time	The last time that Queue Highest Length was set.

---

The following table describes the event-type data that can be monitored by the webMethods Broker probe.

<b>Counters</b>	<b>Description</b>
Events Published	The number of events of this type published.
Events Delivered	The number of events of this type delivered.
Last Published	The time when the last event of this type was published.
Last Delivered	The time when the last event of this type was delivered.
Client Subscriptions	The number of subscriptions open which include this event-type.
Forwards Received	The number of events of this type received by means of forwarding from another broker.
Last Forward	The time the last event of this type was received by means of forwarding from another broker.
Publishing Groups	The number of client groups that can publish this event-type.
Subscribing Groups	The number of client groups that can subscribe to this event-type.

---

The following table describes the client data that can be monitored by the webMethods Broker probe.

<b>Counters</b>	<b>Description</b>
Events Published	The number of events published by this client.
Events Delivered	The number of events delivered by this client.
Events Queued	The number of events queued by this client.
Events Retrieved	The number of events retrieved by this client.
Events Unacknowledged	The number of events unacknowledged by this client.
Last Published	The time when the last event was published by this client.
Last Delivered	The time when the last event was delivered by this client.

---

---

<b>Counters</b>	<b>Description</b>
Last Queued	The time when the last event was queued by this client.
Last Retrieved	The time when the last event was retrieved by this client.
Queue Length	The number of events in the client queue.
Queue Byte Size	The number of bytes worth of events in the client queue.
Queue Highest Length	The highest value of Queue Length.
Queue Highest Length Time	The last time that Queue Highest Length was set.

---

The following table describes the client group data that can be monitored by the webMethods Broker probe.

---

<b>Counters</b>	<b>Description</b>
Events Published	The number of events published by member clients.
Events Delivered	The number of events delivered by member clients.
Last Published	Time when the last event was published by a member.
Last Delivered	Time when the last event was delivered by a member.

---

---

## 4.11 webMethods Integration Server Probe

The webMethods Integration Server probe collects statistics for the server and for specific webMethods services.

The following table describes the integration server statistics that can be monitored by the webMethods Integration Server probe.

---

<b>Counter</b>	<b>Description</b>
Current System Threads	The current number of system threads.
Peak System Threads	The peak number of system threads.
Current Service Threads	The current number of service threads.
Peak Service Threads	The peak number of service threads.
Current Total Sessions	The current number of open sessions.
Peak Total Sessions	The peak number of open sessions.
Current Licensed Sessions	The current number of open licensed sessions.
Peak Licensed Sessions	The peak number of open licensed sessions.
Completed Requests	The total number of completed server requests.
Average Request Time	The current average server request time.
Total Average Request Time	The total average server request time.
Service Error Count	The number of service errors registered.
Started Requests per Minute	The number of started requests per minute.
Completed Requests per Minute	The number of completed requests per minute.
Total JVM Memory	The total amount of memory available to the IS JVM.
Used JVM Memory	The current amount of memory used by the IS JVM.
Free JVM Memory	The amount of available memory unused by the JVM.
Server Uptime	The number of seconds this server has been running.
Total Service Invocation Count	The total number of service invocations.
Total Cached Services	The number of services that have been cached.

---

---

<b>Counter</b>	<b>Description</b>
Total Prefetched Services	The number of services that have been prefetched.

---

The following table describes the services data that can be monitored by the webMethods Integration Server probe.

<b>Counter</b>	<b>Description</b>
Invocation Count	The number of times the service has been invoked.

---

---

## 4.12 Probe Error Handling

If a probe has been set to run in a test but Rational Performance Test Server is unable to run it, the following error will be displayed in the test console:

```
Probe definition does not have a Hosting Agent.
```

Usually, this error is caused when the Systat probe has been selected to monitor an infrastructure component (for example, a messaging server) but an agent for the host has not been configured in the Physical View and has not been included in the **Engines** tab of the performance test.

When adding probes to performance tests, users can select parts of the system to be monitored within their environment. The list of available system components to monitor comes from dependencies expressed in Architecture School (for example, a service that depends upon a database). The configuration of these probes in the Physical View must be carried out in the appropriate place.

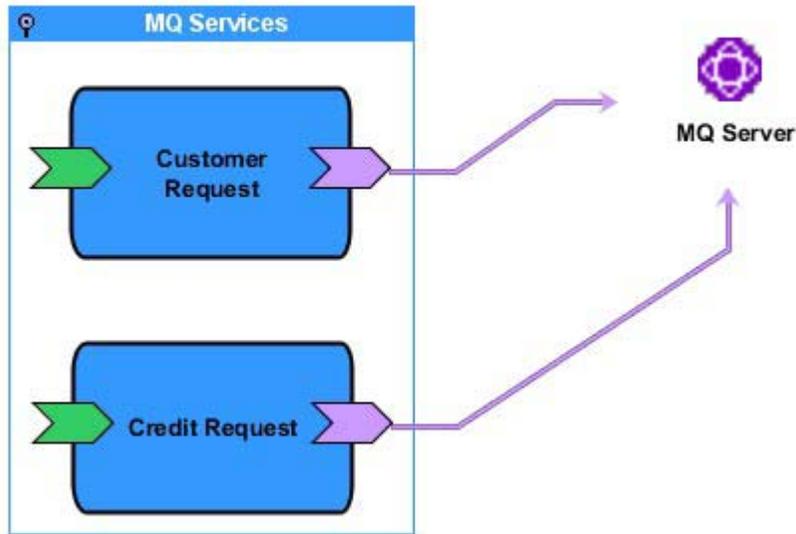
For example, a database server host might be configured to look at system properties, or an EMS broker configuration might be configured to monitor its own statistical data. In both cases, each probe configuration requires a hosting agent (that is, the agent that will execute the probe).

Most probes can be executed on any agent, even if they are not installed on the system under test, because they use remote APIs to capture statistics. However, the Systat probe needs to be installed on the system under test. Therefore, the hosting agent must be configured as that host where the probe is installed.

---

A similar configuration scenario is shown in the following example:

Operations depend on the MQ server in Architecture School's Logical View.

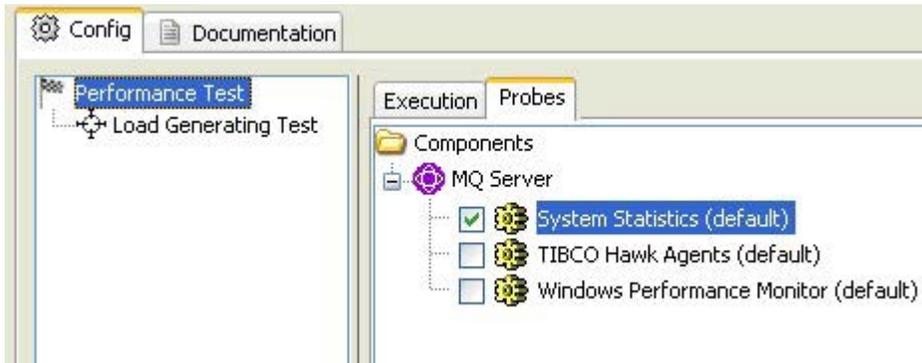


In the Physical View, an agent has been configured on the local testing server, ghc-pc2. The MQ server is on server gh-pc1.



---

A performance test allows probes to be configured for the MQ server.



If the performance test is executed using the agent on `ghc-pc2`, the following hosting agent error message will be displayed:

```
[Error] Error executing test: Probe definition does not have a  
Hosting agent defined: plugin.custom.SysStat
```

The error occurs because the agent on `ghc-pc2` cannot run the Sysstat probe for the MQ server because it is not running in the same location.

To resolve this error, an agent must be added for the MQ server host and included in the performance test.



The next time that the test is executed, successful communication with the probe can be viewed in the test console.

```
Initialised probes on: http://gh-pc1:4476
```

```
Communicating with Probe on http://gh-pc1:4476 via http://gh-  
pc1:59444
```

---

# Glossary

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

---

<b>Term</b>	<b>Description</b>
Agent	A special Rational Integration Tester process running on a host that allows test engine instances and probes to be launched on demand.
Background Test	A test executed on one or more test engines at a constant load level for the duration of the performance test.
Counter	An individual measurement from part of the system, examples include messages per second and CPU utilization.
Load Generating Test	A test that is executed by one or more test engines which may have varying load characteristics.
Host	The computer on which a software process runs.
JMS	Java Message Service, a J2EE technology. Several implementations of JMS exist, for example, IBM WebSphere® MQ, TIBCO EMS, and SonicMQ.
Performance Test Controller	Process that deploys probe and test configuration and orchestrates the performance test during execution. Communicates with Agents to achieve its objectives.
Probe	Measures information from part of the system and exposes it as one or more counters.
Result Set	The results of a performance test execution. One of these is generated every time a performance test is executed.
Server	A host computer on a network shared by more than one user.
Test Engine Instance	An instance of the Rational Integration Tester test engine, started by an agent, to execute a series of tests.

---

---

---

<b>Term</b>	<b>Description</b>
Transport	Informally, the messaging software in use (for example, JMS, TIBCO Rendezvous, TIBCO ActiveEnterprise, IBM WebSphere MQ, and so on).

---

---

# 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.  
1623-14, Shimotsuruma, Yamato-shi  
Kanagawa 242-8502 Japan

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

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

---

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

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

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

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

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

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

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

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

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

---

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

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

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

#### COPYRIGHT LICENSE:

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

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

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

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

---

## Trademarks and service marks

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

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

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

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

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

**IBM**<sup>®</sup>