IBM

# IBM Rational Service Tester for SOA Quality
**10.1.1 Documentation**
**September 2020**

# Special notice

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

# Contents

# Chapter 1. Release notes for IBM® Rational® Service Tester for SOA Quality

This document contains information about what's new, installation, and known issues in IBM® Rational® Service Tester for SOA Quality and contact information of IBM Customer Support.

**Contents**

## Product description

You can find the description of Rational® Service Tester for SOA Quality.

Rational® Service Tester for SOA Quality is a scripting-free environment for automating load and scalability testing of web, ERP, and server-based software applications. Rational® Service Tester for SOA Quality provides rich and customizable reporting to help you identify the presence and cause of system bottlenecks. It captures the network traffic that is rendered when the application under test interacts with a server. This network traffic is then emulated on multiple virtual users while playing back the test. See the Product overview on page 36.

## What's new

You can find information about the features introduced in this release of Rational® Service Tester for SOA Quality.

- **Enhancements to Rational® Service Tester for SOA Quality integrations**

    ◦ You can now use the Resource Monitoring Override Labels parameter when you choose to run tests from other applications such as Jenkins, Ant, Maven, Micro Focus Application Lifecycle Management (ALM), Azure DevOps, and UrbanCode Deploy (UCD). See Integration with other applications on page 75.

    ◦ After you configure the Rational® Test Automation Server URL and if the **Publish result after execution** field is set to **Always**, then when you run tests from other applications, you can now view hyperlinks that point to the corresponding test reports as follows:

| Applications | Location of the report |
|---|---|
| Jenkins | Console page |
| Ant | Log file |
| Maven | Console page |

11

| Applications | Location of the report |
|---|---|
| Micro Focus Application Lifecycle Management | Output window |
| Azure DevOps | Task page |
| UrbanCode Deploy | Output log |

- **Bug fixes**

    Defects reported by clients and internal defects are fixed.

## Installing the product

You can find information about the installation and upgrade instructions for  Rational® Service Tester for SOA Quality.

To download the product from IBM® Passport Advantage®, you must follow the instructions provided in the download document at  Rational® Service Tester for SOA Quality V10.1.1.

For installation instructions, see Installation of the product by using Installation Manager on page 58.

📝 **Notes:**

- You can upgrade to 10.1.1 from 10.1.0 of the product. After you installed  Rational® Service Tester for SOA Quality 10.1.1, at any point in time if you want to use 10.1.0, you can roll back from 10.1.1 to 10.1.0.

- If you installed  Rational® Service Tester for SOA Quality that is earlier than 10.1.0, you cannot upgrade the product to the latest version. You must uninstall the existing version of the product before installing the latest version of the product.

- After you installed  Rational® Service Tester for SOA Quality 10.1.1, at any point in time if you want to use the product that is earlier than 10.1.0, you cannot roll back to the previous version of the product. You must uninstall  Rational® Service Tester for SOA Quality 10.1.1, and then install the required version of the product.

## Known issues

You can find information about the known issues identified in this release of  Rational® Service Tester for SOA Quality.

**Table 1.**

| Product | Download document | Knowledge Base |
| --- | --- | --- |
| Rational® Service Tester for SOA Quality | Release document | Knowledge articles |
|  |  |  |

Known problems are documented in the download document for each product and in the form of individual technotes in the Support Knowledge Base:

The knowledge base is continually updated as problems are discovered and resolved. By searching the knowledge base, you can quickly find workarounds or solutions to problems.

## Contacting IBM Rational Software Support

You can find information about IBM technical support assistance for  Rational® Service Tester for SOA Quality.

- For contact information and guidelines or reference materials that you might need when you require support, read the IBM Support Guide.

- For personalized support that includes notifications of significant upgrades, subscribe to Product notification.

- Before you contact IBM Rational Software Support, you must gather the background information that you might need to describe your problem. When you describe a problem to an IBM software support specialist, be as specific as possible and include all relevant background information so that the specialist can help you solve the problem efficiently. To save time, know the answers to these questions:
    ◦ What software versions were you running when the problem occurred?
    ◦ Do you have logs, traces, or messages that are related to the problem?
    ◦ Can you reproduce the problem? If so, what steps do you take to reproduce it?
    ◦ Is there a workaround for the problem? If so, be prepared to describe the workaround.

# Chapter 2. System Requirements

This document includes information about hardware and software requirements for Rational® Service Tester for SOA Quality.

**Contents**

**Disclaimers**

This report is subject to the Terms of Use (https://www.ibm.com/legal/us/en/) and the following disclaimers:

The information contained in this report is provided for informational purposes only. While efforts were made to verify the completeness and accuracy of the information contained in this publication, it is provided AS IS without warranty of any kind, express or implied, including but not limited to the implied warranties of merchantability, non-infringement, and fitness for a particular purpose. In addition, this information is based on IBM's current product plans and strategy, which are subject to change by IBM without notice. IBM shall not be responsible for any direct, indirect, incidental, consequential, special or other damages arising out of the use of, or otherwise related to, this report or any other materials. Nothing contained in this publication is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software.

References in this report to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. Product release dates and/or capabilities referenced in this presentation may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way. The underlying database used to support these reports is refreshed on a weekly basis. Discrepancies found between reports generated using this web tool and other IBM documentation sources may or may not be attributed to different publish and refresh cycles for this tool and other sources. Nothing contained in this report is intended to, nor shall have the effect of, stating or implying that any activities undertaken by you will result in any specific sales, revenue growth, savings or other results. You assume sole responsibility for any results you obtain or decisions you make as a result of this report.

Notwithstanding the Terms of Use (https://www.ibm.com/legal/us/en/), users of this site are permitted to copy and save the reports generated from this tool for such users own internal business purpose. No other use shall be permitted.

# Hardware

You can find information about the hardware requirements for  Rational® Service Tester for SOA Quality.

**Contents**

**AIX**

| Hardware | Components | Requirement | Notes |
|---|---|---|---|
| Disk space | Rational® Performance Tester Agent | 10 GB | • Disk space requirements can be reduced or increased depending on the features that you install. <br><br> • Large test runs can store several gigabytes of data. Make sure that you have adequate disk space before attempting a large test run. |
| Memory | Rational® Performance Tester Agent | 8 GB | |
| Processor | Rational® Performance Tester Agent | 1.86 GHz Intel Pentium 4 or higher | |

| Hardware | Components | Requirement | Notes |
|---|---|---|---|
|  |  |  | • For best results with large test runs, use 16 GB of RAM. <br><br> • For best results with large test runs, use a 2.30 GHz or higher Intel Core 2 Duo processor. |

**Linux**

| Hardware | Components | Requirement | Notes |
|---|---|---|---|
| Disk space | • Rational® Performance Tester Agent <br><br> • Rational® Service Tester for SOA Quality | 10 GB | • Disk space requirements can be reduced or increased depending on the features that you install. <br><br> • An additional 500 MB of disk space is required in the /tmp directory. |
| Memory | • Rational® Performance Tester Agent <br><br> • Rational® Service Tester for SOA Quality | 8 GB | • For best results with large test runs, use 16 GB of RAM. |
| Processor | • Rational® Performance Tester Agent <br><br> • Rational® Service Tester for SOA Quality | Minimum: 1.86 GHz Intel Pentium 4 or higher | • For best results with large test runs, use a 2.30 GHz or higher Intel Core 2 Duo processor. |

## Mac

| Hardware | Components | Requirement | Notes |
|---|---|---|---|
| Disk space | • Rational® Performance Tester Agent<br><br>• Rational® Service Tester for SOA Quality | 10 GB | • Disk space requirements can be reduced or increased depending on the features that you install.<br><br>• Large test runs can store several gigabytes of data. Make sure that you have adequate disk space before attempting a large test run. |
| Memory | • Rational® Performance Tester Agent<br><br>• Rational® Service Tester for SOA Quality | 8 GB | • For best results with large test runs, use 16 GB of RAM. |
| Processor | • Rational® Performance Tester Agent<br><br>• Rational® Service Tester for SOA Quality | Minimum: 1.86 GHz Intel Pentium 4 or higher | • For best results with large test runs, use a 2.30 GHz or higher Intel Core 2 Duo processor. |

## Windows

| Hardware | Components | Requirement | Notes |
|---|---|---|---|
| Disk space | • Rational® Performance Tester Agent<br><br>• Rational® Service Tester for SOA Quality | 10 GB | • Disk space requirements can be reduced or increased depending on the features that you install.<br><br>• Large test runs can store several gigabytes of data. Make sure that you have adequate disk space before attempting a large test run. |

| Hardware | Components | Requirement | Notes |
|---|---|---|---|
| | | | • Additional disk space is required if you use FAT32 instead of NT-FS. An additional 500 MB of disk space is required in the directory that you specify in the environment variable TEMP. |
| Memory | • Rational® Performance Tester Agent<br><br>• Rational® Service Tester for SOA Quality | 8 GB | • For best results with large test runs, use 16 GB of RAM. |
| Processor | • Rational® Performance Tester Agent<br><br>• Rational® Service Tester for SOA Quality | Minimum: 1.86 GHz Intel Pentium 4 or higher | • For best results with large test runs, use a 2.30 GHz or higher Intel Core 2 Duo processor. |

# Operating systems

You can find the operating systems that are supported, organized by operating system family for  Rational® Service Tester for SOA Quality.

**Contents**

**AIX**

| Operating system | Hardware | Bitness | Components | | Notes |
|---|---|---|---|---|---|
| | | | Desktop | Agent | |
| AIX 7.2 | POWER System - Big Endian | 32 | ✖ | ✔ | |
| AIX 7.1 | POWER System - Big Endian | 32 | ✖ | ✔ | |

**Linux**

| Operating system | Hardware | Bitness | Components | | Notes |
|---|---|---|---|---|---|
| | | | Desktop | Agent | |
| Red Hat Enterprise Linux (RHEL) 8.1 | x86-64 | 64-Exploit | ✔ | ✔ | |
| Red Hat Enterprise Linux (RHEL) 8 | x86-64 | 64-Exploit | ✔ | ✔ | |
| Red Hat Enterprise Linux (RHEL) Client 7 | x86-64 | 64-Exploit | ✔ | ✔ | |
| Red Hat Enterprise Linux (RHEL) Workstation 7 | x86-64 | 64-Exploit | ✔ | ✔ | |
| Red Hat Enterprise Linux (RHEL) Server 7 | x86-64 | 64-Exploit | ✔ | ✔ | |
| Ubuntu 20.04.1 LTS | x86-64 | 64-Exploit | ✔ | ✔ | |
| Ubuntu 18.04 LTS | x86-64 | 64-Exploit | ✔ | ✔ | |

## Mac

| Operating system | Hardware | Bitness | Components | | Notes |
| --- | --- | --- | --- | --- | --- |
| | | | Desktop | Agent | |
| macOS Mojave 10.15 | x86-64 | 64-Exploit | ✔ | ✔ | |
| macOS Catalina 10.14 | x86-64 | 64-Exploit | ✔ | ✔ | |

## Windows

| Operating system | Hardware | Bitness | Components | | Notes |
| --- | --- | --- | --- | --- | --- |
| | | | Desktop | Agent | |
| Windows 10 Enterprise | x86-64 | 32, 64-Exploit | ✔ | ✔ | |
| Windows 10 Pro | x86-64 | 32, 64-Exploit | ✔ | ✔ | |
| Windows 7 Enterprise | x86-32, x86-64 | 32, 64-Exploit | ✔ | ✔ | |
| Windows 7 Professional | x86-32, x86-64 | 32, 64-Exploit | ✔ | ✔ | |
| Windows 7 Ultimate | x86-32, x86-64 | 32, 64-Exploit | ✔ | ✔ | |
| Windows Server 2019 | x86-64 | 32, 64-Exploit | ✔ | ✔ | |
| Windows Server 2016 | x86-64 | 32, 64-Exploit | ✔ | ✔ | |

# Host prerequisites

You can find the host prerequisites that support the operating capabilities for Rational® Service Tester for SOA Quality.

**Contents**

**Licensing**

| Prerequisite | Version | Components | | Notes |
|---|---|---|---|---|
| | | **Desktop** | **Agent** | |
| Rational License Key Server | 8.1.5 and future fix packs | ✔ | Not applicable | |
| | 8.1.4 and future fix packs | ✔ | Not applicable | |
| | 8.1.3 and future fix packs | ✔ | Not applicable | |

**Terminal services**

| Supported software | Version | Components | | Notes |
|---|---|---|---|---|
| | | **Desktop** | **Agent** | |
| Microsoft Windows Server | 2003 Terminal Services and future fix packs | ✔ | Not applicable | For remote terminal access |
| Citrix XenApp | | ✔ | Not applicable | |

| Supported software | Version | Components | | Notes |
|---|---|---|---|---|
| | | Desk-top | Agent | |
| XenDesktop | | ✔ | Not applica-ble | |

## Virtualization Management

| Prerequisite | Version | Components | | Notes |
|---|---|---|---|---|
| | | Desktop | Agent | |
| Docker Community Edition (CE) | 19.3.8 and fu-ture fix packs | ✔ | ✔ | |
| Docker Compose | 1.25.5 and fu-ture fix packs | ✔ | ✔ | |

## Web Browsers

The following versions of web browsers support the viewing of performance reports. See to know the browsers that are supported to record the HTTP tests.

| Prerequisite | Version | Components | | Notes |
|---|---|---|---|---|
| | | Desktop | Agent | |
| Apple Safari | 13 and future fix packs | ✔ | Not applicable | |
| | 12 and future fix packs | ✔ | Not applicable | |
| Google Chrome | 84 and future fix packs | ✔ | Not applicable | |

| Prerequisite | Version | Components | | Notes |
| --- | --- | --- | --- | --- |
| | | **Desktop** | **Agent** | |
| | 83 and future fix packs | ✔ | Not applicable | |
| | 81 and future fix packs | ✔ | Not applicable | |
| | 80 and future fix packs | ✔ | Not applicable | |
| | 79 and future fix packs | ✔ | Not applicable | |
| | 78 and future fix packs | ✔ | Not applicable | |
| Microsoft Edge | Latest release | ✔ | Not applicable | |
| | 44 and future fix packs | ✔ | Not applicable | |
| | 42 and future fix packs | ✔ | Not applicable | |
| Microsoft Internet Explorer | 11 and future fix packs | ✔ | Not applicable | |
| Mozilla Firefox | 79 and future fix packs | ✔ | Not applicable | |
| | 77 and future fix packs | ✔ | Not applicable | |

| Prerequisite | Version | Components | | Notes |
| --- | --- | --- | --- | --- |
| | | Desktop | Agent | |
| | 76 and future fix packs | ✔ | Not applicable | |
| | 75 and future fix packs | ✔ | Not applicable | |
| | 74 and future fix packs | ✔ | Not applicable | |
| | 73 and future fix packs | ✔ | Not applicable | |
| | 72 and future fix packs | ✔ | Not applicable | |
| | 71 and future fix packs | ✔ | Not applicable | |
| | 70 and future fix packs | ✔ | Not applicable | |
| | 69 and future fix packs | ✔ | Not applicable | |
| | 68 and future fix packs | ✔ | Not applicable | |
| | 67 and future fix packs | ✔ | Not applicable | |
| | 66 and future fix packs | ✔ | Not applicable | |

| Prerequisite | Version | Components | | Notes |
| | | Desktop | Agent | |
| --- | --- | --- | --- | --- |
| | 65 and future fix packs | ✔ | Not applicable | |
| | 64 and future fix packs | ✔ | Not applicable | |
| | 63 and future fix packs | ✔ | Not applicable | |
| | 62 and future fix packs | ✔ | Not applicable | |
| | 61 and future fix packs | ✔ | Not applicable | |
| | 60 and future fix packs | ✔ | Not applicable | |
| Mozilla Firefox ESR | 68 and future fix packs | ✔ | Not applicable | |
| | 60 and future fix packs | ✔ | Not applicable | |

# Recording support

You can find information about the web browsers that supports recording capability of HTTP tests for  Rational® Service Tester for SOA Quality.

**Web browsers**

The following versions of web browsers support the recording of HTTP tests. See to know the browsers that are supported to view the performance reports.

| Supported software | Version | Components | | Recording capability | Notes |
|---|---|---|---|---|---|
| | | Desktop | Agent | | |
| Apple Safari | 13 and future fix packs | ✔ | Not applicable | ✔ | To record HTTP tests |
| | 12 and future fix packs | ✔ | Not applicable | ✔ | |
| Google Chrome | 84 and future fix packs | ✔ | Not applicable | ✔ | |
| | 83 and future fix packs | ✔ | Not applicable | ✔ | |
| | 81 and future fix packs | ✔ | Not applicable | ✔ | |
| | 80 and future fix packs | �‖ | Not applicable | ✗ | |
| | 79 and future fix packs | ✗ | Not applicable | ✗ | |
| | 78 and future fix packs | ✗ | Not applicable | ✗ | |

| Supported software | Version | Components | | Recording capability | Notes |
|---|---|---|---|---|---|
| | | Desktop | Agent | | |
| Microsoft Edge | Latest release | ✔ | Not applicable | ✔ | |
| | 44 and future fix packs | ✔ | Not applicable | ✔ | |
| | 42 and future fix packs | ✖ | Not applicable | ✖ | |
| Microsoft Internet Explorer | 11 and future fix packs | ✔ | Not applicable | ✔ | |
| Mozilla Firefox | 79 and future fix packs | ✔ | Not applicable | ✔ | |
| | 77 and future fix packs | ✔ | Not applicable | ✔ | |
| | 76 and future fix packs | ✔ | Not applicable | ✔ | |
| | 75 and future fix packs | ✔ | Not applicable | ✔ | |

| Supported software | Version | Components | | Recording capability | Notes |
|---|---|---|---|---|---|
| | | **Desktop** | **Agent** | | |
| | 74 and future fix packs | ✔ | Not applicable | ✔ | |
| | 73 and future fix packs | ✖ | Not applicable | ✖ | |
| | 72 and future fix packs | ✖ | Not applicable | ✖ | |
| | 71 and future fix packs | ✖ | Not applicable | ✖ | |
| | 70 and future fix packs | ✖ | Not applicable | ✖ | |
| | 69 and future fix packs | ✖ | Not applicable | ✖ | |
| | 68 and future fix packs | ✖ | Not applicable | ✖ | |
| | 67 and future fix packs | ✖ | Not applicable | ✖ | |

| Supported software | Version | Components | | Recording capability | Notes |
| --- | --- | --- | --- | --- | --- |
| | | Desktop | Agent | | |
| | 66 and future fix packs | ✖ | Not applicable | ✖ | |
| | 65 and future fix packs | ✖ | Not applicable | ✖ | |
| | 64 and future fix packs | ✖ | Not applicable | ✖ | |
| | 63 and future fix packs | ✖ | Not applicable | ✖ | |
| | 62 and future fix packs | ✖ | Not applicable | ✖ | |
| | 61 and future fix packs | ✖ | Not applicable | ✖ | |
| | 60 and future fix packs | ✖ | Not applicable | ✖ | |
| Mozilla Firefox ESR | 68 and future fix packs | ✔ | Not applicable | ✔ | |

| Supported software | Version | Components | | Recording capability | Notes |
|---|---|---|---|---|---|
| | | Desktop | Agent | | |
| | 60 and future fix packs | ✖ | Not applicable | ✖ | |

# Supported software

You can find the additional software that are supported for Rational® Service Tester for SOA Quality.

**Contents**

**Application servers**

Support for the following application servers is in reference only to the HTTP Response Time Break Down capability:

| Supported software | Version | Components | | Notes |
|---|---|---|---|---|
| | | Desktop | Agent | |
| Oracle/BEA WebLogic Server | 12 and future fix packs | ✔ | Not applicable | To collect response time breakdown data |
| | 10.3 and future fix packs | ✔ | Not applicable | |
| | 9 and future fix packs | ✔ | Not applicable | |
| WebSphere Application Server | 9.0 | ✔ | Not applicable | |
| | 8.5.5 | ✔ | Not applicable | |

| Supported software | Version | Components | | Notes |
|---|---|---|---|---|
| | | Desktop | Agent | |
| | 8.5 | ✔ | Not applicable | |
| | 8.0 | ✔ | Not applicable | |
| | 7.5 | ✔ | Not applicable | |
| WebSphere Liberty | 17.0.0.1 | ✔ | Not applicable | |
| | 16.0.0.2 | ✔ | Not applicable | |

**Business process management**

| Supported software | Version | Components | | Notes |
|---|---|---|---|---|
| | | Desktop | Agent | |
| SAP GUI | 7.6 and future fix packs | ✔ | ✔ | To record and playback tests of SAP applications built with the SAP GUI client |
| | 7.5 and future fix packs | ✔ | ✔ | |
| | 7.4 and future fix packs | ✔ | ✔ | |
| | 7.3 and future fix packs | ✔ | ✔ | |
| | 7.2 and future fix packs | ✔ | ✔ | |

| Supported software | Version | Components | | Notes |
| --- | --- | --- | --- | --- |
| | | Desktop | Agent | |
| | 7.1 and future fix packs | ✔ | ✔ | |

## Development tools

| Supported software | Version | Components | | Notes |
| --- | --- | --- | --- | --- |
| | | Desktop | Agent | |
| Google Web Toolkit | 2.8 and future fix packs | ✔ | ✔ | To record and playback http tests that against web applica-tions built with Google Web Toolkit |
| | 2.7 and future fix packs | ✔ | ✔ | |
| | 2.6 and future fix packs | ✔ | ✔ | |
| | 2.5 and future fix packs | ✔ | ✔ | |
| | 2.4 and future fix packs | ✔ | ✔ | |
| | 2.3 and future fix packs | ✔ | ✔ | |
| JMeter | 5.3 and future fix packs | ✔ | Not applicable | To integrate and run JMe-ter tests |
| | 5.2 and future fix packs | ✔ | Not applicable | |

| Supported software | Version | Components | | Notes |
| | | Desktop | Agent | |
| --- | --- | --- | --- | --- |
| | 5.1 and future fix packs | ✔ | Not applicable | |
| Rational Application Developer for WebSphere Software | 9.7.0.2 | ✔ | Not applicable | Eclipse shell sharing |
| | 9.7 | ✔ | Not applicable | |
| | 9.6.1 | ✔ | Not applicable | |
| Rational ClearCase | 8.0.1 and future fix packs | ✔ | Not applicable | Eclipse source control |
| Rational® Functional Tester | 10.1.1 | ✔ | Not applicable | Eclipse shell sharing and to run WebUI integrations |
| Engineering Test Management | 7.0.1 | ✔ | Not applicable | To initiate the test runs from Engineering Test Management |
| Rational Quality Manager | 6.0.6 | ✔ | Not applicable | To initiate the test runs from Rational Quality Manager |
| | 6.0.5 | ✔ | Not applicable | |
| | 6.0.4 | ✔ | Not applicable | |
| | 6.0.3 | ✔ | Not applicable | |
| Engineering Workflow Management | 7.0.1 | ✔ | Not applicable | To perform integrations with |

| Supported software | Version | Components | | Notes |
| --- | --- | --- | --- | --- |
| | | Desktop | Agent | |
| | | | | Engineering Workflow Management |
| Rational Team Concert | 6.0.6 | ✔ | Not applicable | To perform integrations with Rational Team Concert |
| | 6.0.5 | ✔ | Not applicable | |
| | 6.0.4 | ✔ | Not applicable | |
| | 6.0.3 | ✔ | Not applicable | |
| Rational® Software Architect Designer | 9.7.0.2 | ✔ | Not applicable | Eclipse shell sharing |
| | 9.7 | ✔ | Not applicable | |
| | 9.6.1 | ✔ | Not applicable | |
| Rational Software Architect Designer for WebSphere Software | 9.7 | ✔ | Not applicable | Eclipse shell sharing |
| Rational® Test Workbench | 10.1.1 | ✔ | ✔ | To integrate and run Rational® Integration Tester tests |
| eGit | 4.10 | ✔ | Not applicable | Eclipse source control |

**DevOps tools**

| Supported software | Version | Components | | Notes |
|---|---|---|---|---|
| | | **Desktop** | **Agent** | |
| IBM UrbanCode Deploy | 7.0.2 | ✔ | Not applicable | To initiate the test runs from UrbanCode Deploy |

# Chapter 3. Getting Started

This guide provides an overview and describes the task flows to get you started with Rational® Service Tester for SOA Quality. This guide is intended for new users.

## Product overview

You can gain the conceptual understanding of the product and its test extensions with these topics.

### Service testing overview

The service testing capabilities of IBM® Rational® Performance Tester or IBM® Rational® Service Tester for SOA Quality automate the creation, execution and analysis of functional, regression and performance tests for SOAP-based web services, including support for Java Messagine Service (JMS), Websphere MQ, WebSphere Java MQ, and Microsoft .NET Windows Communication Foundation (WCF), as well as any service that produces XML, plain text, or binary data.

Informative test results rely upon sound test development. Each of the following stages contributes to generating meaningful test results:

- **Preparation.** Set up your test environment with the libraries and configuration files required for SOAP-based web services or custom security algorithms. You can import Web Service Description Language (WSDL) definition files and digital certificates that are required by the web services to automatically generate your tests. You can create SOAP security profiles with security algorithms for the web service calls and message returns.
- **Test creation:** Create your test by recording the service requests and responses either with the **generic service client**, or with an existing client or a web browser through a recording proxy. When you start the recording, you interact with the service by performing service requests and receiving responses. You can also create service tests manually or from a synchronous Business Process Execution Language (BPEL) model.
- **Test editing:** After recording, you can edit the requests and responses in the test. You can use XML Schema Description (XSD) documents to facilitate XML edition. You can replace recorded test values with variable test data, or add dynamic data to the test.
- **Functional testing:** You can run the test to ensure that service matches the expected behavior defined in *verification points*. During the run, each verification point is checked and receives a *pass*, *fail* or *inconclusive* status.
- **Performance testing:** If you are using IBM® Rational® Performance Tester, you can specify an execution schedule and user groups to emulate a workload that is generated by a large number of virtual users. Then, you can run the schedule, deploying test execution on virtual users that can be hosted on remote computers. Each virtual user runs an instance of the test client. Response times are measured and recorded. Verification points are checked and recorded.
- **Stub simulation:** Service stubs are functional simulations of an existing service. Service stubs are useful for replacing a service that is unavailable or impractical to use in a test environment. They can also be used to

input specific data into a service under test or for prototyping. You can deploy stubs onto a stub server, which can replace the actual server in your test or development environment.

- **Evaluation of results:** You evaluate the results that the tests produce through the performance and verification point reports that are generated during execution. You can also design custom reports by manipulating various counters. Functional reports provide a comprehensive view of the behavior of the service under test. Reports can be exported and archived for validation.

## Service testing tools

The following tools are available in the product:

- The **generic service client** enables you to manually perform service requests for a wide variety of transport protocols, authentication configurations and security profiles, making it an extremely versatile service client. It effectively replaces a dedicated client and can be used to record service calls or for manual testing and debugging a service during development. To open the generic service client, click the **Generic Service Client** toolbar button.
- The **WSDL security editor** allows you to set up sophisticated *algorithm stacks* for your service requests and responses. Algorithm stacks contain digital certificate information and the security algorithms that are applied to messages to perform secure communication with a web service. Algorithm stacks are made of blocks, which can be key definitions, encryption , time stamp, or signature operations which can be associated with any operation in the WSDL file. To open the WSDL security editor, right-click a WSDL file in your workspace and select **Edit WSDL Security** or click the **WSDL Security Editor** button in the generic service client..
- The **test editor** is where you develop your test. After recording, you can modify the test to add data correlation or verification points. You can also add loops and conditions and you can edit every detail of the service requests.
- The **stub editor** enables you to create service stubs. With the stub editor, you can define multiple input conditions, which are similar to verification points. Each condition triggers a predefined simulated response, which is functionally identical to a response from the simulated service.
- In  Rational® Performance Tester, the schedule editor lets you deploy multiple virtual users on local and remote computers to generate a heavy load for performance testing. A schedule typically contains multiple tests and multiple virtual users.

## Generic service client overview

The purpose of the generic service client is to send requests to any service that uses an HTTP, JMS, WebSphere® MQ, or Microsoft™ .NET transport. The generic service client also displays the response returned by the service.

The generic service client is useful for debugging or testing a service when you do not have access to a dedicated client to send the request. You can set up a large variety of transport and security configurations for the service, edit the parameters of the request and send attachments.

When a request is successfully invoked, its message return is added to the **Request History**. You can use this feature to look back at results that were produced at different times.

If you are using IBM® Rational® Performance Tester or IBM® Rational® Service Tester for SOA Quality, you can select requests in the **Request History** and click **Generate Test** to generate a test that will replay all the selected requests. You can edit the test to replace recorded test values with variable test data, or add dynamic data correlation to the test. You can also set verification points on the contents of the XML documents in the service response.

## Supported services

The generic service client enables you to send requests for many types of services that use the following transport protocols:

- HTTP
- Java™ Message Service (JMS), including JBoss and WebSphere® implementations
- WebSphere® MQ
- Microsoft™ .NET Framework Windows™ Communication Foundation (WCF).

**Note:** If you are using IBM® Security AppScan®, only the HTTP transport protocol is supported.

## Encryption and security

The Java™ Runtime Environment (JRE) that the product uses must support the level of encryption required by the digital certificate that you select. For example, you cannot use a digital certificate that requires 256-bit encryption with a JRE that supports only 128-bit encryption. By default, the product is configured with restricted or limited strength ciphers. To use less restricted encryption algorithms, you must download and apply the unlimited jurisdiction policy files (`local_policy.jar` and `US_export_policy.jar`).

For Oracle Java, download the files from this site:[http://www.oracle.com/technetwork/java/javase/downloads/jce8-download-2133166.html](http://www.oracle.com/technetwork/java/javase/downloads/jce8-download-2133166.html).

Before installing these policy files, back up the existing policy files in case you want to restore the original files later. Then overwrite the files in `/jre/lib/security/` directory with the unlimited jurisdiction policy files.

## SSL Authentication

Service tests support simple or double SSL authentication mechanisms:

- Simple authentication (server authentication): In this case, the test client needs to determine whether the service can be trusted. You do not need to setup a key store. If you select the **Always trust** option, you do not need to provide a server certificat key store.

  If you want to really authenticate the service, you can configure an certificate trust store, which contains the certificates of trusted services. In this case, the test will expect to receive a valid certificate.

- Double authentication (client and server authentication): In this case, the service needs to authenticate the test client according to its root authority. You must provide the client certificate keystore that needs to be produced to authenticate the test as a certified client.

When recording a service test through a proxy, the recording proxy sits between the service and the client. In this case, you must configure the SSL settings of the recording proxy to authenticate itself as the actual service to the client (for simple authentication), and as the client to the service (for double authentication). This means that you must supply the recording proxy with the adequate certificates.

When using stub services, you can also configure the SSL settings of the stub service to authenticate itself as the actual server. This means that you must supply the service stub with the adequate certificate.

**NTLM and Kerberos Authentication**

The product supports Microsoft™ NT LAN Manager (NTLMv1 and NTLMv2) and Kerberos authentication. The authentication information is recorded as part of the test during the recording phase.

To enable NTLMv2 support, you must add a third party library to the workbench. For more information, see .

**Digital certificates**

You can test services with digital certificates for both SSL and SOAP security protocol. Digital certificates must be contained in Java™ Key Store (JKS) keystore resources that are accessible in the workspace. When dealing with keystore files, you must set the password required to access the keys both in the security editor and the test editor. For SOAP security you might have to provide an explicit name for the key and provide a password to access the private keys in the keystore.

**Limitations**

Arrays are not supported.

Because of a lack of specification, attachments are not supported with the Java™ Message Service (JMS) transport. The envelope is directly sent using UTF-8 encoding.

All security algorithms are not always available for every Java™ Runtime Environment (JRE) implementation. If a particular security implementation is not available, add the required libraries to the class path of the JRE that this product uses.

The Microsoft™ .NET transport protocol does not support transactions, scopes, or duplex mode requests such as callbacks or two-way services based on the MS-MQ transport.

## Socket API performance testing overview

With IBM® Rational® Performance Tester Extension *for Socket Protocols*, you can test the performance of any application that uses a TCP/IP socket-based protocol.

Informative performance test results rely upon sound test development. Each of the following stages contributes to the generation of meaningful test results:

- **Test creation.** You create your test by recording a session with a client application. Typically, the recorded session starts when you run the client application. You then interact with the application in order to produce relevant network traffic, and the session ends when you close the application or end the recording. The recording is used to generate a performance test that reproduces the behavior of the client application.
- **Test editing.** After recording, you can edit the events that were recorded. You can replace recorded test values with variable test data or add dynamic data to the test.
- **Test validation.** Before deploying the test, you can run the test manually as a single virtual user to make sure that the test runs smoothly and produces the expected results in a nominal environment with minimal server load. You might experience multiple test editing and validation cycles before your test runs as expected.
- **Workload emulation with schedules.** When the test runs repeatedly as anticipated, you specify an execution schedule and user groups to emulate a workload that a large number of virtual users generates.
- **Schedule execution.** You run the schedule, deploying test execution over virtual users that can be hosted on remote hosts. Each virtual user runs an instance of the test. Response time results are collected.
- **Evaluation of results.** You evaluate the results produced by the tests through the various reports that are generated during execution. You can also design custom reports.

## TN3270 performance testing overview

With IBM® Rational® Performance Tester Extension for Socket Protocols, you can test the performance of TN3270 terminal server applications.

Informative performance test results rely on sound test development. Each of these stages contributes to the generation of meaningful test results:

- **Test creation.** You create a test by recording a session with a client application. Typically, the recorded session starts when you run the TN3270 terminal client. You then interact with the application in order to produce relevant network traffic. The session ends when you close the terminal client or end the recording. The recording is used to generate a performance test that reproduces the behavior of the client application.
- **Test editing.** After recording, you can edit the recorded events. You can replace recorded test values with variable test data or add dynamic data to the test.
- **Test validation.** Before deploying the test, you can run the test manually as a single virtual user to make sure that the test runs smoothly and produces the expected results in a nominal environment with minimal server load. You might complete multiple test editing and validation cycles before your test is robust.
- **Workload emulation with schedules.** When the test runs repeatedly as anticipated, you specify an execution schedule and user groups to emulate a workload that a large number of virtual users generates.
- **Schedule execution.** You run the schedule, deploying test execution over virtual users that can be hosted on remote hosts. Each virtual user runs an instance of the test. Response time results are collected.
- **Evaluation of results.** You evaluate the results that the tests produce through the various reports that are generated during execution. You can also design custom reports.

## IBM® Engineering Test Management overview

IBM® Engineering Test Management is a collaborative, web-based, quality management solution that offers comprehensive test planning, manual testing, and integration with other test tools.

Quality Manager is based on the IBM® Rational® Jazz™ platform (http://jazz.net and http://www.ibm.com/software/rational/jazz/) and inherits many characteristics from that platform. Engineering Test Management is designed to be used by test teams of all sizes and supports a variety of user roles, such as test manager, test architect, test lead, tester, and lab manager, as well as roles outside the test organization.

### Comprehensive test planning

A *test plan* that you define in Engineering Test Management drives activity for distributed teams through all phases of the project life cycle. The test plan defines the objectives and scope of the test effort and contains criteria to help teams determine the answer to the question "Are we ready to release?"

The test plan can be configured to meet the needs of your organization. You can use the test plan to do any and all of the following tasks:

- Define business and test objectives
- Establish a review and approval process for the test plan and for individual test cases
- Manage project requirements and test cases and establish the interdependencies between the two
- Estimate the size of the test effort
- Define the schedule for each test iteration and track the dates of other important test activities
- List the various environments to be tested and generate test configurations
- Create a read-only snapshot of the test plan at a particular point in time
- Define quality goals, entrance criteria, and exit criteria
- Create and manage test cases

### Test script construction, execution, and reuse

Engineering Test Management provides a full-featured manual test editor. You can also import manual test scripts from IBM® Rational® Manual Tester. You can add reuse and automation capabilities to your manual tests by using keywords.

With Engineering Test Management, you can manage and execute test scripts that are created with tools such as IBM® Rational® Performance Tester, IBM® Rational® Service Tester for SOA Quality, and IBM® Security AppScan® Tester Edition.

You can also import test artifacts from external test management solutions, such as IBM® Rational® ClearQuest® Test Manager and IBM® Rational® Test Manager.

### Test analysis and reporting

Engineering Test Management includes several standard test reports to help you evaluate test results. Reports are available during all phases of the test process.

You can use reports to perform these tasks:

- Determine the validity of a test run.
- Check feature coverage against test plans, test inputs, configurations, and so on. This can also be used to measure test progress and to analyze trends.
- Run a gap analysis to measure the resources needed to do your testing versus the resources that are available

### Team collaboration

Engineering Test Management makes it easy to share information with other members of your team. With the Jazz-based work-item system, team members can assign tasks and defects to each other and to view everyone's status. Test plan authors and test case designers can distribute their work for review and track the status of each reviewer. New and changed requirements are visible to the team, as are the test cases that are needed to satisfy those requirements. Team members are notified automatically of any changes and milestones that impact their work.

### Lab management

With Engineering Test Management lab management capabilities, you can create requests for the test environments that your test plan specifies. You can then work with the lab manager to ensure that lab resources and test environments are available when needed. Lab managers can track all lab resources from a centralized resource repository and fulfill requests from the test team.

### Web application security

Engineering Test Management helps IT and security professionals protect against the threat of attacks and security breaches through its integration with IBM® Security AppScan® Tester Edition. Security testing for your web applications can result in higher-quality, more secure applications at a reasonable cost.

### Governance

Engineering Test Management helps ensure that your business processes comply with industry, corporate, and departmental standards and regulations. Throughout the testing life cycle, Engineering Test Management provides you with the tools to obtain an up-to-the-minute measurement of software quality and project metrics. With its comprehensive test plan and integration with requirements management and defect tracking tools, Engineering Test Management helps streamline your test strategy and produce reliable records of test results and project history.

## Streamlined Eclipse and full Eclipse overview

When you work in the streamlined Eclipse mode, only those functions that are directly related to the product are enabled in the workbench. When you install the product, by default, the check box to use the streamlined Eclipse mode is selected. With the full Eclipse mode, you have access to all Eclipse functions.

The streamlined Eclipse mode disables options from the menus that are not typically used during testing. Both the fully-enabled and streamlined Eclipse modes can operate using the same workspace, so if you start the product in the streamlined mode and discover that you cannot accomplish all of your tasks, you can close the workbench and restart it in the full Eclipse mode.

The choice of the mode in which to start the product depends on the user's activity and objectives. The streamlined mode is designed for straightforward testing and shows only those menu items that are related to testing. However, this restricts functions. The following list includes use cases where the full Eclipse mode might be preferred:

- You have multiple products installed and you want to use them in the same session.
- You are using the profiling and logging features. The profiling and logging view is not available in the streamlined mode.
- You are using advanced features of custom code including debugging custom code. For more information on custom code, see Extending test execution with custom code on page .

## Starting Rational® Service Tester for SOA Quality in full Eclipse mode

You can start the product in the full Eclipse mode to continue to use native Eclipse features along with Rational® Performance Tester Rational® Service Tester for SOA Quality.

**Before you begin**

You must have installed Rational® Service Tester for SOA Quality.

1. Click **Start > IBM Software Delivery Platform > IBM Rational Service Tester for SOA Quality - Full Eclipse - Full Eclipse**.
2. Perform the following steps to select a working directory, if you are starting the installation of Rational® Service Tester for SOA Quality for the first time.

    a. Enter the path of a `working directory` in the **Workspace** field or click **Browse** to select the directory.

    b. Select **Use this as the default and do not ask again** to make this your default workspace.
    You can change your workspace from Rational® Service Tester for SOA Quality by clicking **File > Switch Workspace**.

3. Click **OK**.

**Results**

You have started Rational® Service Tester for SOA Quality in full Eclipse mode.

## Starting Rational® Service Tester for SOA Quality in streamlined Eclipse mode

If you do not want to view native Eclipse UI, you can start IBM® Rational® Service Tester for SOA Quality in streamlined mode.

**Before you begin**

You must have installed Rational® Service Tester for SOA Quality.

> **Note:** To start  Rational® Service Tester for SOA Quality in the streamlined mode, the streamlined mode must be installed as an optional feature. It is automatically selected during the installation process.

1. Click **Start > IBM Software Delivery PlatformIBM Rational Service Tester for SOA Quality**.
2. Perform the following steps to select a working directory, if you are starting the installation of  Rational® Service Tester for SOA Quality for the first time.

   a. Enter the path of a `working directory` in the **Workspace** field or click **Browse** to select the directory.

   b. Select **Use this as the default and do not ask again** to make this your default workspace.
      You can change your workspace from  Rational® Service Tester for SOA Quality by clicking **File > Switch Workspace**.

3. Click **OK**.

**Results**

You have started  Rational® Service Tester for SOA Quality in streamlined Eclipse mode.

# Chapter 4. Tutorials

This section contains the tutorials which explains the main features of  Rational® Service Tester for SOA Quality.

## Test an SOA application

The movies in this tutorial show you the main features of IBM® Rational® Service Tester for SOA Quality or the SOA extension for IBM® Rational® Performance Tester. The tutorial requires Flash Player to view.

**Learning objectives**
Learn how to perform the following tasks:

- Record service calls and generate service tests
- Configure your test environment to support HTTP, Java™ Message Service (JMS), or WebSphere® MQ transport protocols
- Edit and manage variable data using datasets
- Run tests and view reports
- Create and deploy service stubs
- Generate HTML or PDF functional reports

45 minutes

## Introduction: Testing services

This tutorial introduces you to testing services in an SOA environment.

The service testing capabilities of IBM Rational Performance Tester or IBM Rational Service Tester for SOA Quality automate the creation, execution and analysis of functional, regression and performance tests for SOAP-based web services, XML services or plain text service.

**Learning objectives**
The tutorial is divided into four modules, each with its own learning objectives. Learn to perform the following tasks:

- Creating a service test project
- Importing a WSDL file
- Creating a WSDL security stack
- Invoking a service call and generating a service test
- Simulating services with service stubs

This tutorial requires approximately 20 minutes to finish. If you explore other concepts related to this tutorial, it might take longer to complete.

**Skill level**
Beginner

**Prerequisites**

To complete this tutorial, you need to be familiar with the underlying concepts behind SOA and functional testing. Experience in using the perspectives and views in IBM® Rational® Software Development Platform are also required.

## Module 1: Recording service calls and creating tests

In this module, learn how to create projects and record service calls for testing. The second part of this module helps you understand a service test.

**Learning objectives**

After completing the lessons in this module, you will know how to do the following tasks and understand the associated concepts:

- Create a test project
- Record a service test with the generic service client
- View a test in the test editor

This module requires approximately 12 minutes to complete.

## Lesson 1.1: Creating a project and sending a request to a web service

When you create a service test, the first step is to set up your project and to import all the required resources.

**About this task**

For SOAP-based web services, the main resource is the Web Services Description Language (WSDL) specification. You can import a WSDL from a WebSphere® Service Registry and Repository or Universal Description, Discovery, and Integration (UDDI) repository. Of course, you can also simply import one from the workspace.

If you are testing services that do not use a WSDL file, you can skip this step.

See video

## Lesson 1.2: Using the generic service client to create a service test

The generic service client is a multipurpose tool that is designed to send requests to any kind of service that uses an HTTP, JMS, or WebSphere® MQ transport and to view the message that the service returns.

**About this task**

The generic service client is useful for debugging or testing a service when you do not have access to a dedicated client to invoke the service call. You can set up a wide variety of transport and security configurations for the service, edit the parameters of the call and send attachments.

Although you can record a service test in a variety of ways, this lesson focuses on using the generic service client to send a series of service requests and to generate a service test with the results.

See video

## Lesson 1.3: Creating security stacks

Algorithm stacks contain digital certificate information and the security algorithms that are applied to messages to perform secure communication with a web service.

**About this task**

In this lesson, learn how to use the WSDL security editor to create and edit algorithm stacks for service requests and responses. The WSDL security editor supports most industry standards for encrypting, decrypting, signing, and processing requests and responses from SOAP-based services. The WSDL security editor contains two pages, which relate to the two steps of setting up a security configuration:

- Describing an algorithm stack as a sequence of algorithm blocks that you can customize
- Associating an algorithm stack with each request and response operation in the WSDL

See video

## Module 1 summary

In this module, you learned how to create projects and record service calls for testing.

**Lessons learned**

By completing this module, you learned about the following concepts and tasks:

- Creating a project for the first step in service testing
- Importing a WSDL file and creating transport and SSL configurations
- Invoking service calls with the generic service client and generating a test

## Module 2: Editing service tests

In this module, you see how to edit your service test to include verification points and variable data by using datasets.

With the test editor, you can inspect or customize a test that you recorded. The test editor lists the web service call elements for a test, in sequential order. You can add, remove, and edit test elements in the test editor, including verification points and data correlation.

**Learning objectives**

After completing the lessons in this module, you will know how to do the following tasks and understand the associated concepts:

- Add a verification point
- Create a dataset
- Use a dataset in a test

This module requires approximately 8 minutes to complete.

## Lesson 2.1: Enabling verification points

In this lesson, learn how to add verification points to check whether an expected behavior occurs during a run.

**About this task**

With verification points, you can test the behavior of the service during a test. For example, you can use verification points to ensure that a particular response contains the expected XML content or that a specific binary attachment is returned.

Each verification point returns a `Pass`, `Fail` or `Inconclusive` verdict in the test log. You can view a summary of verification point verdicts in the verification point report after running the test.

See video

## Lesson 2.2: Using datasets

In this lesson, learn how to create a dataset that can provide tests with variable data and how to enable your test to use a dataset during a run.

**About this task**

Datasets provide tests with variable data during a run. When you record a test, you perform a sequence of steps that you expect a typical user to perform. From the recording, a test is generated that exactly reproduces these interactions. When you run this test, it uses the same data that you used during recording. To vary the data in the test, you use a data pool, which is typically a table that contains variable data. At run time, this variable data is substituted for the data in the recorded test.

See video

## Module 2 summary

In this module, you learned how to use variable data to edit and manage testing situations.

**Lessons learned**

By completing this module, you learned about the following concepts and tasks:

  • Adding verification points to check whether an expected behavior occurs during a test run
  • Adding a reference to a dataset so that the test can use variables from a dataset during a run

## Module 3: Running service tests

In this module, you see how to run a service test and to obtain the results of the test run.

You evaluate the results that the tests produce through the performance and verification point reports that are generated during execution. You can also customize reports by manipulating various counters or using custom report

designs. Functional reports provide a comprehensive view of the behavior of the service under test. Reports can be exported and archived for validation.

**Learning objectives**

After completing the lessons in this module, you will understand how to do the following tasks and understand the associated concepts:

- Run a single service test
- View the test log and message contents
- Generate a functional report in HTML or PDF.

This module requires approximately 10 minutes to complete.

## Lesson 3.1: Running a single test and viewing the test log

In this lesson, learn how to run a service test and view the results in the test log.

**About this task**

If you are using IBM® Rational® Performance Tester, then you can also create a schedule that contains service tests. A schedule that contains services tests works in the same way as other performance tests.

See video

## Lesson 3.2: Generating a functional test report

You can generate functional test reports of your tests, which summarize the pass or fail verdicts of elements in the test log. The functional reports from the test run are generated as HTML or PDF files that use predefined report designs.

**About this task**

In this lesson, you learn how to produce a PDF or HTML report that covers the functional behavior of the service. The following report types are available:

- Extensible Stylesheet Language Transformation (XSLT) reports: These reports are faster to generate, but do not contain graphs.
- Business Intelligence and Reporting Tools (BIRT) report: These reports contain graphs but are slower to generate. You can customize and create your own BIRT report designs in the Report Design perspective of the workbench.

See video

## Module 3 summary

In this module, you learned how run a test and display the results of the test run.

**Lessons learned**

By completing this module, you learned about the following concepts and tasks:

- Running a single service test
- Viewing the test log and message contents
- Generating a functional report in HTML or PDF

## Module 4: Simulating services

In this module, you become familiar with simulating services with service stubs.

**Learning objectives**

After completing the lessons in this module, you will know how to do the following tasks and understand the associated concepts:

- Create service stubs
- Deploy and run a stub server

This module requires approximately 15 minutes to complete.

## Lesson 4.1: Creating a service stub

In this lesson, learn how to create a service stub from a Web Service Description Language (WSDL) specification.

**About this task**

Service stubs are simulations of an actual service, which can be used to functionally replace the service in a test environment. A stub server replaces the actual application server in cases where it is not practical to use the server. For example, use a stub server in these instances:

- If you are testing a local service that uses data from a remote service, you might need to inject specific content into the service under test from the remote service. You can simulate the remote service with a service stub to ensure that the local service responds correctly to specific input.
- Some commercial service providers charge users for each request. If you are testing such a service, you can develop and debug your test against a stub service, which is based on the WSDL of the actual service, without being charged by the service provider.
- During integration of a large application that involves multiple clients and services, some services might not yet be operational, although their WSDL specifications are available. You can simulate the missing services with service stubs so that you can proceed with the integration work.

From the point of view of the client application, the service stub looks identical to the actual service that it simulates. To use a service stub as a replacement of the actual service, you must be able to replace the URL of the original service in the client application with the URL of the stub server.

You create a service stub by providing a current WSDL specification. The service stub is generated with the exact same ports and bindings as the original service so that it can be addressed with exactly the same interface. Each operation in the service returns a default response of the type defined by the WSDL.

See video

## Lesson 4.2: Running the service stub server

In this lesson, learn how to deploy a service stub onto a stub server that is running on your local computer.

**About this task**

When you have finished editing the service stub, you can deploy the stub on a local stub server, which runs in the workbench. The stub server simulates an actual application server and can host multiple service stubs. You control the stub server from the stub monitor view.

Finally, to use the service stub instead of the original service, change the URL that the client application uses to point to the local stub server instead of the original application server. This URL and the WSDL of the service stub are provided in the stub monitor view.

See video

## Module 4 summary

In this module, you learned how to simulate a service with a service stub and to validate that the stub server runs correctly.

**Lessons learned**

By completing this module, you learned about the following concepts and tasks:

- Creating service stubs
- Deploying and running a stub server

## Summary

This tutorial introduced you the basics of service testing. You have learned how this product provides the following functions and features:

- The generic service client for sending and receiving service calls
- The test editor for creating and editing service tests
- Datasets for supplying realistic test data.
- The stub editor and stub monitor for simulating services and deploying service stubs.

**Lessons learned**

After completing all of the modules, you can perform these tasks:

- Record service calls and generate service tests
- Configure your test environment to support HTTP, JMS or WebSphere® MQ transport protocols
- Edit and manage variable data using datasets
- Run tests and view reports
- Create and deploy service stubs
- Generate HTML or PDF functional reports

## Resources

To learn more about using Rational® Service Tester for SOA Quality or Rational® Service Tester, visit developerWorks®, IBM's resource for developers, at: http://www.ibm.com/developerworks/

# Chapter 5. Administrator Guide

This guide describes how to install  Rational® Service Tester for SOA Quality. After you install the software, you can perform administration tasks such as license configuration and integration with other products. This guide is intended for administrators.

## Installation of  Rational® Service Tester for SOA Quality

Installing the product involves verifying requirements, planning, managing licenses, and configuring web-based help. This section lists all such topics.

This installation guide covers two independent products:  Rational® Performance Tester and the  Rational® Performance Tester Agent. The  Rational® Performance Tester Agent is a tool that you use with  Rational® Performance Tester. It is included as part of the  Rational® Performance Tester product kit.

Rational® Performance Tester Agent consists of two capabilities:

- Generate load for the application under test by using the virtual users. You can increase the load generation capacity by installing additional agents on remote computers.

- Gather data for the Response Time Breakdown feature and in support of the startup and control of web services stubs in the SOA protocol.

**Note:** The  Rational® Performance Tester and  Rational® Performance Tester Agent are separate offerings and must be installed separately.

### Installation requirements

Installation requires the correct hardware, software, server environment, operating systems, and user privileges for installing and running your software.

**Hardware and Software requirements**

Before you install the product, verify that your system meets the hardware and software requirements.

For information about hardware and software compatibility, see System Requirements on page 14.

**User privileges requirements**

You must have a user ID that meets the following requirements to install  Rational® Performance Tester  Rational® Service Tester for SOA Quality and  Rational® Performance Tester Agent.

**Notes:**

- Your user ID must not contain double-byte characters.
- For Windows operating system, you must have a user ID that belongs to the Administrators group.
- For Linux operating system, you must be able to log in as root.

## Installation conventions and terminology

Understanding these terms and conventions can help you take full advantage of the installation information and your product.

The following conventions are used in this installation information:

- The default installation directory is written as C:\*installation_directory*\product\inst.file.
- The default log location for installation information is C:\*log_file_dir*\log.txt.

These terms are used in the installation topics.

**Installation directory**

The location of product artifacts after the package is installed.

**Package**

An installable unit of a software product. Software product packages are separately installable units that can operate independently from other packages of that software product.

**Package group**

A package group is a directory in which different product packages share resources with other packages in the same group. When you install a package using Installation Manager, you can create a new package group or install the packages into an existing package group. Eclipse-based packages installed in the same package group are able to use the shell-sharing features of Eclipse. Some packages cannot share a package group, in which case the option to use an existing package group is unavailable.

**Repository**

A storage area for installable software packages. A repository can be disc media, a folder on a local hard disk, or a server or web location.

**Shared directory**

In some instances, product packages can share resources. These resources are located in a directory that the packages share.

## Capacity Planning

This document offers a methodology for assessing the CPU and memory characteristics of a given test and provides a set of best practices for scaling up to high volume loads using Rational® Performance Tester.

For information about capacity planning, see Capacity planning for  Rational® Performance Tester.

## Installation Manager overview

Installation Manager is a program for installing, updating, and modifying packages. It helps you to manage the applications or packages that it installs on your computer. Installation Manager also helps you to keep track of what you have installed, determine what is available for you to install, and to organize installation directories.

Installation Manager provides features that help you keep packages up to date, modify packages, manage the licenses for your packages, and uninstall packages.

Installation Manager includes six wizards that make it easy to maintain packages:

- The **Install** wizard walks you through the installation process. You can install a package by simply accepting the defaults or you can modify the default settings to create a custom installation. Before you install, you get a complete summary of your selections throughout the wizard. Using the wizard you can install one or more packages at one time.
- The **Update** wizard searches for available updates to packages that you have installed. An update might be a released fix, a new feature, or a new version of the product. Details of the contents of the update are provided in the wizard. You can choose whether to apply an update. The **Update** wizard searches connected repositories for updates. If you are not connected to the Internet, you may not see newly available updates for your installed products. To apply an update to a computer that is not connected to the Internet, you must download the update and extract it to a local repository.
- The **Modify** wizard helps you modify certain elements of a package that you have already installed. During the first installation of the package, you select the features that you want to install. Later, if you require other features, you can use the modify packages wizard to add them to your package. You can also remove features and add or remove languages.
- The **Manage Licenses** wizard helps you set up the licenses for your packages. Use this wizard to change your trial license to a full license, to set up your servers for floating licenses, and to select which type of license to use for each package.  Rational® Performance Tester requires runtime floating license keys to run tests with multiple virtual users and to use product extensions such as protocols. Runtime floating license keys are not managed using Installation Manager.
- The **Roll Back** wizard helps you to revert to a previous version of a package.
- The **Uninstall** wizard removes a package from your computer. You can uninstall more than one package at a time.

## Installation locations

Installation Manager retrieves product packages from specified repositories and installs the products into selected locations, which are referred to as package groups.

**Package groups**

During installation, you specify a *package group* into which to install a product.

- A package group represents a directory in which products share resources.
- When you install a product by using Installation Manager, you either create a package group or install the product into an existing package group. A new package group is assigned a name automatically; however, you choose the installation directory for the package group.
- After you create a package group you cannot change the installation directory. The installation directory contains files and resources that are shared by the products that are installed into that package group.
- Product resources that are designed to be shared with other packages are installed in the shared resources directory. Not all products can share a package group, in which case the option to use an existing package group is disabled.
- When you install multiple products at the same time, all products are installed into the same package group.

**Shared resources directory**

The *shared resources directory* is where product resources are installed so that they can be used by multiple product package groups. You define the shared resources directory the first time that you install the first product package. For best results, use your largest disk drive for shared resources directories. You cannot change the directory location unless you uninstall all product packages.

## Coexistence

Some products are designed to coexist and share functions when they are installed in the same package group. A package group is a location where you can install one or more software product packages.

**Offering coexistence considerations**

When you install each product package, you select whether to install the product package into an existing package group or to create a package group. Installation Manager prevents you from installing products into package groups products that are not designed to share or do not meet version compatibility and other requirements. To install more than one product at a time, the products must be able to share a package group.

Any number of eligible products can be installed to a package group. When a product is installed, the product functions are shared with all the other products in the package group. If you install a development product and a testing product into one package group, when you start either of the products, you have both the development and testing functions available to you in your user interface. If you add a product with modeling tools, all the products in the package group have the development, testing, and modeling functions available.

## Eclipse instance overview

The product package that you install using Installation Manager comes with a version of Eclipse, which is the base platform of this product package. If you already have Eclipse installed on your workstation, you can add your product package directly to that Eclipse installation and extend the functions of the Eclipse integrated development environment (IDE).

Extending an Eclipse IDE adds the functions of the newly installed product, but maintains your IDE preferences and settings. Previously installed plug-ins are also still available.

In most cases, your current Eclipse IDE must be the same version as the Eclipse that the product you are installing uses. Installation Manager checks that the Eclipse instance that you specify meets the requirements for the installation package and helps you install the latest updates from eclipse.org, if required.

## Increasing the number of file handles on Linux™ workstations

For best product performance, increase the number of file handles above the default setting of 1024 handles.

**About this task**

> ⚠️ **Important:** Before you work with your product, increase the number of file handles. Most of the products use more than the default limit of 1024 file handles per process. A system administrator might need to make this change.

Exercise caution when using the following steps to increase your file descriptors on Linux™. If the instructions are not followed correctly, the computer might not start correctly.

1. Log in as root.

   If you do not have root access, you must obtain it before continuing.

2. Change to the `etc` directory.

   > ⚠️ **Attention:** If you decide to increase the number of file handles in the next step, *do not* leave an empty `initscript` file on your computer. If you do so, your computer will not start up the next time that you turn it on or restart.

3. Use the vi editor to edit the `initscript` file in the `etc` directory. If this file does not exist, type `vi initscript` to create it.
4. On the first line, type `ulimit -n 30000`.
   The point is that 30000 is significantly larger than 1024, the default value on most Linux™ computers.

   > ⚠️ **Important:** Do not set the number of handles too high, because doing so can negatively impact system-wide performance.

5. On the second line, type `eval exec "$4"`.
6. Save and close the file after making sure that you have completed steps 4 and 5.

   > 📝 **Note:** Ensure that you follow the steps correctly. If this procedure is not completed correctly, your computer will not start.

7. **Optional:** Restrict the number of handles available to users or groups by modifying the `limits.conf` file in the `etc/security` directory.

If you do not have this file, consider using a smaller number in step 4 in the previous procedure (for example, 2048). Do this so that most users have a reasonably low limit on the number of open files that are allowed per process. If you use a relatively low number in step 4, it is less important to do this. However, if you set a high number in step 4 earlier and you do not establish limits in the `limits.conf` file, computer performance can be significantly reduced.

The following sample `limits.conf` file restricts all users, and then sets different limits for others afterwards. This sample assumes that you set handles to 8192 in step 4 earlier.

```
*       soft nofile 1024
*       hard nofile 2048
root    soft nofile 4096
root    hard nofile 8192
user1   soft nofile 2048
user1   hard nofile 2048
```

Note that the `*` in the preceding example sets the limits for all users first. These limits are lower than the limits that follow. The root user has a higher number of allowable handles open, while the number that is available to user1 is between the two. Make sure that you read and understand the documentation that the `limits.conf` file contains before making changes.

**What to do next**

For more information on the ulimit command, see the main page for ulimit in the Linux™ documentation.

## Starting the launchpad

You can start the launchpad program to install the product.

**Before you begin**

You must have downloaded the electronic disks from the IBM® Passport Advantage® portal.

1. Open the command-line interface.
2. Change the directory to a location where you extracted the disk images.
3. Run the following command to start the launchpad program:
   - On **Windows**: RPTRST_SETUP\launchpad.exe
   - On **Linux** and **Mac OS X**: RPTRST_SETUP/launchpad.sh

**Results**

You have started the launchpad program.

## Installation of the product by using Installation Manager

The Setup disk includes the launchpad program, which provides you with a single location to start the installation process. You must download the product bits and then from the Installation Manager point to the Setup disk.

Use the launchpad program to start the installation of software by using any of the following methods:

- Installing from an electronic image on your local file system

- Installing from an electronic image on a shared drive

**Note:** For Linux™ computers, you must log in as the root user before you begin installation process.

For products that are installed by Installation Manager, starting the installation process from the launchpad program causes Installation Manager to be automatically installed if it is not already on your computer. Furthermore, the installation process is already configured with the location of the repository that contains the installation package. If you install Installation Manager separately, you must configure the repository preferences manually. Also, you can start the installation of a number of supporting software items from the launchpad.

To install Rational® Service Tester for SOA Quality as a non-administrator, you cannot use the launchpad program to start the installation process. Instead, you must manually run the userinst program from the Setup disk. Running the userinst program provides the same functions as starting the installation of Rational® Service Tester for SOA Quality from the launchpad.

You can also install Rational® Service Tester for SOA Quality from the command-line interface by using response files rather than using the Installation Manager graphical user interface. For more information about silent mode installation, refer to the Installation Manager documentation.

## Installing IBM® Rational® Service Tester for SOA Quality

To test the performance of an application, you must install Rational® Service Tester for SOA Quality.

**About this task**

If you use the Launchpad program to install the product, Installation Manager gets installed automatically if you do not have it on your computer and starts the product installation by using the preconfigured repository that contains the product package.

If you install Installation Manager and then install the product, you must set the repository preferences manually.

For more information about installing the product from the command prompt in the silent mode, see IBM Installation Manager Knowledge Centre.

1. If you are installing from compressed files, such as .zip or ISO files, extract the files into a common directory. Extract the disk images to directories that are named `/disk1,/disk2`, and so on. Extract the Setup disk image to a directory that is named `RST_SETUP`. The Setup disk contains the launchpad program.
2. If you are installing from a CD, insert the first product disc into your CD drive. If autorun is enabled on your workstation, then the launchpad will start automatically. Otherwise, start the launchpad program manually. **Choose from:**

- **Windows** Run the launchpad.exe command, which is located in the root directory of the Setup disk installation image.
- **Linux** Run the launchpad.sh command, which is located in the root directory of the Setup disk installation image.
- **Mac OS X** From the command line terminal, run `open Launchpad.app` from the root directory of the Setup disk installation image.

> **Note:** Mac OS commands are case-sensitive.

3. Select a language in which to run the launchpad and Installation Manager.
4. Select the product to install from the launchpad menu.

   **Result**

   The **Install Packages** window opens.
5. Click a product package to highlight it.

   **Result**

   The description of the package is displayed in the **Details** pane at the end of the screen.
6. To search for updates to the product packages, click **Check for Other Versions, Fixes, and Extensions**. If updates for a product package are found, then they are displayed in the **Installation Packages** list on the **Install Packages** page under their corresponding products. Only recommended updates are displayed by default.

   **Choose from:**
   - To view all updates that are found for the available packages, click **Show all versions**.
   - To display a package description in the **Details** pane, click the package name. If additional information about the package is available, such as a `readme` file or release notes, a **More info** link is included at the end of the description text. Click the link to display the additional information in a browser. To fully understand the package that you are installing, review all information.

     > **Note:** For Installation Manager to search the predefined IBM® update repository locations for the installed packages, the **Search the linked repositories during installation and updates** preference on the **Repositories** preference page must be selected. This preference is selected by default. Internet access is also required. A progress indicator shows that the search is taking place. You can install updates at the same time that you install the base product package.

7. Select the product package and any updates to the package to install. Updates that have dependencies are automatically selected and cleared together. Click **Next** to continue.

📝 **Note:** If you install multiple packages at the same time, then all the packages are installed into the same package group.

8. On the **Prerequisite** page, if a supported version of IBM® Rational® License Key Administrator is not installed, a warning message is displayed. A supported version of Rational® License Key Administrator comes with the product. To administer a license server, you must install a supported version of Rational® License Key Administrator. If you use the launchpad program to install the product, Rational® License Key Administrator is listed on the **Install Packages** page. If you start Installation Manager, you must add the repository for Rational® License Key Administrator to install it at the same time as the product. Click **Next** to continue.

9. On the Licenses page, read the license agreement for the selected package. If you selected more than one package to install, there might be a license agreement for each package. On the left side of the **License** page, click each package version to display its license agreement. The package versions that you selected to install (for example, the base package and an update) are listed under the package name.

   a. If you agree to the terms of all of the license agreements, click **I accept the terms of the license agreements**.

   b. Click **Next** to continue.

10. On the Location page, type the path for the *shared resources directory* in the **Shared Resources Directory** field, or accept the default path. The shared resources directory contains resources that can be shared by one or more package groups. Click **Next** to continue.

    The default path to use follows:
    - ▶Windows `C:\Program Files\IBM\IBMIMShared`
    - ▶Linux `/opt/IBM/IBMIMShared`

    ⚠ **Important:** You can specify the shared resources directory only the first time that you install a package. Use your largest disk for this to help ensure adequate space for the shared resources of future packages. You cannot change the directory location unless you uninstall all packages.

11. On the **Location** page, create a *package group* to install the product package into or if this is an update, use the existing package group. A package group represents a directory in which packages share resources with other packages in the same group. To create a new package group:

    a. Click **Create a new package group**.

    b. Type the path for the installation directory for the package group.
       The name for the package group is created automatically.

       The default path follows:
       - ▶Windows `C:\Program Files\IBM\SDP`
       - ▶Linux `/opt/IBM/SDP`

    c. Click **Next** to continue.

12. On the next **Location** page, you can choose to extend an existing Eclipse IDE that is installed on your computer, which adds the functions in the packages that you are installing. You must have Eclipse Version 3.6 with the latest updates from eclipse.org to select this option.

    **Choose from:**
    - If you do not want to extend an existing Eclipse IDE, click **Next** to continue.
    - To extend an existing Eclipse IDE:
        a. Select **Extend an existing Eclipse**.
        b. In the **Eclipse IDE** field, type or navigate to the location of the folder that contains the eclipse executable file (`eclipse.exe` or `eclipse.bin`). Installation Manager checks whether the Eclipse IDE version is valid for the package that you are installing. The **Eclipse JVM** field displays the Java™ Virtual Machine (JVM) for the IDE that you specified.
        c. Click **Next** to continue.

13. On the **Features** page under **Translations**, select the languages for the package group. The corresponding translations for the user interface and documentation for the product package will be installed.

14. On the next Features page, select the package features to install.
    a. **Optional:** To see the dependency relationships between features, select **Show Dependencies**.
    b. **Optional:** Click a feature to view its brief description under **Details**.
    c. Select or clear features in the packages. Installation Manager automatically enforces any dependencies with other features and displays updated download size and disk space requirements for the installation.
    d. When you are finished selecting features, click **Next** to continue.

15. On the common licensing configuration page, type the TCP/IP port number and host name of the license servers to use to configure licensing on the workbench computer. Separate the port number and host name with the at sign (@). Separate the port-host pairs with semicolons (;). To use the default port, omit the port number. If you do not know the port numbers and names of license servers to use, you can configure the license servers after installation by using Rational® License Key Administrator.

    **Example**

    For example, to configure three license servers that are named license1, license2, and license3 to use port 27000, the default port, and port 1765 respectively, enter this text: `27000@license1;@license2;1765@license3`

    Click **Next**.

16. On the **Summary** page, review your choices before installing the product package. To change the choices that you made on previous pages, click **Back**, and make your changes. When you are satisfied with your installation choices, click **Install** to install the package.

    **Result**

    A progress indicator shows the percentage of the installation that is completed.

17. When the installation process is complete, a message confirms the completion of the process.

a. Click **View log file** to open the installation log file for the current session in a new window. You must close the **Installation Log** window to continue.

b. In the Install Package wizard, select whether to start the product when you exit.

c. Click **Finish** to start the selected package. The Install Package wizard closes and you are returned to the **Start** page of Installation Manager.

18. License the product.

## Installing Rational® Performance Tester Agent

You must install Rational® Performance Tester Agent on different computers to apply load on the server that hosts the application under test.

**Before you begin**

- Before upgrading the 64-bit Rational® Performance Tester Agent from version 8.6 to 8.7 or later, you must first remove the OCR Support feature that was installed with version 8.6 and then upgrade.

**About this task**

If you use the Launchpad program to install the product, Installation Manager gets installed automatically if you do not have it on your computer and starts the product installation by using the preconfigured repository that contains the product package.

If you install Installation Manager and then install the product, you must set the repository preferences manually.

For more information about installing the product from the command prompt in the silent mode, see IBM Installation Manager Knowledge Centre.

1. Complete one of the following steps:

a. If you are installing from compressed files, such as .zip or ISO files, extract the files into a common directory. Navigate to `Common_Directory`/RPTAGENT_SETUP/disk1/`Platform_Directory` and double-click **install.exe**.

b. If you are installing from a CD, insert the first product CD into your CD drive. If autorun is enabled on your workstation, the launchpad starts automatically. Otherwise, start the launchpad program manually.

- **Windows** Run the launchpad.exe command, which is located in the root directory of the Setup disk installation image.
- **Linux** Run the launchpad.sh command, which is located in the root directory of the Setup disk installation image.
- **Mac OS X** From the command line terminal, run `open Launchpad.app` from the root directory of the Setup disk installation image.

> **Note:** Mac OS commands are case-sensitive.

2. Select a language in which to run the launchpad and Installation Manager.
3. Click a product package. Its description is displayed in the **Details** pane at the end of the screen.
4. To search for updates to the product packages, click **Check for Other Versions, Fixes, and Extensions**. Make sure you are connected to the Internet. If updates for a product package are found, they are displayed in the **Installation Packages** list on the **Install Packages** page under their corresponding product. Only recommended updates are displayed by default.

   **Choose from:**
   - To view all of the updates that are found for the available packages, click **Show all versions**.
   - To display a package description under **Details**, click the package name. If additional information about the package is available, such as a `readme` file or release notes, a **More info** link is included at the end of the description text. Click the link to display the additional information in a browser. To fully understand the package that you are installing, review all of the information.

   > **Note:** For Installation Manager to search the predefined IBM® repository locations for the installed packages, select the **Search service repositories during installation and updates** check box on the **Repositories** preference page. This check box is selected by default. Internet access is also required. A progress indicator shows that the search is taking place. You can install updates at the same time that you install the base product package.

5. Select the product package and updates to the package to install. Updates that have dependencies are automatically selected and cleared together. Click **Next** to continue.

   > **Note:** If you install multiple packages at the same time, all the packages are installed into the same package group.

6. On the **Licenses** page, read the license agreement for the selected package:
   a. Read and understand the terms of all of the license agreements before clicking **I accept the terms of the license agreements**.
   b. Click **Next** to continue.
7. On the **Location** page, in the **Shared Resources Directory** field, type the path for the *shared resources directory* or accept the default path. The shared resources directory contains resources that one or more package groups can share. Click **Next** to continue.

The following default paths are provided:

- ◦ **Windows** `C:\Program Files\IBM\IBMIMShared`
- ◦ **Linux** `/opt/IBM/IBMIMShared`
- ◦ **Mac OS X** `/Applications/IBM/IBMIMShared`

> ⚠ **Important:** You can specify the shared resources directory only for the first time when you install a package. You must use your largest disk for this directory to help ensure adequate space for the shared resources of future packages. You cannot change the directory location unless you uninstall all packages.

8. On the **Location** page, create a package group to install the product. If you are updating the product, use the existing package group. A package group represents a directory in which packages share resources with other packages in the same group. To create a package group, complete these steps:

    a. Click **Create a new package group**.

    b. Type the path for the installation directory for the package group.
       The name for the package group is created automatically.

       The following default paths are provided:
       - ▪ **Windows** `C:\Program Files\IBM\SDP`
       - ▪ **Linux** `/opt/IBM/SDP`
       - ▪ **Mac OS X** `/Applications/IBM/SDP`

    c. Click **Next** to continue.

9. Rational® Performance Tester Agent does not support extending an existing Eclipse IDE, so you can ignore this page.

10. On the **Features** page, under **Translations**, select the languages for the package group. The corresponding translations for the user interface and documentation for the product package will be installed.

11. On the next **Features** page, select the features to install.

    > 📝 **Note:**
    > - ◦ The Load Generation Agent is used to generate a load on the system under test and gather data for the Response Time Breakdown feature.
    > - ◦ Installation Manager automatically enforces any dependencies with other features and displays updated download sizes and disk space requirements for the installation.

    a. **Optional:** To see the dependency relationships between features, select **Show Dependencies**.

    b. **Optional:** Click a feature to view its brief description under **Details**.

    c. After you are finished selecting features, click **Next** to continue.

12. On the next **Features** page, configure the agent:

a. `Windows`: The Web UI test panel is applicable only if you are shell-sharing Rational® Performance Tester and Rational® Functional Tester and executing a Web UI test on the remote agent computers. If you are not running a Web UI test, do not select the **The agent will be used primarily to support remote execution of Web UI tests** check box so that Majordomo runs as a service. When Majordomo runs as a service, it starts automatically after the computer reboots. However, if you must run the Web UI test on remote agent computers, select the check box so that, after the installation of the agent, Majordomo runs as a batch file instead of a service. When Majordomo runs as a batch file, it stops after the machine reboots. You must restart Majordomo by double-clicking `AgentInstallDir/Majordomo/Majordomo.bat`.

On Linux, this check box is not available, and by default, the agent runs as a service and can run a Web UI test.

Click **Next**.

b. For Load Generation Agent, type the host name of the workbench . If needed, change the port number. You can also change the port number after you install Rational® Performance Tester Agent. For more information, see Configuring ports for agents.

c. To connect the agent with IBM® Rational® Test Automation Server, specify the fully qualified domain name in **Server Hostname**, the port number, and the offline user token. For information about offline user token, see Managing access to Rational® Test Automation Server.

> 📝 **Note:** Use the default port number 443 to connect to the server.

d. Click **Next**.

13. On the **Summary** page, review your choices before installing the product package. To change the choices that you made on previous pages, click **Back**, and make your changes. When you are satisfied with your installation choices, click **Install** to install the package.
    **Result**
    A progress indicator shows the percentage of the installation that is complete.

14. When the installation process is complete, a message confirms the completion of the process.
    a. Click the **View log file** button to open the installation log file for the current session in a new window. You must close the **Installation Log** window to continue.
    b. Click **Finish** to install the selected package.

**What to do next**

After the installation, the agent starts polling the workbench based on the IP address or the host name provided by you. You can check the status of agents from the workbench. See Checking the status of agents.

# Installing Rational® Performance Tester Agent on an AIX 7.1 or later

IBM® Rational® Performance Tester Agent includes the Load Generation agent to generate load for the application under test by using the virtual users, to gather data for the Response Time Breakdown feature, and in support of the startup and control of web services stubs in the SOA protocol. When you install the Agent, the Data Collection Infrastructure application is also installed. This application is used to instrument the servers for Response Time Breakdown.

**About this task**

By starting the installation process from the launchpad program, Installation Manager is automatically installed if it is not already on your computer. Furthermore, Installation Manager starts preconfigured with the location of the repository that contains the product package.

If you install and start Installation Manager directly, then you must set repository preferences manually. To learn how to install the product from a command prompt in silent mode, see the Installing Silently section in the Installation Manager Knowledge Center.

1. Log in as the root user before you begin installing the product.
2. If you are installing from compressed files, such as .zip or ISO files, extract the files into a common directory. Extract the disk images to directories named /disk1, /disk2, and so on. Extract the Setup disk image to a directory. The Setup disk contains the launchpad program.
3. Run the `launchpad.sh` file, which is located in the root directory of the Setup disk installation image.
4. Select a language in which to run the launchpad and Installation Manager, and click **OK**.
5. Select the product to install from the launchpad menu. The Install Packages window opens.
6. 
7. Click a product package to highlight it. The description of the package is displayed in the Details pane at the end of the screen.
8. **Optional:** To search for updates to the product packages, click **Check for Other Versions and Extensions**. If updates for a product package are found, then they are displayed in the Installation Packages list on the Install Packages page under their corresponding products. Only recommended updates are displayed by default.
9. **Optional:** To view all updates that are found for the available packages, click **Show all versions**.
10. **Optional:** To display a package description, under Details, click the package name. If additional information about the package is available, such as a readme file or release notes, a **More information** link is included at the end of the description text. Click the link to display the additional information in a browser. To fully understand the package you are installing, review all information beforehand.
11. Click **Next** to continue.

    **Note:** If you install multiple packages at the same time, then all the packages are installed into the same package group.

12. On the Licenses page, read the license agreement for the selected package. If you select more than one package to install, there might be a license agreement for each package. On the left side of the License page,

click each package version to display its license agreement. The package versions that you select to install (for example, the base package and an update) are listed under the package name.

13. If you agree to the terms of all the license agreements, click **I accept the terms of the license agreements**; then click **Next** to continue.

14. On the Location page, type the path for the shared resources directory in Shared Resources Directory, or accept the default path. The shared resources directory contains resources that can be shared by one or more package groups. The default path is `/opt/IBM/IBMIMShared`. Click **Next** to continue.

> 📝 **Note:** You can specify the shared resources directory only the first time that you install a package. Use your largest disk for this operation to help ensure adequate space for the shared resources of future packages. You cannot change the directory location unless you uninstall all packages.

15. On the Location page, create a package group to install the product package into, or, if this is an update, use the current package group. A package group represents a directory in which packages share resources with other packages in the same group. The default path is .

16. To create a package group, click **Create a new package group**, and then type the path for the installation directory for the package group. The name for the package group is created automatically. Click **Next** to continue.

17. On the Features page under Translations, select the languages for the package group. The corresponding translations for the user interface and documentation for the product package will be installed.

18. On the next Features page, select the package features to install and click **Next**.

19. On the next Features page, for Load Generation Agent, type the workbench host name. If you do not want to use 7080 as the port number, change the port number. You can also change the port number after you install Rational® Performance Tester Agent.

20. On the Summary page, review your choices and click **Install**.

## Uninstalling the product by using Installation Manager

When you no longer require  Rational® Service Tester for SOA Quality, you can use Installation Manager to uninstall Rational® Service Tester for SOA Quality that you have installed.

**Before you begin**

You must have completed the following tasks:

- Installed Installation Manager.

- Closed any open windows of  Rational® Service Tester for SOA Quality.

- Closed any open web browsers.

- Closed all the other applications that are enabled by  Rational® Service Tester for SOA Quality.

1. Open Installation Manager.
2. Click **Uninstall**.

3. Select the Rational® Service Tester for SOA Quality package checkbox on the **Uninstall Packages** window, and then click **Next**.

4. Review the list of packages that are ready to uninstall, and then click **Uninstall**.
   **Result**
   The **Complete** page is displayed after the uninstallation process is complete.

5. Click **Finish** to exit the Installation Manager wizard.

**Results**

You have uninstalled Rational® Service Tester for SOA Quality from your computer.

# License management

Licensing for your installed IBM® software and customized packages is administered through the Manage Licenses wizard in the IBM® Installation Manager. The Manage Licenses wizard displays license information for each installed package.

Using the Manage Licenses wizard, you can apply a license to a product or upgrade trial versions of an offering to a licensed version by importing a product activation kit. You can also enable floating license enforcement for offerings with trial or permanent licenses to use floating license keys from a license server.

For more information about managing licenses for your Rational® product, see these resources:

- Product Activation of Eclipse-based Rational products

- Rational licensing support

- Rational License Key Server Knowledge Center

## License descriptions

As a purchaser of an IBM® Rational® software product, you can choose from four types of product licenses:

- Authorized User licenses on page 69
- Floating licenses on page 70
- Token licenses on page 71
- Processor value unit (PVU) licensing on page 71

The best choice for your organization depends upon how many people use the product, how often they require access, and how you prefer to purchase software.

**Authorized User licenses**

An IBM® Rational® Authorized User license authorizes an individual to use a Rational® software product. You must obtain an Authorized User license for each individual user who accesses the product in any manner. An Authorized User license cannot be reassigned unless you replace the original assignee on a long-term or permanent basis.

For example, if you purchase one Authorized User license, you can assign that license to one individual who can use the Rational® software product exclusively. The Authorized User license does not authorize a second individual to use that product at any time, even if the licensed individual is not using the product.

> **Note:** If you purchase IBM® Rational® Performance Tester as part of IBM® Rational® Test Workbench, you do not require runtime licenses such as protocol key to run a schedule.

An IBM® Rational® Authorized User Fixed Term License (FTL) authorizes an individual to use a Rational® software product for a specific length of time (the term). Purchasers must obtain an Authorized User FTL for each individual user who accesses the product in any manner. An Authorized User FTL cannot be reassigned unless the purchaser replaces the original assignee on a long-term or permanent basis.

> **Note:** When you purchase an Authorized User FTL under the IBM® Passport Advantage® Express® program, IBM® automatically extends the license term for an additional year at the prevailing price unless you notify IBM® before the license expires that you do not want an extension. The subsequent FTL term starts when the initial FTL term expires. The price for this subsequent term is currently 80% of the initial FTL price, but is subject to change.

## Floating licenses

An IBM® Rational® Floating license is a license for a single software product that can be shared among multiple team members; however, the total number of concurrent users cannot exceed the number of floating licenses you purchase. For example, if you purchase one floating license for a Rational® software product, any user in your organization can use the product at any given time. Another person who wants to access the product must wait until the current user logs off.

To use floating licenses, you must obtain floating license keys and install them on a Rational® License Server. The server responds to user requests for access to the license keys; the server grants access to the number of concurrent users that equals the number of licenses the organization purchased.

Floating license enforcement provides these benefits:

- License compliance enforcement throughout the organization
- Fewer license purchases
- License keys served for IBM® Rational® Team Unifying and IBM® Rational® Software Delivery Platform desktop products from the same license server
- Some versions of Rational® products require an upgraded version of the Rational® License Server. See this support article for license upgrade information: http://www.ibm.com/support/docview.wss?uid=swg21250404

**Token licenses**

The token-based license model means that you can buy a certain number of token licenses. If you use a Rational® tool that checks out a feature that is token-based, the feature line in the license file specifies the number of tokens that are checked out. Token-based licenses can only be used with floating licenses. They cannot be used for authorized user license. For more details about token licensing, contact your local IBM® marketing representative.

**Processor value unit (PVU) licensing**

Processor value unit (PVU) licensing is for providing you pricing structures that are responsive to both the type and number of processors that are available to installed products.

Entitlements can be full capacity or subcapacity. Under the processor value unit licensing structure, you license software based on the number of value units assigned to each processor core. For example, processor type A is assigned 80 value units and processor type B is assigned 100 value units. If you license a product to run on two type A processors, you must acquire an entitlement for 160 value units. If the product is to run on two type B processors, the required entitlement is 200 value units.

The processor value units table, which assigns a number of value units to each supported processor type, is regularly updated to provide for the introduction of new processor technologies. Agents retrieve information about the number and type of processor on the monitored computer or partition and the table is used to determine the level of license use in terms of processor value units.

For more information about managing licenses for your Rational® product, see the IBM License Metric Tool Knowledge Center to learn about processor value unit licensing.

## Runtime license examples

To run tests, you must have the correct license keys installed.

The trial licensing key permits you to run  Rational® Performance Tester for 30 days from the initial installation. After 30 days from the first time you install the product, the trial license expires.

The following table shows the number of virtual users you can use, depending on the license keys that you have installed:

| Protocol | Trial license | Permanent license | Permanent with Protocol key | 500-user virtual tester key pack and protocol key |
|---|---|---|---|---|
| HTTP | 5 | 5 | n/a | 505 |
| Siebel | 5 | 1 | 5 | 505 |
| SAP(GUI) | 5 | 1 | 5 | 505 |
| SAP(Web) | 5 | 1 | 5 | 505 |
| Citrix | 5 | 1 | 5 | 505 |

| Protocol | Trial license | Permanent license | Permanent with Protocol key | 500-user virtual tester key pack and protocol key |
|---|---|---|---|---|
| TN3270 | 5 | 5 | n/a | 505 |
| Socket | 5 | 5 | n/a | 505 |
| SOA | 5 | 1 | 5 | 505 |

## Purchasing licenses

You can purchase new licenses if your current product license is about to expire or to acquire additional product licenses for team members.

1. Determine the type of license to purchase.
2. Go to ibm.com® or contact your IBM® sales representative to purchase the product license. For details, visit the IBM® web page on How to buy software.
3. Depending on the type of license you purchase, use the Proof of Entitlement that you receive and complete one of these steps to enable your product:
   **Choose from:**
   - If you purchase Authorized User licenses for your product, go to Passport Advantage®, and follow the instructions there for downloading your product activation kit. After you have downloaded the activation kit, import the product activation `.jar` file by using Installation Manager.

     Back up the product activation `.jar` file. If you uninstall the product and then install the product again, you might need to use the product activation `.jar` file to license the product again.

   - If you purchase floating licenses for your product, go to the IBM® Rational® Licensing and Download Center, and then click the link to connect to the IBM® Rational® License Key Center. There you can use your Proof of Entitlement to obtain floating license keys for your license server.

     Optionally, you can go to IBM® Passport Advantage® to download the activation kit for your product. After importing the activation kit, you can switch from a floating to a permanent license type if you use your computer offline for long periods.

**What to do next**

To import the activation kit or enable floating license support for your product, use the Manage Licenses wizard in IBM® Installation Manager.

## Enabling licenses

If you are installing the software for the first time or want to extend a license to continue using the product, you have options on how to enable licensing for your product.

Licenses for this product are enabled in two ways:

- Importing a product activation kit
- Enabling Rational® Common Licensing to obtain access to floating license keys

**Activation kits**

The Product Activation Kit CD contains the permanent license key for you product. You use IBM® Installation Manager to import the activation kit to your product.

To enable the PVU licensing capability, download the PVU activation kit from the Rational® Common Licensing server, unzip it, and use Installation Manager to import the activation kit.

**Floating license enforcement**

Optionally, you can obtain floating license keys, install IBM® Rational® License Server, and enable floating license enforcement for your product. Floating license enforcement provides these benefits:

- License compliance enforcement throughout the organization
- Fewer license purchases
- License keys served for IBM® Rational® Team Unifying and Software Delivery Platform desktop products from the same license server

> **Note:** Some 7.0 and later versions of Rational® products require an upgraded version of the Rational® License Server. See this support article for license upgrade information: http://www.ibm.com/support/docview.wss?uid=swg21250404

For more information about obtaining activation kits and floating licenses, see Purchasing licenses on page 72.

## Viewing license information for installed packages

You can review license information for your installed packages, including license types and expiration dates, from IBM® Installation Manager.

1. Start IBM® Installation Manager.
2. On the main page, click **Manage Licenses**.

**Results**

The package vendor, current license types, and expiration dates are displayed for each installed package.

## Product upgrade and migration

When you want to use the enhanced functionalities of  Rational® Performance Tester Rational® Service Tester for SOA Quality, you must upgrade to the latest version of the product software.

You can upgrade to  Rational® Performance Tester Rational® Service Tester for SOA Quality 10.1.1 only from 10.1.0 of the product.

**Remember:**

- You can upgrade to 10.1.1 from 10.1.0 of the product. After you installed  Rational® Service Tester for SOA Quality 10.1.1, at any point in time if you want to use 10.1.0, you can roll back from 10.1.1 to 10.1.0.

- If you installed  Rational® Service Tester for SOA Quality that is earlier than 10.1.0, you cannot upgrade the product to the latest version. You must uninstall the existing version of the product before installing the latest version of the product.

- After you installed  Rational® Service Tester for SOA Quality 10.1.1, at any point in time if you want to use the product that is earlier than 10.1.0, you cannot roll back to the previous version of the product. You must uninstall  Rational® Service Tester for SOA Quality 10.1.1, and then install the required version of the product.

## Migrating test assets to new version of the product

After you install a later version of the product and you choose to open the product from an old workspace, you are prompted to migrate test projects, tests, schedules, rules, and reports. Tests and schedules are migrated automatically when you modify and save them.

You cannot have two versions of the products installed on your computer at one time. Before you install a new version, uninstall the previous version of the product. If you update the product with IBM® Installation Manager, you do not have to uninstall the previous version. Uninstalling a previous version does not delete your test assets.

**Note:** When you want to uninstall the previous version of the product and then install the latest version, you must install the latest version in the same folder that contained the previous version. Thereby, you can avoid compilation errors in the project in the latest version.

When you open a project that contains an older test asset, a message is displayed in the **Test Navigator** view. Typically, you upgrade your tests, schedules, and rules.

⚠ The workspace contains resources produced in a     ✕
previous release. Click here to upgrade.
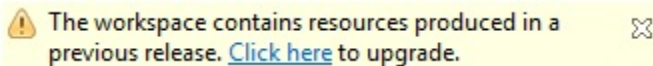
**Important:** Ensure that you back up the test assets before upgrading them for the new version of the product. Do not open a migrated test project with a previous version of the product.

If you leave tests, schedules, and rules unchanged, they will not have the new functions that current release adds. You can always save a modified test asset under a new name, which preserves the older asset. You can identify an older asset by its version, which is listed in brackets:

visitIBM [7.0.1.0]  ✕
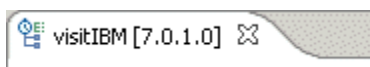
> **Note:** A new release might include enhancements to the default reports. When you run a test or schedule or open a report, you are prompted to upgrade reports to the latest version. If you upgrade the default reports to the latest version, you lose customizations that you have made to the reports.

If you encounter errors when you open a workspace from a different version of the product, reset the perspective. To reset the perspective, click **Window > Reset Perspective**.

# Integration with other products

You can integrate  Rational® Performance Tester Rational® Service Tester for SOA Quality with certain products to run tests, manage test assets, and create defects.

The following topics provides more information about the integration of  Rational® Performance Tester Rational® Service Tester for SOA Quality with other products:

## Integration plugin compatibility matrix

You can find information about the versions of the integration plugin that are compatible with .

The following table lists the versions of the integration plugin that are required to integrate Jenkins, Ant, and UrbanCode™ Deploy with .

> **Note:** You must download the required version of the integration plugin from the portal based on the existing version of . You can then integrate Jenkins, Ant, and UrbanCode™ Deploy with .

|  | **Ant plugin** | **Jenkins plugin** | **UrbanCode™ Deploy plugin** |
|---|---|---|---|
| 10.1.0 | RPT-Ant-5.0 | RPT-Jenkins-6.0 | RPT-UCD-5.0 |
| 10.1.1 | RPT-Ant-6.0 | RPT-Jenkins-7.0 | RPT-UCD-6.0 |

## Testing with Ant

You can use `ant` to run performance and compound tests from the command line. Starting from V2.0.0 of the `ant` plugin, you can run multiple tests simultaneously. of the `ant` plugin is supported in V10.0.2 of the testing products.

**Before you begin**

You must have completed the following tasks:

- Installed Installation Manager.

- Installed .

- Verified that you have test assets residing within .

- Downloaded the Ant plugin from the  portal on to the computer where you install the product.

  For more information about specific versions of plugin, see .

- Added `ant` to the *PATH* environment variable.

**About this task**

To run performance tests on Mac OS, you must add an environment variable that points to the installation directory of .

For example, `export TEST_WORKBENCH_HOME=/opt/IBM/SDP`.

> **Note:** For Windows™ and Linux®, the environment variable is set when you install the product.

1. Extract the following files from the downloaded plugin:

   ◦ Where, *x* is the version number of the `Ant` plugin.

   ◦

   ◦ `README.txt`
2. Open the file and provide required parameter values.
   For example,

   > **Note:** You can add an additional task and provide the details for each test to run multiple tests simultaneously.

   The following table explains each field.

   | Field | Description |
   | --- | --- |
   | **- name** | Required. The name of the test for the particular test product. |
   | **- work-space** | Required. The complete path to the Eclipse workspace. |
   | **- project-name** | Required. The path, including the file name of the project relative to the workspace. |
   | **- suite** | Required. The path, including the file name of the test to run relative to the project. A test can be a performance schedule or a compound test. |
   | **- varfile** | Optional. The complete path to the XML file that contains the variable name and value pairs. |
   | **- configfile** | Optional. The complete path to a file that contains the parameters for a test or schedule run. |

| Field | Description |
|---|---|
| - results | Optional. The name of the results file. The default result file is the test or schedule name with a time stamp appended. |
| - overwrite | Optional. Determines whether a results file with the same name is overwritten. The default value is true, which means the results file can be overwritten. |
| - quiet | Optional. Turns off any message output from the launcher and returns to the command shell when the run or the attempt is complete. |
| - vmargs | Optional. Java™ virtual machine arguments to pass in. |
| - swap-datasets | Optional. For a test or schedule, the default value is the dataset specified in the test editor or schedule editor. Overrides the default dataset value to run if required.<br><br>**Note:** You must use the **swapdatasets** option to replace dataset values during a test or schedule run. You must ensure that both original and new datasets are in the same workspace and have the same column names. You must also include the path to the dataset. For example: `/project_name/ds_path/ds_filename.csv:/project_-name/ds_path/new_ds_filename.csv`. You can swap multiple datasets that are saved in a different project by adding multiple paths to the dataset separated by a semicolon. For example: `/project_name1/ds_path/ds_filename.csv:/project_-name1/ds_path/new_ds_filename.csv;/project_name2/ds_path/ds_-filename.csv:/project_name2/ds_path/new_ds_filename.csv` |
| - export-stats | Optional. The complete path to a directory that can be used to store exported statistical report data. |
| - export-statre-portlist | Optional. A comma-separated list of absolute paths to custom report format files (.view files) to use when exporting statistical report data with the -exportstats option. |
| - export-statshtml | Optional. The complete path to a directory that can be used to export web analytic results. The results are exported in the specified directory. Analyze the results on a web browser without using the test workbench. |
| - user-comments | Optional. Add text within the double quotation mark to display it in the **User Comments** row of the report. |
| -imshared-loc | Optional. The complete path to IBMIMShared location, if it is not at the default location. |

3. Open a command prompt and navigate to the directory where you downloaded the `Ant` plugin.
4. Enter to run the test.

**Results**

You have run the test by using the `Ant` plugin.

**What to do next**

You can view that the `Ant` execution output is logged into the `logfile.txt` file, and a test log is created in a temp directory called .

## Integration with Azure DevOps

When you use Azure DevOps for continuous integration and continuous deployment of your application, you can create tests for your application in and run those tests in Azure DevOps pipelines. You can integrate Azure DevOps with by using the *IBM Rational Test Workbench* extension that is available in the **Visual Studio Marketplace** portal.

**Prerequisites**

Before you integrate Azure DevOps with , you must have completed certain tasks. See Prerequisites for Azure DevOps Integration.

**Overview**

You can use the *IBM Rational Test Workbench* extension that enables you to select any type of test created in that you can add to your task for the job in the Azure DevOps pipelines.

## Running tests in an Azure DevOps Pipeline

After you create the tests in for the application that you are testing, and after you install the *IBM Rational Test Workbench* extension in your organization, you can run the tests in Azure DevOps pipelines.

**Before you begin**
You must have completed the following tasks:

- Installed the *IBM Rational Test Workbench* extension in your organization. See Installing the *IBM Rational Test Workbench* extension.
- Installed an agent in your pipeline. See Azure Pipelines agents.

**About this task**

After you add the *IBM Rational Test Workbench* extension in your Azure DevOps organization, you can use an existing pipeline or create a new one to add test tasks. You can install an agent or use the one that you installed in your default agent pool. You can add the tests to your task for the agent job, configure the task, and then run the task in the Azure DevOps pipeline.

1. Open your **Organization** page in Azure DevOps and perform the following steps:

    a. Click the project you want to use.

    b. Initialize the repository by performing the following steps:

        i. Click **Repos** from the left pane.

       ii. Click **Initialize** from the **Initialize with a README or gitignore** section.

> ✏️ **Note:** Select the **Add a README** check box if it is not selected.

  c. Click **Pipelines** from the left pane.

  d. Click **Create Pipeline**.

  e. Click **Use the classic editor** to create a pipeline without YAML.

  f. Verify the project, repository, and branch for manual and scheduled builds, and then click **Continue**.

  g. Click **Empty job**.

2. Select **Pipeline** and complete the following steps:

  a. Change the name for the build pipeline if required.

  b. Select the **Agent pool** for your build pipeline.

    You can use the agent from the default agent pool or use the one you have installed.

  c. Select the **Agent Specification** for the agent if required.

3. Add a task to the agent job by completing the following steps:

  a. Click the **Add Task** icon ➕ for the agent job.
    **Result**

    The **Add tasks** pane is displayed.

  b. Search for the IBM tasks defined in the *IBM Rational Test Workbench* extension.
    **Result**

    The tasks that you can select are displayed.
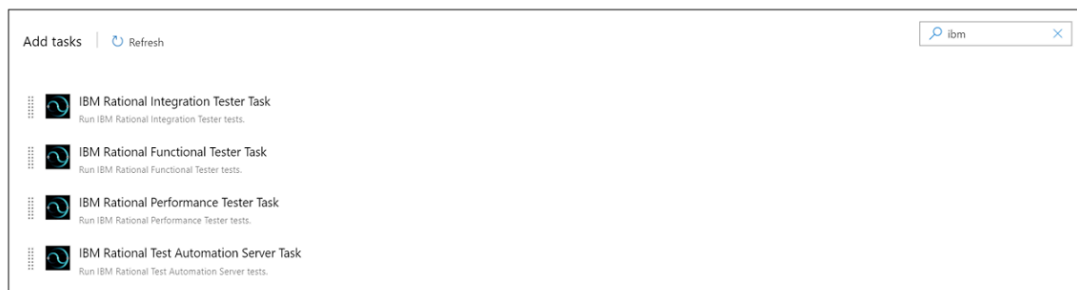


    Depending on the type of test that you have created in , you can select the type of task. You must use the following table to identify the task you must select:

| Type of test | Task to select |
| --- | --- |
|  |  |

c. Select the **IBM Rational Performance Tester** option, and then click **Add** to add the task to the agent job.
**Result**

The selected task is added to the agent job and it is displayed with a warning that some settings require attention. You must configure the settings mentioned in .

You can also remove the tasks that are not required in your job. Select the tasks in the list that you want to remove. You can then right-click the tasks, and click **Remove selected task(s)** to remove them.

4. Configure the settings by performing the following steps:

   a. Select the task version from the list if required.

   b. Follow the action for the performance task by referring to the following table:

   > **Note:** All mandatory fields are marked with an asterisk (*) in the UI.

   c. Expand **Control Options** and configure the settings for your task if required.

   d. Expand **Output Variables** and configure the settings for your task if required.

5. Select the following options:

   a. Click **Save** to save the configured settings for the task.

   > **Note:** The task is not queued for a run.

   You can save the task to a build pipeline and opt to run the build at a later time.

   b. Click **Save & queue** to save the configurations and queue the run in the pipeline.
   **Result**

   The **Run pipeline** dialog box is displayed.

6. Complete the following steps:

   a. Enter a comment for the test in the **Save comment** field.

   b. Select the agent that you configured for the test from the **Agent pool** list.

   c. Select the agent specification from the **Agent Specification** list for the agent if required.

   d. Select the branch from the **Branch/tag** list.

e. Add the variables and demands for the task run from the **Advanced Options** pane if required.

f. Select the **Enable system diagnostics** check box for a detailed log view.

g. Click **Save and run**.
   **Result**

   The `pipeline summary` page displays the progress of the job run.

**Results**

You have run the tests for the application by using the ,*IBM Rational Test Workbench* extension in the Azure DevOps pipeline.

**What to do next**

You can open the job to view the task logs from the `pipeline summary` page.

You must click the task to open the **Task** page to view the test results.

---

Related information

## Integrating with Apache JMeter

Starting from 9.5, you can use JMeter tests extension with IBM® Rational® Performance Tester to execute JMeter tests.

In  Rational® Performance Tester, you have the option to import JMeter tests, add tests to a schedule or compound test to run them. Additionally, JMeter test helps to simulate a heavy load on a server, group of servers, or to investigate overall sample response time under different load types.

JMeter samples are terminal elements in JMeter tests that inform JMeter to send requests to a server and wait for a response. When you execute a JMeter test, a JMeter performance report is generated during a run and saved after a run. This report contains the data most significant to the run, shows the response trend of the lowest 25 samples in the test, and graphs the response trend of each sample for a specified interval.

With JMeter test, you can load and test the performance of an application that uses the following protocols:

- HTTP/HTTPS
- SOAP/REST
- FTP
- LDAP
- MOM
- SMTP/POP3/IMAP
- TCP

**Installing the JMeter**

To work with JMeter tests, you must download the JMeter executable from https://jmeter.apache.org/download_jmeter.cgi and unzip it. To run the JMeter test as part of schedule, you must install JMeter on the remote agent machine and set the JMETER_HOME environment variable to the root installation folder.

To run the JMeter tests, you must either specify the Apache JMeter path in **Preferences** or set the environment variable.

> **Note:** When you run the compound test or schedule, an error message is displayed, if you have not specified the preferences or set the environment variable.

- To access the preference settings for Apache JMeter, click **Window > Preferences > Test > Apache JMeter** and point it to the apache-jmeter-5.0 directory.
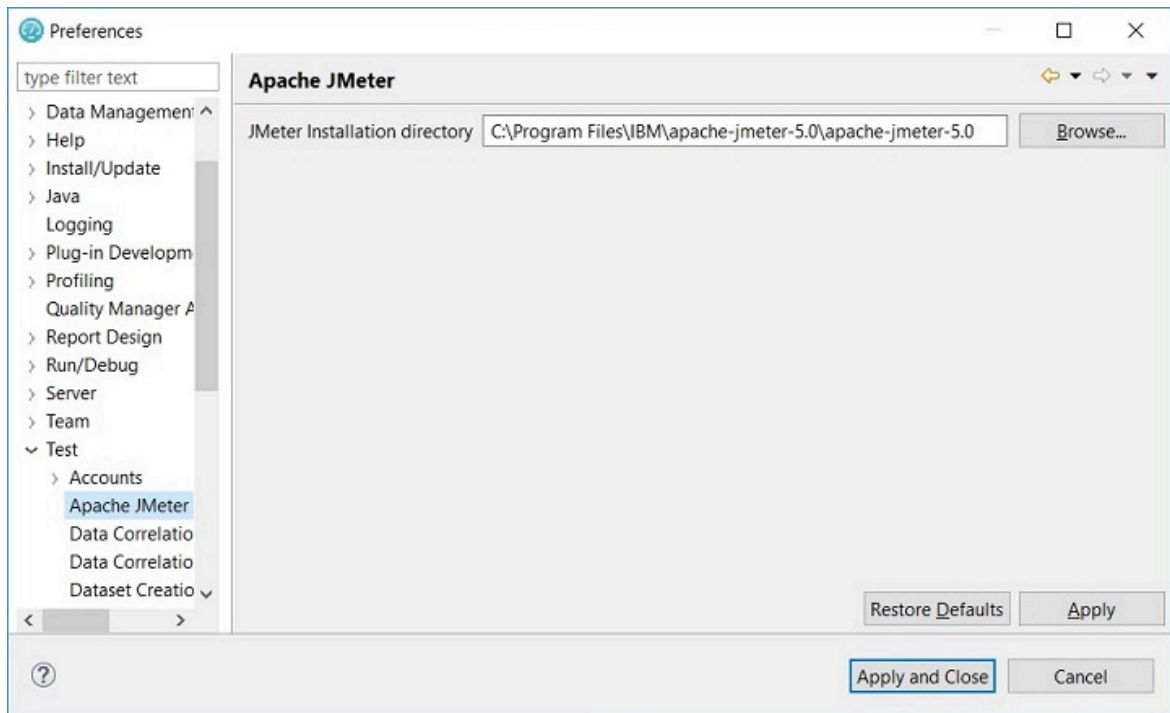
• Set the environment variable JMETER_HOME and point it to the apache-jmeter-5.0 directory.
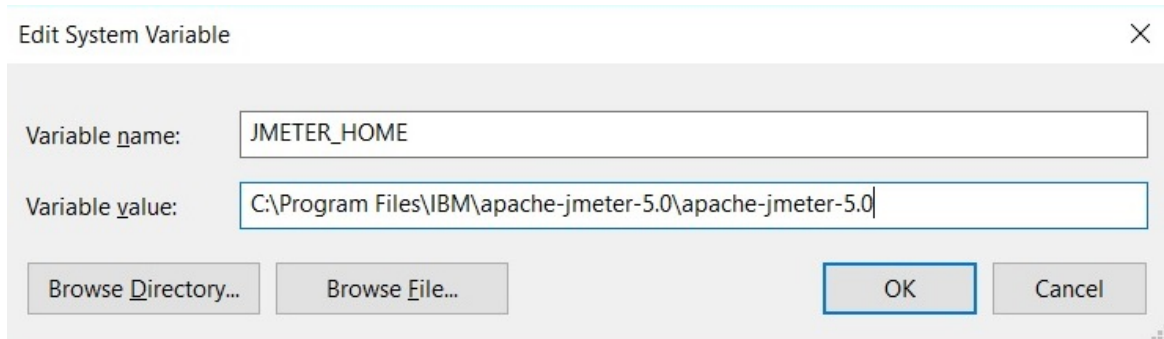


## Importing a JMeter test to a Test Workbench project

You can import your JMeter tests to IBM® Rational® Performance Tester to run them as part of VU schedule.

**About this task**

If you have an existing JMeter test, you can import the test by dragging and dropping a JMeter file (JMX file) into the project in the test navigator. Alternatively, you can use the following procedure to import the tests.

1. In the Test Navigator, right-click and click **Import**.
2. In the Import dialog box, expand **General** in the source list, select **File System** and then click **Next**.
3. Specify the directory where the JMeter test resides.
   Click **Browse** to select a directory from where you can import the JMeter test. The JMeter test assets in the folder you selected are displayed.
4. Select the JMeter test you want to import.
5. Click **Browse** to choose the location to import JMeter test.
6. Click **Finish**. The imported JMeter test is displayed in the **JMeter Tests** folder.

## Adding a JMeter test to an existing VU schedule

You can add a JMeter test to an existing VU schedule to test the performance of both static and dynamic resources and web applications.

**About this task**

When you add a JMeter test to a VU schedule, a user group with a loop is created and this loop contains the JMeter test invocation.

1. In the Test Navigator, browse to the schedule and double-click it. The schedule editor opens.
2. In the VU schedule editor, right-click the Schedule, and then click **Add > JMeter Test**.
3. In the Select Tests window, expand the project name and **JMeter Tests** folder and choose the test that you want to add.
4. Click **OK**. A new User Group with a loop that contains the JMeter test invocation is created.

**What to do next**

You must run the schedule or compound test, to view the statistics on the executed sequences.

- To install the JMeter, see Installing the JMeter on page 82.

- To run the schedule or compound test, see Running a local schedule or test on page 287 or Running compound tests.

## Converting JMeter tests to VU schedule

You can convert a JMeter test to a VU schedule to load and test the performance of an application under test.

**About this task**

Rational® Performance Tester analyzes the selected JMeter test to add the number of users and loop iteration count in the VU schedule. When you convert the JMeter test to a VU schedule, the following events occur:

- The load information identified within the **Thread Group** nodes from the original JMeter test is examined to build a new VU schedule.

- The content of each **Thread Group** node is extracted from the original JMeter test and copied into a new JMeter test.

- The new JMeter test is then invoked by the VU schedule as an external test.

  **Note:** The extracted JMeter test does not contain any load information such as the number of users and loop count, because the VU schedule manages all the information.

  **Remember:**

  - The content of the original **Thread Group** is not considered during the conversion process. Therefore, if there are any loops in the JMeter **Thread Group**, those cannot convert into a loop element in the new VU schedule.

  - If you have a JMeter test with more than one **Thread Group** node, each **Thread Group** is extracted to separate the JMeter test.

  - If you have a complex JMeter test, you must extract the functionalities that are included in the **Modules** or **Include controllers** into another JMeter tests. You must then add those JMeter tests to a VU schedule to run it.

1. Browse and select the JMeter test from the **Test Navigator**.
2. Right-click the selected test, and then click **Convert to VU Schedule**.
3. Verify that the name of the schedule is same as name of the JMeter test.

4. Click **Finish**.

   **Result**

   The schedule editor opens.

**What to do next**

You must perform the following tasks:

- Install JMeter on the remote agent machine and set the JMETER_HOME environment variable to the root installation folder to run the JMeter test as part of schedule. See Installing the JMeter on page 82.

- Run the converted test assets against successive builds of the application under test. See Running a local schedule or test on page 287.

- Analyze the test results that are recorded. See Running compound tests.

## JMeter Performance report

The JMeter performance report summarizes the validity of the run, shows the average sample response time for the requests in the test, and graphs the sample response time of each sample for a specified interval.

### Overall page

The Overall page provides a progress indicator that shows the status of the run and a bar chart that shows percentage of passed JMeter samples.

### Summary page

The Summary page provides important information about the run. This page shows the following Run Summary information:

- The name of the test.

- The number of active users and the number of users who completed testing. This number is updated during the run.

- The elapsed time is the run duration, which is displayed in hours, minutes, and seconds.

- The status of the run. For example, the status can be Initializing Computers, Adding Users, Running, Transferring data to test log, Stopped, or Complete.

### JMeter Samples page

JMeter samples are terminal elements in JMeter tests that informs JMeter to send requests to a server and wait for a response. The JMeter Samples page shows the average sample response time for all the requests in the test. The bar chart shows the average sample response time for all the requests. Each bar represents a sampler of the JMeter test. The corresponding table provides the following additional information:

- The minimum, average, and maximum duration for each sample in the run.

- The standard deviation of the duration.

- The completed sample rate and total number of completed samples per request.

### JMeter Transaction page

The JMeter Transaction page shows the average transaction response time for all the requests in the test. The bar chart shows the average transaction response time for all the requests. Each bar represents a page that you visited during recording. The corresponding table provides the following additional information:

- The minimum, average, and maximum duration for each transaction in the run.

- The standard deviation of the duration.

- The completed transaction rate and total number of completed transaction per request.

### Samples versus Time Summary page

The Samples versus Time Summary page shows the sample response trend as graphed for a specific interval. The Sample Response versus Time graph shows the sample response time for all the requests during the run. Each point on the graph is an average of what has occurred during that sample interval. The table after the graph lists the total average duration for all requests in the run and the standard deviation. To set the sample interval value, open the schedule, choose the **Statistics** tab from the drop-down menu, and then view or modify **Statistics sample interval**.

### Samples versus Time Detail page

The Samples versus Time Detail page shows sample response trend for each of the request in the test. The line graph shows the average sample response time of each requests for a specific interval. The table after the graph provides the minimum, average, and maximum duration for the run and the standard deviation in the average sample response time.

### Sample Throughput page

The Sample Throughput page summarizes the frequency of requests that are transferred per sample interval. The line graph on the left side shows the sample rate and passed sample rate per interval for all samples. The summary table after the graph lists the passed rate of total samples and counts for each passed samples. The line graph on the right side shows active users and the users who completed testing, per interval, over the course of a run. You can set the Statistics sample interval value in the schedule, as a schedule property. As the run nears completion, the number of active users decreases and the number of completed users increases. The summary table after the graph lists the active and completed users for the entire run.

To set the sample interval value, open the schedule, choose the **Statistics** tab from the drop-down menu, and then view or modify **Statistics sample interval**.

**Server Throughput page**

The Server Throughput page lists the rate and number of bytes that are transferred per interval and for the entire run. The page also lists the status of the virtual users for each interval and for the entire run. The line graph on the left side shows the rate of bytes sent and received per interval for all intervals in the run. The summary table after the graph lists the total number of bytes sent and received and bytes sent and received throughput rate for the entire run.

The line graph on the right side shows active users and users who are completed testing, per interval, over the course of a run. You set the Statistics sample interval value in the schedule, as a schedule property. As the run nears completion, the number of active users decreases and the number of completed users increases. The summary table after the graph lists the active and completed users for the entire run.

**Server Health Summary page**

The Server Health Summary page gives an overall indication of how well the server is responding to the load. The bar chart shows the total number of samples and total number of passed samples for the run. The table under the bar chart lists the same information.

**Server Health Detail page**

The Server Health Detail page provides specific details for 25 samples with the lowest success rates. The summary table lists the number of samples completed and passed in the run, and the passed sample percent and completion rate.

# EGit integration

EGit is an Eclipse plugin for the Git version control system. You can store your test assets in the Git repository and use EGit for the daily version control operations.

When you install the product, by default EGit is part of the product. To avoid the clutter, EGit is a separate Eclipse perspective in the product. For more information about EGit, read its documentation.
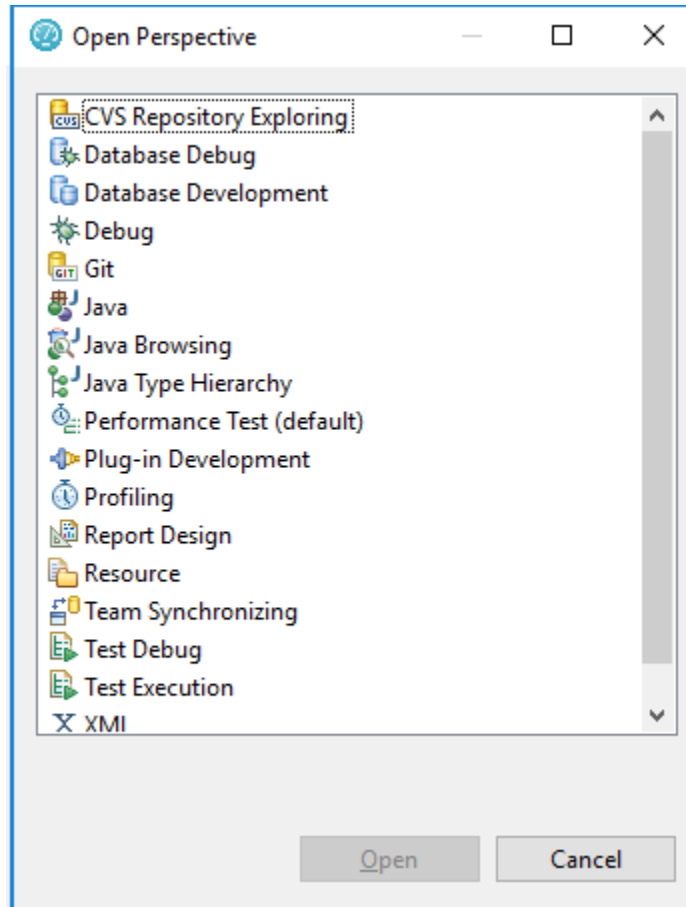
In Eclipse perspective, when you initialize a new Git repository for a project, a `.gitignore` file is created in the project folder, by default. While committing the contents of a project to Git, the `.gitignore` file ignores the report files.

✏️ **Note:**

- After you pull a project in Git perspective, 'Project is missing required Library' error appears in the **Verify Problems** tab. This error occurs due to the .classpath file which is specific to a workspace or system and needs to be pointed to the newly imported location. To resolve this issue:
    1. Under **Package Explorer**, from the required project, navigate to **Java Build Path > Libraries**.
    2. Delete all the jar files that are missing after the pull. These files are marked with a red cross.
    3. Playback the project. The required jar files are added to the project.

- You can also specify additional file types in the `.gitignore` file so that these file types are ignored while committing the project contents to Git.

## Integration with Engineering Test Management

You can integrate IBM® Engineering Test Management (formerly known as IBM® Rational® Quality Manager) with to initiate test runs from Engineering Test Management.

To run tests from Engineering Test Management, you must configure the default adapter that is installed when you install .

You can run the adapter in the following modes:

- GUI

- CLI

### Engineering Test Management reports

When you run a test script from Engineering Test Management, the default report that is displayed during a test run is attached to the results of Engineering Test Management. You can customize the reports based on your requirements. See Customizing reports on page 340.

If you use Engineering Test Management 4.0 or later, you can view and analyze the test reports in Engineering Test Management. You can analyze the test reports while the test is in running state and after the test run is complete. You can click the **Analyze Results Interactively using** Rational® Performance Tester option from the **Execution Results** dialog box to view the test reports in Engineering Test Management.

> **Note:** To access reports from outside of Rational® Performance Tester, you must enable the remote access from Rational® Performance Tester. See Access reports remotely on page .

The result completion state that is reported to Engineering Test Management reflects the overall verdict of the test log that is associated with the run. See Logs overview on page 353. In many cases, a test might contain a failed verification point, but still is considered as passed. You can view the attached report in the execution result of Engineering Test Management, and then set the execution results status accordingly.

You can view the full run results from within by opening in the workspace that is configured to be used by the adapter.

If the adapter is running from the command line , you must stop the adapter before opening . When is opened, you can access the full test reporting and test log capabilities. The test results for the runs that are initiated from Engineering Test Management are under the Engineering Test Management **Results** page.

For Rational® Performance Tester schedules, the result completion state that is reported to Engineering Test Management is based on the overall **Requirements** status. Only performance requirements for the last user stage that is defined in the schedule are covered by the report. If no requirements are specified, the result completion state

in Engineering Test Management is set to *inconclusive*. In this case, you can view the attached performance reports and manually set the completion state in Engineering Test Management. See Defining requirements in schedules.

**Known limitations**

- You cannot run tests from Engineering Test Management with encrypted datasets. When using such datasets, a password prompt is not displayed in the adapter service or in the command-line interface. The use of encrypted datasets are not recommended in the GUI mode, because it requires user interaction with to initiate test runs from Engineering Test Management.

- You can start only one adapter per product installation on a given computer. If you use multiple adapters on the same computer, it requires you to install each product as its own software package in its own directory. If you want to run multiple adapters on the same computer, you must ensure that adapters are using different workspaces.

For information about using Engineering Test Management, refer to the IBM Engineering Lifecycle Management documentation.

Refer to the following topics to learn more about integrating Engineering Test Management with .

## Configuring the Engineering Test Management adapter

You must configure the Engineering Test Management adapter to establish a successful connection between and Engineering Test Management.

**Before you begin**

You must have the following information:

- The URL of the Engineering Test Management server.

- A user credential and valid license to access Engineering Test Management.

- The user account must be added to the project area that is being accessed by the adapter with write permissions to the project.

For more information about Engineering Test Management, refer to the IBM Engineering Lifecycle Management documentation.

1. Open .
2. Click **Window > Preferences > Quality Manager Adapter**.
3. Enter the following information of the Engineering Test Management:

| Fields | Actions |
|---|---|
| **Server URL** | Enter the URL of Engineering Test Management. |

| Fields | Actions |
|---|---|
| | For example, `https://<hostname>:<portnumber>/qm`<br><br>**Note:** If you rename the Engineering Test Management server, you must perform the following tasks:<br><br>    a. Update the Engineering Test Management server name in the hosts file with a new name.<br><br>    b. Update the **Server URL** field with the new name.<br><br>    c. Configure the adapter to point to the new URL. |
| **Adapter name** | Enter a unique name to identify the Engineering Test Management adapter. The Engineering Test Management adapter uses the name of the computer as the default name of the adapter. |
| **Project area** | Enter the name of the project area in Engineering Test Management. |

4. Select one of the following **Authentication type** from the drop-down list to connect to Engineering Test Management:

| Authentication type | Actions |
|---|---|
| **Username and Password** | Perform the following steps:<br><br>    a. Enter the username associated with Engineering Test Management in the **User ID** field.<br><br>    b. Enter the password associated with the username of Engineering Test Management in the **Password** field. |
| **KERBEROS** | Click **Browse** to locate and select the `kerberos.ini` file in the **Configuration File** field.<br><br>**Note:** The `kerberos.ini` file is automatically created when you set up Kerberos.<br><br>For example, on Windows systems, you can locate the file in the `c:\windows\krb5.ini`. The file name and the location might change based on the operating systems. |

| Authentication type | Actions |
|---|---|
| **SSLCERT** | Perform the following steps:<br><br>   a. Enter the location of the SSL certificate keystore in the **Certificate Location** filed.<br><br>   b. Enter the keystore password in the **Password** field.<br><br>**Note:** The expected format of the keystore is p12. The keystore must contain the client certificate that the adapter uses when you authenticate with Engineering Test Management. |
| **SMARTCARD** | Select a certificate from the drop-down list from the **Certificate Selection** field. |

5. **Optional:** Select the **Enable Proxy** checkbox to connect through a proxy computer and perform the following steps to enter the **Proxy Details** of the computer:
   a. Enter the hostname of the proxy computer in the **Host** field.
   b. Enter the port number of the proxy computer in the **Port** field.
   c. Enter the username and password of the proxy computer in the **User** and **Password** fields.
6. Click **Apply and Close** to save and close the configuration.

**Results**

You have configured the details of Engineering Test Management on .

**What to do next**

Related information

## Configuring the workspace directory of the adapter

You must configure the workspace directory of the adapter to start or stop the Engineering Test Management adapter from command-line interface.

**About this task**

If the **Use resources that are local to a test machine** option is set in Engineering Test Management, then the WORKSPACE_DIR must be set to the same workspace where your test assets are located.

1. Locate the `adapter.config` file in the *product_install_dir*`\RPT-RST_RQMAdapter\config\`
   directory.
   Where *product_install_dir* is the directory where is installed.

   For example, `C:\Program Files\IBM\SDP`.

2. Edit the WORKSPACE_DIR variable in the `adapter.config` file to point to the same test workspace that you
   want the adapter to use.

   For example, `WORKSPACE_DIR= C:\Users\`*username*`\IBM\rationalsdp\`*my_adapter_workspace*.

**Results**

You have configured the workspace directory of the adapter.

**What to do next**

You can start or stop the Engineering Test Management adapter from command-line interface.

## Connecting and disconnecting the Engineering Test Management adapter from the GUI mode

You can use the **Quality Manager Adapter** view to connect, disconnect, and view adapter activities from .

**Before you begin**

You must have configured the Engineering Test Management adapter. See Configuring the Engineering Test
Management adapter on page 90.

**About this task**

In the GUI mode, when a script is run from Engineering Test Management, you can see the test run in progress inside
as though the test were run manually in .

Push buttons to connect and disconnect to the Engineering Test Management server are located in the upper-right
corner of **Quality Manager Adapter** view. This view also has a local preferences menu that you can use to control
some behavior of the GUI mode adapter. If you see errors or warnings, use the **Error Log** view for further investigation.

> **Note:** You must not use while the adapter is running. If you do so, you might interfere with the ability of the
> adapter to run test scripts. You must stop the adapter before you open .

The following image displays the activities of the adapter in the **Quality Manager Adapter** view:

1. Open .

2. Click **Window > Show View > Quality Manager Adapter**.

3. Perform the following actions either to connect or disconnect the adapter:

   ◦ Click the **Connect to RQM** icon  to connect the adapter.

   ◦ Click the **Disconnect from RQM** icon  to disconnect the adapter.

**Results**

You have connected or disconnected the Engineering Test Management adapter from .

Related information

IBM Engineering Test Management overview on page 40

Configuring the Engineering Test Management adapter on page 90

Starting and stopping the Engineering Test Management adapter from the command line on page 94

Importing test assets into Engineering Test Management on page 97

# Starting and stopping the Engineering Test Management adapter from the command line

You can use the command-line interface to start, stop, and view activities of the Engineering Test Management adapter that you configured in .

**Before you begin**

You must have performed the following tasks:

- Configured the adapter in . See .

- Configured the workspace directory of the adapter. See .

**About this task**

When you run test assets from the command-line interface, the adapter activities are printed to the `adapter.log` file that can be accessed from `product_install_dir`\RPT-RST_RQMAdapter\logs.

To print the current status of the adapter, you must navigate to the `product_install_dir`\RPT-RST_RQMAdapter\bin directory, and then you can run the RQMAdapter.bat STATUS command.

Where, `product_install_dir` is the installation directory of .

⚠️ **Warning:** You must not use while the adapter is running. You must stop the adapter before you open for any reason.

1. Open a command-line interface.
2. Navigate to the `product_install_dir`\RPT-RST_RQMAdapter\bin\ directory.
3. Perform the following step either to start or stop the adapter:

   ◦ Run the following command to start the adapter from the command line:

   | Operating system | Command to be run |
   | --- | --- |
   | Windows™ | RQMAdapter.bat START |
   | Linux™ | RQMAdapter.sh START |

   ◦ Run the following command to stop the adapter from the command line:

   | Operating system | Command to be run |
   | --- | --- |
   | Windows™ | RQMAdapter.bat STOP |
   | Linux™ | RQMAdapter.sh STOP |

**Results**

You have started or stopped the Engineering Test Management adapter from the command-line interface.

Related information

## Starting and stopping the Engineering Test Management adapter as a Windows™ service

You can use the Windows™ service to start, stop, and view adapter activities.

**Before you begin**

You must have completed the following tasks:

- Configured the adapter in . See .

- Installed Microsoft™ .NET Framework 3.5.x in Windows systems for the adapter service.

- Configured the workspace directory of the adapter. See .

**About this task**

When you install  Rational® Performance Tester, you must install the adapter as a Windows™ service. The default status of **Startup Type** is set to **Manual**.



Optionally, to configure the service to start automatically, you can right-click the adapter listing, and then select **Properties**. You can then select **Automatic** from the drop-down list in the **Startup Type** field. Therefore, the adapter can start automatically when you restart your computer and does not require you to log in.

> **Notes:**

> • When you start the adapter as a service, you cannot run the Web UI tests of IBM® Rational® Test Workbench from IBM® Engineering Test Management.
>
> • You must not open in the same workspace while the adapter is running as a Windows™ service. You must stop the adapter before you open in the configured workspace.

When you run the adapter as a service, the status of the adapter is printed to the `adapter.log` file that can be accessed from *product_install_dir*`\RPT-RST_RQMAdapter\logs`.

You can also print the current status of the adapter by navigating to the *product_install_dir*`\RPT-RST_RQMAdapter\bin\` directory, and then run the RQMAdapter.bat STATUS command.

1. Open the Windows™ services.
2. Perform the following step either to start or stop the adapter:

   ◦ Right-click **IBM RTW-RPT-RST adapter for RQM** and, then click **Start** to start the service.

     > **Note:** The adapter is same for Rational® Service Tester for SOA Quality, Rational® Test Workbench and Rational® Performance Tester.

   ◦ Right-click **IBM RTW-RPT-RST adapter for RQM** and, then click **Stop** to stop the service.

**Results**

You have started or stopped the Engineering Test Management adapter as a Windows service.

Related information

IBM Engineering Test Management overview on page 40

Configuring the Engineering Test Management adapter on page 90

Connecting and disconnecting the Engineering Test Management adapter from the GUI mode on page 93

Starting and stopping the Engineering Test Management adapter from the command line on page 94

Importing test assets into Engineering Test Management on page 97

## Importing test assets into Engineering Test Management

You can import the performance and service tests into Engineering Test Management by using an adapter.

**Before you begin**

The adapter must be running on a computer where the test assets are located.

**About this task**

From Rational® Functional Tester, you cannot import AFT suites into Engineering Test Management because AFT suites are not supported with the Engineering Test Management integration.

1. Log in to Engineering Test Management.
2. Click **Construction > Import Test Scripts**.
3. Select one of the following test scripts in the **Script Type** field:
   a. **IBM Rational Performance Tester** to import a performance test or schedule from  Rational® Performance Tester.
   b. **Service Test** to import a service test from  Rational® Service Tester for SOA Quality
   c. **Rational Functional Tester** to import a functional test from Rational® Functional Tester.
   d. **Rational Test Workbench** to import a Web UI test from Rational® Functional Tester or import a test from Rational® Test Workbench.
4. Select **Use test resources that are local to a test machine**, and click **Select Adapter**.
5. Select the computer on which the adapter is running, and click **Next**.
6. Enter the name of the project in the **Project Path** field, and then click **Go**.

   📝 **Note:** You must specify only the project name and not the entire path to the project.

7. Select the test assets that you want to import, and then click **Finish**.
8. Select those test assets to import again, and then click **Import**.

**Results**

You have imported the test assets to Engineering Test Management by using the adapter.

Related information

IBM Engineering Test Management overview on page 40

Configuring the Engineering Test Management adapter on page 90

Connecting and disconnecting the Engineering Test Management adapter from the GUI mode on page 93

Starting and stopping the Engineering Test Management adapter from the command line on page 94

Testing shared assets with Engineering Test Management on page 98

## Testing shared assets with Engineering Test Management

You can make test projects and assets shareable in Engineering Test Management. By sharing assets, any computer with your product, that is connected to Engineering Test Management can execute a test or schedule.

**Before you begin**

When you are working with tests from a remote shared location,  or  Rational® Service Tester for SOA Quality uses a local workspace for the Engineering Test Management adapter. This adapter workspace is different from normal

workspaces because the test assets are stored remotely. This means that every asset that is related to the test is downloaded from the shared location into the local workspace before execution. The following limitations apply:

- Assets in the adapter workspace might be deleted or overwritten with newer versions when updates are made to the shared location.

- If you change the shared location in the adapter workspace, the entire project is removed from the adapter workspace.

- Test results are stored in a different project, called `RQM_Results`, and are never deleted. The Engineering Test Management test result page links to the correct location.

> **Note:** Do not edit test assets in the adapter workspace because you might lose your work. You must use these assets only for running tests.

If you are using source control and want to include only the minimum required assets, then include the following files:

- All `*.testsuite` tests files

- The `/src` directory if you use custom code

- All `*.dp` dataset files

- All `*.location` location files

- All digital certificates

- All WSDL and SOA security files

> **Note:** All other assets, such as test results, are not required.

Custom code Java™ classes in the shared assets cannot use libraries that are outside the workspace. If your custom code must use such a library, then copy the library into the project, and update the classpath to use the local copy.

1. Create a shared directory on the computer that hosts the UNC file system that contains the test projects to share.
   **Example**
   For example, create a directory called: `C:\MyRemoteWorkspace`.
2. Copy the test projects to share into the shared directory.
   If a project is stored in source control software, then copy it from there.
3. Check that the Engineering Test Management server can access the shared directory by using UNC paths.
   **Example**
   For example, the `\\MyServer\RPTRemoteAssets\` path must be mapped to the `C:\MyRemoteWorkspace` directory.

4. In Engineering Test Management, specify the directory that contains the actual test projects that are located in the shared directory.

5. Verify that you have correctly specified the UNC shared directory by browsing for the shared resource. Ensure that the first dialog box contains the projects at the first level.

   You must not have intermediate directories between the UNC shared directory and the project directory.

---

Related information

Configuring the Engineering Test Management adapter on page 90

Importing test assets into Engineering Test Management on page 97

## Integration with IBM® Engineering Workflow Management

You can integrate  Engineering Workflow Management (formerly known as  Rational® Team Concert™) to create and track defects (bugs) or other work items, as a defect tracking tool in  Rational® Performance Tester.

You can use  Rational® Performance Tester to record and play back tests for the application that you develop and view their results. When you discover that you might want to raise defects, issues, or other types of work items for the test assets, you can create defects, issues, or other work items without the need to open  Engineering Workflow Management.

For more information about  Engineering Workflow Management, refer to the IBM® Engineering Workflow Management documentation.

The following table lists the tasks that you must perform to integrate  Rational® Performance Tester with Engineering Workflow Management:

| Tasks | Go to... |
|---|---|
| Install  Rational® Performance Tester. | Installation of the product by using Installation Manager on page 58 |
| Create any or all of the following types of test assets in Rational® Performance Tester to test your application:<br><br>• Compound tests<br>• Performance tests<br>• Schedules (Rate or VU Schedules) | Test Author Guide on page 119 |
| Install  Engineering Workflow Management and gain access to it. | IBM® Engineering Workflow Management documentation<br><br>**Note:** The System Requirements on page 30 provide more information about specific versions |

| Tasks | Go to... |
|---|---|
| | of  Engineering Workflow Management require-ments. |
| Run the test assets. | Running a local schedule or test on page 287 |
| Configure the  Engineering Workflow Management server URL in  Rational® Performance Tester. | Configuring the URL of Engineering Workflow Management on page 101 |
| Create defects from  Rational® Performance Tester. | Tracking defects with Engineering Workflow Management on page 102 |

## Configuring the URL of  Engineering Workflow Management

You must configure the URL of the  Engineering Workflow Management server to use it as defect tracking tool in Rational® Performance Tester.

**Before you begin**

You must have the URL of the  Engineering Workflow Management server.

**About this task**

Bugzilla is configured as the default defect tracking tool in the **Preferences** window of  Rational® Performance Tester. If you are using  Engineering Workflow Management to create and track defects (bugs), you can provide the URL of the  Engineering Workflow Management server to search, submit, or add work items to test results from Rational® Performance Tester.

1. Open  Rational® Performance Tester.
2. Click **Window > Preferences > Test > Test Log Editor**.
3. Enter the URL of the  Engineering Workflow Management server in the following fields:

| Fields | Format of the URL |
|---|---|
| **Submit URL** | `https://`*`ewm.example.com:9443`*`/ccm/web/projects/`*`projectname`*`#action=com.ibm.team.workitem.newWorkItem` |
| **Search URL** | `https://ewm.example.com:9443/ccm/web/projects/projectname#action=com.ibm.team.workitem.newWorkItem` |
| **Open URL** | `https://ewm.example.com:9443/ccm/web/projects/projectname#action=com.ibm.team.workitem.newWorkItem&id=` |

Where,

- `ewm.example.com:9443` is the URL of the  Engineering Workflow Management server.

- `projectname` is the name of the project in the  Engineering Workflow Management server.

> **Note:** You must update the URL, if there is a change in the name of the project in  Engineering Workflow Management.

4. Click **Apply and Close** to save the configuration and close the **Preferences** window.

**Results**

You have configured the URL of  Engineering Workflow Management in  Rational® Performance Tester.

**What to do next**

You can create defects for the test results that are available in your project in  Rational® Performance Tester. See .

## Tracking defects with  Engineering Workflow Management

You can submit defects to  Engineering Workflow Management from . By default, the test log editor uses Bugzilla as the defect tracking site. You must configure the product to use  Engineering Workflow Management for defect tracking.

1. Click **Window > Preferences > Test > Test Log Editor**.
   **Result**
   The **Test Log Editor** preferences window opens.
2. Specify the URL of  Engineering Workflow Management server in the **Submit URL**, **Search URL**, and **Open URL** fields.
   Contact the administrator of the  Engineering Workflow Management server for more information.
   **Example**
   The following are the example of URLs, where the name of the  Engineering Workflow Management server is *evm.example.com* and the name of the project is *projectname*:

**Table 2. Example of EVM Server URL**

| Field names | Field values |
| --- | --- |
| Submit URL | `https://ewm.example.com:9443/jazz/web/projects/projectname#ac-tion=com.ibm.team.workitem.newWorkItem` |
| Search URL | `https://ewm.example.com:9443/jazz/web/projects/projectname#action=jaz-z.viewPage&id=com.ibm.team.workitem` |

| Field names | Field values |
|---|---|
| **Open URL** | `https://ewm.example.com::9443/jazz/web/projects/projectname#ac-tion=com.ibm.team.workitem.viewWorkItem&id=` |

> **Note:** You must change the URLs, if there is change in name of the  Engineering Workflow Management server.

## Testing with  Rational® Integration Tester

, you can use  Rational® Integration Tester extension to execute Integration tests. In , you can either import the projects from  Rational® Integration Tester or manage them from by establishing the connection between the products. In , you can create a compound test to run the tests by using the Agents.

### Before you begin

To be able to work with Integration tests, you must install Extension for  Rational® Integration Tester.

You also need a  Rational® Performance Tester agent or a  Rational® Integration Tester agent to execute the tests remotely. When installing  Rational® Integration Tester agent, it is recommended to choose **This Agent will only run probes** option.

Moreover, if you update Integration tests in and want to apply the updates back to  Rational® Integration Tester, you must install  Rational® Integration Tester and define the path to its installation directory to set the connection.

Following are the use cases to work with Integration tests in the :

- Both the products are installed and you connect to the Integration project, or you open the Integration resource directly from Test Navigator view and you work directly with the sources files.
- Rational® Integration Tester is not installed and you import the projects in the workspace.

> **Note:** The imported tests must be edited in  Rational® Integration Tester. Similarly, the compounds tests must be edited in .

### Connecting to an existing Integration project

When you connect both the products any change or delete action made in one product workspace is reflected on the other product workspace, if both the products are installed on your machine.

> **Warning:** If you delete a project from the Test Navigator, be sure that the option **Delete project contents on disk**  is not selected in the **Delete Resources** dialog, otherwise the project would be deleted in  Rational® Integration Tester if it is connected.
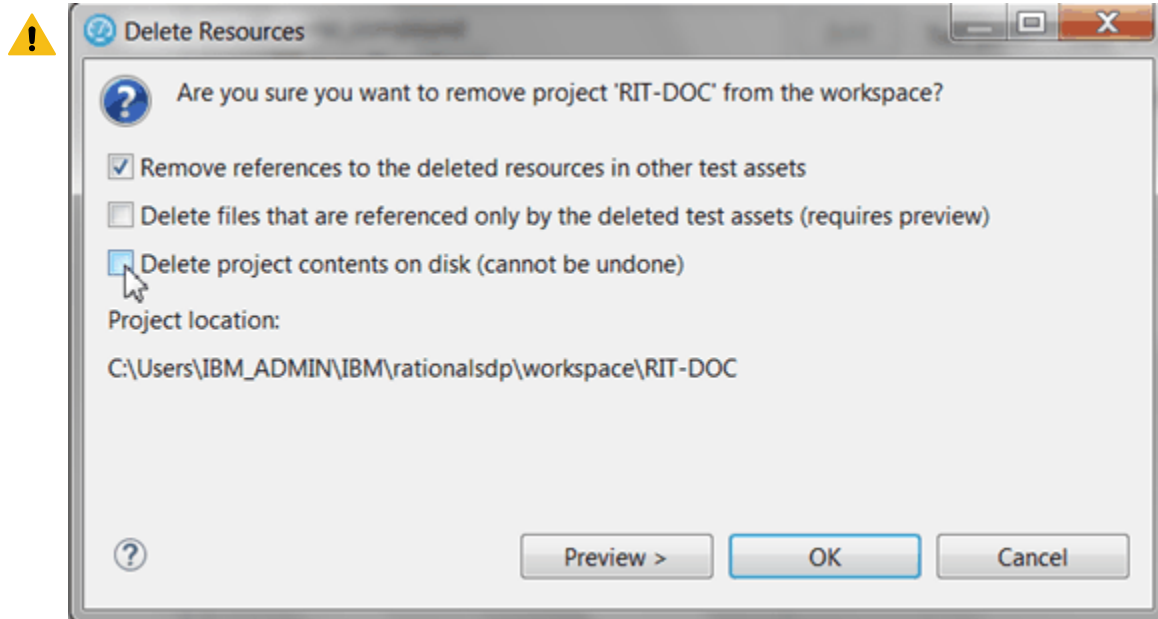
- In , right-click on the **Test Navigator**, select **New > Other**> Rational® Integration Tester **Connect to an** Rational® Integration Tester**Project** and click **Next**.
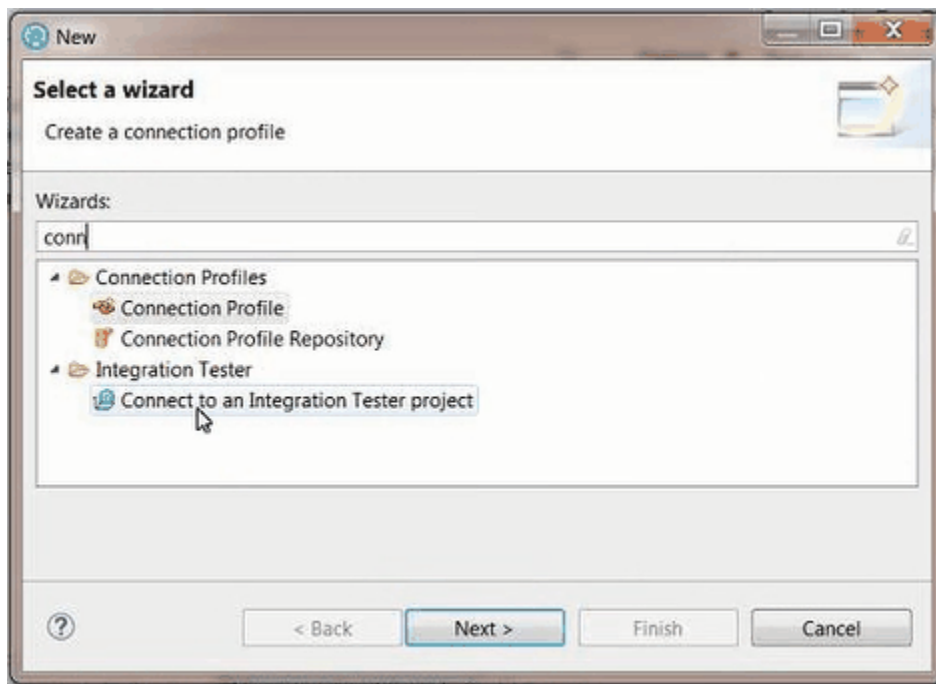


- In the wizard page, click **Browse** and select the root folder that contains the project.

If the path contains a project, its name should automatically appear in **Project Name** and the **Finish** button should be enabled.

- In **When project is connected**, you have to perform one of the following actions:

## Setting Rational® Integration Tester preferences

To be able to open an Rational® Integration Tester project from Test Navigator, you need to have both the products installed on the same computer, and you must set the path to the execution file in the Preferences.

- In , click **Window > Preferences > Test > RIT Integration**.
- Click **Browse** and set the installation path to Rational® Integration Tester execution file. On Windows, the default location would be `C:\Program Files\IBM\IntegrationTester.exe`.
- Click **Apply** and **OK**.

## Opening Rational® Integration Tester resources from the Test Navigator

- Once the preferences are set, you can open an Rational® Integration Tester project.
- In the **Test Navigator**, open the project root node and children nodes, and at any level, right-click and select **Open in** Rational® Integration Tester **Workbench**.
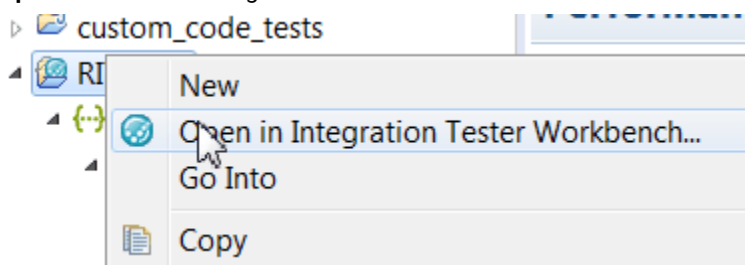


If Rational® Integration Tester is automatically detected, the workspace opens for the selected resources.

If Rational® Integration Tester is not detected, a dialog opens on a Preference page where you need to verify the path to the execution file.

- ⚠️ **Warning:** Rational® Integration Tester cannot open more than one project at a time. If you have another project open, you will get an error. In that case, close Rational® Integration Tester and try to open the project again.

## Importing Rational® Integration Tester project

If both the products are not installed on the same machine, you can import an Rational® Integration Tester project in your workspace. Another reason for the import is when you have Rational® Integration Tester installed but you do not want to connect to the Rational® Integration Tester project. In that case, the project is duplicated, any updates in one product workspace will not be reflected in the other product's workspace.

- To import an  Rational® Integration Tester project:
- Right-click on the **Test Navigator**, choose **Import** and select **Existing project into workspace**.
- Choose **Select root directory** or **Select archive file**; select a project to import and click **Finish**.

   The selected project appears in the **Test Navigator** and the compound test editor automatically opens.

## Modifying  Rational® Integration Tester environments in

In the compound test, you can select  Rational® Integration Tester tests and change the environment of each test. The environments are set in  Rational® Integration Tester, you can only change the selection from the edited compound test.

- Open the compound editor and select a test.
- In the  Rational® Integration Tester details, you can browse and change the properties of the selected test. The **Test path**, the **Environments** and **Decription** are automatically updated accordingly.
- To select another environment for the Integration Tester test, use the dropdown menu.

Alternatively, you can change the environment selection for a test for a collection of tests:

- Right-click on the tree at any level under a node in the compound
   test and select **Replace**  Rational® Integration Tester **Environments**.
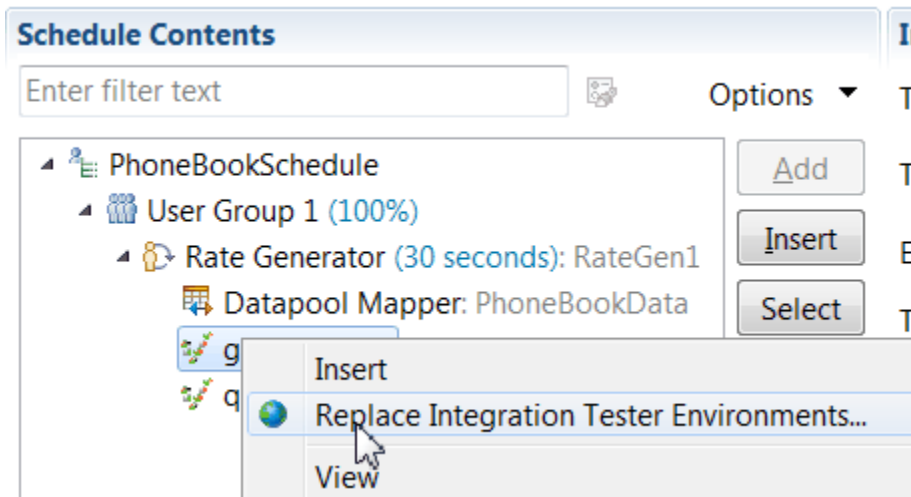


- In the **Set Invocation Tester Environments** wizard, the first page displays the list of projects that use the selected environment and the number of tests from project that use this environment in the compound test.
- Select another used environment in the dropdown list. Click **Finish**. The new choice applies to the selected node and its children.

Next step is to create a compound test or schedule in to , see or Creating a compound test.

You can add a dataset mapper in the compound test for tests that are using multiple tags. See Adding Dataset Mapper to map tags in the  Rational® Integration Tester tests with the variable values of .

## Running Rational® Integration Tester tests

You can use Extension for IBM® Rational® Integration Tester to run IBM® Rational® Integration Tester tests.

You can install both the products on the same machine, establish the connection, and run Rational® Integration Tester tests. See Testing with Rational® Integration Tester on page 103.

You also have the option to just import the projects to Rational® Performance Tester Rational® Integration Tester, add the tests to a compound test to run them. You can either use Rational® Performance Tester Agent or Rational® Integration Tester Agent to generate the load. You need a compound test that contains the Rational® Integration Tester tests.

**Setting environment variable**

To run the tests with Rational® Integration Tester Agent, set the environment variable *INTEGRATION_TESTER_AGENT_HOME* and point it to the Rational® Integration Tester Agent installation directory.

On Linux, set:

```
INTEGRATION_TESTER_AGENT_HOME=/opt/IBM/RIT-Agent
export INTEGRATION_TESTER_AGENT_HOME
echo $INTEGRATION_TESTER_AGENT_HOME
```

If the variable is not set, you will get a test log with error message when the compound test is run.
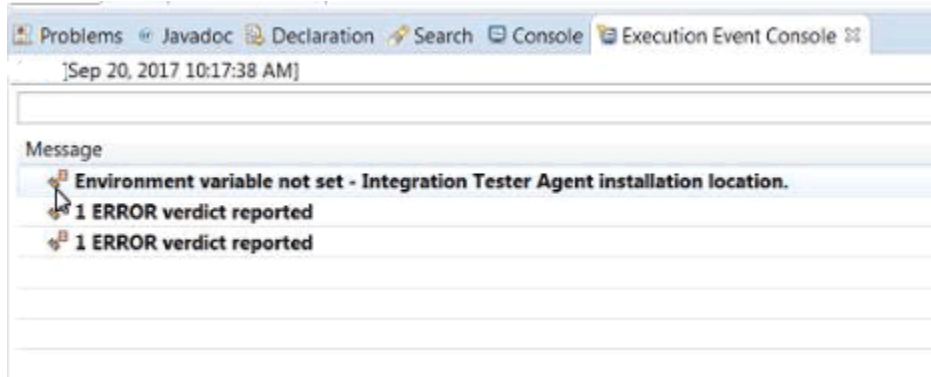


**Running the compound test**

## Integration of Jaeger with the product

Jaeger is software for tracing transactions between distributed services. You can use Jaeger to monitor and troubleshoot complex microservices environments.

You can set up the Jaeger UI in your local environment by using one of the following methods:

- One Jaeger agent shared by all Rational® Performance Tester agents
- One Jaeger agent for each Rational® Performance Tester agent

**One Jaeger agent shared by all Rational® Performance Tester agents**

When you use this method, ensure that the Jaeger agent is accessible by Rational® Performance Tester and all the Rational® Performance Tester agents. You must set the *JAEGER_AGENT_HOST* property as an environment variable by using the command line before running the schedule.

You must also ensure that the Jaeger agent ports 6831, 6832, and 5778 are accessible from other computers to communicate with the Rational® Performance Tester agent via the User Datagram Protocol (UDP). If you want to define any other Jaeger environment variables, set those environment variables only on Rational® Performance Tester.

**One Jaeger agent for each Rational® Performance Tester agent**

When you use this method, you must install the Jaeger agent in the same location where you installed Rational® Performance Tester and on all the Rational® Performance Tester agents.

You must also ensure that the Jaeger agent ports 6831, 6832, and 5778 are accessible from other processes on the same computer to communicate with the Rational® Performance Tester agent via the UDP. If you want to define any other Jaeger environment variables, set those environment variables on all the computers where the Jaeger agent is installed.

For more information about Jaeger, refer to Jaeger documentation.

Related information

## Viewing test logs in Jaeger

You can use the Jaeger UI to view the test logs of the *tests* or *schedules* that you run from the command-line interface to analyze traces of transactions between distributed services.

**Before you begin**

You must have completed the following tasks:

- Downloaded Jaeger components from the Jaeger website.
- Created a test or schedule to run it from the command-line interface. See or Creating a VU Schedule.

**About this task**

While running *tests* or *schedules* by using the command-line interface, you must include the command **-history** *jaeger* in your *test* or *schedule* run. Adding the **-history** *jaeger* enables you to view the test log of the completed *test* or *schedule* from the Jaeger UI in a web browser.

1. Run a *test* or *schedule* from the command-line interface by adding the **-history** *jaeger* option.

   For example: **cmdline.bat -workspace** *workspace_full_path* **-project** *proj_rel_path* **-suite** *suite_rel_path* **-stdout -history** *jaeger*

   **Result**

   The *test* or *schedule* runs and the result of the run is displayed.

2. Open the Jaeger UI in a browser.

   For example: `http://<host IP>:<port>`.

3. Select **Rational Test Product** from the **Service** list.

4. Click **Find Traces**.

   In the Jaeger UI, you can view the entire test log of the *test* or *schedule* that you ran from the command-line.

**What to do next**

- You can use the Jaeger traces to analyze test results.
- You can compare the traces in the Jaeger UI with test logs in  Rational® Performance Tester to confirm that they are the same.

---

Related information

Running a test from a command line on page 295

Viewing test logs on page 354

Integration of Jaeger with the product on page 107

## Testing with Jenkins

You can use the  Jenkins plugin for testing to run tests on a Jenkins server by using a Jenkins build step.

**Introduction**

To automate testing with Jenkins, you must configure two computers: A Jenkins master and a Jenkins slave. This master-slave configuration allows a single Jenkins installation on the master computer to host multiple slave environments for building and running tests. You must install the latest version of the  Jenkins plugin on the master computer, and install the products themselves on the slave computer, where you create the tests. For detailed information about the Jenkins master-slave relationship, see the Distributed Builds section on the Jenkins site.

**Note:** You can download the  Jenkins plugin from the  portal.

For more information about specific versions of plugin, see Integration plugin compatibility matrix on page 75.

You can then install the plugin on the Jenkins server. After you create the tests in , you can run the tests on the Jenkins server by using this plugin. If you have earlier version of the Jenkins plugin, you must uninstall it before installing the latest version of the plugin on the Jenkins server.

**Before you begin**

You must have completed the following tasks:

- Installed Installation Manager, which is required for installing the products.
- Installed the products themselves.
- Verified that you have a Jenkins master computer where Jenkins and the Jenkins plugin is installed.
- Verified that you have a Jenkins slave computer where the products are installed.
- Verified that you have a test residing within an Eclipse workspace on the Jenkins slave where the products are installed.

**Note:**

, you must add an environment variable that points to the installation directory of the product, for example, `export TEST_WORKBENCH_HOME=/opt/IBM/SDP`. For Windows™ and Linux®, this environment variable is set when you install the product.

**Installing the Jenkins plugin on the Jenkins master computer**

1. Download the Jenkins plugin from the portal.
2. From the Jenkins dashboard, perform the following tasks:
   a. Click **Manage Jenkins > Manage Plugins**.
   b. Click **Advanced**.
   c. From the **Upload Plugin** section, click **Choose File** to locate and **Open** the Jenkins plugin.
   d. Click **Upload**.

   The Jenkins plugin is displayed in the **Installed** tab.

3. Configure **Global Security** to allow *Random TCP Ports* for Java™ Network Launch Protocol (JNLP) agents.

**Job Configuration**

1. Create a new Jenkins free-style software project. For more information to create a free-style Jenkins project, see Building a software project in Jenkins.
2. From the Jenkins dashboard, perform the following tasks:
   a. Open the Jenkins free-style software project, and then click **Configure**.
   b. Click **Build > Add build step**.
   c. Click IBM® Rational® Performance Tester.
3. Provide the details about the test run as shown. The following table explains each field.

| Field | Description |
|---|---|
| Name | Required. The name of the test. |
| Work-space | Required. The complete path to the Eclipse workspace. |
| Project | Required. The path, including the file name of the project relative to the workspace. |
| Test Suite Name | Required. The path, including the file name of the test to run related to the project. |
| IMShared Location | Optional. The complete path to IBMIMShared location,if it is not the default location. |
| Var File | Optional. The complete path to the XML file that contains the variable name and value pairs. |
| Config File | Optional. The complete path to a file that contains the parameters for a test or schedule run. |
| Results File | Optional. The name of the results file. The default result file is the test or schedule name with a time stamp appended. The results file is stored in the **Results** directory. If you are running multiple tests, do not provide a name for the results file. |
| Overwrite Results File | Optional. Determines whether a results file with the same name is overwritten. The default value is true, which means the results file can be overwritten. |
| Quiet | Optional. Turns off any message output from the launcher and returns to the command shell when the run or the attempt is complete. |
| VM Args | Optional. Java™ virtual machine arguments to pass in. |
| Dataset Override | Optional. For a test or schedule, the default value is the dataset specified in the test editor or schedule editor. Overrides the default dataset value to run if required. <br><br> **✎ Note:** <br><br> You must use the `Dataset Override` option to replace the dataset values during a test or schedule run. You must ensure that both original and new datasets are in the same workspace and have the same column names. You must also include the path to the dataset. <br><br> For example, <br><br> `/project_name/ds_path/ds_filename.csv:/project_name/ds_path/new_ds_filename.csv`. <br><br> You can swap multiple datasets that are saved in a different project by adding multiple paths to the dataset separated by a `semicolon (;)`. <br><br> For example, |

| Field | Description |
|-------|-------------|
| | ✏️ `/project_name1/ds_path/ds_filename.csv:/project_name1/ds_path/new_ds_file-name.csv;/project_name2/ds_path/ds_filename.csv:/project_name2/ds_path/new_ds_-filename.csv` |
| Exported Statistical Report Data File | Optional. The complete path to a directory to store exported statistical report data. |
| Custom Report Format Files | Optional. A comma-separated list of absolute paths to custom report format files (.view files) to use when exporting statistical report data with the Export Statistical Report Data File option. |
| Exported Statistical Report in html | Optional. The complete path to a directory to export web analytic results. Analyze the results on a web browser without using the test workbench. If you run multiple tests, do not provide a value in this field. The web analytic results will be exported to Jenkins workspace. |
| User Comments | Optional. Add text within the double quotation mark to display it in the **User Comments** row of the report. |

If you do not supply a value for `Exported Statistical Report Data File`, these logs will be saved in Jenkins workspace/temp directory.

4. Click **Save**.
5. Optionally, to run multiple tests under the same job, click **Add build step** again, and provide details for the next test.

## Environment variable configuration

After you set the environment variable in **Manage Jenkins > Configure System > Global properties** on the Jenkins server, you can enter the variable name by using any of the following methods for the corresponding text fields in the IBM® Rational® Performance Tester step:

- Use the dollar sign (`$`) followed by the variable name.

  For example, `$workspace`

- Use the dollar sign (`$`) followed by the variable name between braces.

  For example, `${workspace}`

You can then run the Jenkins build by using environment variables in the IBM® Rational® Performance Tester step. The environment variable that you added to your test is substituted with the actual value associated with the variable

that you set in **Manage Jenkins > Configure System > Global properties** for the job. The Jenkins plugin uses the actual value while running the job.

For example,

If you set the environment variable named `workspace` with the value `C:\Users\IBM\workspace1` in **Manage Jenkins > Configure System > Global properties**, then you can use `$workspace` or `${workspace}` as input to the corresponding text field of your choice for the IBM® Rational® Performance Tester step in the Jenkins build step. When you run the build, `$workspace` or `${workspace}` is substituted with its corresponding value `C:\Users\IBM\workspace1`.

## Master/Slave Configuration

Master and slave configurations are supported by  Jenkins plugin . Refer to Distributed builds for more information.

While creating the job configuration, in addition to the preceding steps, you must select the **Restrict where this project can be run** check box and provide the name of the slave node in the **Label Expression** field. This is the location where the products are installed and where tests can be run.
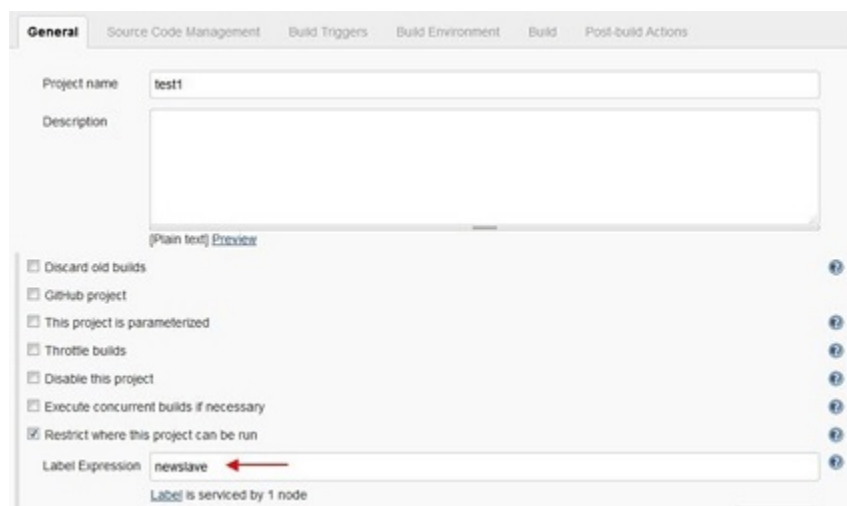


## Running tests

After you save the project, you must open the project, and then click **Build Now**. This action starts the test run on the slave computer. You must specify the relative path from the project to the test including the file name of the test. To run multiple tests from different projects, you must add new build step for each project.

**Building result and logs**

1. After the build completes, click the build number and open the console for the project. Locate the `Test Result` to check the test execution status.

    > **Note:** If you add multiple build steps to run multiple tests, multiple `Test Result` instances are displayed.

2. For product logs, log in to the slave computer and search in the Jenkins slave workspace/temp directory.

## Testing with Maven

Starting from V9.2.0, you can use the Maven plug-in that is provided with the testing product to run tests as part of your Maven build. Apache Maven is a software build tool based on the concept of a project object model (POM).

**Before you begin**

- You must have installed the testing product from V9.2.0 and set an environment variable that points to the installation location.

    For Mac OS, add an environment variable that points to the installation directory of the product: `export TEST_WORKBENCH_HOME=/opt/IBM/SDP`

    For Windows™ and Linux®, this environment variable is set when you install the product.

- You must have installed Maven from V3.2.0 and set up an environment variable that points to the `M2_HOME` installation directory.

**Introduction**

To automate testing with Maven, you must configure a pom.xml file and launch your tests from the command line using Maven command. You can either use your own pom.file, or one that is delivered with the product.

Three files are delivered with the product installation in the `<product install location>\SDP\maven2\` folder:

- `pomCustomSurefireSample.xml` for Windows, Linux and macOS.
- `pomMojoExecPluginSample_Linux.xml` for Linux and MacOS.
- `pomMojoExecPluginSample_Windows.xml` for Windows.

The files contain all types of dependencies as well as arguments required to execute the test scripts. There are two methods to run tests with Maven.

**Method 1**

With this method, you can run one or several tests. If you use your own pom.xml file, edit it with the following lines and indicate which test(s) must be executed, otherwise, use the `pomCustomSurefireSample.xml` file as follows:

- Copy the `pomCustomSurefireSample.xml` to a directory.
- Edit the file and update the lines, enter the name and location of the test(s) that must be run. If the product is installed on a different drive or a different location, or if location has been changed, enter the correct path to the `IBMIMShared` plug-in folder. For aftsuite attribute, you can input aft xml file as the parameter value.

```
<!--test suite="testSources/Test1.testsuite"/-->
 <!--test suite="Test2.testsuite"/-->
 <test suite="C:/Runtimes/runtime-RptMvn/AA/testSources/Test2.testsuite" plugins="C:/Program
 Files/IBM/IBMIMShared/plugins"/>
 <!--test    schedule="Schedule.testsuite" project="AA" workspace="C:/Runtimes/runtime-RptMvn"
 plugins="C:/Program Files/IBM/IBMIMShared/plugins"/-->
 <!--test    suite="Test2.testsuite" project="AA" workspace="C:/Runtimes/runtime-RptMvn"/-->
<!--test aftsuite="Test1.xml" project="AA" workspace="C:/Runtimes/runtime-RptMvn"
 plugins="C:/Program Files/IBM/IBMIMShared/plugins"/-->
```

- Run Maven to update the pom file version command and use the plug-in version currently available on delivered repositories.

```
mvn versions:update-properties -Dincludes=com.hcl.products.test.it -f pomCustomSurefireSample.xml
```

- Run the test(s).

```
mvn clean verify -f pomCustomSurefireSample.xml
```

Fail safe reports are generated in the target directory, especially in `target/failsafe-reports/<ProjectName>/<TestName>_<timestamp>.txt` that will contain the screen capture of the execution.

## Method 2

With this method, no Maven report is generated. If you use your own pom.xml file, copy the following lines and provide your parameter values. Otherwise, you can use the `pomMojoExecPluginSample_Linux.xml` or `pomMojoExecPluginSample_Windows.xml` sample file. Example with the `pomMojoExecPluginSample_Windows.xml` sample file:

- Copy `pomMojoExecPluginSample_Windows.xml` to a directory.
- Edit the file and update the arguments to reflect which test to execute. If the product is installed on a different drive or a different location, or if `IBMIMShared` location has been changed, update the two last lines with the path to the `IBMIMShared` plug-in folder.

```
<argument>/C</argument>
                        <argument>${pt-plugin-cmdline}</argument>
                        <argument>-workspace</argument>
                        <argument>C:\Runtimes\runtime-RptMvn</argument>
                        <argument>-project</argument>
                        <argument>AA</argument>
                        <argument>-suite</argument>
                        <argument>Test1.testsuite</argument>
                        <argument>-plugins</argument>
                        <argument>C:/Program Files/IBM/IBMIMShared/plugins</argument>
```

- In the argument tags, instead of the -suite option, you can use the -aftsuite option and input the aft xml file as the parameter value in the subsequent argument tag to run the AFT test. For example, in the preceding template, `<argument>-suite</argument>` `<argument>Test1.testsuite</argument>` can be replaced with `<argument>-aftsuite</argument>` `<argument>aftfile.xml</argument>`.

- Run the test.

    For Windows:

    ```
    mvn clean verify -f pomMojoExecPluginSample_Windows.xml
    ```

    For Linux or MacOS:

    ```
    mvn clean verify -f pomMojoExecPluginSample_Linux.xml
    ```

Related information

https://maven.apache.org/index.html

## Integrating and running performance test scripts in Micro Focus ALM

To obtain test result details, you must integrate and run performance test scripts in Micro Focus Application Lifecycle Management (ALM) by using a readily available template available in IBM® Rational® Performance Tester installation directory.

**About this task**

The performance test template is available in the installation directory of Rational® Performance Tester. You must copy the contents of the template to a new VAPI-XP VBScript test script in Micro Focus ALM, add your test script details into the VAPI-XP VBScript test script, and then run the test script.

> **Note:** You can use Micro Focus ALM client only on Microsoft™ Internet Explorer. For more information, see Micro Focus ALM system requirements.

1. Navigate to the directory `IBM\SDP\alm` in Rational® Performance Tester installation directory.

    You can use `PT_ALM_Windows.txt` file for performance test scripts.

2. Copy the contents of the template.
3. Log in to the **Micro Focus ALM** portal, if you are not already logged in.
    **Result**
    The Micro Focus ALM dashboard is displayed.
4. From Micro Focus ALM, create an new integration test by performing the following actions:
    a. Expand **Testing** from the left pane and then click **Test Plan**.
    b. Expand **Integration**.
    c. Right-click **Integration** and click **New Test** to create a new test.
    d. Enter a test name in the **Test Name** field.
    e. Select **VAPI-XP-TEST** as test type from the **Type** drop-down list.
    f. Click **OK**.

        The **VAPI-XP Wizard** is displayed.

      g. Select a test script language by performing the following steps:

            i. Select **VBScript** from the **Script Language** drop-down list.

           ii. Enter a script name (for example, script) in the **Script Name** field.

          iii. Click **Next** and select a test type if required.

> ✏️ **Note:** The **COM/DCOM Server Test** test type is already selected as a test type.

          iv. Click **Finish**.

      The test is created.

5. Click the **Test Script** tab.
6. Paste the content of the template that you copied in to the test script.
7. Enter the test details in the VAPI-XP Vbscript test script by referring to the following table:

| Parameter | Description |
| --- | --- |
| **Workspace** | Required. The complete path to the Eclipse workspace. |
| **Project** | Required. The path, including the file name of the project relative to the workspace. |
| **Testsuit-eName** | Required. The path, including the file name of the test to run relative to the project. |
| **IMShared-Location** | Optional. The complete path to IBMIMShared location. |
| **Varfile** | Optional. The complete path to the XML file that contains the variable name and value pairs. |
| **Configfile** | Optional. The complete path to a file that contains the parameters for a test or schedule run. |
| **ResultsFile** | Optional. The name of the result file. The default result file is the test or schedule name with a time stamp appended. |
| **Overwrite-ResultsFile** | Optional. Determines whether a results file with the same name is overwritten. The default value is true, which means the results file can be overwritten. |
| **Quiet** | Optional. Turns off any message output from the launcher and returns to the command shell when the run or the attempt is complete. |
| **VMArgs** | Optional. You can use this parameter to pass Java™ virtual machine arguments. |
| **Dataset-Override** | Optional. For a test or schedule, the default value is the dataset specified in the test editor or schedule editor. Overrides the default dataset value to run if required. <br><br> ✏️ **Note:** <br><br> You must use the `Dataset Override` option to replace the dataset values during a test or schedule run. You must ensure that both original and new datasets are in the same |

| Parameter | Description |
|---|---|
| | workspace and have the same column names. You must also include the path to the dataset. For example, `/project_name/ds_path/ds_filename.csv:/project_name/ds_path/new_ds_filename.csv`. You can swap multiple datasets that are saved in a different project by adding multiple paths to the dataset separated by a `semicolon (;)`. For example, `/project_name1/ds_path/ds_filename.csv:/project_name1/ds_path/new_ds_file-name.csv;/project_name2/ds_path/ds_filename.csv:/project_name2/ds_path/new_ds_-filename.csv` |
| ExportS-tatsFile | Optional. The complete path to a directory that you can use to store exported statistical report data. |
| ExportStat-Reportlist | Optional. A comma-separated list of absolute paths to custom report format files (.view files) that you can use to export statistical report data with ExportStatsFile. |
| ExportS-tatsHtml | Optional. The complete path to a directory that you can use to export web analytic results. The results are exported to the specified directory. You can analyze the results on a web browser without using the test workbench. |
| UserCom-ments | Optional. You can add text within the double quotation mark to display it in the **User Comments** row of the report. |

8. Run the VAPI-XP Vbscript test script.

**Results**

You have integrated and run performance test scripts in Micro Focus ALM.

**What to do next**

The test result details are displayed in the **Output** window of Micro Focus ALM.

# Chapter 6. Test Author Guide

This guide describes how to create test scripts in Rational® Service Tester for SOA Quality and enhances tests by applying different test elements such as dataset, variables, and verification points. This guide is intended for testers.

## Creating tests

To create a test, you record representative interactions with an application.

**About this task**

After you record a test, you can play it back to confirm that the recorded actions do what you expect.

> **Note:** When you record a test that includes a file download, the file is not physically saved to disk. However, you can confirm that the file was retrieved from the server by looking in the response of the request that asked for the file. One method to locate the request for large downloaded files is to look for a request with a large response size.

1. Click the **test name** in test editor.
2. Click **Select**.
3. Select **HTTP Request**.

### Creating a project

The tests that you create, and the assets associated with the tests, reside in a project on your desktop. You can create the project separately, or you can simply record a test, which automatically creates a project named `testproj`.

1. Select **File > New**.
   **Result**
   The **Create a Project** window opens.
2. In the **Project Name** field, type a name for the project.
3. Select **Use default location**.
4. Optional: Click **Next** and select the folders to create in the new project. These folders organize your files by asset (`Tests`, `Results`, and so on).
5. Click **Finish**.
   **Result**

   After you click finish, you are prompted to record a test. You can create a test from a new recording or from an existing recording, or just click **Cancel** to create a test project without recording a test.

### Recording service tests

When you record a test, the test creation wizard records your interactions with the service, generates a test from the recording, and opens the test for editing. You can record a test session by invoking service calls with the generic

service client or by using an existing client. You can also create a service test manually or from a Business Process Execution Language (BPEL) model.

## Service testing guidelines

Before you can test a service, you must set up your test environment and incorporate these guidelines in order to produce reliable tests.

### Test prerequisites

Before creating service tests, you might need to perform some initial tasks. These tasks depend on the transport and security protocols that are implemented by the web service under test.

- **HTTP**: This transport method is supported by default; no additional configuration is required.
- **SSL**: The workspace must contain the certificate keystore (JKS) files that are required for single or double authentication.
- **Java™ Message Service (JMS)**: The Web Services Description Language (WSDL) syntax must be compatible with the requirements of the product. Refer to Verifying WSDL syntax compliance for JMS services on page 122.

### Test generation

When the test is generated, message call envelopes are created according to the XML schema definition (XSD). During this process, mandatory fields are created, and default choices are assumed. You can modify these elements in the test editor.

> **Note:** During recording, you might supply authentication details which are not relevant for the actual application under test. To exclude such actions from the generated test, in **Window > Preferences > Test > Test editor > Service test** ensure that the **Display the 'Skip if Empty' column in XML tree viewer** check box is selected. To select the empty XML elements that you want to skip, in the test editor, select the elements in the **Skip if empty** column.

### Encryption and security

The Java™ Runtime Environment (JRE) that the product uses must support the level of encryption required by the digital certificate that you select. For example, you cannot use a digital certificate that requires 256-bit encryption with a JRE that supports only 128-bit encryption. By default, the product is configured with restricted or limited strength ciphers. To use less restricted encryption algorithms, you must download and apply the unlimited jurisdiction policy files (`local_policy.jar` and `US_export_policy.jar`).

For Oracle Java, download the files from this site:http://www.oracle.com/technetwork/java/javase/downloads/jce8-download-2133166.html.

Before installing these policy files, back up the existing policy files in case you want to restore the original files later. Then overwrite the files in `/jre/lib/security/` directory with the unlimited jurisdiction policy files.

**SSL Authentication**

Service tests support simple or double SSL authentication mechanisms:

- Simple authentication (server authentication): In this case, the test client needs to determine whether the service can be trusted. You do not need to setup a key store. If you select the **Always trust** option, you do not need to provide a server certificat key store.

  If you want to really authenticate the service, you can configure an certificate trust store, which contains the certificates of trusted services. In this case, the test will expect to receive a valid certificate.

- Double authentication (client and server authentication): In this case, the service needs to authenticate the test client according to its root authority. You must provide the client certificate keystore that needs to be produced to authenticate the test as a certified client.

When recording a service test through a proxy, the recording proxy sits between the service and the client. In this case, you must configure the SSL settings of the recording proxy to authenticate itself as the actual service to the client (for simple authentication), and as the client to the service (for double authentication). This means that you must supply the recording proxy with the adequate certificates.

When using stub services, you can also configure the SSL settings of the stub service to authenticate itself as the actual server. This means that you must supply the service stub with the adequate certificate.

**NTLM and Kerberos Authentication**

The product supports Microsoft™ NT LAN Manager (NTLMv1 and NTLMv2) and Kerberos authentication. The authentication information is recorded as part of the test during the recording phase.

To enable NTLMv2 support, you must add a third party library to the workbench. For more information, see Configuring the workbench for NTLMv2 authentication on page 263.

**Digital certificates**

You can test services with digital certificates for both SSL and SOAP security protocol. Digital certificates must be contained in Java™ Key Store (JKS) keystore resources that are accessible in the workspace. When dealing with keystore files, you must set the password required to access the keys both in the security editor and the test editor. For SOAP security you might have to provide an explicit name for the key and provide a password to access the private keys in the keystore.

**Limitations**

Arrays are not supported.

Because of a lack of specification, attachments are not supported with the Java™ Message Service (JMS) transport. The envelope is directly sent using UTF-8 encoding.

All security algorithms are not always available for every Java™ Runtime Environment (JRE) implementation. If a particular security implementation is not available, add the required libraries to the class path of the JRE that this product uses.

The generic service tester displays the envelope as reflected in the XML document. However, security algorithms consider the envelope as a binary. Therefore, you must set up the SOAP security configuration so that incoming and outgoing messages are correctly encrypted but remain decrypted inside the test.

**Performance**

Virtual user performance depends on the implementation of the container application. For an HTTP transport, the product has been tested with a maximum of 900 concurrent virtual users under Windows™ and 600 under Linux™. For JMS, the maximum is 100 concurrent virtual users, although this number can vary due to the asynchronous implementation of JMS. Beyond these values, connection errors might occur and the transaction rate will decrease.

## Verifying WSDL syntax compliance for JMS services

Various Java™ Message Service (JMS) providers vary in the syntax used for describing services. Before testing JMS services, you must ensure that Web Services Description Language (WSDL) files comply with the requirements of the tool.

1. In the project explorer or test explorer, locate and open the WSDL file for the JMS service that you want to test. If necessary, you can import a WSDL file from the file system by clicking **File > Import > File System**.
2. Ensure that the following criteria are met in the syntax of the WSDL file that you use.

   - Namespace: `xmlns:jms="http://schemas.xmlsoap.org/wsdl/jms/"`
   - SOAP bindings are set to: `transport="http://schemas.xmlsoap.org/soap/jms"`
   - JMS transports are defined either as a URL or as `jms:address` element
3. If the WSDL file is not compliant, edit the file so that it meets the criteria, and then save and close the file.

**Exemple**

For example, a JMS defined as a URL looks like this:

```
<soap:address location="jms:/queue?jndiConnectionFactoryName=UIL2ConnectionFactory;
             jndiDestinationName=queue/testQueue;
             initialContextFactory=org.jnp.interfaces.NamingContextFactory;
             jndiProviderURL=9.143.104.47"/>
```

A JMS defined as an address looks like this:

```
<jms:address destinationStyle="queue"
             jndiConnectionFactoryName="myQCF"
             jndiDestinationName="myQ"
             initialContextFactory="com.ibm.NamingFactory"
             jndiProviderURL="iiop://something:900/">
</jms:address>
```

## Configuring the environment for SOAP security

SOAP security profiles require access to the libraries that implement encryption, signature, and other security algorithms that transform the XML messages before sending and after receiving them. You must prepare an environment with these libraries to use SOAP security, set the class path of the Java™ Runtime Environment (JRE) that Eclipse uses, and set the class path of the virtual machine that the Agent Controller uses.

**Before you begin**

Before you can test SOAP-based services that use security algorithms, you must obtain a set of security libraries and configuration files for SOAP.

BouncyCastle (http://www.bouncycastle.org) is a provider of such security libraries. Use of these security libraries is optional for the Rational® test product.

1. Copy the library files into the `jre/lib/ext` of the JRE installation.

   By default, this is the following directory: `C:\Program Files\IBM\SDP\jdk\jre\lib\ext`

2. Add the following VM argument either to the Eclipse launch command line or to the `eclipse.ini` file:
   `-vmargs-Dosgi.parentClassloader=ext`

   The `eclipse.ini` file is located in the same directory as the `eclipse.exe` launcher binary that is used to run the product.

**What to do next**

To configure a remote computer that uses only the Agent Controller and does not require access to the workbench, perform only step 1 and restart the Agent Controller service.

After configuring the environment, you must import a Web Services Description Language (WSDL) file and use the **WSDL security editor** to set up a security profile for the WSDL file.

## Recording a service test with the generic service client

You can record a service test by invoking service requests with the generic service client. After you have sent the requests and received the responses from the service, select the results in the History section of the generic service client to generate a test. If you do not have access to a dedicated client for the service calls, the generic service client is the easiest way to generate the calls and to record a test.

**Before you begin**

If you are testing a SOAP-based web service, ensure that you have access to a valid Web Services Description Language (WSDL) file. The wizard can import WSDL files from the workspace, the file system, a remote repository, or from a URL. Ensure that the WSDL files use the correct syntax for the test environment. The generic service client might not work with some WSDL files.

If you are using Secure Sockets Layer (SSL) authentication, ensure that you have the required key files in your workspace.

If you are using SOAP security, ensure that you have configured the environment with the correct libraries and configuration files. See Configuring the environment for SOAP security on page 123 for more information.

If the response in a recording or test generation is in XML and the size of the XML data is higher than the value set in the **XML Message Received maximum length** field, the response is automatically converted to text to avoid any memory issues. To convert the full response to text, the tool checks the value set for **Text Message Received maximum length**. If the value is lesser than the size of the response, the response is truncated. If you want the response to be in XML when the response size exceeds the value set in **XML Message Received maximum length**, you can manually increase the value for both recording and test generation. To change the value for recording, click **Window > Preferences > Generic Service Client > Message Edition**. To change the value for test generation, click **Window > Preferences > Test > Test Generation > Service Test Generation**.

**About this task**

You can also record and generate a test by using REST APIs. The API documentation to record a test is located at `C:\Program Files\IBM\IBMIMSHARED\plugins\com.ibm.rational.test.lt.server.recorder.jar`. The API documentation to generate a test after the recording completes is located at `C:\Program Files\IBM\IBMIMSHARED\plugins\com.ibm.rational.test.lt.server.testgen.jar`.

To use a WS-SecurityPolicy that is included in a WSDL or an external XML file, you need to configure the security policy as described in Using a security policy on page        . If a recording contains the Security Assertion Markup Language (SAML) token, the WS Security policy file must rely on the Service Token Service (STS) that produces the token. This token can then be used for encryption or other purposes as was designed.

Sample policy file that relies on SAML token:

```
<sp:SupportingTokens xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
<wsp:Policy>
<sp:IssuedToken
 sp:IncludeToken="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy/IncludeToken/AlwaysToRecipient">
<sp:Issuer>
<Address
 xmlns="http://www.w3.org/2005/08/addressing">http://9.143.105.204:8080/axis2/services/STS</Address>
</sp:Issuer>
<sp:RequestSecurityTokenTemplate>
<t:TokenType
 xmlns:t="http://schemas.xmlsoap.org/ws/2005/02/trust">http://
docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0</t:TokenType>
<t:KeyType
 xmlns:t="http://schemas.xmlsoap.org/ws/2005/02/trust">http://
schemas.xmlsoap.org/ws/2005/02/trust/SymmetricKey</t:KeyType>
<t:KeySize xmlns:t="http://schemas.xmlsoap.org/ws/2005/02/trust">256</t:KeySize>
</sp:RequestSecurityTokenTemplate>
<wsp:Policy>
<sp:RequireInternalReference/>
</wsp:Policy>
</sp:IssuedToken>
```

```
</wsp:Policy>
</sp:SupportingTokens>
```

1. In the Performance Test perspective, click the **New Test from Recording** toolbar button or click **File > New > Test from Recording**.

2. In the **New Test from Recording** wizard, click **Create a test from a new recording**, select **Service Test**, and click **Next**.

   If you are recording sensitive data, you can select a **Recording encryption level**.

3. On the **Select Location** page, select the project and folder where you want to create the test, type a name for the test, and click **Next**.

   If necessary, click **Create Parent Folder** to create a project or folder

4. On the **Select Location** page, select **Generic Service Client**.

   This option uses the generic service client if you do not have access to a dedicated client for the service calls. See Recording a service test through a client program on page 126 for information about using other client programs to record the test.

5. Click **Next**. If this is the first time you are recording a web service test, read the Privacy Warning, select **Accept**, and click **Finish** to proceed.

   **Result**

   The generic service client opens.

6. If your service uses a transport or authentication protocol that requires overriding the default settings, then click the **Transport** tab and create an HTTP, Java™ Message Service (JMS), IBM® WebSphere® MQ, IBM® WebSphere® Java MQ, or Java MQ transport.

7. Click the **Requests** tab.

   **Choose from:**

   - Right-click **WSDLs** and select one of the options to get the WSDL file.
   - Right-click **WADLs** and select one of the options to get the WADL file.
   - Right-click **Endpoints** and select one of the options to send the request.

   See Sending service requests with the generic service client on page 259 for more information about using the generic service client.

8. After creating the call, click the **Edit Data** arrow to change the details of the call if necessary.

9. Click the **Invoke** arrow to invoke the service call.

   **Result**

   If the call was successful, the response is displayed under the **View Response** arrow.

10. To record a test with multiple calls, repeat steps 6 through 9.

11. When you have finished sending service requests, stop the recorder. You can do this by closing the generic service client or by clicking the **Stop** push button in the **Recorder Control** view.

    If you changed the network settings of the client program as described in step 8, you can revert to the default settings before closing the program.

    **Result**

    The **Generate Service Test** wizard opens.

12. Click **Finish**.

**What to do next**

Alternatively, you can use the generic service client to create, edit, and invoke the calls without recording. Successful responses are added to the **Request History** list. You can select calls in the **Request History** list, and click the

**Generate Test Suite** icon .

---

Related information

## Recording a service test through a client program

You can record tests for SOAP-based, XML, plain text, or binary services with any client program that uses the HTTP protocol. To record the test, the recorder intercepts the service calls and message returns between the client and the service. You can choose between an HTTP or SOCKS proxy recorder or a low-level socket recorder, depending on the capabilities of the client program.

**Before you begin**

The following recorders are available for recording traffic from an application:

- SOCKS proxy recorder: Use this recorder when no proxy connections are required.
- HTTP proxy recorder: Use this recorder when a proxy connections is required to connect to the network or when the client program does not support SOCKS.
- Socket recorder: Use this recorder for low-level network traffic when the client does not support proxies. This recorder does not support SSL authentication or encryption of any kind and is only available if the IBM® Rational® Performance Tester Extension *for Socket Protocols* is installed.

Regardless of the recorder that you use, the client program must use the HTTP network protocol. For recording Java™ Message Service (JMS) or IBM® WebSphere® MQ tests, see .

If you are using Secure Sockets Layer (SSL), the HTTP or SOCKS proxy can cause authentication problems because the proxy recorder relays traffic between the client and the server. Depending on the authentication method in place, the client might require that the proxy recorder authenticate itself as the server and the server might require that the proxy recorder authenticate as the client. If the client program requires an authenticated server, you must either have access to the server certificate keystore and provide it to the proxy recorder or configure the client to accept the default certificate from the proxy recorder instead of the certificate from the actual server.

If you are testing a SOAP-based web service, ensure that you have access to a valid Web Services Description Language (WSDL) file. The wizard can import WSDL files from the workspace, the file system, a remote repository,

or from a URL. Ensure that the WSDL files use the correct syntax for the test environment. The generic service client might not work with some WSDL files.

If you are using SOAP security, ensure that you have configured the environment with the correct libraries and configuration files. See Configuring the environment for SOAP security on page 123 for more information.

To record a service test with a client program:

1. In the Performance Test perspective, click the **New Test from Recording** toolbar button  or click **File > New > Test from Recording**.
2. In the **New Test from Recording** wizard, click **Create a test from a new recording**, select **Service Test**, and click **Next**.
   If you are recording sensitive data, you can select a **Recording encryption level**.
3. On the **Select Location** page, select the project and folder to create the test in, type a name for the test, and click **Next**.
   If necessary, click **Create Parent Folder**  to create a project or folder
4. On the **Select Client Application** page, select the type of client program to use.
   The program type defines the recorder that can be used. The following client program types are supported for recording a service test:
   **Choose from:**
   - **Managed Application**: This option starts a specified program and uses a proxy or socket recorder to record the traffic.

     On the **Managed Application Options** page, click **Browse** to specify the **Program path**. If necessary, specify the **Working directory**, and type the command line **Arguments** that the program requires.

     If the program requires user input from a command-line interface, select **Open console for user input**.
   - Choose a web browser to record traffic that is sent and received with the web browser.
   - **Unmanaged Application**: This option enables you to record traffic from one or multiple client programs that use a proxy. You must manually start the client programs and the proxy recorder records all traffic that is sent and received through the specified network port.
   - **Generic Service Client**: This option uses the generic service client if you do not have access to a dedicated client for the service calls. See Recording a service test with the generic service client on page 123 for using the generic service client to record service tests.
5. On the **Recorder Settings** page, depending on the type of client program you selected, specify these details:

   a. If you selected **Managed Application**, specify the recording method.
      - Select **Record traffic with the proxy recorder** to record HTTP or SOCKS traffic through a proxy.
      - Select **Record traffic with the socket recorder** to record low-level network traffic for applications where a proxy cannot be used. This recorder does not support SSL authentication or encryption.

> ✎ **Note:** When using proxy recording, you can filter out HTTP or HTTPS requests to a specific endpoints so that any requests to those endpoints are not recorded. See Proxy recording preferences

b. If you selected **Record traffic with the proxy recorder**, specify whether the proxy recorder uses HTTP or SOCKS. Select **HTTP** if a connection to proxy is required or if your application does not support SOCKS.

c. If you are using SSL authentication, specify the authentication settings for the proxy recorder. During the recording, the proxy recorder is between the client and the server.

- If the server requires client SSL authentication, provide the client certificate for the proxy recorder to be authenticated by the server as though the proxy recorder were the client. Select **The server requires a specific client certificate**.

   To provide single certificate keystore, specify the file name and password of the server certificate keystore. If multiple certificates are required, click **Multiple certificates**, and click **Add** to specify a certificate keystore file name and password for each host name and port.

   To provide smart card authentication, specify the certificate alias and PIN code of the smart card. See the Smart card authentication topic for more information.

   > ✎ **Note:** You must ensure that the smart card reader device is connected to the machine that has  Rational® Performance Tester and you have inserted the smart card into the reader.

- To record a secured site using Internet Explorer or Google Chrome on Windows, install the recorder certificate by selecting **Register the recorder root certificate authority**. Before the recording starts, the browser prompts you to install the certificate. After the recording is stopped, the browser prompts you to uninstall the certificate. To avoid multiple prompts for each recording, select **Keep the recorder root certificate authority after recording**.

   > ✎ **Note:** If you already had the certificate from a version prior to 9.2.1 and then install the latest version of the product, you might have to install the certificate again.

   This option is not available when you record by using the Firefox or Safari browser. To record a secured site on these browsers, manually import the certificate in the browser from the default location `C:\Program Files (x86)\__VENDOR_NAME__\__VENDOR_NAME__IMShared\plugins \com.ibm.rational.test.lt.recorder.proxy_version\SSLCertificate.` For information about how to import the certificates, see the browser's documentation.

- If the client requires server authentication, you must provide the server certificate keystore for the proxy recorder to be authenticated by the client as though the proxy recorder were

the server. Select **The client requires a specific server certificate**, and click **Add** to specify a certificate keystore filename and password for each hostname and port. If you do not select this option, the proxy recorder provides its own default certificate.

> 📝 **Note:** The keystore must contain the private certificate of the server.

    d. If you selected to use the HTTP proxy recorder, specify how to connect to the network. If necessary, specify an HTTP or SOCKS proxy or point to a proxy auto-configuration (PAC) file.
Use this option if you are connecting to the service through a corporate proxy or firewall.

6. Click **Next**. If this is the first time you record a service test and you did not select a web browser for the client application, read the Privacy Warning, select **Accept**, and click **Finish** to proceed.

7. If you selected a proxy recorder with a managed or unmanaged application, change the network settings of the client program to use the proxy recorder.
The method for changing the network settings depends on the client program. However, you must be able to set the following proxy settings in the program:
   - SOCKS or HTTP proxy: Specify the protocol that you selected for the proxy recorder in the wizard.
   - Host name: Set to `localhost`.
   - Port: Specify the port number that you selected for the proxy recorder in the wizard.
   To avoid unexpected results, revert to the previous proxy settings before you stop the recording.

8. Use the client program to perform the actions to test.
You can use the **Recorder Test Annotations** toolbar to add comments, record synchronizations, or take screen captures during the recording.
   - To add a comment to the recorded test, click the **Insert comment** icon 📋.
   - To add a screen capture to the recorded test, click the **Capture screen** icon 📷. Screen and window captures make your tests easier to read and help you visualize the recorded test. You can change the settings for screen captures and add a comment to the image.
   - To manually add a synchronization point to the recording, click the **Insert synchronization** icon 📶.
   - To manually add a transaction folder to the recording, click the **Start Transaction** icon 🕐 and **Stop Transaction** 🕐 icon to start and stop the transaction.
   - To insert a split point into the recorded test, click the **Split point** icon 🔀. With split points, you can generate multiple tests from a single recording, which you can replay in a different order with a schedule.

9. After you finish the user tasks in the client program, stop the recorder. You can do this by closing the client program or by clicking the button **Stop** ⬜ in the **Recorder Control** view.
If you changed the network settings of the client program as described in step 8, you can revert to the default settings before closing the program.
**Result**
The **Generate Service Test** wizard opens.

10. If you inserted a split point during the recording, on the **Destination** page, specify the location for the split test or merge the split recordings together.

See Splitting an HTTP test during recording for more information about splitting tests.

11. On the **Service Test Generation Options** page, if you are testing a SOAP-based web service, specify a Web Services Description Language (WSDL) file from the workspace or click **Add** to import a WSDL or to link to a remote WSDL file and click **Next**.

12. Select the domains to include in the test and click **Finish**. The domains that are not selected are not included in the test. You can add them back by generating the test again from the recording.
To include all the domains for all of the recordings, click the **Select all and remember my decision** check box. To enable the filter again for HTTP tests, click **Window > Preferences > Test > Test Generation > HTTP Test Generation**, and, for Service tests, click **Service Test Generation** and then click the **Enable domain review before test generation** check box.

13. Click **Finish**.

**Results**

A progress window opens while the test is generated. On completion, the **Recorder Control** view displays the `Test generation completed` message, the test navigator lists your test, and the test opens in the test editor.

Related information

Recording a service test with the generic service client on page 123

Sending service requests with the generic service client on page 259

Recording sensitive session data on page 154

Sending service requests with WSDL files on page 272

## Preparing to record a test for the HTTP/2 service

To test a web service that is based on the HTTP/2 protocol, record a test by using the SOA extension of  Rational® Performance Tester. Before recording the HTTP/2 service, follow the procedure in this topic to configure your computer.

**About this task**

This configuration is required because this feature is released as Beta and is intended for use in a non-production environment only.

Use Mozilla Firefox or Google Chrome when recording on servers that support the HTTP/2 service.

1. Download the following Application Layer Protocol Negotiation (ALPN) boot jar file https://mvnrepository.com/artifact/org.mortbay.jetty.alpn/alpn-boot/8.1.8.v20160420

2. Create or rename the `productInstallDir\jdk` folder to `..\jdk.ibm`. You can rename the folder back to `jdk` later to test with the IBM JDK.

3. Download Oracle Java 1.8.0u92 from http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html. You can then either extract the compressed file or install Java at `productInstallDir\jdk`.

4. Copy the ALPN jar file to `productInstallDir\majordomo\lib`.

5. From `productInstallDir`, open `eclipse.ini` and add the following flags:

   ```
   -Xbootclasspath/p:<productInstallDir>\majordomo\lib\alpn-boot-8.1.8.v20160420.jar
   ```

   **Note:** If there are any other flags starting with -X, delete those flags.

6. Configure  Rational® Performance Tester Agent to use Oracle Java.

   a. Stop the Majordomo process.

      On Windows systems, run the following command: `cd "c:\program files\ibm\sdp\majordomo" ngastop`

      On Linux systems, run the following command: `cd /opt/IBM/SDP/Majordomo ./MDStop.sh`

   b. Set the environment variable *RPT_JAVA* to the Oracle Java binary or executable file.

      On Windows systems, run the following command: `set RPT_JAVA=c:\program files\java\jdk1.8.0_92\bin\java.exe`

      On Linux systems, run the following command: `export RPT_JAVA=/root/jdk1.8.0_92/jre/bin/java`

   c. Start the Majordomo process.

7. When you record a service by using a web browser against an HTTP/2 client, the HTTP/2 traffic is automatically captured. But, to record an HTTP/2 service by using the GSC client, in the HTTP transport protocol configuration dialog box, you must select the **Activate** check box.

8. Specify the following configuration options for HTTP/2:

**HTTP/2**

> 📝 **Note:** Testing HTTP/2 service is in the Beta mode. For more information, see Preparing to record a HTTP/2 service on page 130.

To test a service that uses the HTTP/2 protocol, select the **Activate** check box. This check box is automatically selected when you record a service by using a browser. If you use the Generic Service Client component to create a HTTP/2 test, you have to manually select the check box.

**HTTP/2 client connection timeout**

Specifies the time limit for the HTTP/2 client to connect to the HTTP/2 server.

**Time out for the HTTP/2 session creations**

Specifies the time limit to create the HTTP/2 session. This time starts after the connection is established.

**Enable HTTP/2 Push**

The Push functionality of HTTP/2 automatically identifies and passes the related objects or requests to the client when a request is sent to the server. Clear the check box to not use the functionality.

**Initial session window**

Specifies the buffer size on the sessions.

**Initial stream window**

Specifies the window size for buffer on each stream after the connection is established.

**HTTP/2 Client Input Buffer Size**

Specifies the buffer size that is used to read the network traffic.

**Maximum Quantity of Messages that can be queued**

Specifies the maximum number of messages that can be queued for the HTTP/2 client on a thread.

**Maximum Quantity of HTTP/2 thread pool**

Specifies the maximum number of thread pools that will be used by the HTTP/2 client to distribute the workload.

**Minimum Quantity of HTTP/2 thread pool**

Specifies the minimum number of thread pools that will be used by the HTTP/2 client to distribute the workload.

**HTTP/2 client bytebuffer pool size**

Specifies the buffer size to receive the unciphered values.

**Server Name Indication**

> **Note:** Not applicable for HTTP/2.

Clear this check box if you do not want to connect to the host computer by using the Server Name Indication protocol. If the host computer is already configured with Server Name Indication protocol, you should keep this check box selected.

**Use HTTP Keep Alive**

Select this option to keep the HTTP connection open after the request. This option is not available if you are using IBM® Rational® AppScan®.

**Use SSL**

Select this option to use an SSL configuration. Click **Configure SSL** to create an SSL configuration or select an existing configuration.

**Platform Authentication**

In this section, specify the type of authentication that is required to access the service. Select **None** if no authentication is required.

**Basic HTTP authentication**

Select this option to specify the **User Name** and **Password** that are used for basic authentication.

**NTLM authentication**

**Note:** Not applicable for HTTP/2.

Select this option to use the Microsoft™ NT LAN Manager (NTLM) authentication protocol. NTLM uses challenge-response authentication. This view lists what is negotiated (supported by the client and requested of the server) and what is authenticated (the client reply to the challenge from the server).

**Kerberos authentication**

**Note:** Not applicable for HTTP/2.

Select this option to use the Kerberos authentication protocol between the client and server.

**Connect through proxy server**

**Note:** Not applicable for HTTP/2.

If the HTTP connection needs to go through a proxy server or a corporate firewall, specify the **Address** and **Port** of the proxy server. If the proxy requires authentication, select either **Basic proxy authentication** or **NTLM proxy authentication**.

**Proxy authentication**

In this section, specify the type of authentication that is required to access the proxy. Select **None** if no authentication is required.

**Basic proxy authentication**

Select this option to specify the **User Name** and **Password** that are used for basic authentication.

**NTLM proxy authentication**

Select this option to use the Microsoft™ NT LAN Manager (NTLM) authentication protocol. NTLM uses challenge-response authentication. This view lists what is negotiated (supported by the client and requested of the server) and what is authenticated (the client reply to the challenge from the server).

**Custom class**

> 📝 **Note:** Not applicable for HTTP/2.

Select this option if the communication protocol requires complex, low-level processing with a custom Java™ code to transform incoming or outgoing messages. Click **Browse** to select a Java™ class that uses the corresponding API. This option is not available in IBM® Security AppScan®.

9. Click **OK**. You have configured the workbench to test an HTTP/2 service.

**What to do next**

You can now record a regular SOA test for the HTTP/2 service. After the recording, in the Version field of request details, the requests are marked with HTTP/2 indicating that the HTTP/2 traffic is captured. If the test playback fails, check if all the steps are correctly followed.

## Optimizing HTTP/2 tests for SOA

HTTP/2 tests require a lot of CPU and memory resources. When you apply load on HTTP/2 tests using computers that do not have enough resources, the tests might fail. You might want to configure or tune the computers that run HTTP/2 tests.

**CPU Usage**

Ensure that the HTTP/2 tests get adequate CPU resources to run. If there are other processes running on the computer and they are not required, you can stop them. For example, the CPU usage statistic in the image below indicates that the other processes (shown in red) on the computer are consuming a lot of resources whereas the test execution process (shown in green) is getting less resources.
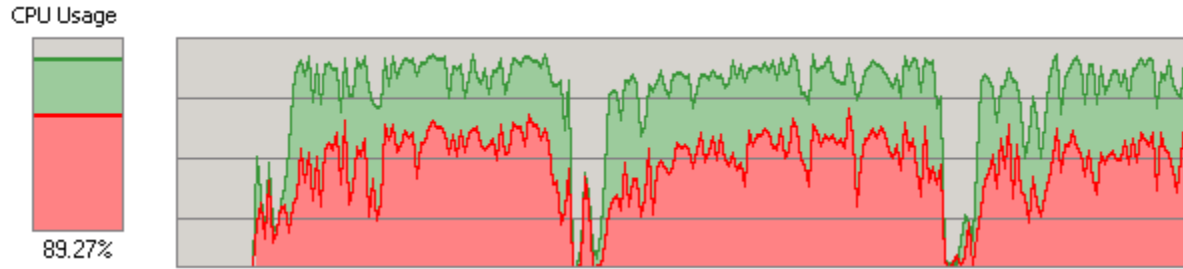
## Memory usage and garbage collection

Ensure that enough memory is available for the test execution. You can configure the garbage collector and adjust the memory heap size.

Garbage Collection - Consider using the following values so that the garbage collector does not allocate large amount of temporary memory. By doing so, you are tuning the number of threads allocated for the garbage collector according to the capability of the computer. You apply the values for each location asset of the schedule.

**RPT_VMARGS**

-XX:MaxGCPauseMillis=250 -XX:ParallelGCThreads=6 -XX:ConcGCThreads=3 -XX:GCTimeRatio=19



Memory heap - Consider using the following values for memory heap:

**RPT_VMARGS**

-Xms11024m

**RPT_DEFAULT_MEMORY**

22412m

## Thread Usage

Ensure that you start load testing with fewer virtual testers and gradually ramp up the workload. This practice helps in observing the changes in the workload, that is, the number of calls per second. In the graph, when the number of calls per second is flat, it indicates that the maximum capacity of the computer is reached and there is no need to add more virtual testers.

## Platform tuning

Configure the TCP/Socket capabilities of your system by following these two links:

https://wiki.eclipse.org/Jetty/Howto/High_Load

https://msdn.microsoft.com/en-us/library/aa560610%28v=bts.20%29.aspx

# Creating a service test from a BPEL model

You can use Business Process Execution Language (BPEL) resources from your workspace to automatically generate a set of service tests that corresponds to the paths that are run in a synchronous BPEL model.

**Before you begin**

Tests are stored in test projects. If your workspace does not contain a test project, the test creation wizard creates one, enabling you to change its name. To store a test in a specific project, verify that the project exists before you record the test.

If you are using Secure Sockets Layer (SSL) authentication, ensure that you have any required key files in your workspace.

If you are using Java™ Message Service (JMS), ensure that you have configured the environment with the correct libraries and configuration files. Ensure that the WSDL files use the correct syntax for the test environment.

If you are using SOAP security, ensure that you have configured the environment with the correct libraries and configuration files.

BPEL models must be synchronous. Asynchronous BPEL models are not supported.

Ensure that the BPEL models refer to the WSDL files in a valid import statement, for example:

```
<bpws:import importType="http://www.w3.org/2001/XMLSchema" location="foo.wsdl" namespace="http://foo"/>
```

Relative file paths, such as: "../../foo.wsdl" are not supported.

Ensure that you have one or more valid Web Services Description Language (WSDL) files and the associated BPEL model in your workspace. Only the calls to services with a valid web service binding are taken into account. For example, if the BPEL model was produced in IBM® Websphere Integration Developer, then services must be exported with the following web service bindings:

```
<bpws:invoke name="myOperation" operation="myOperation" partnerLink="IServicePartner"
 portType="ns3:IService" wpc:displayName="myOperation" wpc:id="20">
```

Only BPEL *invoke* activities are considered for generating tests. Any BPEL *receive* and *reply* activities are ignored.

Websphere Integration Developer does not generate the required `soapAction` attributes for the soap operations in the WSDL files. Please edit the generated WSDL files, as follows for every operation: `<soap:operation soapAction=""/>`.

To create a service test from a BPEL model:

1. In the Performance Test perspective, click **File > New > Other > Test > Test Assets > BPEL to Web service test**, and then click **Next**.
2. Click **Browse** to select a BPEL file from the workspace, and click **Next**.
3. On the **Web service test generation** page, change the number of paths by specifying how activities and sequences from the flow of the BPEL model are processed. Each path generates one test.
   a. In the **Flow** section, select how any concurrent sequences that are found in the flow will be converted into paths.
   b. In the **Switch** section, select whether to test *otherwise* activities from the flow.
   c. In the **Throw** section, select how *throw* activities from the flow are converted into paths.
   d. In the **Invoke** section, select whether to test inline catches inside *invoke* activities from the flow.
   e. Select **Enable data correlation in generated tests** to automatically create references in the generated test elements by propagating variables to the parameters of the web service call and message return elements.
4. Click **Recount paths** to update the number of paths to test, and click **Next**.
   One test is generated for each path.
5. For WSDL operations that are bound to multiple ports, you must select one port that is to be used for the test.

   Under each test that will be generated, the **Operations** list displays the WSDL operations that are bound to multiple ports.

   If no WSDL operations are displayed under the tests, this means that all operations are bound to a single port. In this case, skip step 6.

   a. In the **Operations** list, expand a test and select a WSDL operation that requires binding.
   b. In the **Binding ports** list, select the port that you want to use to test the selected WSDL operation.
   c. Repeat steps a and b for each WSDL operation that requires binding.

6. Click **Next**.
7. Select a location and a name for the new folder where the tests generated from the BPEL model are created, and click **Finish**.

**Results**

A new folder is created in the Test Navigator containing the generated service tests. These tests are generated with default message content and must be edited with valid input values.

## Creating a service test manually

You can create a service test without recording by simply adding the test elements as required and manually editing the test element details in the test editor.

**Before you begin**

Tests are stored in test projects, which are test projects that include a source folder. You must create a test project before creating a test.

Ensure that you have a valid WSDL file in your workspace. Ensure that the WSDL files use the proper syntax for the test environment.

If you are using Secure Sockets Layer (SSL) authentication, ensure that you have any required key files in your workspace.

If you are using SOAP security, ensure that you have configured the environment with the proper libraries and configuration files.

1. In the workbench, click **File > New > Other > Test > Test Assets > Web service test** or click the **New Service Test** toolbar button.
2. Select a project and, in **Name**, type a name for the test, and then, click **Next**.
   The name that you type is the base name for the recording, test, and other required files. You see these files in standard Navigator or the Java™ Package Explorer with their distinguishing suffixes, but you see only the simple (test) name in the Test Navigator.
3. Select a web service request to create the test for.
   If you select **Web service request** or one of the options in **Specification-based structure**, specify a WSDL port and then configuration properties for the HTTP protocol. If you select, **XML request** and **Text request**, specify the configuration properties for the HTTP, JMS, WebSphere MQ, WebSphere Java MQ, and Microsoft.Net protocols.

   For information about the configuration properties of each protocol, see the topics in Sending service requests with the generic service client on page        .
4. Click **Finish**. The service test is created.

## Creating a service test for WebSphere® MQ

You can create an IBM® WebSphere® MQ test by adding the test elements as required and editing the test element details in the test editor.

**Before you begin**

Tests are stored in test projects, which are Java™ projects that include a source folder. You must create a test project before creating a test.

Ensure that you have a valid Web Services Description Language (WSDL) file for a WebSphere® MQ service in your workspace.

If you are using Secure Sockets Layer (SSL) authentication, ensure that you have any required key files in your workspace.

If you are using SOAP security, ensure that you have configured the environment with the correct libraries and configuration files.

1. In the workbench, click **File > New > Other > Test > Test Assets > Web service test** or click the **New Service Test** toolbar button.
2. Select a project, and then, in **Test file name**, type a name for the test and click **Next**.
   The name that you type is the base name for the recording, test, and other required files. You see these files in the standard Navigator or the Java™ Package Explorer with their distinguishing suffixes, but you see only the simple (test) name in the Test Navigator.
3. In the **Select a service request interface** page, complete one of the following steps:
   a. To test a service that use a WSDL file, select **Web service request** or **Specification-based structure**, click **Next**, and select a WSDL file.
   b. To test a service that does not use a WSDL file, select **XML Request**, **Text Request**, **Binary Request** or an **Empty test**.
4. Click **Next** and select the **WebSphere MQ** protocol.
5. In **SOAP Action**, specify the SOAP action to be used to invoke the MQ request.
6. To override the message header and descriptor that was specified in WebSphere MQ transport configuration, click **Override MQ Protocol Configuration values** and specify the customize header and message descriptor.
7. Click **Finish**. The service test is created.
8. On the web service call, click **Update Response**.
   This opens the **Response Preview** window, displaying the data that will be used to perform the call.
9. Click **Update Test**.
   This action calls the web service and creates a message return element with the return data. If a message return element already exists, then it is updated with latest return data. With the message return test element, you can implement data correlation and content-based verification points.

## Creating a service test for WebSphere Java MQ

To test Java-based applications, create a service test and add the WebSphere Java MQ messages. You can create a service test by using Generic Service Client option ▧ or the New Service Test wizard ⛁.

**Before you begin**

Connect to a WebSphere MQ server.

If you are using Secure Sockets Layer (SSL) authentication, ensure that you have any required key files in your workspace.

If you are using SOAP security, ensure that you have configured the environment with the correct libraries and configuration files.

1. In the workbench, click **File > New > Other > Test > Test Assets > Web Service Test** or click **Create a Service Test** ⛁.
2. Select a project, and then, in **Test file name**, type a name for the test.
   The name that you type is the base name for the recording, test, and other required files. You see these files in the standard Navigator or the Java™ Package Explorer with their distinguishing suffixes, but you see only the simple test name in the Test Navigator.
3. In the **Select a service request interface** page, complete one of the following steps:
   a. To test a service that use a WSDL file, select **Web service request** or **Specification-based structure**, click **Next**, and select a WSDL file.
   b. To test a service that does not use a WSDL file, select **XML Request**, **Text Request**, **Binary Request** or an **Empty test**.
4. Click **Next**, select the **WebSphere Java MQ** protocol, and specify a transport configuration. If necessary, click **New** to create the transport configuration for the call. See Creating a WebSphere Java MQ transport configuration on page 266.
5. Complete the following information in the **General** tab:

   📝 **Learn more about the UI elements in the General tab:**

   **Queue**

   Name of the queue as defined on the WebSphere MQ server.

   **Message type**

   The types of messages are these:
   - *Datagram* means that the message does not require a reply.
   - *Request* means that the message requires a reply.
   - *Reply* means that the message is a reply to an earlier request message.
   - *Report* means that the message is reporting on some expected or unexpected occurrence, usually related to some other message. An example is a request message that contained data that was not valid.

**Message Persistence**

This value indicates whether the message is persistent or not. If the message is persistent, it survives the system failures and restarts of the queue manager. If the message is not persistent, it survives a restart if it is present on a queue having the NPMCLASS(HIGH) attribute. However, even with the NPMCLASS(HIGH) attribute a message does not survive a QMGR class. Nonpersistent messages on queues having the NPMCLASS(NORMAL) attribute are discarded at queue manager restart, even if the message is found on the auxiliary storage during the restart procedure.

**Dynamic Reply**

Select this check box for the WebSphere MQ server to dynamically create a temporary queue as a reply. If this check box is not selected, the message in Reply Queue is used.

**Reply Queue**

This is the name of the message queue to which the application that issued the get request for the message should send the reply and report messages.

**Reply Manager**

This is the name of the queue manager on which the reply-to queue is defined.

**Additional properties**

Specify the additional properties for the queues.

6. **Optional:** If necessary, complete the following information on the **Config** tab:

**Learn more about the UI elements in the Config tab:**

**Message Priority**

This is the priority of the message. The lowest priority is 0.

**Encoding**

This is the numeric encoding of numeric data in the message. This value does not apply to numeric data in the MQMD structure itself.

**Expiry Interval**

This is the period of time, in tenths of a second, after which the message becomes eligible to be discarded if it has not already been removed from the target queue. The expiry interval is set by the application that put the message.

> ✏️ **Character set**
>
> This is the character set identifier of the character data in the application message data.

7. **Optional:** In the **Report** tab, select the report messages to receive.
8. **Optional:** If necessary, complete the following information in the **Context** tab:

> ✏️ **Learn more about the UI elements in the Context tab:**
>
> **Application Identity Data**
>
> This information is defined by the application suite. Use it to provide information about the message or its originator.
>
> **Application Origin Data**
>
> This information is defined by the application suite. Use it to provide additional information about the origin of the message.
>
> **Accounting Token**
>
> This information is needed by the application to appropriately charge for the work that is done as a result of the message.
>
> **User ID**
>
> This is the user identifier of the application that originated the message.

9. **Optional:** In the **Identifiers** tab, for the messages that require binary input, specify the ID in the string format in the second column. The first column is filled automatically in the hexadecimal format.
10. **Optional:** In the **Segmentation** tab, select the segment of the message and click **Next**.
11. If you had selected **XML Request**, click **Next**, select a XSD file and click **Finish**.
    **Result**
    The new service test is created.

**What to do next**

You can now enhance the test and run it.

## Creating a service test for a plain XML call

You can create a test for a plain XML call over HTTP, JMS, or IBM® WebSphere® MQ, by simply adding the test elements as required and editing the test element details in the test editor.

**Before you begin**

Tests are stored in test projects, which are Java™ projects that include a source folder. You must create a test project before creating a test.

If you are using Secure Sockets Layer (SSL) authentication, ensure that you have any required key files in your workspace.

If you are using SOAP security, ensure that you have configured the environment with the correct libraries and configuration files.

1. In the workbench, click **File > New > Other > Test > Test Assets > Service Test** or click the **New Service Test** toolbar button.
2. Select a project, and then, in **Test file name**, type a name for the test and click **Next**.
   The name that you type is the base name for the recording, test, and other required files. You see these files in the standard Navigator or the Java™ Package Explorer with their distinguishing suffixes, but you see only the simple (test) name in the Test Navigator.
3. On the **Select Service Call Interface** page, select whether you want to create a test using a plain **XML call** interface or a **Web service call** interface.
   If you select web service call interface, select or add a WSDL file and then, select port to which the call will be binded. Click **Next**.
4. On the **Configure Protocol** page, select either **HTTP**, **JMS** or WebSphere® **MQ** as the protocol and then, specify the options for the selected **Protocol configuration**.
5. On the **Select Root Element** page, you can select an XSD and then, select a root element for the call.
6. Click **Finish**.

## Changing service test generation preferences

You can change default test generation values by changing the preference settings. The default settings, however, are appropriate for recording in most cases.

1. Click **Window > Preferences > Test > Web Services Test Generation**
2. Select the setting to change.

   **Time out delay used for call**

   > This is the default time out for web service calls. If the web service does not respond within this period, an error is produced.

   **Think time default value**

   > This is the default think time for generated tests.

3. After changing a setting, click **Apply**.

## Digital certificates overview

The digital certificates feature enables you to run tests against servers that use Secure Sockets Layer (SSL) for applications that require client-side digital certificates to authenticate users.

A *digital certificate* is a file that binds a public cryptographic key with an identity (a user or an organization). Trusted certificate authorities issue digital certificates, which are then used to authenticate users and organizations for

access to websites, email servers, and other secure systems. A *certificate store* is an archive file that contains almost any number of digital certificates, possibly certificates that are issued from different certificate authorities.

To use digital certificates in tests:

1. Create a digital certificate store. For more information about this subject, see Digital certificate creation overview on page 146 and Creating a digital certificate store on page 149.
2. Record a test that requires that you use a digital certificate. For more information about this subject.
3. Associate a digital certificate with a test for playback. For more information about this subject, see Playing back a test with a digital certificate on page 152.
4. Optionally, you can associate the digital certificates in one or more digital certificate stores with a dataset. For more information about this subject, see Using a digital certificate store with a dataset on page          .

Create a certificate store by running the supplied KeyTool command-line program. The program creates a certificate store that contains digital certificates.

Record a test that requires using a digital certificate. Specify the certificate and password that to use, and then begin recording the test. Browse the website as you typically would to record a test.

After you have finished recording, open the test for editing. On the Common Options page, under Digital Certificates, click **Add**. Type the name of the certificate store that you created previously; then select the certificate that you want to use. Save the test. When you run this test, the digital certificate from the certificate store is submitted to the server.

If you have recorded a test that does not use SSL, you can convert that test to be secure by adding an SSL object to the corresponding Server Access Configuration in the test.
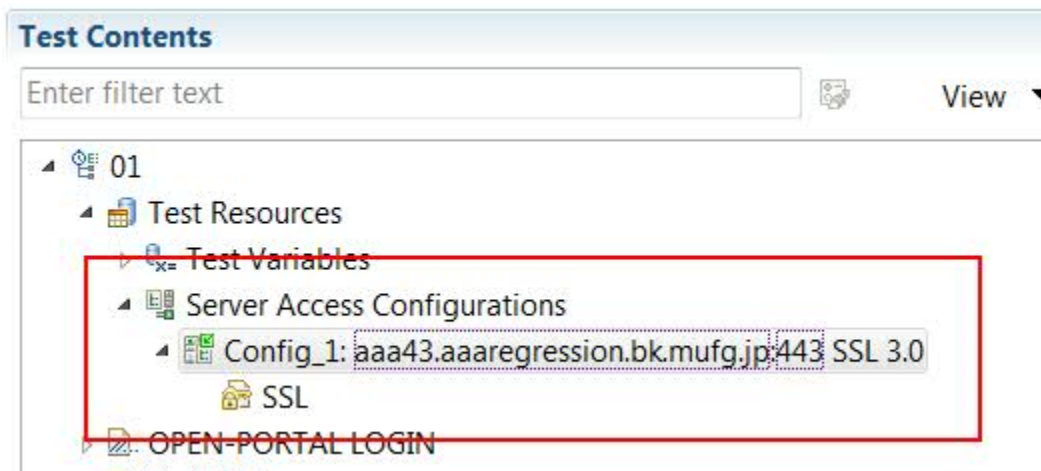


To use a certificate store with a dataset, open the test for editing. On the **Common Options** page, click **Add Dataset**. Create a dataset with two columns that contains a list of the certificates in the certificate store and a list of passphrases for the certificates. Select **Fetch only once per user**. Save the dataset. On the Common Options page, under Digital Certificates, click **Add**. Select the certificate store that you created previously from the **Certificate Store** column. Insert a **Certificate Name** for the digital certificate. Highlight this name, and then select **Substitute from dataset**. Choose the dataset added previously, and then choose the column with the certificate name. Repeat this

process to substitute passphrases from the dataset column containing passphrases. Save the test. Add the test to a schedule. When you run this schedule, the certificates from the certificate store are submitted to the server.

## Digital certificate creation overview

If you want to use digital certificates to run tests against applications that require client-side digital certificates to authenticate users, work with the appropriate server administrators to determine the types of certificates that you need to create.

In cryptography, a public key certificate is a document that uses a digital signature to bind a public key with a physical identity. These certificates are often referred to generically as digital certificates or client digital certificates. The most common standard for digital certificates is the X.509 standard.

In public key cryptography, each certificate has two associated keys: a public key and a private key. The public key is incorporated into the X.509 certificate and is always available with the certificate itself. The private key is always kept private (meaning, it is never transmitted). For ease of portability, the two keys (and the certificate) can be included in one, encrypted and passphrase-protected, format known as PKCS#12.

In order to verify the authenticity of a certificate, it is digitally signed by another certificate, known as a Certificate Authority (CA). This CA certificate may be one created (and kept secure) by a company hosting a secure application, or it could be created by a company such as Verisign.

When a web application requires digital certificates, an administrator typically creates digital certificates for each authorized user. The administrator digitally signs each certificate using the system CA certificate. These certificates, along with the public and private keys, are distributed to users. Often these keys will be distributed in the PKCS#12 format. Users then import these certificates into their web browsers. When the browser is challenged by the server, it will produce its certificate.

When importing certificates for web applications, select the check box that indicates that the keys be exportable. With this indication, the certificate can be exported to a PKCS#12 formatted file for later use by other programs.

Do not use certificates that are assigned to actual users for performance testing purposes. Use test certificates that do not correspond to actual users.

There are four types of certificates that can be used in testing:

- Self-signed certificates
- Signed certificates
- Certificate authority (CA) certificates
- Unsigned certificates (rarely used)

*Self-signed certificates* are used when no entity needs to vouch for the authenticity of the certificate. These are the simplest certificates to create and use. Typically, however, a signed certificate is used to represent a particular user.

*Signed certificates* are used when a certificate needs to be created for and issued to one, and only one, user. Signed certificates are signed by a certificate authority (CA).

*Certificate authority (CA) certificates* are self-signed certificates used to sign (certify) certificates.

*Unsigned certificates* are certificates that are neither signed by a CA nor self-signed. Most web applications do not use unsigned certificates.

When you create a self-signed or signed certificate (including CA certificates) you can specify a *subject*. The subject of a certificate is the set of attributes of an X.500 Distinguished Name that is encoded in the certificate. The subject enables the recipient of a certificate to see information about the owner of the certificate. The subject describes the certificate owner, but is not necessarily unique. Think of subjects as entries in a telephone book; there can be multiple entries for Patel Agrawal, but each entry refers to a different person.

The subject can contain many different types of identifying data. Typically, the subject includes the following:

| Attribute | Example |
|---|---|
| COMMON NAME (CN) | `CN=Patel Agrawal` |
| ORGANIZATION (O) | `O=XYZ Corporation` |
| ORGANIZATIONAL UNIT (OU) | `OU=XYZ Software Group` |
| COUNTRY (C) | `C=IN` |
| LOCALITY (L) | `L=Bangalore` |
| STATE or PROVINCE (ST) | `ST=Kanataka` |
| E-MAIL ADDRESS (emailAddress) | `emailAddress=agrawal@xyz.com` |

This information can be typed as one string, using forward slashes to separate the data.

For example, the above subject would be typed as follows:

```
/CN=Patel Agrawal/O=XYZ Corporation/OU=XYZ Software Group/C=IN/L=Bangalore/ST=Karnataka/
emailAddress=agrawal@xyz.com
```

To learn more about using the supplied command-line program to create certificates, see Creating a digital certificate store on page 149.

## Creating a digital certificate with OpenSSL

You can use the OpenSSL program to create digital certificates for use with tests.

**Before you begin**
OpenSSL is available from the OpenSSL Project at http://www.openssl.org/.

1. Create a certificate authority (CA).
   For the purposes of testing, this CA takes the place of a recognized CA on the Internet, such as VeriSign. You use this CA to digitally sign each certificate that you plan to use for testing.

a. Create a certificate request (CSR) file. The "subject" (-subj) describes the user of the certificate. Enter dummy values as shown. The following command line sets the password for the certificate to `abcdefg`.
**Example**

```
openssl req -passout pass:abcdefg -subj "/C=US/ST=IL/L=Chicago/O=IBM Corporation/OU=IBM Software
Group/CN=Rational Performance Tester CA/emailAddress=rpt@abc.ibm.com" -new > waipio.ca.cert.csr
```

b. Create a key file, `waipio.ca.key`, to store the private key.
This removes the password protection from the certificate request file so that you do not have to type the password every time you sign a certificate. Because the password protection has been removed, use the certificate request file for testing purposes only.
**Example**

```
openssl rsa -passin pass:abcdefg -in privkey.pem -out waipio.ca.key
```

c. Create an X.509 digital certificate from the certificate request. The following command line creates a certificate signed with the CA private key. The certificate is valid for 365 days.
**Example**

```
openssl x509 -in waipio.ca.cert.csr -out waipio.ca.cert -req -signkey waipio.ca.key -days 365
```

d. Create a PKCS#12-encoded file containing the certificate and private key. The following command line sets the password on the P12 file to `default`. Rational® Performance Tester uses password of `default` for all PKCS#12 files by default.
**Example**

```
openssl pkcs12 -passout pass:default -export -nokeys -cacerts -in waipio.ca.cert -out
waipio.ca.cert.p12 -inkey waipio.ca.key
```

**Result**
You now have a CA certificate (`waipio.ca.cert`), which can be installed into the web server under test and a private key file (`waipio.ca.key`) that you can use to sign user certificates.

2. Create a digital certificate for a user.

a. Create a CSR file for the user. Set the initial password to `abc`. Optionally, provide an appropriate subject.
**Example**

```
openssl req -passout pass:abc -subj "/C=US/ST=IL/L=Chicago/O=IBM Corporation/OU=IBM Software
Group/CN=John Smith/emailAddress=smith@abc.ibm.com" -new > johnsmith.cert.csr
```

b. Create a private key file without a password.
**Example**

```
openssl rsa -passin pass:abc -in privkey.pem -out johnsmith.key
```

c. Create a new X.509 certificate for the new user, digitally sign it using the user's private key, and certify it using the CA private key. The following command line creates a certificate which is valid for 365 days.

**Example**

```
openssl x509 -req -in johnsmith.cert.csr -out johnsmith.cert -signkey johnsmith.key -CA
waipio.ca.cert -CAkey waipio.ca.key -CAcreateserial -days 365
```

d. **Optional:** Create a DER-encoded version of the public key. This file contains only the public key, not the private key. Because it does not contain the private key, it can be shared, and does not need to be password protected.

**Example**

```
openssl x509 -in johnsmith.cert -out johnsmith.cert.der -outform DER
```

e. Create a PKCS#12-encoded file. The following command line sets the password on the P12 file to `default`.

**Example**

```
openssl pkcs12 -passout pass:default -export -in johnsmith.cert -out johnsmith.cert.p12 -inkey
johnsmith.key
```

Repeat this step to create as many digital certificates as needed for testing. Keep the key files secure, and delete them when they are no longer needed. Do not delete the CA private key file. You need the CA private key file to sign certificates.

**Results**

Now you can install the CA certificate (`waipio.ca.cert`) into WebSphere®. Optionally, create a user certificate specifically for your web server, and install it into WebSphere®.

You can use user certificates individually to record tests. To use the user certificates (`johnsmith.cert.p12`) during test editing and playback, compress them in ZIP format to a file with the .rcs extension. This creates a digital certificate store. To learn more digital certificate stores, see Creating a digital certificate store on page 149. You can also import user certificates into your web browser to interactively test them in your environment.

## Creating a digital certificate store

The KeyTool command-line program enables you to create a Rational® Certificate Store (RCS) file that contains digital certificates for use with tests. A Rational® Certificate Store (RCS) file is a compressed archive file that contains one or more PKCS#12 certificates. You can also use the KeyTool program to remove certificates from a certificate store.

**About this task**

Rational® Performance Tester acts as a proxy between the browser and the server application to record the data exchange. When a secured page is recorded using Rational® Performance Tester, the proxy certificate of the product is presented to the browser.

1. In the command line tool, navigate to the directory that contains the Keytool utility. By default, the utility is located at `C:\Program Files\IBM\SDP\jdk\jre\bin`.
2. Type the following command:

   **Example**
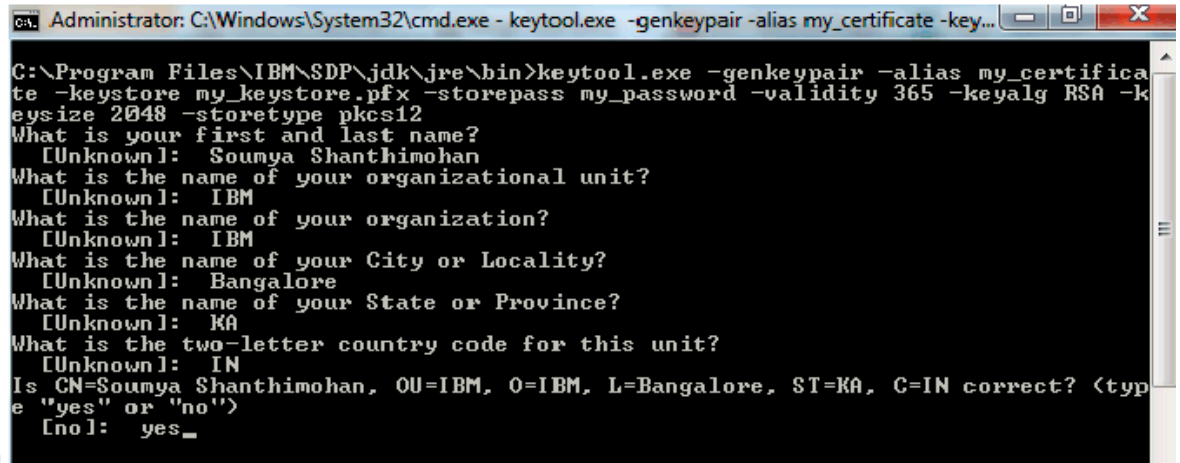
   ```
   keytool.exe -genkeypair -alias certificateName -keystore keystoreName -storepass password -validity 365
   -keyalg RSA -keysize 2048 -storetype pkcs12
   ```

   For additional information about parameters by certificate generation, review the official keytool [documentation](#).

| Option | Description |
|---|---|
| **-genkeypair** | Generate public and private keys for key pair. |
| **-alias** | Alias for your certificate in the key store. You may never use it, but every new certificate in your key store must have its own alias. |
| **-keystore** | Name of the key store file, which will be generated as the result of the command. It holds your certificate and a corresponding private key. You can reuse this key store for next certificates that you might generate. One key store can contain many certificates. |
| **-storepass** | Password that protects your key store file. You will have to enter it every time you want to sign a document. |
| **-validity** | Number of days the certificate is valid. You can enter more than 365. |
| **-keyalg** | Algorithm to generate the cryptographic keys that is corresponding to your certificate. You can use RSA or DSA. |
| **-keysize** | Length of the cryptographic keys. The more the length the stronger the signature. |
| **-storetype** | Format of the key store file. PKCS#12 (a.k.a PFX) key stores can be understood by a lot of different programs and you can also import a PKCS#12 file in |

| Option | Description |
|---|---|
| | your Windows key store (just double click it and follow the instructions). |

3. The certificate generation process prompts you to enter some information about you. Enter the information as



prompted.

4. If prompted for a password when using the keystore, enter the same password as provided on the command line.

   **Result**

   The key store file (.pfx) is stored in your current directory.

**Results**

You now have a digital certificate store that you can use with tests. Because the KeyTool program has many options, you might want to create an alias or script file to use to invoke KeyTool. Use KeyTool to create and add as many digital certificates as you want. If you want to create a dataset of the names of certificates in the certificate store, run KeyTool again with the `-list` option. This option writes a list of names that can then be imported to a dataset.

**What to do next**

Before you start recording the application that requires client certification, import the certificate to the Rational® Performance Tester project. For information about how to import the certificate and record a test, see the Recording a test on page topic.

You do not have to use the KeyTool command-line program to create a certificate store. It is possible to use existing PKCS#12 certificates with Rational® Performance Tester. PKCS#12 certificates can be exported from a web browser. PKCS#12 certificates encode the private key within the certificate by means of a password.

**Note:** Do not use certificates associated with real users. Certificates associated with real users contain private keys that should not become known by or available to anyone other than the owner of the certificate. An intruder who gained access to the certificate store would have access to the private keys of all certificates

in the store. For this reason, you must create, or have created for you, certificates that are signed by the correct certificate authority (CA) but that are not associated with real users.

## Playing back a test with a digital certificate

After you create a digital certificate store and record a test using a digital certificate, you must associate the digital certificate with the test for playback.

**Before you begin**

You need to record a test using a digital certificate, and you need a digital certificate store file containing one or more PKCS#12 certificates.

If your certificate extension is not .rcs, then you need to zip the certificate, rename the extension to .rcs, and copy it to the root directory of the project.

To associate a digital certificate with a test for playback:

1. Open the test for editing.
2. On the Security tab, under Digital Certificates, click **Add**.
3. Select or type the name of the certificate store file that you created previously.
   You must type or select the file name. You cannot browse to locate the file. The certificate store must be a Rational® Certificate Store (RCS) file. A Rational® Certificate Store file is a compressed archive file that contains one or more PKCS#12 certificates.

   > **Note:** For more information about using certificates that do not have a .rcs extention see: Using non-rcs digital certificates.

4. Select the digital certificate that you want to use, and then click **Select**.
5. When prompted to place the digital certificate in a dataset, click **No**. To learn more about substituting digital certificates, see Using a digital certificate store with a dataset on page        .
   **Result**

   > **Note:** If you add multiple certificates to the **Digital Certificates** list on the Common Options page, the first certificate that satisfies the request from the server (in the order by which the certificates were entered) is used during playback.

6. Save the test.

**Results**

When you run this test, the digital certificate from the certificate store is submitted to the server.

## Entrust TruePass authentication overview

Entrust provides digital identity and encryption technologies to governments and private industry. With Entrust TruePass software users can authenticate with secure web applications without installing a digital certificate in their browsers. This makes it convenient for use in kiosks and other public user environments.

You can now run tests against servers that require Entrust TruePass authentication. Roaming mode with TruePass applet version 7.0 and later are supported. Local mode, and versions of the TruePass applet prior to 7.0, are not supported. Recording tests with Entrust TruePass applications works just as regular HTTP recording does.

The Entrust TruePass Authentication object is displayed in the test editor for tests that you record with Entrust TruePass applications. The **Version** field displays the recorded version number of the Entrust TruePass applet. The **Server Name** and **Port** are correlated fields. Click **Substitute** to use the **Data Sources** view to change the server or port number for playback. The **Application Context** displays where the Entrust application is mapped to in the application server. The **User Name** and **Passphrase** fields can be substituted with values from a dataset.



## Annotating a test during recording

You can add comments, add transactions, or change a page name while you record a test. The advantage of adding these elements during (rather than after) recording is that you can place the annotations in the test exactly where you want. In addition, because annotations are part of the recorded test, they are regenerated when you regenerate the test. You can also insert split points into a test during record.

1. Start recording the test. The **Recorder Test Annotations** toolbar opens near the top of the screen.
2. Click the appropriate icon.

You can use the **Recorder Test Annotations** toolbar to add comments, record synchronizations, or take screen captures during the recording.

- To add a comment to the recorded test, click the **Insert comment** icon . You are prompted for a comment.
- To add a screen capture to the recorded test, click the **Capture screen** icon . Screen and window captures make your tests easier to read and help you visualize the recorded test. You can change the settings for screen captures and add a comment to the image.
- To manually add a transaction folder to the recording, click the **Start Transaction** icon  and **Stop Transaction** icon  to start and stop the transaction. Transactions can be nested.
- To insert a split point into the recorded test, click the **Split point** icon . With split points, you can generate multiple tests from a single recording, which you can replay in a different order with a schedule. See Splitting a test during recording for more information about splitting a test.
- When recording an HTTP test, to change the page name, click the **Change page name** icon . In the resulting test, the page element in the test editor uses the new name, however the original name is preserved in the **Page Title Verification Point** area so that page title verification points still work correctly.

3. Close the client program to stop the recording.
4. If you inserted a split point during the recording, on the **Destination** page, in the **Test Generation** wizard, specify the location for the split test or merge the split recordings together.

**Results**

The test is generated with the comments, transactions, and page names that you added.

## Recording sensitive session data

You can keep recording session (`.recsession`) files to view the contents of a recording or to regenerate tests. However, if a recorded test contains sensitive information, you can choose to obfuscate, or encrypt, text strings in the recsession file.

To protect test data in a recording session file:

1. In the Performance Test perspective, click the **New Test from Recording** toolbar button  or click **File > New > Test from Recording**.
2. In the **New Test from Recording** window, select **Create a Test from a New Recording**, and select the type of test to create.
3. In **Recording encryption level**, select one of these options:
   **Choose from:**

◦ **Obfuscated**: This setting hides text strings to prevent viewing the raw data in recsession files with a text editor outside of the workbench. You can still use recsession file to generate tests and to view recording information.

◦ **Passphrase**: This setting uses an AES-128-bit algorithm to encrypt text strings in the recsession files. The encryption strength depends on the length of the passphrase. The recording session file is unrecoverable if the passphrase is lost.

4. On the **Select Location** page, select the project and folder locations to contain the new test, type a name for the test, and click **Next**.

5. If you selected **Passphrase**, on the **Passphrase Protection** page, type the passphrase twice in **Passphrase** and **Confirm passphrase**.

   For solid protection, make the passphrase longer than 24 characters if using English words or at least 12 random characters.

6. Click **Next**, and continue the recording session for the type of test that you selected.

Related information

## Splitting an HTTP test during recording

You can insert split points when you record a test. With split points, you can generate multiple tests from a single recording that you can replay in a different order with a schedule. You can also create a schedule that contains all of the tests that are generated from the split points.

**About this task**

During the recording process, you can select the option to create a schedule for the tests that are generated from the split points. The schedule will contain these attributes:

- One user
- One user group for the local computer
- All of the tests from the recording, in serial order
- One stage: Run until finished
- Recorded think times, with the maximum think time set to 2 seconds
- Statistics:

  Statistics log level: All

  Statistics sample interval: 5 Seconds

  Only store All Hosts statistics
- Test Log:

  Show errors and failures: All

  Also show warnings: All

  And also show all other types: All
- Problem Determination log level: Warning

To insert split points when you record a test:

1. Start recording the test. The **Recorder Test Annotations** toolbar opens near the top of the screen.

2. To insert a split point into the recorded test, click the  **Split point** icon ⤴. The **Insert Split Point** window is displayed.

   **Choose from:**

   - Click **Test name**, and then type a name for this section of the test.

     ⓘ **Tip:** You are naming the previous section of the test, not the upcoming section of the test.

   Repeat this step between recorded user actions as needed to split tests.

3. Close the client program to stop the recording. The **Generate HTTP Test** window is displayed.
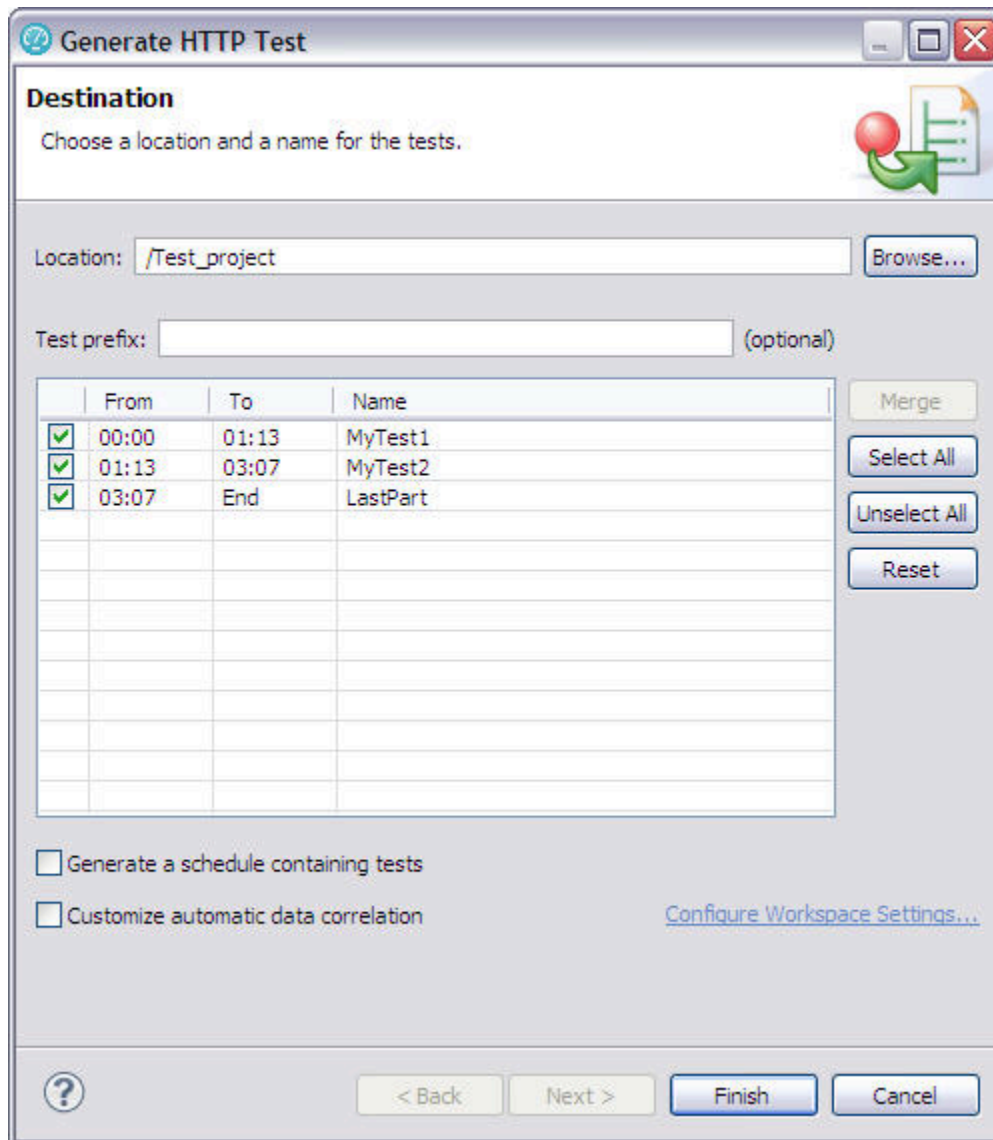


   **Choose from:**

- ◦ Type a **Test prefix** for all the tests in the schedule.
- ◦ Select **Generate a schedule containing tests** to create a schedule for the tests. When you select this option, you can modify the name of the schedule.
- ◦ Select **Customize automatic data correlation** to choose automatic data correlation or rule-based data correlation. If you select this option and choose rule-based data correlation, you can specify which data correlation rule sets to use.

4. Click **Finish**.

   **Result**

   The **Test Generation** window displays the status of generating the tests and schedule and the data correlation. You can view the test generation log from this window.

5. When test generation is complete, you can select the test to open and then click **Open Selected Tests**, or you can click **Close** to finish this process.

**Results**

The schedule and tests are generated using the names that you specified in the wizard.

## Generating a new test from a recorded session

You can generate a new test from a recorded session. For example, if you accidentally damage a test during editing, or if you want to change a test preference, you can regenerate the test instead of re-recording it. If split points were inserted in the recording, you can choose to generate a single test without split points.

To regenerate a complete test from a recording that contains split points:

1. In the test navigator, select the .recsession file of the test recording to regenerate.
2. Right-click, and then select **Generate Test**.

   **Result**

   The **Generation Test** wizard is displayed.
3. If the .recsession file is compatible with multiple test types, select the type of test that you want to generate and click **Next**.

   **Example**

   For example, select **HTTP Test** to generate an HTTP performance test.
4. On the **Select Location** page, select the project and folder where you want to create the test, type a name for the test, and click **Next**.

   If necessary, click **Create the parent folder** to create a new performance test project or a folder
5. If the .recsession file contains split points, on the **Options** page, select **Generate test without split points** if you want to regenerate the test as a single test.
6. Click **Finish**.

**Results**

The test is regenerated and opened in the test editor.

## Organizing test assets by type

By clicking an icon, you can view your test assets in a logical order, in separate folders for tests, schedules, results, locations, and datasets.

**About this task**

In the Test Navigator view, you can click **Show Java Content** to see custom code that you created, click **Show Missing Resources** to view unresolved references, and click **Show File Extenstions** to view file extensions of test assets.

1. On the Test Navigator toolbar, click the **Show the logical test navigator** ⬛ icon.
   To see the Test Navigator view, click **Windows > Show view** and click **Test Navigator**.
2. Your assets are now grouped logically. To see them, open the appropriate folder.
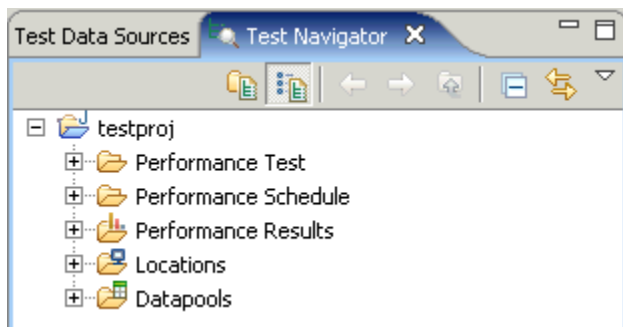


> **Note:** In the Logical view, only if the asset is available the appropriate folder is displayed. For example, if the Results folder is displayed only after you have executed a test.

# Editing tests

After you record a test, you can edit it to include datasets (to provide variable data rather than the data that you recorded), verification points (to confirm that the test runs as expected), and data correlation (to ensure that returned data is appropriate for the corresponding request). You can also add protocol-specific elements to a test. When you edit a test, the modified items appear in italic type. The italic type changes to regular type after you save the test.

## Editing service tests

After you record a service test, you can edit the calls and message returns to include variable data (rather than the data that you recorded). You can add verification points (to confirm that the test runs as expected), transactions, conditional processing, and custom code.

## Web service test editor overview

With the test editor, you can inspect or customize a test that you recorded.

The test editor lists the web service call elements for a test, in sequential order.

There are two main areas in the test editor window. The area on the left, **Test Contents**, displays the chronological sequence of test elements in the test. The area on the right, **Test Element Details**, displays details about the currently selected item (test, call, message return, or verification point) in the test hierarchy.

Window events are the primary test elements in a Citrix test and represent graphic objects that are drawn by the Citrix server, such as actual window, dialog boxes, menus, or tooltips. A Window event is recorded each time a window is created, destroyed, moved, or resized. The first occurrence of a window, a create window event, is displayed in bold. Window objects are typically identified by their title. If there is no window title, for example on menus or tooltips, then the test editor uses the window ID number.

A service request node name can be updated automatically or you can use custom code or dataset to supply different names. To apply a dataset to a node name, in the test editor, select the node name. In the Request Details area of the test editor, clear the **Update node name automatically**, select the name and substitute it with dataset.

Web service calls can contain web service message return elements, which display the results of the web service call. The XML message content can be displayed either in Form, Tree or Source view. Each of these views displays the same message content in different forms:

- **Form** view provides a simplified view of the call elements focused on editing the values of the XML message content.
- **Tree** view provides a hierarchical view of the XML structure, including elements, namespaces, and the associated values. Tree view also allows you to manipulate XML fragments.
- **Source** view displays the XML contents of a web service or XML call or the plain text contents of a simple text message.

Message return elements can contain verification point elements that check that the actual return results match expected criteria.

Some actions contain data that is highlighted. This highlighting indicates that the data can be used as a dataset candidate or as a reference. See Data correlation overview on page       for more information.

In service calls and message returns, you can use datasets and data correlation on values contained in the XML or on XML fragments. To use data correlation on XML fragments, switch to the Tree view, right-click the XML element and select **Create XML Fragment**.

To view or modify the color coding in web service tests, click **Window > Preferences > Test > Test Editor**, and then click the **Fonts and Colors** tab.

Click **Add** to add elements to the selected test element. Alternatively, you can right-click a test element and select an action from a menu.

The choices that you see depend on what you have selected. For example, inside a web service call, you can add a web service message return. The **Insert** button works similarly. Use it to insert an element before the selected element. The **Remove** button allows you to delete an item.

## Verifying application behavior

To check the expected behavior of the application during a service test, you can add verification points after a message return. During the run, verification points produce a pass, fail, error, or inconclusive status in the Web Service Verification Point report.

## Adding equal verification points

Equal verification points enable you to check that the contents returned by a service match exactly the contents specified in the verification point.

**About this task**

When you add verification points, the results from a service response are compared with the expected data specified in the verification point test element. *Equal* or *contain* verification points enable you to directly compare the XML document that the service returns.

- Contain verification points return a Pass status when the response XML document contains the expected XML data.
- Equal verification points return a Pass status when the response XML document matches exactly the expected XML data.

Complex service requests or verification points might have empty XML elements that are not needed in a test script. When playing back the test, you can skip such empty XML elements. In **Window > Preferences > Test > Test editor > Service test** ensure that the **Display the 'Skip if empty' column in XML tree viewer** check box is selected. This option displays a **Skip if empty** column in the tree view of the request. You can then choose the XML elements to skip.

1. Open the test editor, and right click a response element and select **Add > Equal Verification Point**.
2. Select the verification point, and in the **Test Element Details** area of the test editor, type a name for the verification point.
3. Select the verification options:
   - Select **Test using XML namespaces** to perform the verification on the qualified structure of the XML document, including the namespace tagging, instead of the simple name. Disable this option to check only the simple name of the element and the final return value.
   - Select **Text XML text nodes** to include the content of text elements in the verification.
   - Select **Text XML attributes** to include the content of attributes in the verification.
4. On the Message page, select the **Form**, **Tree**, or **Source** view to specify the expected XML data.

   For an equal verification point, the expected XML data contains the XML document from the response test element. If necessary, you can edit the expected XML data.

   You can specify standard Java™ regular expressions in the **Tree** view. To do this, select the **Regular expression** column on the line of an attribute or text value and type the regular expression in the **Value**

column. For example, the following regular expression checks for a correctly formatted email address: `/^([a-zA-Z0-9_\.\-])+\@(([a-zA-Z0-9\-])+\.)+([a-zA-Z0-9]{2,4})+$/`

When using regular expressions, the number of XML nodes or XML fragments in the verification point must match the quantity of expected nodes.

**What to do next**

You can enable or disable each verification point by right-clicking the verification point in the test editor and clicking **Enable** or **Disable**.

## Adding contain verification points

With contain verification points, you can check that one or several elements of the XML content returned by a service match the XML fragment that is specified in the verification point.

**About this task**

When you add verification points, the results from a service response are compared with the expected content that is specified in the verification point test element. *Equal* or *contain* verification points enable you to directly compare the XML contents that the service returns.

- Contain verification points return a Pass status when the response XML contents contain the expected XML fragment.
- Equal verification points return a Pass status when the response XML contents match exactly the entire expected XML content.

Complex service requests or verification points might have empty XML elements that are not needed in a test script. When playing back the test, you can skip such empty XML elements. In **Window > Preferences > Test > Test editor > Service test** ensure that the **Display the 'Skip if empty' column in XML tree viewer** check box is selected. This option displays a **Skip if empty** column in the tree view of the request. You can then choose the XML elements to skip.

1. Open the test editor, and select a service response element.
2. In the **Test Element Details** area, click the **Message** tab and select the **Form** or **Tree** view.
3. Expand the envelope line, right click the element that you want to check, and then click **Create Contain Verification Point**. This action creates a contain verification point that includes the XML element from the recorded response.

   > ✏️ **Note:** You can also create a contain verification point with the message response by selecting the message response in the **Test Contents** pane and clicking **Add > Contain Verification Point**. However, the result is effectively the same as an equal verification point because the verification point contains the entire XML content of the message response.

4. Select the verification point, and in the **Test Element Details** pane, type a name for the verification point.
5. Select the verification options:

- ◦ Select the **Test using XML namespaces** check box to perform the verification on the qualified structure of the XML document, including the namespace tagging, instead of the simple name. Disable this option to check only the simple name of the element and the final return value.
- ◦ Select the **Test XML text nodes** check box to include the content of text elements in the verification.
- ◦ Select the **Test XML attributes** check box to include the content of attributes in the verification.

6. If necessary, select the **Form**, **Tree**, or **Source** views to edit the expected XML fragment.

For an equal verification point, the expected XML data contains the XML document from the response test element. If necessary, you can edit the expected XML data.

You can specify standard Java™ regular expressions in the **Tree** view. Select the **Regular expression** column on the line of an attribute or text value and type the regular expression in the **Value** column. For example, the following regular expression checks for a correctly formatted email address: `/^([a-zA-Z0-9_\.\-])+\@(([a-zA-Z0-9\-])+\.)+([a-zA-Z0-9]{2,4})+$/`

When using regular expressions, the number of XML nodes or XML fragments in the verification point must match the number of expected nodes. The verification point returns a Pass status when all regular expressions in the XML fragment are matched.

**Exemple**

You can use a contain verification point to check that the message response contains only a specific element with a specific value. For example, consider the following message response:

```
<s:Envelope
 xmlns:a="http://www.w3.org/2005/08/addressing"
 xmlns:s="http://www.w3.org/2003/05/soap-envelope">
  <s:Header>
    <a:Action
     s:mustUnderstand="1">http://www.w3.org/2005/08/addressing/soap/fault</a:Action>
    <a:RelatesTo>uuid:ed9bc447-d739-452f-989d-cd48344d494a</a:RelatesTo>
  </s:Header>
  <s:Body>
    <s:Fault>
      <s:Code>
        <s:Value>s:Sender</s:Value>
        <s:Subcode>
          <s:Value
           xmlns:a="http://schemas.xmlsoap.org/ws/2005/02/sc">a:BadContextToken</s:Value>
        </s:Subcode>
      </s:Code>
      <s:Reason>
        <s:Text
         xml:lang="en-US">The message could not be processed. This is most likely because the action
&apos;http://Samples.ICalculator/Add&apos; is incorrect or because the message contains an invalid or
expired security context token or because there is a mismatch between bindings. The security context
token would be invalid if the service aborted the channel due to inactivity. To prevent the service
from aborting idle sessions prematurely increase the Receive timeout on the service endpoint&apos;s
binding.</s:Text>
      </s:Reason>
      <s:Node>http://www.w3.org/1999/xlink</s:Node>
      <s:Role>http://www.w3.org/1999/xlink</s:Role>
```

```
      <s:Detail
       xmlns:tns0="http://schemas.com/2003/10/Serialization/"
       xmlns:tns15="http://Samples.Windows"
       tns0:Id="id"
       tns0:Ref="idref">
        <tns15:GetCallerIdentityResponse>
          <tns15:GetCallerIdentityResult>str</tns15:GetCallerIdentityResult>
        </tns15:GetCallerIdentityResponse>
      </s:Detail>
    </s:Fault>
  </s:Body>
</s:Envelope>
```

To check for the `Subcode` element, the expected content of the contain verification point is the following XML fragment:

```
<s:Subcode
  xmlns:a="http://www.w3.org/2005/08/addressing"
  xmlns:s="http://www.w3.org/2003/05/soap-envelope">
   <s:Value
   xmlns:a="http://schemas.xmlsoap.org/ws/2005/02/sc">a:BadContextToken</s:Value>
</s:Subcode>
```

By default, the contain verification point checks whether an element named `Subcode` contains one element named `Value`. You can use the following options:

- **Test using XML namespaces**: With this option, the verification point checks whether an element named `"http://www.w3.org/2003/05/soap-envelope":SubCode` contains one element named `"http://www.w3.org/2003/05/soap-envelope":Value`.
- **Test XML text node**: With this option, the verification point also checks whether the element named `Value` contains the text `a:BadContextToken`.
- **Test XML attributes**: With this option, the verification point also checks that the attributes match the expected XML fragment. In this example, the **Test XML attributes** option is not necessary because the `Subcode` element does not have any attributes.

To check that the `Detail` element properly returns a specific value for `GetCallerIdentityResult`, the expected content of the contain verification point is the following XML fragment:

```
<s:Detail
 xmlns:a="http://www.w3.org/2005/08/addressing"
 xmlns:s="http://www.w3.org/2003/05/soap-envelope"
 xmlns:tns0="http://schemas.com/2003/10/Serialization/"
 xmlns:tns15="http://Samples.Windows"
 tns0:Id="regular_expression"
 tns0:Ref="idref">
  <tns15:GetCallerIdentityResponse>
    <tns15:GetCallerIdentityResult>IdentityValue</tns15:GetCallerIdentityResult>
  </tns15:GetCallerIdentityResponse>
</s:Detail>
```

You can use the following options:

- **Test XML text node**: With this option, the verification point also checks whether the element named `GetCallerIdentityResult` contains the text `IdentityValue`.
- **Test XML attributes**: With this option, the verification point also checks that the attribute `Id` referred to by `tns0:Id` has the expected value. You can specify a regular expression for this value by using the **Regular expression** column in the **Tree** view of the verification point. For example, `tns0:Id="[a-zA-Z]"` checks that the value does not contains numbers.

**What to do next**

You can enable or disable each verification point by right-clicking the verification point in the test editor and clicking **Enable** or **Disable**.

## Adding Xpath query verification points

With service query verification points, you can check that a response matches an Xpath query.

**Before you begin**

When you add verification points, the results from a service response are compared with the expected data that is specified in the verification point test element. With *query* verification points, you can check that the number of nodes returned by an XML Path language query matches the expected number of nodes specified in the verification point.

Refer to the XPath specification for details on expressing an XPath query: http://www.w3.org/TR/xpath.

You can use the test editor to create or edit verification points.

1. Open the test editor, and select a web service response element.
2. Click **Add**, and select **Query verification point**.
3. In the **Test Element Details** area of the test editor, type a name for the verification point.
4. Type a valid **XPath expression** or click **Build Expression** to open the **XPath Expression Builder**.
   The **XPath Expression Builder** helps you build and evaluate XPath expressions based on the recorded contents of the response.
5. Specify a **Comparison operator** (**=**, **>**, or **<**), and the expected number of nodes that the query should return. Click **Evaluate** to update the Expected Count with the actual result based on the recorded contents of the response.

**What to do next**

You can enable or disable each verification point by right-clicking the verification point in the test editor and clicking **Enable** or **Disable**.

**Note:** Because XPath expressions require that the qualified name have a prefix, XPath expressions will return null for the default namespace declared with *xmlns*.

## Adding attachment verification points

Service attachment verification points enable you to check that the attachment of a service response matches the specified criteria.

**Before you begin**

When you add verification points, the results from a service response are compared with the expected data that are specified in the verification point test element. *Attachment* verification points enable you to verify that an expected attachment is delivered with the response.

Attachment verification points return a Pass status when all the criteria of an attachment match the expected criteria specified in the verification point test element. If any of the criteria do not match, the verification point returns a Fail status.

You can use the test editor to create or edit verification points.

To add attachment verification points to a performance test:

1. Open the test editor and select a service response element.
2. Click **Add** and select **Attachment Verification Point**.
3. In the **Test Element Details** area of the test editor, type a name for the verification point, and specify the criteria to be verified. All criteria must match in order for the verification point to pass.
   a. In the case of multiple attachments, set the **Index of attachments** to the index number of the attachment to be checked. Type 1 if there is only one attachment in the response.
   b. Specify the expected size in bytes of the attachment.
   c. Specify the MIME type and encoding of the attachment.

**What to do next**
You can enable or disable each verification point by clicking **Enable verification point** in the test editor.

## Adding Text verification points

To check the text content that is returned by the service response, you can add a text verification point in the service test. When you add the verification point, you can check whether the text matches equally with the response or whether the response contains the text.

1. Open the Test editor, right-click a response element and select **Add > Text Verification Point**.
2. In **Verification Point Name**, specify a name for the verification point.
3. In the **Operator** field, select the basis of comparison between the text to be verified and the response content.
4. To search between the offset values, select **From Offset** and **To Offset** check boxes and specify the offset values.

5. To search between two string values, select **From String** and **To String** check boxes and specify the strings. You must also specify the number where the strings occurred.

   For example, if there are four occurrences of 'My Text' in the content and you want to verify the text that is between second and third occurrence, you should specify 2 and 3 in **From String** and **To String** respectively.

6. To do a case-sensitive match, select the **Case sensitive** check box.

7. To ignore carriage return/ line feed in the response, select the **Ignore CL/LF when matching** check box.

8. Save the test and run it.

**Results**

The Service Verification Point Report shows the number of Text Verification Points that passed or failed.

## Adding **properties verification points** to a test response

You can add verification points for the properties in a service test so that these properties in the test response are verified and validated when you play back the test.

**Before you begin**

You must have recorded or created a service test using the test editor.

**About this task**

When you add verification points, results from a service response are compared with the expected data specified as the verification point test element. You can add the verification point for the properties to an existing test response when the test is manually created or recorded. After you add the verification point for the properties to a test response, you can verify the selected response properties during the test run.

1. Identify the service test from **Test Navigator** and double-click the service test to open it in the test editor.

2. Select a service response for a service request from the service test.

3. Right-click the service response, click **Add > Properties Verification Point**.
   **Result**

   The **Properties Verification Point** is added based on the existing properties of the service response.

   📝 **Note:** You can add multiple verification points for the properties, if required.

4. Perform any of the following on the verification points for the properties in the **Properties Verification Point Details** pane.

| To Do... | Do This... |
| --- | --- |
| To add a new property and its value | Click **Add**. |
| To edit the value of an existing property | Click **Edit**. |

| To Do... | Do This... |
|---|---|
| To remove the property that you do not want to verify during the test run | Click **Remove**.<br><br>📝 **Note:** You can remove multiple properties in a group at the same time. |

5. Select or clear the **Apply And Operator** check box based on the requirement as follows:
   - To verify all the listed properties, select the **Apply And Operator** check box.
   - To verify one of the listed properties, clear the **Apply And Operator** check box.
6. Optionally, you can substitute the value of one or more properties in the verification point by using a test variable, data set, custom java code, or built-in variables.
7. Verify all the verification points for the properties that you entered, and then click **Save**.

**Results**

The verification points that you added for the response properties are added to the service test.

**What to do next**

You can run the test and after the test run, you can view and analyze the `properties verification point` details from the following page and reports:

- **Verdict List** pane in the **Test Log** page. Click any of the verification point from the list and view the details.
- **Response Properties Verification Points** tab in the **Service Verification Point Report**. Click the **Response Properties Verification Points** tab and view the verification point details.
- **Verification points verdicts** pane from the **Functional Test** report page. Click any of the verdict status to verify the expected value and actual value of the verification point for the properties.

---

Related reference

Web Service Verification Points report

## Adding XSD verification points

XSD verification points enable you to check that the XML content of a service response comply with the rules defined in an XML Schema Definition (XSD) file.

**Before you begin**

When you add verification points, the results from a service response are compared with the expected data that are specified in the verification point test element. *XSD* verification points return a Pass status when the XML contents of the response are compliant with the associated XSD or a Web Service Description Language (WSDL) file that contains XSD information.

If you add multiple XSD files to the verification, then the XML content of the response must comply with all of the XSD files.

You can use the test editor to create or edit verification points.

To add an XSD verification point to a test:

1. Open the test editor and select a service response element.
2. Click **Add** and select **XSD Verification Point**.
3. In the **Test Element Details** area of the test editor, type a name for the verification point.
4. Click **Add XSD** to add a an XSD file to the validation list or **Add WSDL** to add a WSDL that contains XSD information.
   Click **Open** to display the XSD or WSDL contents.

**What to do next**

You can enable or disable each verification point by right-clicking the verification point in the test editor and clicking **Enable** or **Disable**.

## Working with Server Name Indication (SNI) recordings

If you have recorded against a server that supports Server Name Indication (SNI), an extension of the TLS protocol, the recording session file displays `true` for the **SNI Extension** field. There might be a need for you to access both SNI and non-SNI applications from the same server. To run the same test without using the SNI extension, you can manually change the value to `false`.

**About this task**



The **Server Access Configurations** resource of the test script also have SSL entries. Each SSL entry displays which TLS version and Cipher value was used. To edit multiple SSL entries, select them and in the Detail area, right-click the entries and click **Edit multiple SSLs**.

## Adding elements to a service test

To improve the reliability of the test script, add the different elements to the service test.

## Adding a service request

You can use service request elements in tests to send a request to the service.

**About this task**

Complex service requests or verification points might have empty XML elements that are not needed in a test script. When playing back the test, you can skip such empty XML elements. In **Window > Preferences > Test > Test editor > Service test** ensure that the **Display the 'Skip if empty' column in XML tree viewer** check box is selected. This option displays a **Skip if empty** column in the tree view of the request. You can then choose the XML elements to skip.

1. Open the test in the test editor, and select the first element in the test.
2. Click **Add** and select a service request.
3. If you selected WSDL service request, select one or several WSDL files in your workspace for the web service that you want to test and click **Next**.
   If necessary, you can import a WSDL file into the workspace with the **Add** button.
4. Select either **HTTP**, **JMS**, or **WebSphere MQ** depending on the transport protocol used by the web service, and provide the correct transport protocol configuration to perform the call.
   You can create a **New** transport configuration or reuse an existing one.
5. Click **Finish**.
   **Result**
   This creates the web service request in the test editor.
6. On the **Message** page of the request, select the **Form**, **Tree**, or **Source** views to edit the service request contents.
7. If any resource files are to be attached to the request, select the **Attachment** tab. Use **Add**, **Remove**, or **Edit** to specify the resources that are to be attached to the request.
8. If the service uses encryption, signature or other security protocols, select the **Security for Request** and **Security for Response** pages to configure the security for this particular service request or to open the WSDL security editor.

**What to do next**

After creating elements, you can use the test editor to edit service requests. You can create a service response element to test the performance and behavior of the service. You can also replace some content values with dataset variables or a references.

## Updating a service response from the service

While you are developing a service test, you can send a request from the test editor to record or update the response element.

**Before you begin**

Service response elements are children of service request elements. Service tests use response elements to measure the response time between a call and the corresponding response. Response elements can also contain verification points.

You can click **Update Response** in the request element to complete one of the following actions:

- Record a response from the service: This method sends the request and records the actual response from the service. For services that use the IBM® WebSphere® MQ or JMS transport protocols, multiple responses can be recorded.
- Update the current response content: If a response exists, its contents are replaced. If multiple responses are received, the number and order of the responses are updated.

> ⚠️ **Important:** After updating the response content, data correlation or verification points that referred to replaced content might no longer work.

You can use the test editor to create or edit response elements in a service test. There are three methods of adding a service response:

- Generate a response from Web Services Description Language (WSDL): If the service uses WSDL, then the response is created with the content structure that the WSDL specifies.
- Add a text response: In this response type, you specify free formatted content for the response.
- Record a response from the service: This method sends the request and records the actual response from the service.

WebSphere® MQ and JMS requests can contain multiple response elements.

To add a response element to a service test:

1. Open the test in the test editor, and select a service request element.
2. On the **Test Element Details** page, click **Update Response**.
   Alternatively, right-click the service request element, and click **Add > Response from Request**.
   **Result**
   This action performs the service request. If the request is valid, the **Update Response** window opens and displays the response data.
3. In the **Return Preview** window, review the content of the response to ensure that it is correct.
   For the WebSphere® MQ and JMS protocols, if multiple responses are received, then click the arrows to view each response.
   a. Click the **Message** tab to view the contents of the response in the **Form**, **Tree** or **Source** view.
   b. Click the **Attachment** tab to view any resource files that were attached to the response.
   c. Click the **Response Properties** tab to view the properties of the response.
4. To use the received response in the test, click **Update Test**.
   This creates the response elements as a child of the request element or updates the existing response elements with the new data.

**What to do next**

After creating or updating response elements, you can create verification points on the response contents to test the behavior of the service.

---

Related information

## Manually adding a response element

You can add service response elements to specify the received content of a service request. You can use the test editor to create or edit response elements in an existing service test.

**Before you begin**

Service response elements are children of service request elements. Service tests use response elements to measure the response time between a call and the corresponding response. Response elements can also contain verification points. IBM® WebSphere® MQ and JMS requests can contain multiple response elements.

Depending on the type of request, you can manually create several types of response elements:

- Response from Web Services Description Language (WSDL): For web services, this response type uses the WSDL file to create the specified XML structure of the response.
- XML response: This response type creates an empty response element in which you must manually create the expected XML structure. You can use an XML Schema Definition (XSD) document from the XSD catalog to assist you.
- Text response: This response type creates an empty response element, which can contain freely formatted text.

Alternatively, you can automatically create and update response content by recording the actual response content that the service returns. See for more information.

To add a response element to a service test:

1. Open the test in the test editor, and select a service request element.
2. Create one of these elements:
   **Choose from:**
     ◦ For web service requests, click **Add > Response from WSDL**.
     ◦ If the expected response contains XML content, click **Add > XML Response**.
     ◦ If the expected response contains plain text, click **Add > Text Response**.
   **Result**
   This action creates the corresponding response element in the test. If the request uses the WebSphere® MQ or JMS format, then you can create multiple responses.

3. Edit the message content of the response element to reflect to actual content that the service returns.

    a. Click the **Message** tab to view the contents of the response in the **Form**, **Tree** or **Source** view.

    b. Click the **Attachment** tab to view any resource files that were attached to the response.

    c. Click the **Response Properties** tab to view the properties of the response.

**What to do next**

After creating a message return, you can create verification points on the contents to test the behavior of the service.

Related information

## Managing JMS/MQ connections in a service test

When you run a service test that includes JMS or MQ protocol, the socket connections are created and closed in the background. When you include multiple tests in a compound test or a A schedule, in this context, is used to refer to both VU Schedule and Rate Schedule., multiple connections are created and closed. Starting from 9.2, when you run service tests in a schedule, you can select a pooling strategy for these JMS/MQ connections so that when the connections are created, they do not close and are reused subsequently for the other JMS/MQ calls, if required.

**About this task**

You can set the scope of JMS/MQ connections to a test, compound test, or schedule. When you set the scope to test, existing behavior comes into play wherein duplicate connections could be created and closed. When you set the scope to a compound test or a schedule, the connections are reused for JMS/MQ calls within a compound test or a schedule.

1. In the Test Navigator, browse to the schedule and double-click it. The schedule is displayed.
2. Select a schedule. In the VU Schedule Details area, click the **Advanced** tab and under **Protocol-specific options**, click **Edit Options**.
3. In **JMS/MQ connections scope**, select the scope of the connections.
4. Use the following options to control the underlying MQ Connection Manager to create only the specified number of connections. These options are generally used by the MQ expert:

    a. In **Maximum quantity of connections**, specify a number to ensure that a certain number of connections are open at a time only for MQ Java.

    b. In **Maximum quantity of unused connections**, specify the maximum number of connections that should be unused among the open connections.

    c. In **Connection timeout (ms)**, specify a time after which there is no attempt to establish the connection.

**Results**

When you run a schedule, the JMS/MQ connections are reused.

## Editing WSDL security profiles

To ensure that your service test uses the correct security protocols to access a SOAP-based service, you must specify a security profile for the (Web Service Description Language) WSDL file. After a security profile is set up, it can be reused in multiple web service calls.

## WSDL security editor overview

With the WSDL security editor you can create the SOAP algorithm stacks that are associated with a web service operation. Algorithm stacks contain digital certificate information and the security algorithms that are applied to messages to perform secure communication with a web service.

After you create an algorithm stack, you associate it with an operation that is specified in the Web Services Description Language (WSDL) file of the web service. Algorithm stacks remain available in the workspace and you can reuse them with other WSDL files. You can also edit a test to make the same web service call several times with different security configurations.

You use the **WSDL security editor** to create and edit security configurations. The WSDL security editor contains two pages that correspond to the steps of setting up a security configuration:

- Describing a security stack
- Associating a security stack with each WSDL operation

**Algorithm stacks**

Algorithm stacks contain one or several algorithm blocks that are arranged in a sequence of steps. Each algorithm block modifies or transforms the message content. Algorithm blocks can add timestamps to, add tokens to, encrypt, or sign messages.

Use the **Algorithm Stacks** page of the WSDL security editor to create stacks for service requests and responses. When a message is sent or received, each algorithm block in the stack is executed in the specified order. For example, you can define a request stack for outgoing requests that adds a timestamp, signs, and then encrypts the message content, and you can define a response stack that decrypts incoming responses. You can create as many algorithms as your application requires.

You can edit algorithm blocks and move them up and down in the stack. Encryption and signature blocks can use keystores for digital certificates. Some algorithm blocks display messages that help you enter correct information. If the contents of the algorithm block are invalid, an error icon is displayed.

**Raw transaction data view**

When a stack is associated with a service request or response, viewing the results of each transformation step that is applied to the XML message content can be useful. You can use the **Raw Transaction Data** view to look at the message content before and after each algorithm in the stack.

**Digital certificate keystores**

You can add digital certificate keystores to a security stack to use with encryption or signature algorithms. Keystores must be declared with their associated passwords before the algorithms that use them. Digital certificates are contained in Java™ keystore files (KS, JKS, JCEKS, PKCS12, and PEM) that must be located in your workspace.

**Associating stacks with WSDL operations**

Use the **Algorithms by WSDL operations** page of the WSDL security editor to associate a security algorithm stack with each web service call and message return in the WSDL file.

## Creating security profiles for WSDL files

You can create SOAP security profiles for the web service calls or message returns that require message encryption, signature or other advanced security algorithms.

**Before you begin**

You must have a Web Services Description Language (WSDL) file in your workspace.

If the security profile uses digital certificates for encrypting or signing requests or responses, you must have the corresponding keystore files (KS, JKS, JKECS, PKCS12, or PEM) in your workspace.

**About this task**

If the WSDL is simple and you want to check its security, in the **Request Stack** tab of the test editor, click **Override Stack > Tools > Analyze Security from Pasted Content**. Paste the SOAP XML message and click **Next**. The next page shows the different security algorithms used in the XML. Click **Finish** to add the security algorithms to the editor.

> **Note:** When you add a secured SOAP XML message in **Message > Source** tab of the test editor, certain security related warnings are displayed in the **Error Message** view. If you are aware of the secured SOAP XML message and do not want to view the warnings, click **Window > Preferences > Generic Service Client > Message Edition** and select the **Analyze pasted SOAP content** check box.

If the WSDL uses WS-Policy, you must configure security as follows:

1. In the test navigator or project explorer, right-click the WSDL file and select **Edit WSDL Security**.
   **Result**
   The WSDL security editor is displayed.
2. Click the **Security Algorithms** tab.
   Security profiles are described by adding elements to a stack. When a service request is sent or a response is received, each element in the stack is applied to the message in a specified order. If necessary, create one security profile for outgoing requests and one for incoming responses.
3. In the **Security Algorithms** area, click **Add** to create a new algorithm stack, and click **Rename** to change the default name.
4. In the **Algorithm Stack Details** area, click **Add** to add a new algorithm element to the stack.

You can add time stamps, username tokens, encryption, or signatures.

5. Edit each element in the stack according to the requirements of the web service.

   You can apply encryption and signature stack elements to portions of the web service call or message return document by specifying an Xpath query in **User Xpath part selection**. For example, you can encrypt one XML element with one encryption stack element, and another element with another stack element. You can use the **Web Service Protocol Data** view to help identify the correct Xpath query for this option.

   You can check whether the security stack is valid by clicking **Tools > Validate Selected Algorithm**.

6. When all the stack elements are complete, ensure that the execution order is correct.

   If necessary, use the **Up** and **Down** buttons to change the order of elements in the stack.

7. Repeat steps 4 through 7 to create as many algorithms as are required for security profile.

8. Click the **Algorithms by WSDL Operations** tab.

   This page enables you to associate a security profile with each request or response operation in the WSDL.

9. In the **WSDL Contents** column, select a service request or response.

10. In the **Algorithm Stack** column, select a security profile from the list.

    If necessary, click **<<** to open the stack on the Security Algorithms page.

**Results**

After saving the security profile, the **Web Service Protocol Data** view displays the effect of the security profile on the XML data of the web service.

---

Related reference

Related information

## Using a security policy

The WS-Policy specification enables web services to use XML to publish their security policies either as part of the Web Services Description Language (WSDL) file (compliant with the WS-PolicyAttachment specification) or as a separate XML document. With the WSDL Security Editor, you can create a security profile that uses a policy that complies with the WS-Policy specification.

**Before you begin**

Before creating a security configuration, you must have a WSDL file in your workspace.

If the security policy uses digital certificates for encrypting or signing requests or responses, you must have the corresponding keystore files (KS, JKS, JKECS, PKCS12, or PEM) in your workspace.

When you import a WSDL that contains a policy (with WS-PolicyAttachment), a security profile is automatically generated for each operation in the WSDL security editor.

1. In the test navigator or project explorer, right-click the WSDL file, and select **Configure WSDL Security**.
   **Result**
   This opens the WSDL security editor.
2. Click the **Security Algorithms** tab.
   Security profiles are described by adding elements to a stack. When a service request is sent or a response is received, each element in the stack is applied to the message in the specified order.
3. In the **Security Algorithms** area, click **Add** to create a profile, and click **Rename** to change the default name.
4. In the **Algorithm Stack Details** area, click **Add > WS-Policy** to add the WS-Policy element to the stack.
   You can also add time stamps, user-name tokens, encryption, or signatures.
5. If the policy is included in the WSDL file, click **Use policy included in WSDL (WS-PolicyAttachment)**, and edit the WS-Policy settings as required:

   **Policy**

   If you are not using the WS-PolicyAttachment specification, specify the XML policy file. Click **Browse** to add a policy file from the workspace or to import a policy file.

   **Signature configuration**

   Select this option to specify a keystore for any signature that is specified in the policy. Click **Edit Security** to add a keystore from the workspace or to import a keystore.

   **Encryption configuration**

   Select this option to specify a keystore for any encryption that is specified in the policy. Click **Edit Security** to add a keystore from the workspace or to import a keystore.

   **Decryption configuration**

   Select this option to specify a keystore for any decryption that is specified in the policy. Click **Edit Security** to add a keystore from the workspace or to import a keystore.

   **Retrieve token from security token server (WS-Trust and WS-SecureConversation)**

   Select this option, and click **Configure** to specify a Security Token Server (STS) to use with the policy.

   **Additional properties**

   Use this table to specify settings for the advanced properties or specific implementations of the WS-Security specification. Click **Add** to add a property name and to set a value.

6. Check that the security profile is valid by clicking **Tools > Validate Selected Algorithm**.
7. Click the **Algorithms by WSDL Operations** tab.
   On this page, you can associate a security profile with each request or response operation in the WSDL.
8. In the **WSDL Contents** column, select a web service request or response operation.
9. In the **Algorithm Stack** column, select a security profile from the list.
   If necessary, click **<<** to open the stack on the **Security Algorithms** page.

**What to do next**

After saving the security profile, the **Web Service Protocol Data** view displays the result of the security profile on the XML data of the web service.

Related information

## Adding security stacks

To provide better WSDL security, you can make use of many security algorithms in the service test.

**About this task**

1. From the Test Navigator view or from the Request Library section of Generic Service Client, right-click the WSDL file and select **Edit WSDL Security**.
2. In the **Security Algorithms** area of **Algorithm Stacks** tab, click **Add** to create a profile.
3. In the **Stack Contents** area, click **Add** and add any of the following security algorithms:

   **Custom Security Algorithm**

   If you want to use a Java™ class as a custom security algorithm, then use this stack element to apply the custom algorithm to the service.

   **Java™ Project**

   If you have not implemented a custom Java™ class, select **Java Project**, type a name for the new project, and click **Generate** to create a new Java™ class with the default structure for custom security implementations.

   **Note:** If you are using IBM® Security AppScan®, this field is not available.

   **Implementation class**

   Specify the name of the class that implements the custom security algorithm. Click **Browse Class** to select an existing Java™ class from the workspace.

   **Properties**

   Use this table to send any specific properties and associated values to the custom security algorithm.

   **WS-Addressing Algorithm**

   Use this block if your service uses either WS-Addressing 2004/08 or the WS-Addressing 1.0 Core standard.

**Namespace**

Specify the namespace for either WS-Addressing 2004/08 or WS-Addressing 1.0 Core.

**Action if request uses WS-Addressing**

Select the action to complete if WS-Addressing is already in the request.

**Replace anonymous address in Reply-to with:**

Select this option to generate the specified address in the Reply-to header instead of an anonymous address.

**Remove WS-Addressing from response**

Select this option to strip any WS-Addressing headers from the response.

## Encrypted Key

This block defines an encrypted key that can be used in an XML signature or XML encryption block. The encrypted key block must be before a block that uses the encrypted key.

**Actor / Role name**

Specify the name of the recipient of the algorithm header element, if required.

**Must understand**

Select whether it is mandatory that the algorithm header is processed by the recipient, if required. The recipient is either the Actor name or the server.

**Key name**

Specify the name of the encrypted key.

**Identifier type**

Select the type of key identifier to be used for the key. The following key identifiers are available, as defined in the the Web Service Security (WSS) specification X509 profile and OASIS WSS 1.1 specification:

- ISSUER_SERIAL
- BST_DIRECT_REFERENCE
- X509_KEY_IDENTIFIER
- THUMBPRINT_IDENTIFIER
- SKI_KEY_IDENTIFIER

**Key size**

Specify the size of the key in bits.

**Key encoding algorithm name**

Specify the algorithm to be used for encoding the key.

**Keystore**

Select a keystore or click **Edit Security** to define a new keystore or to manage the existing keystores.

**Name**

Select a key contained in the specified keystore.

**Password**

Type the password for the selected key name.

**XML Signature**

The XML signature security algorithm specifies how the XML document is signed. For details on security algorithms, refer to the web service security specification.

**Actor / Role name**

Specify the name of the recipient of the algorithm header element, if required.

**Must understand**

Select whether it is mandatory that the algorithm header is processed by the recipient, if required. The recipient is either the Actor name or the server.

**Security token**

Select the type of key identifier to be used for the signature. The following key identifiers are available, as defined in the the Web Service Security (WSS) specification X509 profile and OASIS WSS 1.1 specification:

- ISSUER_SERIAL
- BST_DIRECT_REFERENCE
- X509_KEY_IDENTIFIER
- SKI_KEY_IDENTIFIER
- KEY_VALUE
- USER_NAME_TOKEN
- CUSTOM_SYMM_SIGNATURE

In addition, the following identifiers are available when the signature is based on a UsernameToken profile:

- USER_NAME_TOKEN
- CUSTOM_SYMM_SIGNATURE

**User XPath part selection**

Specify an XPath query that describes parts of the XML document that can be the subjects of the algorithm. By default, the body is the subject. Click the **XPath Helper** button to build the Xpath expression.

**Key**

Select the key used for the encryption. The details of each key vary.

◦ **x509 key**: This key specifies the name and password of the x509 key and the keystore where it is located.

◦ **User name token key**: This specifies a user name and password for the signature.

◦ **Encrypted key**: This specifies a reference to an encrypted key that was previously defined in the security stack. Click **Insert a new encrypted key** to create a new encrypted key definition block.

**Signature algorithm name**

Specify the signature method algorithm as described in the XML Signature Syntax and Processing specification.

**Canonicalization**

Specify the canonicalization method to be used as described in the XML Signature Syntax and Processing specification.

**Digest algorithm method**

Specify which digest method to be used based on the algorithm method used on the server side.

**Inclusive namespaces**

Specify whether the canonicalization is exclusive as described in the Exclusive XML Canonicalization specification.

**XML Encryption**

The XML encryption security algorithm specifies how the XML document is encrypted. For details on security algorithms, refer to the web service security specification.

**Actor / Role name**

Specify the name of the recipient of the algorithm header element, if required.

**Must understand**

Select whether it is mandatory that the algorithm header is processed by the recipient, if required. The recipient is either the Actor name or the server.

**Identifier type**

Select the type of key identifier to be used for the encryption. The following key identifiers are available, as defined in the Web Services Security (WSS) specification X509 profile and the OASIS WSS 1.1 specification:

◦ ISSUER_SERIAL

◦ BST_DIRECT_REFERENCE

◦ X509_KEY_IDENTIFIER

- SKI_KEY_IDENTIFIER
- EMBEDDED_KEYNAME
- THUMBPRINT_IDENTIFIER
- ENCRYPTED_KEY_SHA1_IDENTIFIER

**User XPath part selection**

This enables you to specify an XPath query that describes parts of the XML document that can be subjects of the algorithm. By default, the body is the subject.

**Key**

Select the key used for the encryption. The details of each key vary.

- **x509 key**: This specifies the name and password of the x509 key and the keystore where it is located.
- **Raw key**: This specifies the name and the byte value of your SecretKey in hexadecimal.
- **Encrypted key**: This specifies a reference to an encrypted key that was previously defined in the security stack. Click **Insert a new encrypted key** to create a new encrypted key definition block.

**Encoding Algorithm Name**

Specify the encryption method to be used as defined in the XML Encryption Syntax and Processing specification.

**Key Encoding Algorithm**

Specify the standard algorithm for encoding the key as defined in the XML Encryption Syntax and Processing specification.

**User name token**

The user name token security algorithm adds a user name token to the XML document in the message. For details on security algorithms, refer to the web service security specification.

**Actor / Role name**

Specify the name of the recipient of the algorithm header element, if required.

**Must understand**

Select whether it is mandatory that the algorithm header is processed by the recipient, if required. The recipient is either the Actor name or the server.

**Name**

Type the name of the user.

**Password**

Type the password of the user.

**Password type**

Specify the password type for the security algorithm as defined in the Web Services Security UsernameToken profile.

**Use nonce**

Select this check box to add the Nonce element to the User Name Token XML code. In most cases, the Nonce ID is required.

**Use created**

Select this check box to add current timestamp to the Created XML element in the User Name Token XML.

**Time Stamp**

The time stamp security algorithm adds time stamp information to the XML document in the response. For details on security algorithms, refer to the web service security specification.

**Actor / Role name**

Specify the name of the recipient of the algorithm header element, if required.

**Must understand**

Select whether it is mandatory that the algorithm header is processed by the recipient, if required. The recipient is either the Actor name or the server.

**Expiration delay**

Specify the delay after which the time stamp expires.

**Millisecond precision**

Select this option to produce a time stamp that uses millisecond precision instead of the default (1/100th second).

**SAML Assertion Block**

To use the self-signed SAML assertion security algorithm, add the SAML Assertion stack to the request or WSDL files.

**User XPath part selection**

Specify an XPath query that describes parts of the XML document that can be the subjects of the algorithm. By default, the body is the subject. Click the **XPath Helper** button to build the Xpath expression.

**Key**

Select the key used for the encryption. The details of each key vary.

- **x509 key**: This key specifies the name and password of the x509 key and the keystore where it is located.
- **User name token key**: This specifies a user name and password for the signature.
- **Encrypted key**: This specifies a reference to an encrypted key that was previously defined in the security stack. Click **Insert a new encrypted key** to create a new encrypted key definition block.

**Signature algorithm name**

Specify the signature method algorithm as described in the XML Signature Syntax and Processing specification.

**Canonicalization**

Specify the canonicalization method to be used as described in the XML Signature Syntax and Processing specification.

**Digest algorithm method**

Specify which digest method to be used based on the algorithm method used on the server side.

**Inclusive namespaces**

Specify whether the canonicalization is exclusive as described in the Exclusive XML Canonicalization specification.

**Signed Assertion**

Select this check box to self-sign the SAML Assertion.

**Issuer**

Specify the description of the issuer of the SAML Assertion or protocol message.

**Subject**

Specify the principal that is the subject of all of the statements in the assertion. It might contain an identifier or a series of one or more subject confirmations.

**Subject Qualifier**

Specify the Name Qualifier of the Subject

**Subject Format**

Specify the format used for the Subject.

**Subject Locality DNS**

Specify the DNS domain name for the system from which the assertion subject was authenticated.

**Subject Locality IP**

Specify the IP address for the system from which the assertion subject was authenticated.

**Statement Type**

Specify the authentication method to use for the assertion.

**Authentication**: The assertion subject was authenticated

**Attribute**: The specified subject is associated with the supplied attributes.

**Authorization decision**: Permission to allow a subject to access the specified resource.

**Requested Resource**

When **Authorization decision** option is used, specify the resource for which you need access.

**Action**

Specify what action to take to access the resource.

**Confirmation number**

Confirmation methods define the mechanism by which an entity provides evidence (proof) of the relationship between the subject and the claims of the SAML assertions.

**Sender vouches**: Select this option when a server needs to share the client identity with SOAP messages on behalf of the client. This method is similar to identity assertion, but it has the added flexibility of using SAML assertions to share not only the client identity, but also client attributes.

**Holder of key**: Select this option when the proof of the relationship between the subject and claims is established by signing part of the SOAP message with the key specified in the SAML assertion. Because there is key material associated with a holder-of-key token, this token can be used to provide a message-level protection (signing and encryption) of the SOAP message.

**Bearers**: Select this option when the proof of the relationship between the subject and claims is implicit. No specific steps are taken to establish the relationship. Because there is no key material associated with a bearer token, protection of the SOAP message, if required, must be performed using a transport-level mechanism or another security token, such as an X.509 or Kerberos token, for message level protection.

**Version**

Specify the SAML version used.

4. **Optional:**  To verify simple SAML code, use the **Analyze Security from Pasted Content** option.
For more information about that option, see Creating security for WSDL profiles on page 174.

## Implementing a custom security algorithm

You can define your own security algorithms for SOAP security profiles by implementing custom security Java™ interfaces that can be used in the WSDL security editor. With custom security algorithms, you can implement proprietary security algorithms that transform the XML before sending and after receiving message content.

**Before you begin**

The custom security interface and the JAR file that contains it are provided with the product in the `customsecuritydefinition` folder of the `com.ibm.rational.ttt.common.models.core` plugin. You need these interfaces to create your own algorithms. If you are using IBM® Rational® Performance Tester or IBM® Rational® Service Tester for SOA Quality, see Extending test execution with custom code on page        for more information about extending test capabilities with Java™ code.

1. In the test navigator or project explorer, create a new Java™ class in your web service test project folder.
2. Implement a security algorithm in Java™ using the following interface:

```
/**
 * ******************************************************************
 * IBM Confidential
 *
 * (c) Copyright IBM Corporation. 2008. All Rights Reserved.
 *
 * The source code for this program is not published or otherwise
 * divested of its trade secrets, irrespective of what has been
 * deposited with the U.S. Copyright Office.
 * ******************************************************************
 *
 */

package com.ibm.rational.test.lt.models.wscore.datamodel.security.xmlsec;

import java.util.Properties;
import org.w3c.dom.Document;


public interface ICustomSecurityAlgorithm {

 /**
  * The following methods can be used in both case:
  * Execution in the workbench and execution of the test.
  */


 /**
  * Called to process de Document that is sent over a transport.
  * @param subject
  */
 void process(Document subject);
 /**
  * Called to un process a document that is received from a server.
  * @param subject
  */
 void unProcess(Document subject);

 /**
  * Properties defined in the UI of the CustomSecurityAlgorithm.
  * @param map
  */
 void setProperties(Properties map);
```

```
/**
 * The following methods can only be used in terms of cast to test service interface,
 * or in terms of access to the previous XML  information, when the jar containing
 * the custom security algorithm is deployed in the performance test project. In
 * this case you cannot use the algorimth directly from the workbench.
 */

/**
 * This object corresponds to the ITestExecutionService object.
 * This applies only to an algorithm that must link to the execution of the test.
 * If you plan to use this object you will need to deploy the jar containing the
 * implementation into your performance test project and not directly into the JRE.
 *
 * In case of a need of the previous xml document received from the execution you can
 * obtain the value using:
 * IDataArea area =
((ITestExecutionService)executionObject).findDataArea(IDataArea.VIRTUALUSER);
 *String previousXML = (String) area.get("PREVIOUS_XML"); //$NON-NLS-1$
 *
 */
void setExecutionContext(Object executionObject);
```

The `process` method modifies the XML before it is sent to the server.

The `unprocess` method modifies the XML after it is received from the server.

The `setProperties` method retrieves any properties that are defined in the security editor for this custom security interface.

The `setExecutionContext` method is called during test with the object `ITestExecutionServices` that corresponds to the message using this custom security interface.

3. The custom security interface can be used either in the **WSDL security editor** for web services or in XML call elements in the **Local XML security** tab.
   **Choose from:**
   - If you are testing a WSDL-based web service, right-click the WSDL file in the test navigator or project explorer to open the WSDL security editor, select the **Security Algorithms** page; then, under **Details of selected security algorithm stack**, click **Add > Custom Security Algorithm**.
   - If you are testing an XML call, open the XML call element in the test editor, select the **Local XML Security** tab, and then, click **Add > Custom Security Algorithm**
4. In custom security, click **Browse Class** to select the class name of the custom security algorithm, for example : `ICustomSecurityAlgorithm`.
5. Type an **Algorithm name** for the custom security algorithm.
6. In the properties list, use **Add**, **Remove**, or **Edit** to specify any properties that are used by the `setProperties` method in your custom security algorithm.

**What to do next**
After saving the security configuration or the call element, the **Web Service Protocol Data** view displays the effect of the security algorithm on the XML data of the web service.

Related reference

## Adding WS-Addressing to a security configuration

The WS-Addressing specification provides transport-neutral mechanisms that enable SOAP-based web services to communicate addressing information. You can use WSDL security algorithms to add WS-Addressing to your service tests.

**Before you begin**

Before adding WS-Addressing to a security configuration, you must have a service test with requests and responses that are related to a valid WSDL.

To add WS-Addressing to a WSDL security algorithm:

1. Open the test, select a service request, and in the **Raw Transaction Data** view, select **Enable the display of the XML document after the security processing**.
2. On the **Request Stack** page, click **Edit WSDL Security** .

   > **Tip:** If you need to edit separate security or processing algorithms for incoming responses, click **Show Response Stack** to add a **Response Stack** page to the editor.

   **Result**

   The **WSDL security editor** opens.
3. Select the **Algorithm Stacks** page of the WSDL security editor, and in the **Security Algorithm** list, select or create a security algorithm.
4. In the **Stack Contents** list, click **Add > WS-Addressing** and specify the settings that are implemented by the service.

   **WS-Addressing Algorithm**

   Use this block if your service uses either WS-Addressing 2004/08 or the WS-Addressing 1.0 Core standard.

   **Namespace**

   Specify the namespace for either WS-Addressing 2004/08 or WS-Addressing 1.0 Core.

   **Action if request uses WS-Addressing**

   Select the action to complete if WS-Addressing is already in the request.

   **Replace anonymous address in Reply-to with:**

   Select this option to generate the specified address in the Reply-to header instead of an anonymous address.

**Remove WS-Addressing from response**

Select this option to strip any WS-Addressing headers from the response.

5. Save the WSDL security algorithm, and select the test editor.

   **Result**

   The WS-Addressing namespace and header XML content is displayed in the **Raw Transaction Data** view.

---

Related reference

Related information

## Testing asynchronous services

Use the asynchronous callback services for inter-object communications in a service test.

## Asynchronous service testing overview

Asynchronous services use a callback interaction pattern for inter-object communications. Asynchronous services can be used, for example, in publish-subscribe systems that are provided by message-oriented middleware vendors or in system and device management domains.

**WS-Notification services**

Asynchronous services are standardized in the *WS-Notification* specifications:

- *WS-BaseNotification* defines the web services interfaces for *NotificationProducers* and *NotificationConsumers*. This specification includes standard message exchanges that are implemented by service providers that want to act in these roles, along with the associated operational requirements.
- *WS-BrokeredNotification* defines the web services interface for a *NotificationBroker*. A NotificationBroker is an intermediary which, among other things, enables entities that are not service providers themselves to publish messages. It includes standard message exchanges that are implemented by NotificationBroker service providers, along with the associated operational requirements of service providers and requestors that participate in brokered notifications.
- *WS-Topics* defines a mechanism to organize and categorize items of interest for subscription known as *topics*. These are used in conjunction with the notification mechanisms defined in WS-BaseNotification and WS-BrokeredNotification.

You can test web services and XML services that implement the WS-Notification specification by creating an asynchronous request inside a test. The asynchronous request contains the interfaces for the corresponding WS-Notification specification, along with a callback structure.

**Proprietary asynchronous services**

You can test proprietary asynchronous services that do not implement WS-Notification specifications. To test these services, you manually create a service request that contains the interfaces for the service, and then, you can add the asynchronous callback structure to the request.

The XML data of the asynchronous request must contain an endpoint that specifies the URL of the callback receiver. During the test, this endpoint is used to redirect the callback to the tester instead of the real receiver.

**Callback structure**

To test asynchronous services, you must create an asynchronous request structure in your test as shown in the following diagram:

```
□  MySOA Test
   ⊞   getLastNameRequest( "Nicholas" )
   □   Subscribe
         SubscribeResponse
      □   Callback
         □   Parallel
               // Elements here run while waiting for callback
            □   Loop (infinite )
                  Anonymous Call( ? )
         □   Receive
               // Elements here run after callback is received
         □   Timeout
               // Elements here run after timeout occurs
```
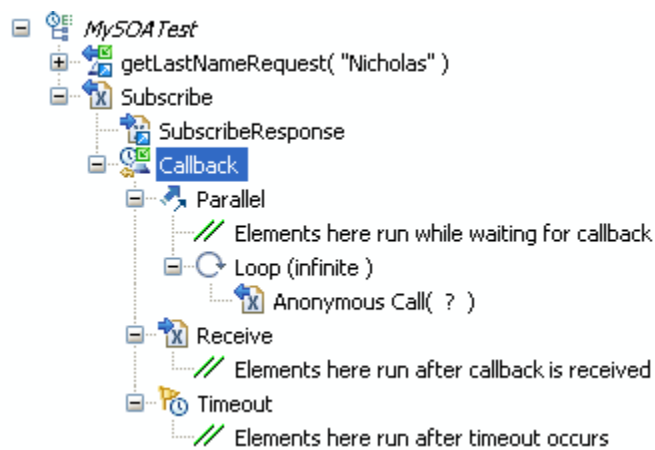
A web service request or a plain XML request provides the subscription action and contains a callback element, which describes the behavior of the test in three states:

- *Parallel* contains test elements that are run after the subscription request and while waiting for the notification response.
- *Receive* contains test elements that are run when the notification response has been received from the service.
- *Timeout* contains test elements that are run if the notification response is not received after a delay that is specified in the callback element.

When everything contained in the parallel, receive, and timeout elements have finished running, the run continues with the next element in the test after the asynchronous request.

The method for generating the asynchronous callback structure in the test depends on whether the asynchronous service uses the WS-Notification specification:

- WS-Notification services: Create the asynchronous request in the test.
- Proprietary services: Manually create a web service request or XML request in the test, and then add the asynchronous callback structure to the request.

## Creating an asynchronous request structure

You can create an asynchronous request based on the *WS-Notification* specification, which contains an callback structure.

1. In the test editor, select the test, and click **Add**, and then click **Specification-based Structure**.
   **Result**
   The **New Web Service Test** wizard opens.
2. On the **Web Services Specification Selection** page, Select **WS-Notification**, and click **Next**.
3. On the **WS-Notification Details** page, if the service has a Web Services Description Language (WSDL) file, click **Add** to associate it with the call.
4. Specify the **Subscription identifier**.
   You can select default identifiers for Websphere Application Server or Apache Muse; or if your service does not use a standard identifier, you can select **Custom**, and type the **Name** and **Namespace** of the identifier.
5. In the **Topic** area, replace the default **Name** and **Namespace** values with those of topic of your service.
6. Specify the **Subscription duration**.
   Because this is a test environment, the subscription expires after the specified delay to save server resources.
7. If this is a WS-BrokeredNotification service, which implements a notify call when the subscription is received, you can select **Add notify call**, and type the message to be sent.
8. Click **Next**.
9. On the **Configure Protocol** page, select a **Protocol configuration**, and specify the options of the configuration. Select **Generate SOAP 1.2 envelope** if you are testing a SOAP 1.2 web service.
10. Click **Finish**.
    **Result**
    This action generates in the test editor a web service call or an XML request with a callback structure that contains a parallel, a receive, and a timeout element.

**What to do next**

In the callback structure, add test elements to the parallel, receive, and timeout elements to specify the behavior of the test:

- *Parallel* contains test elements that are run after the asynchronous call has been sent.
- *Receive* specifies the message return of the callback and contains test elements that are run after the callback is received.
- *Timeout* contains test elements that are run if the callback is not received after a specified delay.

## Adding an asynchronous callback to a service request

To test a proprietary asynchronous service that does not implement the *WS-Notification* specification, you can add an asynchronous callback to a service request or XML request.

**Before you begin**

Manually create a web service call or XML call that invokes the asynchronous service. The call must contain an endpoint that specifies the URL of the callback receiver. This endpoint is used to redirect the callback to the tester.

If the service implements the WS-Notification specification, create the asynchronous call structure with the **Create New WS-Notification Request and Callback** wizard instead. See Creating an asynchronous request structure on page 190.

1. In the test editor, select a web service or XML request, click **Add**, and then click **Asynchronous Callback**.
   **Result**
   The **Create New Asynchronous Callback** wizard opens.
2. On the **Select Callback Endpoint** page, select the XML element of the request where the endpoint URL of the callback is located.
3. If you have a web Services Description Language (WSDL) file for the web service, click **Next**. Otherwise, skip to step 5.
4. On the **Bind Message to WSDL Port** page, select a port from the WSDL file. If the WSDL file for the service is not listed, click **Add** to add a WSDL file from the workspace or to import a WSDL file.
5. Click **Finish**.
   **Result**
   This generates a callback structure that contains a parallel, a receive, and a timeout element, in the test editor.

**What to do next**

In the callback structure, you can add test elements to the parallel, receive, and timeout elements to specify the behavior of the test:

- *Parallel* contains test elements that are run after the asynchronous request has been sent.
- *Receive* specifies the message return of the callback and contains test elements that are run after the callback is received.
- *Timeout* contains test elements that are run if the callback is not received after a specified delay.

## Creating a reliable messaging call structure

You can create a test structure dedicated to testing service calls based on the *WS-ReliableMessaging* specification.

**Before you begin**
The WS-ReliableMessaging specification provides for a series of SOAP messages to be delivered reliably between distributed applications in the presence of software component, system, or network failures. In the context of a service test, a reliable messaging call structure consists of a series of calls that conform to the specification. The structure can be created either as a sequential list of unique service calls or a loop that contains a call element and uses a dataset to identify the unique calls.

1. In the test editor, select the test, and click **Add**, and then click **Specification-based Structure**.
   **Result**
   The **New Web Service Test** wizard opens.

2. On the **Web Service Specification Selection** page, Select **WS-ReliableMessaging**, and click **Next**.

3. Select one or several Web Services Description Language (WSDL) files in your workspace for the web service that you want to test, and click **Next**.

   If necessary, you can import a WSDL file into the workspace with the **Import** push button.

4. On the**Configure Protocol** page, select an existing HTTP transport configuration, or click **New** to create a new configuration.

   a. Specify the **URL** of the service, the HTTP **Method**, and **Version**.

   b. In the **Header** table, click **Add** to specify any specific headers that need to be added to the call.

   c. In the **Cookies** table, click **Add** to specify any specific cookies that need to be used by the call.

   d. Click **Next**.

5. On the **Sequence Options** page, specify how the sequence structure will be created in the test.

   a. In **Message count**, specify the number of calls in the list or the number loop iterations.

   b. Select **Create service call list** to generate a list of calls with the number of messages or **Create loop with dataset** to generate a loop with a dataset.
   The dataset defines the call number for each call in the loop.

   c. Select **Shuffle sequence** if you want the call numbers to be generated in a random order.

6. Click **Finish**.

   **Result**

   This action generates a reliable messaging service call structure in the test.

## Searching within tests

Search request data or response content by right-clicking in the data or content and selecting **Find**. To search for specific element types and to display the results in a table, click **Select**. For a still more powerful search and replace, use the **Test Search** function.

**About this task**
You can use a number of different methods to search within a test.

- Use the **Find** option to search within the **Test Details** area and, optionally, to replace text.
- Use the **Select** button to search within the **Test Contents** area and display a table of like test elements.
- Use the powerful **Test Search** function to search within the **Test Contents** and the **Test Details** areas. For example, you can search for a type of verification point and also declare whether the result should include enabled verification points, disabled verification points, or verification points in both states. The specific data that you can search for and the search options are protocol-dependent.

## Locating specific types of test elements

A test script can include multiple requests and responses with many test elements and attributes. To locate elements of a specific type in the **Test Contents** area, click **Select**. The results are displayed in a table, and you can sort the table columns. This option is also useful for viewing attributes of test elements that are the same type.

1. In the Test Navigator, browse to the test and double-click it.

   **Result**

   The test opens.

2. To restrict the search to specific instances of elements, select them in the test. For example, you might want to search for text in specific responses, rather than in all responses.

3. Click the test editor tree to give it focus, and then click **Select**.

   **Result**

   A list of element types opens. This list is generated dynamically based on the contents of the test. For example, if a test does not contain verification points, they are not listed.
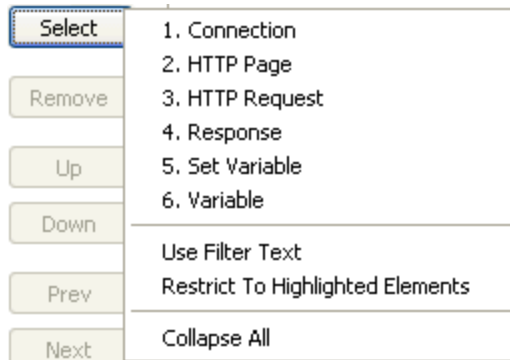


4. To include only the elements that you selected, select **Restrict To Highlighted Elements**.
5. To include only the elements that contain specific text from the **Test Contents** area, select **Use Filter Text**.
6. Select the type of test element to view from the list.
7. If you selected **Use Filter Text**, enter the filter text in the prompt, and then click **OK**.

   To use regular expressions in the filter text, click the **Search Options** icon to the right of the prompt, and then select **Regular expression**. By default, if **Regular expression** is not selected, the asterisk (*) and question mark (?) are interpreted as wildcard characters. To search for a URL that contains an asterisk or question mark, type a backslash (\) before the asterisk or question mark.

**Results**

The **Test Element Details** area displays the results under the heading **Multiple Items**.

**Exemple**

The following example shows results for an HTTP request. Other protocols might display less detailed information. Double-click a table row to locate the element within the test.

**Test Element Details**

Multiple Items

| | URL | Delay | | Response | Size |
|---|---|---|---|---|---|
| GET | www.ibm.com/ | 0 ms. | | 302 - Found | 206 |
| **GET** | **www.ibm.com/us/** | **0 ms.** | | 200 - OK | 31,458 |
| GET | www.ibm.com/common/v16/hp/js/h... | 0 ms. | | 200 - OK | 5,742 |
| GET | www.ibm.com/common/v16/css/all.css | 0 ms. | | 200 - OK | 138 |
| GET | www.ibm.com/common/v16/css/scr... | 0 ms. | | 200 - OK | 18,188 |
| GET | www.ibm.com/common/v16/css/scr... | 156 ms. | | 200 - OK | 99,371 |
| GET | www.ibm.com/common/v16/css/us/... | 0 ms. | | 200 - OK | 15,847 |

## Searching and replacing text in tests

With the **Test Search** function, you can search for text in a test or search within specific test elements and optionally replace the found text.

1. In the **Test Navigator**, browse to the test and double-click it.
   **Result**
   The test opens.
2. Right-click the test name, and then select **Test Search**.
3. In **Search for text**, type the text to locate.

   You can leave this field blank, depending on your search strategy. For example, if you know that a string occurs in elements or element instances that you are not interested in, by using the options described in steps 4, 6, and 8, you can locate the elements or element instances of interest before entering the search text into this field.

4. If you have selected pages or requests within the test, click **More Options**, and then select **Restrict search to elements highlighted in Test Contents**.
   This restricts the search to the selected pages and requests.
5. To perform a case-sensitive search, select **Case sensitive**. To search with regular expressions, select **Regular expression**.
   In regular expression mode, press Ctrl and type a space in **Search for text** for content assistance. Content assistance lists the regular expression patterns and the content that they match.
6. To highlight found elements in the **Test Contents** area, click **More Options**, and then select **Highlight found elements in Test Contents**.
   You can use this option with the option that is described in step 4 to designate the element instances of interest before specifying the text of interest.
7. To have the search include children of the selected element, click **More Options**, and then select **Recursive**.
   This option is selected by default. If **Recursive** is cleared, then only the selected element is searched.
8. To have the search locate both encoded and decoded versions of the specified text, click **More Options**, and then select **Match encoded and decoded values**.

This option is selected by default. The type of encoding that the search supports varies depending on the protocol.

**Example**

For example, when searching in HTTP data, `abc%123` and `abc%25123` match.

9. In the **Elements to search** list, select all test elements to search.

Selecting the check box in step 4 restricts the elements that you can select in this step to the instances that are selected in the **Test Contents** area. For example, if you select **HTTP Pages** here and only one page is selected in the **Test Contents** area, only that page is found. If the check box in step 4 is cleared, every test page is found.

10. **Optional:** Click selected elements to define how to search them.

A new area opens, where you can define how to search a selected element.

To locate items, continue to the next step. To replace found strings, click **Replace**, and go to step 12.

11. Click **Search**.

**Result**

The results of your search are displayed in two views

   ◦ The Search view, which lists the objects that contain matches
   ◦ The Test Search Match Preview view, which displays the matches that were found

12. In the Search view, complete any of these search actions:

   ◦ To preview a found string in the Test Search Match Preview, click the object.
   ◦ To open your test at the location where an instance is found, double-click the object.
   ◦ To perform a different search action (such as proceed to the next match or previous match, replace), right-click the object, and select your choice.

13. If you clicked **Replace** in step 9, the **Replace** window opens. In the **With** field, type the replacement text.

14. Select the replacement action by clicking the appropriate push button.

**Result**

If you are making selective replacements, found instances are displayed in the same order as in the **Test Search Match Preview** view. Click **Replace** or **Skip** until all found instances have been displayed.

## Exporting a test

To share the test scripts with manual testers or reviewers who do not have the workbench, export the test scripts to text files. You can export one file at a time.

1. In the Test Navigator view, double-click a test.
2. In the test editor, right-click the root node of the test and click **Export Contents to File**.
3. Select a project and specify the name of the file to export to.
4. To add a separator between two steps or lines, select the **Add line separators after each step** check box and click **Finish**.

**Results**

The text file opens on another tab in the workbench and is saved in the directory it is exported to.

## Copying test assets with dependencies

You can export test assets, and then import them into another project or workspace without losing any dependencies. Test assets include projects, schedules, and tests. You can export and import test assets to collaborate with other testers.

**Before you begin**
If you plan to export assets with dependencies, make sure that you have migrated the test assets to the current version of the product before you start to export.

**About this task**

When you copy a test with dependencies, any datasets or custom code referred to by the test are also copied. When you copy a schedule with dependencies, any locations or tests referred to by the schedule are also copied. When you copy results, any schedules or tests referred to by the results are copied.

1. In the **Test Navigator** view, right-click the test assets to export, and then click **Export**.
   You can export projects, schedules, tests, and test results with dependencies.
2. In the **Export** window, expand the **Test** folder, and then click **Test Assets with Dependencies**.
   You can export test assets with dependencies if the test assets were created in the current version of the product. You cannot export test assets with dependencies if the test assets were created in a previous version of the product and the assets have not been migrated to the current version of the product.
3. Click **Next**.
4. Specify the path and name of the archive file into which you want to export the selected test assets.
5. Click **Finish**.
   The assets are exported to the archive file. You are prompted if the total size of the test assets is larger than 4 GB, or if any individual test asset file is larger than 4 GB. To copy test asset files that are larger than 4 GB, copy the files manually.
6. If the target workspace is on a different computer, transfer the archive file to a location that is accessible to that computer.
7. In the **Test Navigator** view, select the test project into which you want to import the test assets.
   The target project must have the same name as the source project. Optionally, you can import test assets with dependencies into a workspace where no projects exist. If you import test assets with dependencies into a workspace where no projects exist, the Import wizard creates projects based on information from the archive file. To import test assets into a project with a different name, you must first import the test assets into a project with the same name, and then manually move the assets into the project with the different name.
8. Click **File > Import**.
9. In the **Import** window, expand the **Test** folder, and then click **Test Assets with Dependencies**.
10. Click **Next**.
11. In the **Import with dependencies** window, click **Browse**, and then select the archive file.
    The test assets are displayed in the **File contents** list.
12. Click **Finish** to import the test assets with dependencies from the archive file into the target project.

If a file that you are attempting to import already exists in the target workspace, you will be prompted to choose whether to overwrite the file. You can also choose to overwrite all files that already exist in the target workspace, or not to overwrite any files that already exist in the workspace. If you choose to overwrite all files that already exist in the target workspace, you will be prompted again if the import process encounters a `.classpath` or `.project` file in the source archive file.

## Copying projects

You can export a test project from a workspace and import it into another workspace.

**About this task**

If you export test assets to an archive file and then import them to another project, ensure that both project names are the same. Otherwise, you might not be able to locate your imported test assets.

> **Note:** You can also export the test project with all the dependent assets in to an archive file. See Copying test assets with dependencies on page 196 for the instructions.

1. Start Rational® Performance Tester, and select the source workspace.
2. Export the test project to an archive file.
   For instructions, see Exporting resources to an Archive file. Datasets can be located either in the same project as the tests that use them or in different projects. Be sure to export all the datasets that the exported tests require.
3. If the target workspace is on a different computer, transfer the archive file to a location that is accessible to that computer.
4. Start Rational® Performance Tester, and select the target workspace.
5. Click **File > Import**. Expand the **General** folder, and click the **Existing Projects into Workspace** icon; then click **Next**.
6. Click **Select archive file**, and then click **Browse** to select the archive file. Click **Finish** to import the source project from the archive file into the target workspace.
7. **Optional:** If the imported project contains custom code or tests that have been run, you might need to change the Java™ build path.
   The following examples are cases that might require a change to the Java™ build path:
   ◦ The Java™ build path was manually changed in the project from which it was exported. In this case, the same changes need to be made in the imported project. While importing, you are asked whether to overwrite the class path file, which stores the Java™ build path for project. Answering **Yes** reduces the likelihood that the build path will need to be changed.

◦ The project was imported onto a different computer with a different Java™ installation configuration. In this case, missing libraries must be removed from the build path.

◦ The project was imported into a workspace on a different disk drive. When you are asked whether to replace the class path file, answering **No** reduces the likelihood that the build path will need to be changed.

For instructions on changing the build path, see the Java™ Build Path page.

**What to do next**

If you encounter errors after importing a test project or when using an existing workspace with a new version of the product, you might need to delete .java files from the `src` folder in the workspace:

1. Click **Window > Open Perspective > Resource** to open the Resource perspective.
2. In the **Navigator** window, expand the test project folder, and locate the `src` folder.
3. Delete all .java files in the `src` folder, except for those that contain custom code.
4. Return to your test perspective: Click **Window > Open Perspective**, and select **Performance Test** (or **Service Test**, if you are using Rational® Service Tester).

## Disabling portions of a test

When you disable portions of a test, you can still see the disabled portion, but it is not executed during a run.

To disable an element:

1. In the Test Navigator, browse to the test, and double-click it.
   **Result**
   The test opens.
2. Right-click the element that you want to disable, and select **Disable**.
   The element and the dependent child elements, which are disabled automatically, are shaded and preceded by two forward slashes (`//`) to remind you that they are disabled.

   > ✎ **Note:** To change the color or symbol that represents disabled elements, click **Windows  > Preferences > Test > Test Editor**, and then click the **Colors and Fonts** tab.

   **Result**

   Although a disabled test element does not run, you can still work with it. For example, you can insert a test into a disabled user group for later use.

3. To enable a disabled element, right-click it, and select **Enable**. Select **Enable All** to enable all disabled elements.

**Exemple**
Disabling an element affects other elements in the following ways:

| Disabled element | Result |
| --- | --- |
| User group (percentage) | The percentages in the remaining user groups are recalculated. When you enable the user group again, remember to return all of the affected user groups to their original percentage. |
| Request or step that contains a data correlation reference | Substitution in the remaining actions that depend on this request does not work. |
| Request or step that contains a data correlation substituter | Substitution does not occur because the entire action is omitted. The substituter that uses the disabled data source is also disabled. To re-enable the substituter, select an enabled data source for substitution. |
| HTTP request that contains a server connection | No effect. The connection is automatically created in the next request. |
| Portion of custom code | Custom code with disabled arguments is flagged. If the disabling causes an unexpected number of arguments passed to custom code elements, you receive an error at runtime. To fix this, modify the custom code to check the number of arguments. |
| IF *data_source* construct | An IF construct is marked as invalid if it contains a disabled data source. |
| Test element and child are disabled | If you disable a child element and then disable its parent (for example, a request and then a page), the disabled child element will have two prefixes: one created manually and one inherited. In the following example, the first request has inherited the disabled state. The second request has been manually disabled and has also inherited the disabled state:<br><br>```<br>//disabled page<br>  //request<br>  ////disabled request<br>```<br><br>Do one of the following to re-enable the second request:<br><br>• Re-enable the request, and then re-enable the page.<br>• Right-click the page and select **Enable All**. |
| A data source or a range of text that will be replaced | The **Data** table displays this text in gray. |

## Running test elements in random order

You can record multiple user scenarios in a test and then run each scenario in a random order. To do this, you put each scenario under a random selector and then select the proportion of time that the scenario should be run.

**About this task**

For example, you can record a test that includes logging on to a system, browsing through items in the system, buying various items, and then totaling the order. In this case, you could run the logging in and the totaling scenarios once, but put the browsing and buying scenarios under a random selector.

1. In the Test Navigator, browse to the test and double-click it.
   **Result**
   The test opens.
2. Click the test element that will be controlled by the random selector, and then click **Insert > Random Selector**. Use Shift+Click to select multiple elements.
3. You are asked whether you want to move the selected elements into a new random selector. Click **Yes**. Click **No** to insert an empty random selector into the test.

   **Note:** To set whether or not elements are moved automatically, or whether you are prompted, click **Window > Preferences > Test > Test Editor**, and click the **General** tab.

4. Set the weight of the random selector. The weight determines the statistical probability that a specific element will be selected.
   a. If you have added a number of test elements, the **Create weighted blocks** window is displayed. You can select adjacent elements and group them. Each element—whether in a group or by itself—must be weighted.
   b. If you have added only one test element, the weighted block is displayed in the **Test Element Details** area with a default of 1.

**Exemple**

When a selector contains many different weights, you can mathematically determine the likelihood that a block will be executed. To do this, add the weights together and divide the weight for each block by that total.

For example, assume a selector contains six blocks set to the following weight:

- two blocks set to a weight of 1
- one block set to a weight of 2
- two blocks set to a weight of 5
- one block set to a weight of 9

The total of the weights is: 1 + 1 + 2 + 5 + 5 + 9 = 23. Therefore, the statistical likelihood of selection is:

| Weight of block | Likelihood of block being selected |
| --- | --- |
| 1 (two blocks) | 1/23. = 0.0435, or about 4.35% (for each block) |
| 2 | 2/23 = 0.0870, or about 8.70% |
| 5 (two blocks) | 5/23 = 0.2174, or about 21.74% (for each block) |
| 9 | 9/23 = 0.3913, or about 39.13% |

Note that a higher weight increases the likelihood, but does not guarantee, that a block will be executed. Some variation might occur. For example, if you run a test 23 times, you cannot predict that the first and second blocks will execute exactly once, the third block exactly twice, the fourth and fifth blocks exactly five times, and the sixth block exactly nine times. However, the more times that the blocks are executed, the more accurate this prediction is.

## Renaming test assets

As your test assets increase and become more complex, you might want to rename them. Use the Eclipse **Rename** function or save the assets under a different name.

Use either of the following steps to rename a test asset:

1. When you use the Eclipse **Rename** function, the new name is visible in the Test Navigator, but the underlying file system name is not changed. To use the Eclipse **Rename** function:

   a. In the Test Navigator, right-click the test asset, and then select **Rename**

   b. Type the new name, and then click **Enter**
      Be sure to click **Enter**, or the file will not be renamed.

2. When you rename a test asset by saving it under another name, the underlying file system name is changed, but you must perform manual cleanup. To save a test asset under another name:
   a. In the Test Navigator, browse to the test and double-click it. The test asset opens.
   b. Click **File > Save As**, and save the asset under a different name.
   c. Delete the original asset.

**Exemple**
The following table summarizes how renaming an asset affects the other assets in your workspace.

| Renamed asset | Effect on other assets |
| --- | --- |
| Project | Do not rename a project. Renaming a project might result in lost or corrupted project assets. |
| Schedule | Renaming a schedule has no affect on other assets, but note that results cannot be renamed. |

| Renamed asset | Effect on other assets |
|---|---|
| Test | When you use **Rename**, schedules that contain the old test name will still run correctly. To avoid confusion, manually update the schedule to use the new test name.<br><br>After you use **Save As**, manually update each schedule that uses the renamed test. |
| Custom code | If you rename the custom code class (.java file), then the reference to the class in the custom code action of the test will not work. Typically rename the custom code class in the Resource perspective or the Java™ perspective.<br><br>If you change the name of the custom code class in the test editor that implements the custom code action, the modification does not change the corresponding .java file; instead the modification causes the custom code action to refer to a different (and possibly new) custom code class. |
| Dataset | When you use **Rename** and open a test that contains the dataset, you are prompted to save the changes (in this case, the renamed dataset that the test now uses).<br><br>After you use **Save As**, manually update each test that uses the dataset. |
| Location | When you use **Rename**, locations (agent computers) are automatically updated in the schedules that use them.<br><br>When you use **Save As**, manually update each schedule that uses the test. |
| Results | You cannot rename results. |
| Weighted block | Renaming a weighted block has no affect on other assets. To rename a weighted block, click the block in the test, and type the new name in the **Name** field. |

## Deleting test assets

As your test assets grow and become more complicated, you might want to delete the assets that you no longer use.

In the Test Navigator, right-click the test asset, and then select **Delete**.

**Result**

The following table summarizes how deleting an asset affects the other assets in your workspace.

> 📝 **Note:** If you are deleting a test asset, you can choose to delete it from other test assets that refer it and you can choose to delete other test assets that are referenced only by the test asset that you are deleting.
>
> For example, if you delete a dataset, the **Remove references to *test asset name* in other test assets** option lets you delete the dataset from all the test assets that uses it.
>
> If you delete a test, the **Delete files that are referenced only by *test asset name*** option lets you delete all the test assets such as recsession and dataset that are referenced only by the test that you are deleting. If the dataset is used by another test too, it will not be deleted. The **Preview** button lets you see the assets that are referenced by the test.

| Deleted asset | Effect on other assets |
|---|---|
| Project | You are prompted whether to delete the project contents. If you click **Yes**, the contents are physically deleted. If you click **No**, you will not see the contents in the Test Navigator, but the project remains in your workspace, which is, by default, `C:\Documents and Settings\`*`user-`**`name`*`\__VENDOR_NAME__\__SDP_FULL_SHORT-`*`NAME__n.n`*`/workspace`). |
| Schedule | Deleting a schedule has no effect on other assets. |
| Test | If you delete the test in the Test Navigator, the test is physically deleted.<br><br>If you open a schedule and delete a test, the test is deleted from the schedule, but the test remains available as a test asset. |
| Custom code | If you delete the custom code class (.java file), then the reference to the class in the custom code action of the test will not work. Typically you delete the custom code class from the Resource perspective or the Java™ perspective.<br><br>If you delete the name of the custom code class that implements the custom code action, the deletion does not change the corresponding .java file. |

| Deleted asset | Effect on other assets |
|---|---|
| Dataset | If you delete a dataset in the Test Navigator, the dataset is physically deleted. When you open a test that uses the dataset, you are prompted to take one of these actions:<br><br>• Locate the dataset<br>• Remove the dataset reference from the test<br>• Leave the invalid reference in<br><br>You must correct or delete the reference to run the test successfully.<br><br>If you open a test and delete the dataset from the **Common Options** tab, only the reference to the dataset is deleted. |
| Location | You are not asked to confirm the deletion, nor are you warned if a user group uses the location. The user group is marked with a red x when you open the schedule that contains it. |
| Results | You are asked to confirm the deletion, and the results are physically deleted. |

## Debugging custom code for tests and compound tests

If you have custom code added to a test or a compound test, you can debug the custom code for any errors by clicking the **Debug** button.

**Before you begin**

This procedure can only be done when custom code is part of a test or compound test. If there are multiple custom code classes added to a test or compound test, the debug action debugs all the custom code classes.

You can use the debug option only in full Eclipse mode of the product. You cannot debug in the streamline mode.

1. Open a test or a compound test from the Test Navigator view.
2. Click the **Debug** button.
   **Result**
   The Debug view opens. If there are any breakpoints in the custom code, the test run pauses at the breakpoint. Press F8 to resume the test run.

## Providing tests with variable data (datasets)

You can produce more realistic tests by changing them to use datasets. During execution, a test that uses a dataset replaces a value in the recorded test with variable test data that is stored in the dataset. This substitution allows each virtual user to generate a different request to the server.

## Dataset overview

Datasets provide tests with variable data during a run. When you record a test, you perform a sequence of steps that you expect a typical user to perform. After the recording, a test is generated that captures these interactions. When you run this test, it uses the same data that you used during recording. To vary the data in the test, you use a *dataset*, that contains variable data. At run time, this variable data is substituted for the data in the recorded test.

If you need to create a dataset with many records, you can initialize the dataset quickly by importing data from a comma-separated-value (CSV) file. Also, you can export test data from your dataset into a CSV file to enable you to maintain large volumes of test data as a spreadsheet for reuse. Earlier to 9.5, the dataset (formerly known as datapool) was in .datapool format and starting from the 9.5 release, the dataset is in the csv format.

You can copy the CSV file and paste into your project to import the data from a CSV file and create a dataset. Similarly, to export the dataset values as a CSV file, you must copy the dataset from your project and paste it into your local machine.

> **Note:** Alternatively, you can use the **Import** option available in the **CSV editor** to import the data from a CSV file. For more information, see Editing datasets on page 217.

Perform the following steps should you plan to create a test that searches the IBM® website for three items: IBM® Rational® Performance Tester, IBM® Rational® Functional Tester, and IBM® Rational® Manual Tester:

1. Record a test that searches for one item.
2. Create a dataset and associate it with the test. See Creating a dataset associated with a test on page 205.
3. Associate a request in the test with a column in the dataset. See Associating a test value with a dataset column.
4. Add a loop in the test to fetch the values from different rows of a dataset. A test without a loop fetches the value only from the first row of the dataset. See Adding a loop to a test.

## Creating a dataset associated with a test

You can create a dataset that contains variable data for tests to use when they run. This is the preferred way to create a dataset because the dataset is automatically associated with a test. You can create anything from an empty dataset that contains one column, which you can edit later, to a fully functioning dataset.

1. In the Test Navigator, browse to the test and double-click it.
   **Result**
   The test opens.
2. In the **Test Contents** area, click the name of the test.

3. In the **Common Options** tab, click **Add Dataset**.

The options listed in the following table, enable you to create anything from a simple dataset that you can edit later to a complete dataset.

| To create | Do this in the Test Editor - Add Dataset window |
|---|---|
| A one-column dataset with a default access mode. | In **Existing datasets in workspace**, select *New Dataset<testname>.csv*, and click **Finish**. You can optionally name the dataset column in this session, and you can add other columns and data later. |
| A one-column dataset and choose the access mode. | In **Existing datasets in workspace**, select *New Dataset<testname>.csv*, and click **Next**. You can optionally name the dataset column in this session and you are prompted for the access mode. You can add other columns and data later. |
| An association between the test and an existing dataset. | Select the dataset. The dataset is associated with the test, and you can optionally set the access mode in this session. |
| A new, fully functioning dataset. | Select a project and click **Use wizard to create new dataset**. |

4. Select **Open mode** for the dataset. This mode determines the view that virtual users have of the dataset. Different tests can open the same dataset differently, and you can change the open mode later by opening the test and double-clicking the dataset title.

| Option | Description |
|---|---|
| **Shared (per test execution) (default)** | When you choose the Shared (per test execution) option, the virtual users running in the test share the dataset values in sequential order.<br><br>For example, if your dataset has 10 rows, the dataset values are taken from row 1, row 2, row 3, and so on when you select this option. |
| **Private** | Virtual users draw from a private view of the dataset, with dataset rows apportioned to each user in sequential order.<br><br>This option ensures that each virtual user gets the same data from the dataset in the same order. However, because each user starts with the first row of |

| Option | Description |
| --- | --- |
|  | the dataset and accesses the rows in order, different virtual users will use the same row. The next row of the dataset is used only if you add the test that is using the dataset in a loop with more than one iteration. |
| **Shared (for all test executions)** | When you choose the Shared (for all test executions) option, the virtual users running in multiple tests share the dataset values from the current row.<br><br>For example, if your dataset has 10 rows and when you set the current row as row 5, the dataset values are taken from row 5 instead of row 1 when you select this option. If you had set the current row as row 1 and used the dataset values until row 5, the dataset values are retrieved from row 6 when you run the test next time. |

5. If you are setting how the test accesses the dataset during this session, select one of the following options:
    ◦ **Sequential**: The rows in the dataset are accessed in the order in which they are physically stored in the dataset file, beginning with the first row and ending with the last.
    ◦ **Random**: The rows in the dataset are accessed in any order, and any given row can be accessed multiple times or not at all. Each row has an equal chance of being selected each time.
    ◦ **Shuffled**: Before each dataset access, the order of the rows is changed, and a different sequence results. Rows are accessed randomly but all rows must be selected once before a row is selected again.
6. Select one of the following options.

| Option | Description |
| --- | --- |
| **Wrap when the last row is reached** | By default, when a test reaches the end of a dataset or dataset segment, it reuses the data from the beginning. To force a test to stop at the end of a dataset or segment, clear the check box **Wrap when the last row is reached**. Forcing a stop might be useful if, for example, a dataset contains 15 records, you run a test with 20 virtual users, and you do not want the last five users to reuse information. Although the test is marked as *"Fail"* because of the forced stop, the performance data in the test is still valid. However, if it does not matter to your application if data is |

| Option | Description |
|---|---|
| | reused, the default of wrapping is more convenient. With wrapping, you need not ensure that your dataset is large enough when you change the workload by adding more users or increasing the iteration count in a loop. <br><br> **Note:** <br> ◦ With **Random** access order, **Wrap when the last row is reached** option is unavailable because you never reach the end of the row. <br> ◦ With **Shuffled** access order, if you select **Wrap when the last row is reached** option, you resume selecting from the beginning of the row with the same access order after each row has been selected once. No more selections are required if you clear the **Wrap when the last row is reached** option. |
| **Fetch only once per user** | By default, one row is retrieved from the dataset for each execution of a test, and the data in the dataset row is available to the test only for the duration of the test. Select **Fetch only once per user** to specify that every access of the dataset from any test being run by a particular virtual user will always return the same row. |

**Example**

To illustrate how these options affect the rows that are returned, assume that a test contains a loop which accesses a dataset. The loop has two iterations. The following table shows the row that is accessed in each iteration:

| Dataset option | Iteration 1 | Iteration 2 |
|---|---|---|
| **Sequential** and **Private** | row 1 | row 2 |
| **Shared** and **Shuffled** | row x | row y |

| Dataset option | Iteration 1 | Iteration 2 |
|---|---|---|
| **Fetch only once per user** | row x | row x |

7. If you are creating a fully functioning dataset, you can optionally import the data from a CSV file during this session by copying the CSV file and pasting into your project. For more information on importing dataset, see Editing datasets on page 217.

**What to do next**

After you have created a dataset and added data to it, the next step is to associate a value in the test with a column in the dataset, as discussed in .

## Creating a dataset in a workspace

You can create datasets in a workspace containing variable data that tests use when they run. You can use this method to create a dataset if you have not yet created the test that will use it.

1. Click **File > New > Dataset**.
2. In the **New Dataset** window, click the project that contains the dataset. The project is displayed in the **Enter, create, or select the parent folder** field.
3. In the **Name** field, type the name of the dataset, and then click **Next**.
4. In the window for describing the dataset, add a description.
5. In the **Dimensions** field, specify the number of rows and columns for the dataset that you want to create.
6. Click **Finish**.

**Results**

The new dataset opens in a browser. For instructions on how to add data to or edit the dataset, see teditdp_perf.dita on page 217.

**What to do next**

After you have created a dataset and added data to it, you must associate a value in the test with a column in the dataset.

## Converting an existing datapool to a dataset

Starting from 9.5 the dataset formerly known as datapool is in the CSV format. You can convert any existing datapool to a dataset.

**About this task**

When you open the workspace earlier to 9.5 in IBM® Rational® Performance Tester 9.5, the existing datapools in the workspace are stored in the Datapools (Legacy-Unsupported) folder as shown in the following figure.
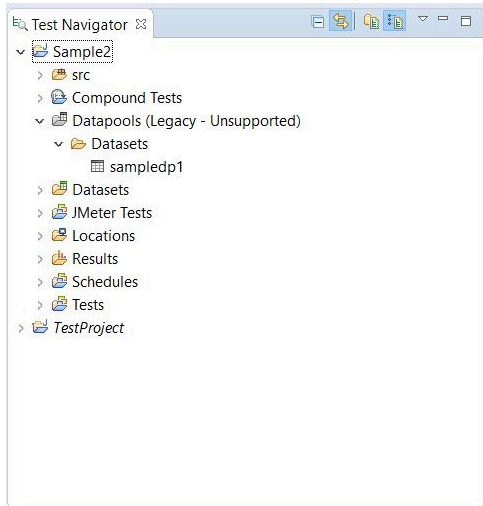
To convert the existing datapool to a dataset:

1. In the Test Navigator, browse and select the existing datapool.
2. Right-click and select **Convert to Dataset…**. Verify that the name of the dataset is the name of the existing datapool and format is .csv.
3. Click **Finish**. The converted datapool opens in a CSV editor.

**What to do next**

After you have created a dataset and added data to it, you must associate a value in the test with a column in the dataset.

## Creating datasets with multiple substitutions

Earlier to 9.2, you could substitute one dataset value at a time. Starting from 9.2, after the test is generated, you can view all the dataset candidates, add multiple candidates as dataset values, substitute values, and create a new dataset out of it. You can also substitute multiple dataset candidates for an existing dataset.

**About this task**

When you substitute multiple dataset candidates to create a new dataset, the same number of columns are created in the dataset. The names of the candidates become the names of columns and values in the dataset. When you substitute multiple dataset candidates in an existing dataset, the column names in the dataset are retained. If the number of substitutions chosen was greater than the number of columns in the dataset, the extra number of substitutions are added as columns in the dataset. For instance, if a dataset has three columns and you substitute five dataset candidates, two new columns are created by using the names of the dataset candidates.

To create a dataset from multiple dataset candidates:

1. In the Test Editor, select the name of the test and from the Test Details section, select **Common Options** and click **Show Dataset Candidates**.
   **Alternative**: After the test generation when you open the test, you are prompted that *"Some test data may need to be correlated or substituted"*. If you click **Yes**, you can see the list of dataset candidates.
2. Select the dataset candidates that you want to add as values to the dataset and click **Substitute multiple candidates**.
   The **Add Dataset** dialog box shows the list of datasets that are in the project but not associated with the test.
3. To associate an existing dataset with the test and assign the selected dataset candidates as values and substitutions, select a dataset and click **Next**. To associate a new dataset with the test, click the **Use wizard to create new Dataset** and click **Next**.
4. Select **Open mode** for the dataset. This mode determines the view that virtual users have of the dataset. Different tests can open the same dataset differently, and you can change the open mode later by opening the test and double-clicking the dataset title.

| Option | Description |
|---|---|
| **Shared (per test execution) (default)** | When you choose the Shared (per test execution) option, the virtual users running in the test share the dataset values in sequential order.<br><br>For example, if your dataset has 10 rows, the dataset values are taken from row 1, row 2, row 3, and so on when you select this option. |
| **Private** | Virtual users draw from a private view of the dataset, with dataset rows apportioned to each user in sequential order.<br><br>This option ensures that each virtual user gets the same data from the dataset in the same order. However, because each user starts with the first row of the dataset and accesses the rows in order, different virtual users will use the same row. The next row of the dataset is used only if you add the test that is using the dataset in a loop with more than one iteration. |
| **Shared (for all test executions)** | When you choose the Shared (for all test executions) option, the virtual users running in multiple tests share the dataset values from the current row.<br><br>For example, if your dataset has 10 rows and when you set the current row as row 5, the dataset values |

| Option | Description |
|---|---|
| | are taken from row 5 instead of row 1 when you select this option. If you had set the current row as row 1 and used the dataset values until row 5, the dataset values are retrieved from row 6 when you run the test next time. |

5. If you are setting how the test accesses the dataset during this session, select one of the following options:
    ◦ **Sequential**: The rows in the dataset are accessed in the order in which they are physically stored in the dataset file, beginning with the first row and ending with the last.
    ◦ **Random**: The rows in the dataset are accessed in any order, and any given row can be accessed multiple times or not at all. Each row has an equal chance of being selected each time.
    ◦ **Shuffled**: Before each dataset access, the order of the rows is changed, and a different sequence results. Rows are accessed randomly but all rows must be selected once before a row is selected again.
6. Select one of the following options.

| Option | Description |
|---|---|
| **Wrap when the last row is reached** | By default, when a test reaches the end of a dataset or dataset segment, it reuses the data from the beginning. To force a test to stop at the end of a dataset or segment, clear the check box **Wrap when the last row is reached**. Forcing a stop might be useful if, for example, a dataset contains 15 records, you run a test with 20 virtual users, and you do not want the last five users to reuse information. Although the test is marked as *"Fail"* because of the forced stop, the performance data in the test is still valid. However, if it does not matter to your application if data is reused, the default of wrapping is more convenient. With wrapping, you need not ensure that your dataset is large enough when you change the workload by adding more users or increasing the iteration count in a loop. |

> ✏️ **Note:**
>
> ◦ With **Random** access order, **Wrap when the last row is reached** option is unavailable because you never reach the end of the row.

212

| Option | Description |
|---|---|
|  | ◦ With **Shuffled** access order, if you select **Wrap when the last row is reached** option, you resume selecting from the beginning of the row with the same access order after each row has been selected once. No more selections are required if you clear the **Wrap when the last row is reached** option. |
| **Fetch only once per user** | By default, one row is retrieved from the dataset for each execution of a test, and the data in the dataset row is available to the test only for the duration of the test. Select **Fetch only once per user** to specify that every access of the dataset from any test being run by a particular virtual user will always return the same row. |

**Example**

To illustrate how these options affect the rows that are returned, assume that a test contains a loop which accesses a dataset. The loop has two iterations. The following table shows the row that is accessed in each iteration:

| Dataset option | Iteration 1 | Iteration 2 |
|---|---|---|
| **Sequential** and **Private** | row 1 | row 2 |
| **Shared** and **Shuffled** | row x | row y |
| **Fetch only once per user** | row x | row x |

7. Click **Finish**.

## How dataset options affect values that a virtual user retrieves

The Open, Access, and Wrap modes that you select for a dataset affect the values that a virtual user retrieves.

The following table lists the most common types of datasets and the options that you select to create them.

| Dataset purpose | Open mode selection | Access mode selection | Wrap mode selection |
|---|---|---|---|
| The virtual user retrieves the value from the current row of the dataset in a random order for every attempted transaction. Note that before accessing each row of the dataset the order of the rows is rearranged. | Shared (for all test executions) | Shuffled | Fetch only once per user |
| The virtual user retrieves the value from the current row of the dataset in sequential order for every attempted transaction. | Shared (for all test executions) | Sequential | Fetch only once per user |
| The virtual user retrieves the value from the beginning of the row of a dataset in a random order for every attempted transaction. | Shared (per test execution) | Random | Wrap when the last row is reached |
| The virtual user retrieves the value from the current row of a dataset in sequential order for every attempted transaction. When a test reaches the end of a dataset, it reuses the data from the current row selection of the dataset. | Shared (for all test executions) | Sequential | Wrap when the last row is reached |

## Enabling a test to use a dataset

Before a test can use variable data from a dataset, you must update the test to include a reference to that dataset.

1. In the Test Navigator, browse to the test and double-click it. The test opens.
2. Right-click the test name, and click **Add > Dataset**.
   **Result**
   The **Select Dataset File** window is displayed listing the datasets available to the test. If a test is already using a dataset, it does not appear in the list.
3. In the **Existing Dataset in workspace** list, click the name of the dataset that your test will use and click **Next**.
4. Select the **Open mode** for the dataset. This mode determines the view that virtual users have of the dataset. This option is useful when you do a parallel test run.

| Option | Description |
|---|---|
| **Shared (per test execution) (default)** | When you choose the Shared (per test execution) option, the virtual users running in the test share the dataset values in sequential order. |

| Option | Description |
| --- | --- |
| | For example, if your dataset has 10 rows, the dataset values are taken from row 1, row 2, row 3, and so on when you select this option. |
| **Private** | Virtual users draw from a private view of the dataset, with dataset rows apportioned to each user in sequential order.<br><br>This option ensures that each virtual user gets the same data from the dataset in the same order. However, because each user starts with the first row of the dataset and accesses the rows in order, different virtual users will use the same row. The next row of the dataset is used only if you add the test that is using the dataset in a loop with more than one iteration. |
| **Shared (for all test executions)** | When you choose the Shared (for all test executions) option, the virtual users running in multiple tests share the dataset values from the current row.<br><br>For example, if your dataset has 10 rows and when you set the current row as row 5, the dataset values are taken from row 5 instead of row 1 when you select this option. If you had set the current row as row 1 and used the dataset values until row 5, the dataset values are retrieved from row 6 when you run the test next time. |

5. Select the **Access mode** for the dataset:
   ◦ **Sequential**: The rows in the dataset are accessed in the order in which they are physically stored in the dataset file, beginning with the first row and ending with the last.
   ◦ **Random**: The rows in the dataset are accessed in any order, and any given row can be accessed multiple times or not at all. Each row has an equal chance of being selected each time.
   ◦ **Shuffled**: Before each dataset access, the order of the rows is changed, and a different sequence results. Rows are accessed randomly but all rows must be selected once before a row is selected again.
6. Select whether the test will reuse data when it reaches the end of the dataset.

   By default, when a test reaches the end of a dataset or dataset segment, it reuses the data from the beginning. To force a test to stop at the end of a dataset or segment, clear the check box **Wrap when the**

**last row is reached**. Forcing a stop might be useful if, for example, a dataset contains 15 records, you run a test with 20 virtual users, and you do not want the last five users to reuse information. Although the test is marked as *"Fail"* because of the forced stop, the performance data in the test is still valid. However, if it does not matter to your application if data is reused, the default of wrapping is more convenient. With wrapping, you need not ensure that your dataset is large enough when you change the workload by adding more users or increasing the iteration count in a loop.

7. Select whether the test will make the data in the dataset record permanent for each virtual user.

By default, one row is retrieved from the dataset for each execution of a test, and the data in the dataset row is available to the test only for the duration of the test. Select **Fetch only once per user** to specify that every access of the dataset from any test being run by a particular virtual user will always return the same row.

To illustrate how these options affect the rows that are returned, assume that a test contains a loop which accesses a dataset. The loop has two iterations. The following table shows the row that is accessed in each iteration:

| Dataset option | Iteration 1 | Iteration 2 |
|---|---|---|
| **Sequential** and **Private** | row 1 | row 2 |
| **Shared** and **Shuffled** | row x | row y |
| **Fetch only once per user** | row x | row x |

8. Click **Finish**.
   **Result**
   A reference to the dataset is added to the test, and the **Test Details** area is updated with the dataset information.
9. Save the test.

**What to do next**

Now that you have created a reference between the test and the dataset, the next step is to associate a value in the test with a column in the dataset.

## Viewing dataset candidates when you open a test

Dataset candidates are displayed automatically when you open a test for the first time. From the dataset candidates window you can view the dataset candidates in the test, bookmark locations of interest, and add or remove dataset references.

1. Record a test.
   **Result**
   When the test opens for the first time in the Test Navigator, the **Show Dataset Candidates** window is displayed. The **Show Dataset Candidates** window is displayed only if there are dataset candidates and if **Always display this dialog when a test is first opened** is selected. To prevent the **Show Dataset Candidates**

from being displayed when a test opens, clear the **Always display this dialog when a test is first opened**
check box in the **Show Dataset Candidates** window.

2. Do one of the following:

| Option | Description |
|---|---|
| **To view details about the dataset candidates in a test** | Navigate through the **Dataset Candidates** field to see them previewed in the **Preview** pane. Click the **Next** and **Previous** icons to move the selection down or up in the list of dataset candidates. Click the **Show as Tree** icon to toggle between tree format and list format. Click the **Sort** icon to sort the list of dataset candidates. Click the **Bookmark** icon to bookmark a location for later review. |
| **To select a data source for a dataset candidate** | Select the dataset candidate in the **Dataset Candidates** field, and then click **Substitute**. The **Select Data Source** window opens. |
| **To find more values in the test that have the same value as the selected dataset candidate** | Click **Find More and Substitute**. These values can be reviewed and substituted interactively as needed. |
| **To remove a substitution** | Select a substitution site, and then click **Remove Substitution**. |

3. Click **Close** to close the **Show Dataset Candidates** window and proceed to the test in the test editor.
To display the **Show Dataset Candidates** window again while in the test editor, click the root node of the test.
Then click the **Common Options** tab under **Test Element Details**, and then click **Show Dataset Candidates**.

## Editing datasets

You can add, modify, or remove data from a dataset by using the CSV Editor. The working principle of the CSV Editor
is similar to that of a spreadsheet.

**Before you begin**

You must have created a dataset. See Creating a dataset in a workspace on page 209.

**About this task**

In V9.5.0 or later, you can use the CSV Editor to view and edit data in the dataset.

To have seamless access to a dataset CSV editor use any one of the following web browsers on Windows or Linux or
Mac operating systems:

- Mozilla Firefox
- Google Chrome
- Microsoft Edge based on Chromium

You can also view the datasets in other editors by right-clicking the dataset and selecting the **Open With** option.

When you want to run a test asset with different dataset values, you can either edit the existing dataset or create a new dataset to use it during the test asset run. When the number of edits is minimal, it is easier to edit the dataset within the CSV Editor.

You can perform basic tasks in the CSV Editor by right-clicking any row, column, or cell of the dataset to organize your data in a better way. For example, updating the data in a cell, inserting or deleting rows and columns, or renaming column names.

When you edit the dataset in a CSV Editor, you can use the following keyboard shortcuts to control the cursor selection in the CSV Editor:

- **Tab** - To move the cursor control to the next available option.

- **Shift-Tab** – To move the cursor control to the previous option.

- **Shift+F10** – To open the context menu from the dataset cell.

📝 **Note:**

You cannot resize the width of rows in the CSV Editor. When you have a large amount of data in a cell, you can right-click the cell and select **Copy** (or Ctrl+C), and then paste it into a text-editing program to view the content. Alternatively, you can hover the mouse over the cell to view the content.

1. Double-click the dataset that you want to edit in the **Test Navigator**.
   **Result**
   The dataset opens in the CSV Editor in a bowser.
2. Perform the following actions to use the options available in the CSV Editor:

| Option | Description |
|---|---|
| **Set as current row** | Right-click any cell in a row and select **Set as current row**. <br><br> **When rows are deleted:** <br><br> If you delete any row between row 1 to current row, the current row data is taken from the next row. <br><br> For example, when you set the current row as 6, and then you delete any row between row 1 to row 6, the current row remains at row 6, but the content of row 7 is moved to row 6. <br><br> **When rows are inserted:** |

| Option | Description |
| --- | --- |
| | If you insert any new row between row 1 to the current row, the current row data is taken from the previous row. |
| | For example, when you set the current row as 6, and then you insert any row between row 1 to row 6, the current row remains at row 6, but the content of row 5 is moved to row 6. |
| **Find and Replace**  | To find: <br><br> a. Click the **Find and Replace** icon  . <br> b. Enter the content that you want to search in the **Find** field. <br> c. Select any or all the following options to find the search content more effectively: <br><br> ▪ Select the **Case sensitive** check box to search the content that is the exact letter case of the content entered in the **Find** field. <br> ▪ Select the **Match entire cell contents** check box to search for cells that contain only the characters that you have entered in the **Find** field. <br> ▪ Select the **Search using regular expression** check box to search the pattern that matches strings. <br><br> For example, to search a cell that contains any number between 0 to 9, do the following: <br><br> i. Enter \d in the **Find** field. <br> ii. Select the **Search using regular expression** check box. <br> iii. Click **Find**. <br><br> d. Click **Find**. If the text is found, the cell containing that text is selected. <br> e. Click **Find** again to find further instances of the search text. |

| Option | Description |
|---|---|
| **Find and Replace** Q | To find and replace:<br><br>a. Click the **Find and Replace** icon Q .<br>b. Enter the content that you want to search in the **Find** field.<br>c. Enter the content that you want to replace in the **Replace** field.<br>d. Select any or all the following options to find and replace the content more effectively:<br>▪ Select the **Case sensitive** check box to find the content that is the exact letter case of the content entered in the **Find** field.<br>▪ Select the **Match entire cell contents** check box to find and replace for cells that contain only the characters that you have entered in the **Find** and **Replace** fields.<br>▪ Select the **Search using regular expression** check box to find and replace the pattern that matches strings.<br>e. Click **Replace** to replace the individual instances.<br>f. Click **Replace All** to replace every instance of the content throughout the dataset. |
| **Undo** ↺ | a. Click the **Undo** icon ↺ .<br>b. Select the recent changes from the list that you want to undo, and then click the list.<br><br>The **Undo** option undoes anything you do in the dataset. The CSV Editor saves the unlimited undo-able action. You can perform the undo action even after you save your changes made to the dataset. |
| **Redo** ↻ | a. Click the **Redo** icon ↻ .<br>b. Select the recent changes from the list that you want to redo, and then click the list. |

Test Author Guide

| Option | Description |
|---|---|
| | The CSV Editor saves the unlimited redo action. |
| **Import**  | a. Click the **Import** icon  . <br> b. Click **Choose File** and select the CSV file that you want to import in the **Import File** dialog box. <br><br> **Note:** <br><br> If the CSV file contains test data with Unicode characters in it, you must save the CSV file in `UTF-8` format. You can then choose the CSV file and import the test data from the CSV file into the dataset. <br><br> c. Optional. Click **Overwrite** to add the rows and columns from the selected CSV file from the beginning of the dataset. <br> d. Optional. Click **Append** to add rows and columns from the selected CSV file to the end of the dataset. <br> e. Optional. Select the **First row contains headers** check box if your CSV file contains the header. |
| **Export**  | Click the **Export** icon  to download the dataset as a CSV file. |
| **Dataset configuration settings**  | In the **Configure Dataset** window, you can change the row and column settings and configure the string values in the dataset that contains variable data for tests to use when they run. <br><br> a. Click the **Menu** icon  and select **Configure** option. <br> b. Configure the following options in the **Configure Dataset** window: |

| Option | Description |
|---|---|
| | <ul><li>**Column header** - To change the column header, use an up-down control button to increment or decrement the value.</li><li>**Data start point** - To change the data starting pointer, use an up-down control button to increment or decrement the value.</li><li>**Current row** - To set the current row, use an up-down control button to increment or decrement the value. You can also see the Current row number on the upper-right side.</li><li>**Treat empty text as null** -If a dataset contains any blank cells, the value of those blank cells is interpreted as null when you select this field.</li><li>**Treat as null** - Enter a string value that is to be treated as null when running test.</li><li>**Treat as empty** - Enter a string value that is to be treated as empty when running test.</li></ul> |
| **Discard** | Click the **Menu** icon and select **Discard** to discard the changes made to the dataset. |

3. Click the **Save** icon to save the changes made to the dataset and close the CSV editor.

## Encrypted datasets overview

You can encrypt one or more columns in a dataset. If you want to encrypt confidential information such as a set of passwords or account numbers that are used during a test, you can use an encrypted dataset.

Dataset columns are encrypted using the RC4 private-key algorithm. You can use only one password to encrypt columns in any given dataset. Encrypted datasets are not supported on agent computers that are running the z/OS® or AIX® operating systems.

⚠️ **Important:** If you forget the password to a dataset, there is no way to recover the password.

When you run a test that uses a dataset that contains encrypted variables, you are prompted for the dataset password. If the test uses multiple encrypted datasets, you must enter the password for every encrypted dataset that the test uses.

When you run a test that uses a dataset with an encrypted column, the value of the column is decrypted at a run time. The column value is sent as a cleartext string in the requests to the server. The actual values of the encrypted dataset variables are not displayed in the test log. The test log displays asterisks for the encrypted dataset variables.

To see the actual values of variables that are sent to the server at run time, you must use custom code. You can send the dataset column value to custom code that writes the value to a file other than the test log. If the custom code writes to the test log using `tes.getTestLogManager().reportMessage()`, then asterisks are displayed instead of the decrypted variables.

## Encrypting a dataset column

To secure test data, you must encrypt datasets. You can encrypt data in the columns of a dataset by using an encryption key. When you run a test that utilizes a dataset with encrypted variables, you must enter the encryption key for the encrypted column that the test uses.

**Before you begin**

You must have created a dataset. See Creating a dataset in a workspace on page 209.

1. Double-click the dataset in the Test Navigator.
   **Result**
   The dataset is displayed in a browser.
2. Right-click any cell in a column that you want to encrypt and select **Encrypt column data**.
   **Result**
   The **Encrypt Column** window is displayed.
3. Enter an encryption key in the **Encryption Key** field to encrypt the data in the column.

   🔔 **Remember:** When you have already encrypted other columns in the dataset, you must enter the same encryption key that you used previously. You can use only one encryption key to encrypt columns in a dataset.

   ⚠️ **Important:** The encryption keys you use to encrypt data in a dataset are not stored on the server nor can be retrieved from the server. Therefore, you must remember to store the encryption keys in a secure location. You must use the same encryption keys to view the encrypted values, to decrypt data, or enable the use of the encrypted dataset during test runs.

4. Click **Encrypt Column**.

**Result**

Asterisks are displayed instead of actual data for the encrypted column.

**Results**

The dataset column is encrypted.

## Decrypting a dataset column

To view the content of an encrypted dataset, you can decrypt the dataset. Removing encryption from a dataset revokes the protection offered to the test data.

**Before you begin**

You must have created a dataset with at least one encrypted column. See Creating a dataset in a workspace on page 209 and Encrypting a dataset column on page 223.

1. Double-click the dataset in the Test Navigator.
   **Result**

   The dataset is displayed in a browser.
2. Right-click encrypted cells that display the contents with asterisks, and then select **Decrypt column data**.
   **Result**

   The **Decrypt Column** window is displayed.
3. Enter the encryption key that you used to encrypt the data in the column in the **Encryption Key** field.
4. Click **Decrypt Column**.
   **Result**

   Asterisks are replaced with the actual data in the decrypted column.

**Results**

The encryption is now removed from the selected column in the dataset. When you run a test that uses a dataset that contains decrypted data, the variable data is substituted for the data in the recorded test without prompting for the encryption key.

## Navigating between a dataset and a test

After you have created a dataset or imported a comma-separate values (CSV) file into a dataset, you can navigate between the dataset and associated tests in the test editor. You can enlarge the test and the dataset, list the datasets that a test uses, navigate from a row in a dataset to the corresponding element in the test, see the data for a page or request, and add or remove dataset references.

1. In the Test Navigator, browse to the test and double-click it. The test opens.
2. Do one of the following actions:

| Option | Description |
|---|---|
| **Maximize the test window** | Double-click the test tab (for example, ). Do not click the **x**, or you will close the test. To return to the default perspective, click **Window > Reset Perspective**. |
| **View the datasets that a test uses** | In the **Test Contents** area, click the first line of the test, which is the test name. |
| **Navigate from a row in a dataset to its corresponding element** | a. In the **Test Contents** area, click the test name, which displays the dataset.<br>b. Expand the dataset to display the rows.<br>c. Double-click the row. |
| **View the data for a page or request** | In the **Test Contents** area, click the page or request. |
| **To add a reference to a dataset** | In the **Test Element Details** area, drag your cursor over the candidate, right-click, and select **Substitute > Select Data Source**. The **Select Data Source** window opens. If you have not already added the dataset to the test, click **Dataset**, and then add the new dataset. |
| **Remove a reference to a dataset** | In the **Test Element Details** area, drag your cursor over the reference, right-click, and select **Remove Substitution**. |

3. Save the test, if you have made any changes.

## Test variables

A test variable is a user-defined, name-value pair that stores and refers to information throughout a test and between tests.

A variable is declared in the test variables section of the test. You can use it throughout the test as a reference for any field that can be substituted. Substituting data from a test variable is achieved using the **Test Variables** page of the Test Data Source view. You can do the following actions to a test variable:

- Provide a default value to the variable during declaration.

- Change the value of the variable using Set Variable statement. You can use the Add and Insert menus of the Test Editor to create Set Variable statements.

- Set hard-coded value or value retried from a data source, such as dataset or reference that appears before the Set statement to the variable.

If a variable is initialized at various places such as test, compound test, schedule, or user group, the product uses the following order to initialize the value of the variable when running the test. The variable set in the variable table of the compound test editor takes the highest precedence followed by others:

1. Compound test setting in the variable table UI
2. Compound test specified in a var file
3. User group setting in the variable table UI
4. User group specified in a var file
5. Schedule specified setting in the variable table UI
6. Schedule specified in a var file
7. Command line

**Note:** You must select **All tests for this user** from the **Visible in** drop-down list to take the precedence of variable initialization.

## Sharing variables among tests

In order to share variables between tests, all the tests must contain the variables with the same name. The variables must also have **Visible in** set to **All tests for this user**. When these conditions are met and multiple tests have been placed in a schedule, then variable in the dataset of one test can be used in the other test.

A common reason to share data between tests is to perform data correlation. With data correlation, a variable is set to a response that comes from a request in one test and is used in requests performed in a different test. Assume that you are testing an employee database. The Create Employee test creates an employee record and the Modify Employee test modifies an employee record. When a new record is created, it is assigned a record ID. Variables can be used to pass the record ID from a response in the Create Employee test to the Modify Employee test. A user-defined variable is not shared among different virtual users. The variable is shared only among the different tests of the same virtual user. If you set **Visible in** to **This test only**, then dataset from a test is not available to another even if both tests contain variables with the same name.

If you want to share variables between the different types of test scripts in your product, consider the following points:

- Declare the test variables with the same name across all the test scripts for the variables to communicate with each other. Set **Visible in** to **All tests for this user**.

- Include the required test scripts into a compound test.

Using test steps, you can share the default values of the variables to another test script. You can also assign new values to the variables and use the latest values in another test script.

If you want to share variables between the test scripts of different testing products such as Rational® Functional Tester, Rational® Performance Tester, or Rational® Integration Tester, you must consider the following points:

- If you are using Installation Manager, you must shell-share or install the products in the same package group.

- Declare the test variables with the same name across the Rational® Functional Tester and  Rational® Performance Tester test scripts. Set **Visible in** to **All tests for this user**.

- Include the required test scripts into a compound test.

- If you are using  Rational® Integration Tester test scripts, you must map your tags with the test variables of Rational® Functional Tester or  Rational® Performance Tester.

**Using variables to access datasets**

You can define variables so that they share data from a dataset throughout tests. This is achieved by having the value field of a Set Variable statement substituted from a dataset. By doing so, the first test which appears in the schedule can set the variable from a dataset and share it with the other test in a schedule.

Assume that you have two tests that log in to an application using a user ID from a dataset. The first test can set the value of a variable from the dataset, and both tests can use the variable, instead of directly using the dataset. In this case both use the same record from the dataset. This is similar to the fetch-only-once-per-user behavior of a dataset. However, fetching once means that during playback, a virtual user gets only one record from the dataset. The one-record limit holds even if the tests are in a loop, and are run several times by the virtual user. By using the user-defined variables, the virtual user retrieves a new record each time through the loop, and both tests can use the same record.

> **Note:** Assignment (set) operators can not only have a variable value substituted from a dataset, but also in the declaration of a variable. You can substitute the assignment operator and variable value from any data source, and thus that value can be shared between tests as well.

**Array variables**

You create an array variable to add multiple values to a variable. If you create a secondary HTTP request, add complete paths of the requests in the array variable that can be used a custom code during playback.

## Declaring and assigning test variables

When you declare a variable, you can create a container for it, initialize it to a string or a dataset value, and set its scope. Then, within the test, you can reassign another value to the variable.

**About this task**

If the data that you want to assign to a variable is only available after a specific test step, instead of initializing the variable, you need to add a variable assignment further down in the test, so that when the assignment occurs, the data that you need to use is available. Otherwise, when you try to initialize the variable (or do the assignment), the value that you want to use will not be available and will not show up as an option to select.

If a variable is initialized at various places such as test, compound test, schedule, or user group, the product uses the following order to initialize the value of the variable when running the test. The variable set in the variable table of the compound test editor takes the highest precedence followed by others:

1. Compound test setting in the variable table UI
2. Compound test specified in a var file
3. User group setting in the variable table UI
4. User group specified in a var file
5. Schedule specified setting in the variable table UI
6. Schedule specified in a var file
7. Command line

**Note:** You must select **All tests for this user** from the **Visible in** drop-down list to take the precedence of variable initialization.

To create, initialize, and assign a value to a test variable:

1. In the Test Navigator, browse to the test and double-click it.
   **Result**
   The test opens.
2. To create a container for the test variables that you create in a test:

   a. Open the test, and in the**Test Contents** area, click **Test Variables**.

   b. Select **Add >  Test Variable Container**.
   **Result**
   A container named **Test Variables** is created for the user-defined variables.

   c. Select the container to rename it.
   **Result**
   The **Test Element Details** area opens for you to type a new name in the **Name** field.

3. To declare or define a test variable:

   a. Open the test, and in the **Test Contents** section, click the user-defined container to contain the variable.

   b. To create a variable, select **Add >  Variable Declaration**. To create an array variable, select **Add >  Array Variable Declaration**.

   c. Type the name of the variable, and click **OK**.
   **Result**
   The variable is added as the last element in the container and the **Test Element Details** area opens.

   d. In the **Test Element Details** area, set the scope and initial value for the variable.

**Visible in**: Select **This test only** to restrict data to the current test only. Even if another test has a variable with the same name, that variable will not change. Select **All tests for this user** to share the value of this variable when the test runs in a schedule. For the variable to be shared, both tests must have a variable with the same name and must have this option enabled.

**Check Value**: Select **When first used** to check whether or not a variable is initialized only after the test execution reaches the first request that uses a variable. Select **At test start** to check whether or not a variable is initialized when starting the execution of the test. If the variable is not initialized, then an error message is displayed, depending on the behavior set.

**If not initialized, set to**: Select **Text** to initialize the variable to a specific value whenever the test runs in the schedule. Select **Dataset value** and, in the **Select Data Source** window, select the dataset that will initialize the variable.

**Run-time error if variable not initialized**: Select the action for the run when it encounters an uninitialized test variable. If you select **Issue test log warning** or **Issue test log error**, verify that the **Test log** page in the schedule sets errors, failures, and warnings to **All**, which is the default setting. If you select **Exit the test**, the schedule continues to run although the virtual users that have the uninitialized variable stop. If you select **Do nothing**, the test continues to run.

4. To assign or initial a value to a test variable:

   a. Open the test, and in the **Test Contents** area, select a test element.

   b. Select **Insert >  Variable Assignment**, which inserts the assignment before the selected element.
   **Result**
   The **Test Editor** window opens and lists the variables available to the test.

   c. Select the variable that you are assigning a value to and, in the **Set to** box in the **Test Element Details** area, set the value for the variable.
   You can set the value to a text string, to any data source that exists in the test before the assignment statement, or to **Not initialized**.
   **Result**
   A `Set` statement is added to the test, with the value you chose.

## Initializing variables from the command line

To initialize test variables from an XML file, you can run the test from the command-line interface using the `varfile` option.

**Before you begin**

- Declare the variables using IBM® Rational® Performance Tester.
- Create an XML file that contains the variables with values. The XML file would have a structure similar to the following image

```xml
<?xml version="1.0" encoding="UTF-8"?>
<inits>
    <variable_init value="9.999.99.999" name="hostname"/>
</inits>
```

**About this task**

If a variable is initialized at various places such as test, compound test, schedule, or user group, the product uses the following order to initialize the value of the variable when running the test. The variable set in the variable table of the compound test editor takes the highest precedence followed by others:

1. Compound test setting in the variable table UI
2. Compound test specified in a var file
3. User group setting in the variable table UI
4. User group specified in a var file
5. Schedule specified setting in the variable table UI
6. Schedule specified in a var file
7. Command line

**Note:** You must select **All tests for this user** from the **Visible in** drop-down list to take the precedence of variable initialization.

1. Navigate to the directory that contains the `cmdline.bat` and `cmdline.sh` files. On Windows™ operating systems, this directory is typically found as *productInstallationDirectory*/`cmdline`. For example, `C:\Program Files\IBM\SDP\cmdline`.
2. Issue the following command:
   **Example**

   **Note:**
   - The workspace is locked after you issue the command. To check the progress of the test during the run, invoke another workspace and open the project through that workspace.
   - On Linux operating system, the command must start with `cmdline.sh`.

   If a value contains spaces, enclose the value in quotation marks. To see the online help for this command while you are in the directory that contains the .bat file, type `cmdline -help`.

   The following table explains each option:

| -work-<br>space | Required. The complete path to the Eclipse workspace. |
|---|---|
| -project | Required. The path, including the file name of the project relative to the workspace. |
| -eclipse-<br>home | Optional. The complete path to the directory that contains `eclipse.exe`. |
| -plugins | Optional. The complete path to the folder that contains the plugins. Typically, on Windows operating systems, this folder is located at `C:\Program Files\IBM\IBMIMShared\plugins`.<br><br>Required. This option is required only if the folder is at a different location. |
| -suite | Optional. However, in a command, it is mandatory to use one of the following options:<br><br>    ◦ -suite<br><br>You must not use the -suite option along with the other options. The path includes the file name of the suite to run relative to the project.<br><br>Starting from V9.2.1.1, you can execute multiple tests simultaneously.<br><br>For example, **-suite** *test1:test2:test3*. |
| -im-<br>portzip | Optional. To import the project as test assets with dependencies into your workspace, use the -importzip option. This command is available from V9.2.1.1 and later.<br><br>For example, `C:\User\Desktop\test1.zip` |
| -varfile | Optional. You can use this option to specify the complete path to the XML file that contains the variable name and value pairs. |
| -config-<br>file | Optional. You can use this option to specify the complete path to a file that contains the parameters for a test run. Each parameter must be on a single line. To create a configuration file, you must use an editor that does not wrap lines. Any parameters, whether required or optional, can be set in the configuration file. The command line parameters override the values in this file.<br><br>    📝 **Notes:** |

| | ◦ If you are creating a config file manually, the file must be in the UTF-8 format. You must not use quotation marks in this file even for values that contain spaces.<br><br>◦ You can create command line config file from the product, which you can use while running tests or schedules from the command-line interface or Maven. For more information about how to create a command line config file from the product, see related links. |
|---|---|
| **-results** | Optional. You can use this option to specify the name of the results file. The default result file name is the test name with a time stamp appended. You must specify a folder name that is relative to the project to store the test results.<br><br>For example, **-results** *folder/resultname* |
| **-over-write** | Optional. Determines whether a results file with the same name is overwritten. The default value, `false`, indicates that the new results file is created. If the value is `true`, the file is overwritten and retains the same file name. You must use double quotes "" for values `true` or `false`. |
| **-quiet** | Optional. Turns off any message output from the launcher and returns to the command shell when the run or the attempt is complete. |
| **-vmargs** | Optional. To specify the Java™ maximum heap size for the Java™ process that controls the command line playback, use the **-vmargs** option with the *-Xmx* argument.<br><br>For example, when you use **-vmargs** *-Xmx4096m*, specify a maximum heap size of 4096m. This method is similar to specifying *-Xmx4096m* in the eclipse.ini file for the workbench when playing back the test from the user interface. |
| **-publish** | Optional. You can use this parameter to publish test results to the IBM® Rational® Test Automation Server. You can use the following options along with the **-publish** parameter:<br><br>◦ *serverURL*#project.name=*projectName*: You can use this option to specify the project name. If the project name is not specified, the name of the current project is used.<br><br>    **Note:** If you provide the server and the project details under **Window > Preferences > Test > Rational Test Automation Server** in the product, and if you use *serverURL*#project.name=*projectName* along with the **-publish** parameter, the server details in the command-line interface takes precedence over the product preferences.<br><br>◦ `no`: You can use this option if you do not want to publish test results after the run. This option is useful if the workbench preferences are set to publish the results, but you do not want to publish them. |

| | |
|---|---|
| | ⚠️ **Important:** You must provide the offline user token for the server by using the **RTCP_OF-FLINE_TOKEN** environment variable before you use the **-publish** parameter in the command-line interface. |
| **-pub-lish_for** | Optional. You can use this option to publish the test results based on the completion status of the tests:<br><br>◦ ALL - This is the default option. You can use this option to publish test results for any text execution verdict.<br>◦ PASS - You can use this option to publish test results for the tests that have passed.<br>◦ FAIL - You can use this option to publish test results for the tests that have failed.<br>◦ ERROR - You can use this option to publish test results for the tests that included errors.<br>◦ INCONCLUSIVE - You can use this option to publish test results for the tests that were inconclusive.<br><br>You can add multiple parameters separated by a comma. |
| **-export-log** | Optional. You can use this parameter to specify the file directory path to store the exported HTTP test log.<br><br>Starting from V10.0.1, by using the **-exportlog** parameter, you can provide multiple parameter entries when running multiple tests. You must use a colon to separate the parameter entries.<br><br>For example: **-exportlog** *c:/logexport.txt:c:/secondlogexport.txt*<br><br>If there are multiple **-suite** parameter entries with a single **-exportlog** parameter entry, then the **-exportlog** parameter generates the appropriate number of test logs by appending 0, 1, 2, and so on to the **-exportlog** parameter entry name.<br><br>For example: **-suite** *"sampletest1:sampletest2:sampletest3"* **-exportlog** *c:/logexport.txt* The command generates the following test logs:<br><br>◦ logexport_0.txt<br>◦ logexport_1.txt<br>◦ logexport.txt<br><br>The last test log generated has the same name as that of the initial **-exportlog** entry.<br><br>📝 **Note:** If there are multiple **-suite** and **-exportlog** parameter entries, the number of **-suite** entries must match with the number of **-exportlog** entries. Otherwise, the following error message is displayed: |

| | |
|---|---|
| | *Error, number of -suite and -exportlog entries do not match.* |
| **-export-stats** | Optional. You can use this option to export reports in comma-separated values (CSV) format, with the file name derived from the report name. This directory can be relative to the project or a directory on your file system. If the **-exportstatreportlist** option is not specified, the reports specified on the **Export Reports** page of the **Performance Test Report** preferences are exported. |
| **-export-stat-shtml** | Optional. When you want to export web analytic results, you can use this option. The results are exported in the specified directory. You can then analyze the results on a web browser without using the test workbench. |
| **-com-pare** | You can use this argument along with **-exportstatshtml** and **-execsummary** to export the result in compare mode. The value can be paths to the runs and are relative to the workspace. You must separate the paths by a comma. |
| **-export-statre-portlist** | Optional. You can use this option to specify a comma-separated list of report IDs along with **-exportstats** or **-exportstatshtml** to list the reports that you want to export in place of the default reports, or the reports selected under **Preferences**. To view this setting, navigate to **Window > Preferences > Test > Performance Test Reports > Export Reports**.<br><br>To copy the report IDs list into your command line, navigate to **Window > Preferences > Test > Performance Test Reports > Export Reports**. Under **Select reports to export**, select the required reports, and click **Copy ID to clipboard**. You can then paste the clipboard content on to your command line editor |
| **-exec-summa-ry** | Optional. You can use this option to export all of the reports for the test run in a printable format, also known as an executive summary, to the local computer. You must specify the path to store the executive summary. |
| **-exec-summa-ryreport** | Optional. You can use this option to export a specific report as an executive summary for the test run to the local computer. You must specify the ID of the report to export.<br><br>For example, to export an HTTP performance report, specify `http`. You must use this option along with **-execsummary**.<br><br>To copy the report IDs list into your command line, navigate to **Window > Preferences > Test > Performance Test Reports > Export Reports**. Under **Select reports to export**, select the required reports, and click **Copy ID to clipboard**. You can then paste the clipboard content on to your command line editor |
| **-user-com-ments** | Optional. You can add text within double quotation mark (*""*) to display it in the **User Comments** row of the report. |

> 📝 **Note:** You can use the file `CommandLine.exe` to run the command to add comments in a language that might not support Unicode characters on Windows operating system.

| | |
|---|---|
| **-publishreports** | Optional. You can use this option to publish test results in Rational® Test Automation Server. The parameters that you can use with it are the following:<br><br>   ◦ FUNCTIONAL<br>   ◦ MOBILE_WEBUI<br>   ◦ STATS<br>   ◦ TESTLOG<br><br>For example, **-publishreports** *"STATS, TESTLOG"*<br><br>You must prefix with *"!"* to publish all the reports except the specified one.<br><br>For example, **-publishreports** *"! TESTLOG"*<br><br>All the reports except the TESTLOG report is published to Rational® Test Automation Server after executing the command. |
| **-stdout** | Optional. You can use this option to display the information about the test on the command line.<br><br>After you run a test from the command line, the following outputs are displayed to give you the overall information of the test :<br><br>   ◦ --VERDICT: The verdict of the test .<br>   ◦ --REMOTE_RESULT: The URL of the result published to Rational® Test Automation Server.<br>   ◦ --REMOTE_RESULT_UI: The URL of the result published to Rational® Test Automation Server and can be opened in a browser to analyze the result.<br>   ◦ --LOCAL_RESULT: The path of the result saved locally.<br><br>For example, **-workspace** *workspace_full_path* **-project** *proj_rel_path* **-publish***publish_url* **-stdout** |
| **-swapdatasets** | Optional. Use this option to replace dataset values during a test . If a test is associated with a dataset, you can replace the dataset at run time while initiating the run from the command line.<br><br>You must ensure that both original and new datasets are in the same workspace and have the same column names. You must also include the path to the dataset when you run the -swap-datasets command.<br><br>For example, **-swapdatsets** */project_name/ds_path/ds_filename.csv:/project_name/ds_-path/new_ds_filename.csv*<br><br>You can swap multiple datasets that are saved in a different project by adding multiple paths to the dataset separated by a semicolon. |

| | |
|---|---|
| | For example, **-swapdatsets** */project_name1/ds_path/ds_filename.csv:/project_name1/ds_-path/new_ds_filename.csv;/project_name2/ds_path/ds_filename.csv:/project_name2/ds_-path/new_ds_filename.csv* |
| **-history** | Use this command when you want to view a record of all events that occurred during a *test* run. However, you must use the command suffixed with any of the following options:<br><br>    ◦ jaeger: To send test logs to the Jaeger UI during the *test* run.<br>    ◦ testlog: To send test logs as traditional test logs in during the *test* or *schedule* run.<br>    ◦ null: To send no test logs either to the Jaeger UI or during the *test* run.<br><br>For example:<br><br>`-workspace` *workspace_full_path* `-project` *proj_rel_path*<br>`-suite` *suite_rel_path* `-stdout` `-history` *comma delimited list of modes*<br><br>`-workspace` *C:/Users/IBM/rationalsdp/test_ws* `-project` *Project1*<br>`-suite` *test1.testsuite* `-stdout` `-history` *jaeger*<br><br>**Note:** You can add multiple options separated by a comma to send test logs during the *test* run to and the Jaeger UI.<br><br>For example:<br><br>`-workspace` *C:/Users/IBM/rationalsdp/test_ws* `-project` *Project1*<br>`-suite` *test1.testsuite* `-stdout` `-history` *jaeger, testlog*<br><br>For more information about how to view test logs in the Jaeger UI and , see related links. |

To stop the test run, you can open another command prompt window and use one of the following options with the cmdline option:

| Com-mand | Description |
|---|---|
| **-sto-prun** | Optional. Stops the test run after the specified number of seconds. The block is executed, and the test log is transferred before stopping the run. You must use the **-workspace** command and specify the location of the workspace. |
| **-aban-** | Optional. Stops the test run immediately. You must use the **-workspace** command and specify the lo-cation of the workspace. |

| Com-mand | Description |
| --- | --- |
| don-run | |

> **Note:** Messages are displayed to indicate when the test is launched and when it is completed unless you include the -quiet option.

**Exemple**

```
cmdline -workspace C:/RPTWorkspace -project testProj -eclipsehome C:\Program Files\__BRAND_NAME__\__SDP_PATH__
\eclipse.exe -schedule MySchedule -varfile C:/Assets/testProjVar.xml
```

## Initializing variables from Engineering Test Management

If you want to run an IBM® Rational® Performance Tester test from IBM® Rational® Quality Manager, you can pass the execution variables defined in Engineering Test Management to the Rational® Performance Tester test.

**Before you begin**

- Configure the Engineering Test Management adapter in Rational® Performance Tester. For more information, see the Configuration on page 90 topic.
- Variable names must be the same in Engineering Test Management and Rational® Performance Tester.
- The **Visible in** value for the variable in the Rational® Performance Tester test must be set to **All tests for this user**.

**About this task**

When you pass an execution variable to a Rational® Performance Tester test, the value initialized in the test is replaced by the value in the execution variable. If you modify the value that is initialized in the test, after the test is executed, the modified value is passed back to the execution variable in Engineering Test Management.

To initialize an execution variable value to a test, run the test from Engineering Test Management. For information about execution variables, see Using execution variables in manual test.

## Correlating response and request data

For a test to run correctly, a request that is sent to a server might need to use a value that was returned by a previous request. By ensuring that this data is correlated accurately, you can produce better performance tests.

## Data correlation overview

A request can include data that was returned in the response to a previous request. Associating data in this manner is called *data correlation*.

Video: Data correlation

Interactions with an application are typically related to each other. For example, consider the following interactions with a web-based application, in which each request depends on information returned from a previous response:

1. A payroll clerk types the web address for an application, which sends a login prompt. When the clerk logs in, the web server returns a page that indicates that login has succeeded and a unique session ID to the web browser that the clerk is using.
2. The clerk clicks a link on the returned page, which requests that the web server open the page for searching the employee database. The web browser includes the session ID when sending the request. Based on the session ID, the web server knows that the request comes from someone who is already logged on, and so opens the search form for the employee database. The clerk then searches for a specific employee. The web server returns a photograph of that employee and the employee's unique ID.
3. The clerk clicks a link that requests the web server to return the payroll record for the employee. With this request, the web browser sends two IDs:
   ◦ The session ID, so that the web server knows that the request comes from some who is logged on
   ◦ The employee ID, so that the web server can locate and return the correct information

In this example, request 2 depends on request 1, and request 3 depends on requests 1 and 2.

If you record these interactions in a test, before running the test with multiple users, you would vary the test data. For example, you would replace the user name and password values, the employee name search values, or both, with values that datasets contain. When you run the test, each virtual user returns a different employee payroll record, based on the contents of the datasets.

In a generated test, where data in a request depends on data that is contained in the response to a previous request, the request data is substituted from the response data on which it depends. The term for this internal linking of response and request data is *data correlation*. When you run a test with multiple users and varied data, data correlation is required to ensure that the test runs correctly.

A *reference* is a value in a test (typically in a response) that can be used by a subsequent value in the test (typically in a request). When the test generator detects that a request value must be substituted from a previous value, it designates the earlier value as a reference and correlates the subsequent request value with the reference. This process is called *automated data correlation*. You can also manually correlate any two values in a test or unlink existing correlations.

**Note:** You can change or disable automated data correlation. To do so, click **Window > Preferences**, expand **Test**, and then click **Test Generation**.

By default, the empty strings are not correlated because it might increase the time taken to generate a test. However, sometimes empty strings such as spouse name or middle initial become important to correlate. To correlate the empty strings, click **Window > Preferences > Test > Test Generation > HTTP Test Generation > Data Correlation** and select the **Create substitutions for empty strings** check box.

Generally, the HTML response content after the recording appears as *<input type="username" name="User" id="aaa" value="John"/>*. Some applications dynamically update the *name* attribute. Therefore, when you play back the test,

the HTML response content appears as *<input type="username" name="idt020" id="aaa" value="John"/>*. Because the *name* attribute changes dynamically, data correlation does not occur and the playback fails. For data correlation to correlate the response content based on the *ID* attribute, ensure that you have selected **ON** in the **Priortize correlation based on ID** option at **Window > Preferences > Test > Test Generation > HTTP Test Generation > Data correlation**.

To help you work with correlated data, the test editor uses color coding and provides navigational aids:

- When you click a page, you see a Test Data table for that page. By default, related dataset candidates are shown in green text on a light green background, values that are already associated with a dataset are shown in white text on a green background, and references are shown in blue text.



- If correlated data is not displayed, right-click the table and verify that **Show References** is selected. To navigate directly to a page request containing correlated data, double-click a table row. To associate correlated data from this table with a dataset, click the row, click **Substitute**, and then click **Select Data Source** to open the **Select Data Source** window. You can also use the **Test Data Sources** view to make substitutions. In the test editor, right-click the **Test Data** table, and then select **Link with Test Data Sources View**. When you click a row in the **Test Data** table, the **Test Data Sources** view displays information about the selected substitution site.
- When you expand a page, green text indicates page requests that contain dataset data or candidates. Blue text indicates page requests that contain references.

- When you click a highlighted request, dataset candidates are highlighted in light green, data that is associated with a dataset is highlighted in dark green, and correlated data is highlighted in red. If you right-click a value for correlated data, as shown in the example, you can then click **Go To** to see its reference:
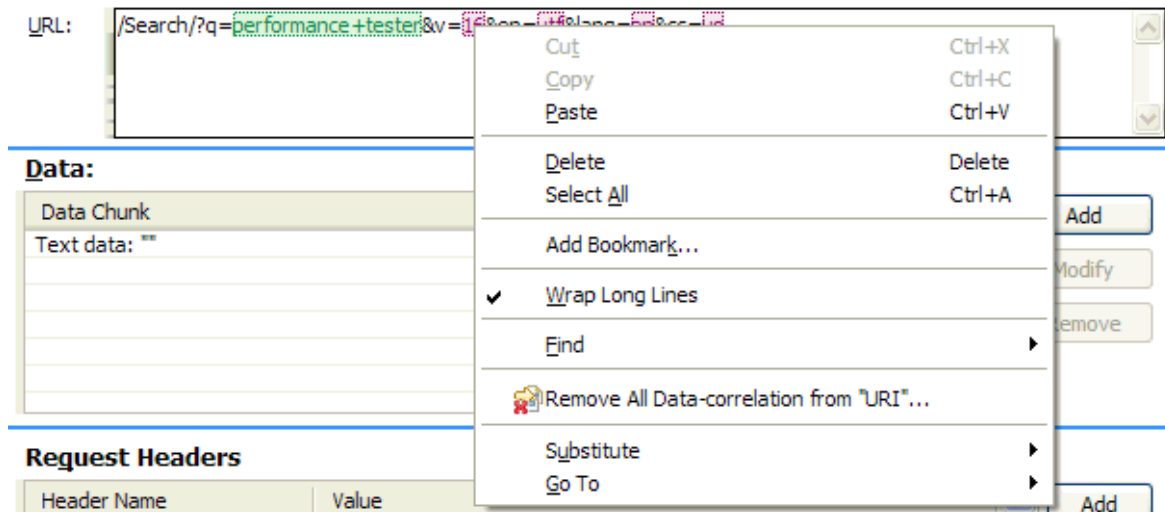


- References are highlighted in dark blue.

## Viewing data correlation

You can switch between viewing all test elements in the test editor and viewing only elements related to data correlation in the test editor. Viewing only data correlation elements makes it easier to add and remove substitutions.

1. In the Test Navigator, browse to the test, and double-click it. The test opens.
2. In the **Test Contents** area, click **Options**.
3. Click **Show  > Data Correlation**.
   **Result**
   The test editor window displays only elements that are related to data correlation. Alternately, click **View** under **Test Contents** to switch between **Display all Test Contents** and **Show Substitutions**.
4. Select a single test element in the **Test Contents** area to see the current data source and to remove or change the substitution in the **Test Element Details** area. Select multiple elements in the **Test Contents** area to see the data in tabular form in the **Test Element Details** area. Different controls are available depending on the type and number of elements that you select in the **Test Contents** area.
5. **Optional:** In the Test Elements Details area, click **Substitute > Select Data Source** to open the **Select Data Source** window, where you can specify the data source for the selected substitution site.

**What to do next**
To view all test elements, click **Options > Show > Data Correlation** again.

> **Note:** If you select a test element while viewing all test contents, and then switch to viewing only data correlation elements, then the corresponding substituters and dataset candidates are selected. For example,

if you select an HTTP page in the test editor, and then switch to viewing only data correlation elements, then all substituters and dataset candidates for all requests from the HTTP page are selected.

## Test data sources overview

Use the **Test Data Sources** window to add or change data correlation for any supported test element.

The test generator attempts to perform automated data correlation. You can also manually correlate request values with other test data. The **Test Data Sources** window displays the following types of test data that you can substitute from:

- References
- Datasets
- Custom code
- Test variables
- Built-in datasources

You can right-click references, custom code, and built-in data sources to display a menu that contains commonly used commands. In addition, you can use the toolbar and menu at the top of the **Test Data Sources** window to complete common operations, such as creating a dataset or switching from tree view to list view. The **Substitute From** push button is enabled when you select a substitution site in the test editor and a data source from one of the five categories in the **Test Data Sources** window. Click **Substitute From** to correlate the data source and the substitution site.
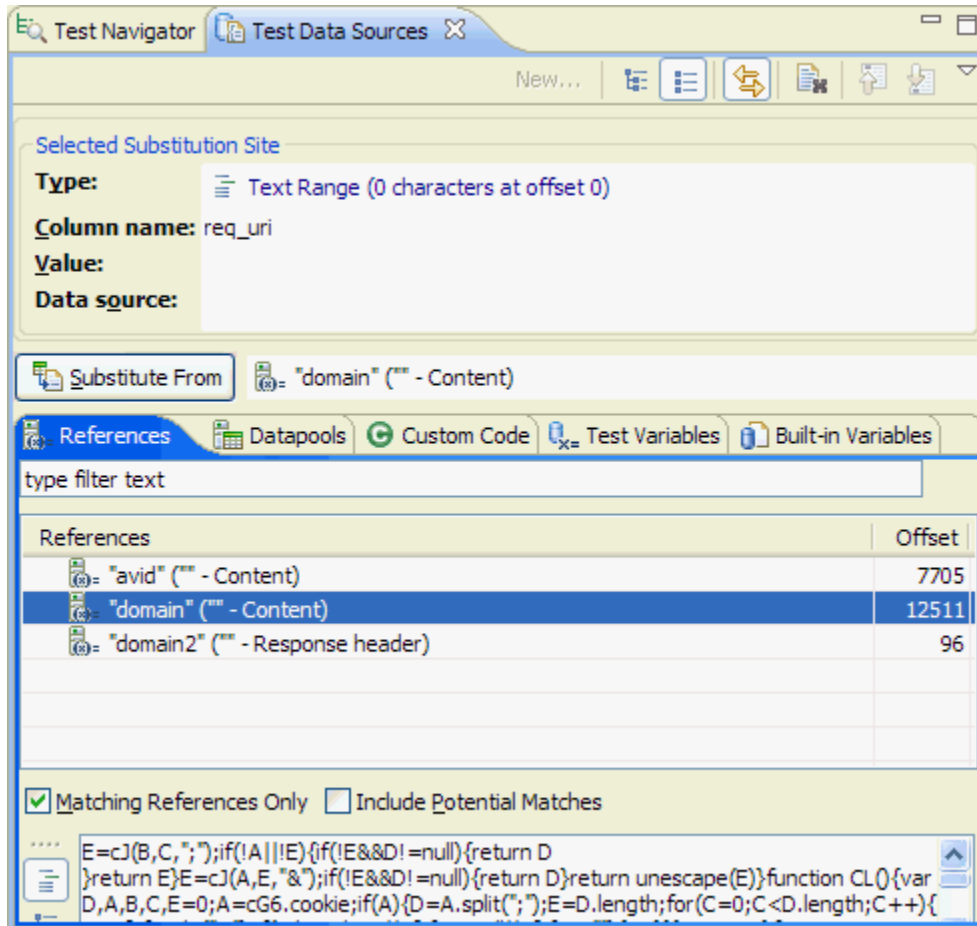
**References**

> The data sources that have been created in the test and the potential matches that are available for a selected substitution site. For example, text that is found in a response and used in a subsequent request is automatically created as a reference. Clear the **Matching References Only** check box to see all the references that occur before the substitution site in the test. Select **Include Potential Matches** to see a list of locations that might match the substitution site. Replace the `type filter text` string with keywords to filter the list of references. Select a reference in the **References** table to display the reference in the pane at the bottom of the window. Use the controls to the left of the preview pane to switch between inline view and tree view.

**Datasets**

> The datasets that have been added to the test. To add a new dataset, click the **Add new Dataset** icon ( ).

**Custom Code**

> The custom code that is available as data sources for this substitution site, if you have written Test Execution Services (TES) custom code.

**Test Variables**

> The test variables and their types that are available as data sources for this substitution site. Replace the `type filter text` string with keywords to filter the list of test variables.

**Built-in Datasources**

> The dynamically calculated data sources, such as **Current Date**, **Random Number**, **Sequential Number** and **Timestamp**, that are available as data sources for this substitution site. To create a built-in data source, right-click inside the **Built-in Datasources** page, and select **New**, or click the **New** push button at the top of the window.
>
> To assign a unique value for every substitution, when creating the built-in data sources, select the **Get New Value Each Time Used** check box.

### Detaching the Test Data Sources window

The **Test Data Sources** window is detachable. To detach the **Test Data Sources** window, right-click the **Test Data Sources** tab, and then select **Detached**. When detached, the **Test Data Sources** window is always displayed in front of the other windows that make up the workbench.

## Correlating a request value with test data

If a test runs without error but does not generate the results that you expect, you might need to correlate a value in a request with other test data.

### About this task

You can correlate a request value with the following types of test data:

- References
- Datasets
- Custom code
- Test variables
- Built-in datasources

For example, if you recorded a test and searched on a date, you might want to substitute the built-in data source *Current Date* so that the test will search on the playback date, not on the recorded date. For information on the different types of test data, see .

1. In the **Test Navigator**, browse to the test, and double-click it. The test opens in the test editor.
2. Locate the value that the other test data will replace.
3. Highlight the value: Press and hold the left mouse button and drag your mouse over the value.
4. In the **Test Data Sources** window, click the appropriate tab: **References**, **Datasets**, **Custom Code**, **Test Variables**, or **Built-in Datasources**. To see all references, clear the **Matching Only** check box.
5. In the **Test Data Sources** window, select the test data to use. For references and custom code, you can double-click the entry in the **Test Data Sources** window to find the data source in the test editor.
6. Click **Substitute From**.

**Result**

The value is shown in purple text to indicate that it has been correlated and the correlation is added to the Test Data table, which contains the substitution sites for the page.

## Substituting request content with file contents

You can substitute the content portion of a protocol request with the contents of a file. This feature is only supported in certain sections of a test, depending on the protocol. For example, HTTP tests support file content substitution only in the POST data section of a request. SOA tests support file content substitution for MIME and DIME attachments, XML node values and fragments, and text content. File content substitution works in the same manner as other substitutions. All standard test data sources such as test variables, datasets, and references can be used. The data from the source is treated as a full path to a file. The file is opened, its contents are read, and then those contents are used in the substitution.

**Before you begin**

You must copy the files that contain the substitution content to the agent computers. You must record a test with locations for substitution from a file. For example, record an HTTP test that contains multipart MIME data in a POST request.

1. Create a data source that contains the full path to the file from which you want to substitute content. Specify an absolute path to the file. Use path separator characters appropriate to the operating system of the agent computer running the test. Optionally, specify a character set to use in reading the file. The existence of the file is not validated. If the file cannot be opened when the test runs, a message is written to the test log. If you use path separator characters that are not appropriate for the operating system of the agent computer, the substitution cannot be completed. For example, if you use a path of `D:\DataFiles\file1` on an agent computer running Linux™, the substitution cannot be completed, because Linux™ uses forward slashes as path separator characters.
2. In the test editor, navigate to the request where you want to substitute content, and then select the request data that you want to substitute.
3. Right-click, and then select **Substitute > Select Data Source**.
4. Select the data source that contains the path to the file from which you want to substitute.
5. Right-click the substitution site, and then select **File Contents Substituter**.

**Results**

When the test runs, the content in the protocol request is substituted with the specified file contents.

HTTP POST data is displayed in the test editor in chunks. You can create a file contents substitution in the POST data of an HTTP POST request by selecting the data chunk that you want to correlate, and then clicking **Substitute**. The test data source that you select is automatically treated as a file contents substituter. The entire data chunk is replaced with the contents of the file when the test runs, even if only a portion of the text in a text data chunk is selected by the substituter.

## Built-in Datasources

You can use built-in data sources instead of creating custom data sources to substitute the recorded values. The Built-in Datasources section in the Test editor displays the data sources that have been used and unused. You can modify their properties from a location.

The built-in data sources are Current Date, Random Number, Sequential Number, and Timestamp. The values of these data sources are dynamically calculated and submitted to the test.

## Assigning random numbers to users or clients

To assign unique random numbers to all the virtual users in a test, you can create a random number data source in the workbench. This data source will generate unique integers or floating point numbers for the users or clients. You can choose to distribute the numbers of the virtual users or clients in a uniform, normal, or negative exponential way.

1. In the Test editor, select the root node (name of the test) and from the Test Details area, select **Built-in Datasources**.
2. Select the **Random Number** data source and click **Add Built-in Datasource**.
3. Type a name for the data source.
4. Select how do you want to generate the numbers:
   - **Uniform**: Click this option to generate random numbers with a uniform distribution. Specify the minimum and maximum values for the generated numbers.
   - **Normal**: Click this option to generate random numbers with a normal or Gaussian distribution. Specify the average and the standard deviation for the generated numbers.
   - **Negative Exponential**: Click this option to generate random numbers with an exponential distribution. Specify the average for the generated numbers.
5. Select how do you want to format the numbers. You can select **Common** to format the numbers in decimal or scientific notations.
   You can choose **Custom** to specify a **Format mask** by using the standard Java formatting syntax. The changes that you specify can be previewed in **Formatted output**.
6. To substitute the built-in data source every time with a new value for the requests, select the **Get new value each time used** check box.

## Assigning sequential numbers to users

To assign unique sequential numbers to all the virtual users in a test, you can create a sequential number data source in the workbench. This data source will generate unique integers or floating point numbers for the users.

**Before you begin**
You can use the sequential built-in data source option wherever data correlation substitutions are permitted, such as Transactions or Delays.

**About this task**
You define an initial value that should be assigned to the first virtual user and a step value that is a number by which the current value increments after each retrieval by a virtual user. If initial value is 1 and step value is 5, the workbench

generates numbers in the sequence of 1, 5, 10, 15, and so on and each number is mapped to a virtual user. If a test is run on multiple agent machines, the workbench assigns a sequential value to all the users in all the agent machines.

You can also assign a full sequence of numbers of one virtual user. The sequence number increments in the request for each time the request in the multi-request generator is executed.

1. In the Test Contents area of the test, click an element in the test where data correlation substitution is permitted, such as a transaction name or delays.
2. In the Test Element Details area, right-click the name of the element and click **Substitue > Built-in Datasources**.
3. In **Built-in Datasource Selection Wizard**, click **Sequential Number** and click **Next**.
4. Assign a name for the data source.
5. In **Initial Value**, type a number to be assigned to the first virtual user.
6. In **Step Value**, type a number.
7. In **Formatting Options**, you can format the number in the manner you want to use.
8. **Optional:** To assign a full sequence of numbers to one virtual user, select the **Execute for individual user** check box.
9. **Optional:** To substitute the built-in data source every time with a new value for the requests, select the **Get new value each time used** check box.
10. Click **Finish**.
11. Save and run the test.

**Results**

After you add the test to a schedule and run the schedule, the test log displays each element name where the data source is applied with the unique sequential number assigned to the virtual user.

## Creating a reference or field reference

When you designate a test value as a reference or designate a set of test data as a field reference, you can use the data elsewhere in the test.

**About this task**

A *reference*, which is typically located in response data, points to a specific value that you want to use in a subsequent test location, typically a request. You can substitute a request value with a reference. This substitution is called *data correlation*. You can also use a reference as input to an IF-THEN condition in a test or as input to custom Java™ code that your test calls.

A *field reference* points to an entire block of test data. For example, an entire HTTP response can be designated as a field reference. You can use a field reference as input to custom Java™ code that your test calls.

1. In the Test Navigator, browse to the test, and double-click it. The test opens.
2. Locate the value or set of data to designate as a reference or field reference.

Different protocols support different references. For HTTP tests, you can create references and field references in these fields:

   ◦ A response header value, the Value column of a Response Headers table

   ◦ Response content, the Content field

For HTTP responses, you can create field references in these fields:

   ◦ The Status field

   ◦ The Reason field

3. Create the reference:

   a. For response contents, highlight the value. For response header contents, click the row in the Response Headers table, and then click **Modify**.

   b. Right-click, and then click **Create Reference**.
   **Result**
   The value is highlighted in light blue to indicate that it is an unused reference. When you use it, the highlight changes to dark blue. The reference is given a name automatically. To see the name of the reference, right-click the value, and then select **Properties**. To edit the regular expression that is used to locate the reference, click the **Toggle regular expression assistant** push button on the **Properties** window. The regular expression assistant displays the response content matched by the regular expression and the groups captured by the regular expression. To ensure that the details about the reference is always logged, select a reference and click **Properties**, and then click the **Always log details** checkbox. To create a reference that will be used by the HTTP secondary request, you must select **All occurrences**. You can also match the reference within a given range of all the occurrences.

   **Note:**

   A reference that is created to be used by the HTTP secondary request cannot be used by custom code or other data sources.

   You can always log the details of Substituters, Data Sources, and Requests.

4. To create a field reference, do not highlight the value. Instead, right-click the value, and then click **Create Field Reference**.

   a. Field references are not automatically given names. To name a field reference, right-click the field reference, and then select **Properties**. Type a name in the **Name** field, and then click **OK**.

   **Result**
   The entire field is highlighted in yellow to indicate that it is a field reference.

## Selecting a reference in a response

When a response contains multiple matches for the regular expression that defines a reference, you can select which match is used subsequently as the data source. You can specify a particular occurrence, or you can specify a random occurrence.

**About this task**

An application under test might return responses that contain multiple matches for a regular expression that defines a reference. For example, a response might contain multiple links to rows of data, where each row represents a different user. You can control which occurrence of the regular expression is used as the data source in subsequent data correlation.

If you edit the **Regular Expression** that is associated with a reference, and then click **Verify** or **OK**, and the new regular expression still connects to the highlighted string in the preview window, then the **Specific occurrence number** is updated automatically, overwriting any changes.

1. In the Test Navigator, browse to the test, and double-click it. The test opens.
2. Locate the response that contains the reference that you want to specify.
3. In the **Content** field under **Test Element Details**, right-click the reference, and then select **Properties**.
4. **Optional:** To edit the regular expression that is used to locate the reference, click the **Toggle regular expression assistant** push button on the **Properties** window. The regular expression assistant displays the response content that is matched by the regular expression and the groups that are captured by the regular expression.
5. On the **Properties** page for the reference, select which **Occurrence** to use as the data source. By default, the first occurrence of a match for the **Regular Expression** is used as the data source.
   **Choose from:**
   - To specify a particular occurrence, select **Specific occurrence number**, and then type the number of the match. For example, type 4 to specify the fourth match of the regular expression in the response.
   - To specify a random occurrence, select **Random occurrence**.
   - To specify the last occurrence, select **Last occurrence**.
6. Click **OK**.
   **Result**
   The occurrence that you specified is used as the data source for data correlation when you run the test.

## Correlating multiple fields in a test

Some tests are structured in such a way that you must correlate data for multiple fields.For example, assume that you plan to dataset an item that a virtual user is buying. For the test flow to be correct, you must also dataset all occurrences of that item in the test. You can find and correlate all instances of that item in one procedure. Typically, you use **Find More and Substitute** in the **Show Dataset Candidates** window to correlate data for multiple fields. See Viewing dataset candidates when you open a test on page 216. Alternatively, you can use the **Test Search** page to correlate data for multiple fields.

To find all instances of a field in a test and correlate some or all of the instances with a data source, such as a dataset:

1. In the Test Navigator, browse to the test, and double-click the test. The test opens.
2. Locate the item or the substitution site to change or create a reference for. If the item is plain text, select the item. If the item is an existing reference, click the highlighted area.
3. Right-click, and then click **Find > More Substitution Sites**.

4. Click **OK**.

5. On the **Test Search** page, select **Case sensitive** to perform a case-sensitive search or **Regular expression** to perform a search using regular expressions. In regular expression mode, press Ctrl+spacebar key in **Search for text** for content assistance. Content assistance lists the regular expression patterns and the content that they match.

6. Click **More Options**, and then select the appropriate options:

    **Restrict to elements highlighted in Test Contents**

    Search only in elements that are selected in the **Test Contents** area.

    **Highlight found elements in Test Contents**

    Highlight found elements in the **Test Contents** area.

    **Recursive**

    Searches the child test elements in addition to the element. For example, if you search an HTTP page, select this option to search the requests and responses within the page.

    **Match encoded and decoded values (protocol-specific)**

    When selected, searches for matches of the unencoded and URL-encoded versions of the specified text. For example, when searching in HTTP data, `abc%123` and `abc%25123` match.

    **Include matches with overlapping data correlation**

    Include sites that are contained in, or overlap with, an existing substitution site. If you decide to substitute, the conflicting substitutions are automatically removed.

    **Include matching substituters**

    Click to return elements that originally matched the search string but have since been substituted. Clear to skip existing substitution sites when results are returned.

7. Click **Close**.

8. Click **Search**. The search results are displayed in the **Search** view.

9. In the **Search** view, select the matches to substitute, and then right-click the selection.

10. Optional: To select all matches, right-click the test name.

11. Click **Substitute in DataSource View**.

    **Result**

    This action sends the selected matches to the **Test Data Sources** window.

12. In the **Test Data Sources** window, click the tab that corresponds to the type of data source to use:

| Option | Description |
|---|---|
| **References** | The data sources that have already been created in the test and the possible matches that are available as data sources for the selected substitution site. For example, text that is found in a response and used in a subsequent request is automatically created as a reference. Clear the **Matching References Only** check |

| Option | Description |
|---|---|
| | box to see all the references that occur before the substitution site in the test. |
| **Datasets** | The datasets that have been added to the test. To add a new dataset, click the **Add new Dataset** icon ( ). |
| **Custom Code** | If you have written test execution services (TES) custom code, the custom code that is available as data sources for this substitution site. |
| **Test Variables** | The test variables and their types that are available as data sources for this substitution site. |
| **Built-in Datasources** | The dynamically calculated data sources (**Current Date**, **Random Number**, **Sequential Number**, and **Timestamp**) that are available for this substitution site. To create a new built-in data source, right-click inside the **Built-in Datasources** page, and select **New**. |

13. Select the data source, and click **Substitute From**.
    **Result**
    The **Substitute Multiple Items** window is displayed, showing information about the data source and substitutions sites that you selected.
14. For each site with a selected check box, click **Substitute Checked** to substitute the data source or clear the check box to skip the site.
    Click **Always Prompt** to examine every substitution site one at a time. Click **Prompt on overlapping data correlations** to examine a site only if the site you are substituting into is contained in, or overlaps with, an another substitution site. If you decide to substitute, the conflicting substitutions are automatically removed.

**Results**

The selected instances of the field are correlated with the data from the data source.

## Guidelines for adjusting data correlation

When you run a test, you might notice that the server is not under the expected load or that your database is not being updated as expected. Incomplete or incorrect data correlation can cause these problems.

To identify data correlation problems:

1. Use the **Potential Correlation Errors** view to find missing or incorrect data correlations. See Finding data correlation errors on page 253 for more information.
2. Run a test individually or in a schedule with the **Log Level** for errors, failures, and warnings set to **All**.
3. After the run, open the test log as explained in Viewing the test logs on page 354.

The data correlation algorithms that are used during test generation are based on well known best practices. However, because these practices continually evolve, various types of errors can occur during automated data correlation:

- **Insufficient correlation:** Test values that must be correlated are not. Some possible causes follow:
    - Two parameters that must be correlated have different names.
    - A value must be correlated with a previous value that does not occur in the expected location.
    - A parameter or value must be correlated with a previous parameter or value that does not occur in the test because it is a computed value.
- **Superfluous correlation:** Unrelated test values are correlated.
- **Incorrect correlation:** Test values that must be correlated are correlated incorrectly.

## Insufficient correlation: Parameters have different names or occur in unexpected locations

When two parameters that must be correlated have different names, automated data correlation does not recognize that the two parameters are related. For example, consider this request: http://www.example.com?id=12345. Suppose that this request must be correlated with the server response that contains `customer_ID=12345`, not `ID=12345`. In this case, the `ID` parameter must be correlated with `customer_ID`.

Data correlation typically links a response value that was returned from the server with a subsequent request value. The automated correlation algorithms search in the URL and the POST data for potential matches; however, other schemes for returning parameters are possible. For example, consider this request: http://www.example.com?id=12345. Suppose that this request must be correlated with the server response that contains the name and entity pair `href name="customer_ID" entity="12345"`, not `ID=12345`. In this case, the `ID` parameter must be correlated with `name="customer_ID"` and value `12345` must be correlated with `entity="12345"`.

Here are some additional causes of insufficient correlation:

- Siebel uses the star array format. Standard correlation algorithms can neither retrieve from this format nor substitute into this format.
- SOAP designates correlation parameters in external XML files. The correlation algorithms cannot correlate parameters in the external file with parameters in the test.

To manually correlate data in these cases:

1. In the test editor, use search or browse to locate the two parameters for correlation.
2. Navigate to the parameter that occurs later in the test, and select the parameter. This is the substitution site.
3. In the **Test Data Sources** window, click the **References** tab.
4. Select the data source to use as a reference, and then click **Substitute From**.

## Insufficient correlation: One parameter is unnamed

Sometimes a parameter or value must be correlated with a previous parameter or value that is not named in the test, because it is computed, for example, by a JavaScript™ program. In this case, in order to correctly correlate the data,

you must understand how and where the parameter or value is computed, and then use a custom code block. See Extending test execution with custom code on page 304 for more information about custom code.

For example, consider the web address http://www.example.com?login_stamp=12345_Apr_11_07, where the value for `login_timestamp` is the concatenation of the login ID and the current date. In this case, you must generate a custom code that concatenates the login ID and the date.

For another example, suppose that the server returned the login ID and date as separate entities: `href "customer_id=12345" Date="Apr_11_07"`. In this case, you can put these parameters in separate references and, in subsequent requests that use customer ID and date, substitute them separately.

### Superfluous correlation

Automated data correlation is based on pattern matching: A parameter or parameter value is correlated with a subsequent parameter or parameter value with an exact or similar name. But sometimes parameters with exact or similar names are in fact unrelated. In the best case, unneeded correlation is either harmless or adds a slight load that is inappropriate. In the worst case, the application does not expect a correlation and fails during playback.

To remove a superfluous data correlation:

1. In the test editor, search or browse to locate the substitution site that must not be correlated. By default, purple letters indicate correlated data.
2. Right-click the substitution site.
3. Click **Remove Substitution**.

### Incorrect correlation

A parameter that requires data correlation might occur many times throughout a test. For example, a session ID parameter that is used initially when a user logs in might also be used in every subsequent request. If multiple instances of a parameter in a test are not same, the correlation algorithms might use the wrong instance.

With the HTTP Test Generation preferences, you can optimize automatic data correlation for accuracy or for efficiency.

- **Accuracy**: Each occurrence of a parameter is correlated with the nearest previous occurrence. This is the default setting.
- **Efficiency**: Each occurrence of a parameter is correlated with a single previous occurrence.

**Note:** If you do not manually apply a correlation in the Referer field in an HTTP request header, then the Referer field is automatically correlated as needed. If you manually apply a correlation in the Referer field in an HTTP request header, then no automatic correlation is performed.

Incorrect correlations are more likely to happen when **Optimize automatic data correlation for execution** is set to **Efficiency**. To fix an incorrect correlation:

1. In the test editor, search or browse to locate the value that is incorrectly correlated.
2. Right-click the substitution site.
3. Click **Remove Substitution**.
4. Right-click the substitution site again.
5. Click **Substitute**, and select the correct parameter.

Generally, the HTML response content after the recording would look like `<input type="username" name="User" id="aaa" value="John"/>`. Some applications dynamically update the name attribute. So, when you play back the test the HTML response content would look like `<input type="username" name="idt020" id="aaa" value="John"/>`. Because the name attribute is changing dynamically, data correlation would not occur and the playback would fail.

Such correlations are the result of the tool using the *name* attribute as the basis for correlating other attributes in the response code instead of the *ID*. To correlate the responses based on ID, select **ON** in **Window > Preferences > Test > Test Generation > HTTP Test Generation > Data correlation types > Prioritize correlation based on ID**.

## Finding data correlation errors

You can use the **Potential Correlation Errors** view to find missing or incorrect data correlations.

**Before you begin**
Run a test or a single-user schedule. The **Potential Correlation Errors** view does not support multiple-user schedules. If verification points fail while you are running a test, you are prompted to open the **Potential Correlation Errors** view when the test run is complete.

To find data correlation errors:

1. In the **Test Navigator**, select the result of the test run where you want to find correlation errors.
2. Right-click the result, and then select **Find Data Correlation Errors**. You can choose **Missing Correlation**, **Incorrect Correlation**, or **All**.
3. The **Potential Correlation Errors** view opens.
   After the test log is processed, the view is populated. Depending on the size of the test log, it can take significant time to populate the view. The potential missing or incorrect data correlations are displayed, in descending order of the likelihood that the correlation is incorrect. Selecting an item in the **Potential Correlation Errors** view automatically selects the corresponding element in the test editor, so that you can fix the potential error.
4. Use the **Compare with Test Log** toolbar button in the upper-right corner of the view to compare the request or response in the test with the same object in the test log.
5. For missing correlations, use the **Suggest Fix** toolbar button in the upper-right corner of the view to search for other instances of the value in all responses in the test. If a matching value occurs in an earlier response in the test, create a reference in that response.

## Disabling data correlation

You can disable a data correlation source or a substitution site. When you disable a data source, none of the substitution sites that use the source will be correlated when you run tests. When you disable a substitution site, only

that specific substitution site is disabled. Other substitution sites that use the same reference will be correlated when you run tests. You can also disable data correlation entirely for subsequent tests that you record.

To disable a data correlation source or substitution site:

1. In the **Test Navigator**, browse to the test and double-click it. The test displays in the test editor.
2. In the **Test Contents** area, click a request.
3. In the **Test Element Details** area, locate the data correlation source or substitution site.
4. Right-click the data value and select **Disable** from the menu.
   To re-enable a disabled data source or substitution site, right-click the data value and select **Enable** from the menu.

**Results**

The data correlation source or substitution site is disabled.

> **Note:** To disable data correlation for the entire workspace, click **Window > Preferences >  Test Generation**, and clear **Enable automatic data correlation**. Subsequent tests that you record or regenerate will not include data correlation.

## Recorrelating test data

If you disabled automatic data correlation before recording a test, you can regenerate the test with automatic data correlation enabled.

1. Click **Window > Preferences > Test > Test Generation**.
   **Result**

   The **Test Generation** preferences window opens.
2. Click the **Data Correlation** tab.
3. Select the types of data correlation to enable, and then click **OK**.
4. In the **Test Navigator**, browse to the test and double-click it. The test displays in the test editor.
5. Click **Edit > Re-correlate test data**.

**Results**

The test is regenerated with the types of automatic data correlation that you selected.

# Simulating services with stubs

Service stubs enable you to simulate the behavior of an actual service for a wide variety testing or integration purposes.

## Service stub overview

Service stubs are simulations of an actual service, which can be used to functionally replace the service in a test environment. A stub server replaces the actual application server.

From the point of view of the client application, the service stub looks identical to the actual service that it simulates. To use a service stub in replacement of the actual service, you must be able to replace the URL of the original service in the client application with the URL of the stub server.

> ⚠️ **Important:** For version 8.7 and later, you cannot use the schedule option of IBM® Rational® Performance Tester to deploy stub servers remotely. If you have already deployed stub servers remotely, you must install Rational® Performance Tester on those computers and then deploy the stub servers locally.

### Use case examples

There are several cases where it can be useful to deploy a stub services instead of using the actual services for your tests:

- If you are testing a local service that uses data from another remote service, you might need to inject specific content to the service under test from the remote service. You can simulate the remote service with a service stub to ensure that the local service responds properly to some specific input.
- Some commercial services charge users for each call. If you are testing such a service, you can develop and debug your test against a stub service, which is based on the WSDL of the actual service, without being charged by the commercial service.
- During integration of a large application involving multiple clients and services, some services might not yet be operational, although their WSDL specifications are available. You can simulate the missing services with service stubs, which will allow you to proceed with the integration work.

### Service stub architecture

You create a service stub by providing an existing WSDL specification. The service stub is generated with the exact same ports and bindings as the original service so that it can be addressed with exactly the same interface. Each operation in the service returns a default response of the type defined by the WSDL.

You can edit the service stub in the stub editor to change the default response or to create conditional responses that simulate the actual responses of the original service.

When you have finished editing the service stub, you can deploy it on a local stub server, which runs in the workbench. The stub server simulates an actual application server and can host multiple service stubs. You control the stub server from the stub monitor view.

Finally, to use the service stub instead of the original service, you change the URL used by the client application to point to the local stub server instead of the original application server. This URL, as well as the WSDL of the service stub, is provided in the stub monitor view.

## Creating a service stub

You can use a WSDL (Web Service Description Language) specification file to generate a service stub that can simulate the behavior of the original service and uses the exact same interface.

**Before you begin**

Service stubs are stored in test projects. If your workspace does not contain a test project, the test creation wizard creates one, enabling you to change its name. To store a service stub in a specific project, verify that the project exists before you create the stub.

If you are using Secure Sockets Layer (SSL) authentication, ensure that you have any required key files in your workspace.

The wizard can import WSDL files from the workspace, the file system, a remote repository, or from a URL. Ensure that the WSDL files use the correct syntax for the test environment. Service stub generation might not work with some Web Services Description Language (WSDL) files.

1. In the workbench, click **File > New > Other > Test > Test Assets > Service Test** or click the **New Service Stub** toolbar button.
2. Select the WSDL of the service that you want to simulate. If necessary, you can import the WSDL from the file system, a URL, or a WSRR or UDDI repository.
3. Click **Next**.
4. Select a project location and a name for the new service stub. Click **Finish**.

**Results**

The wizard generates a working service stub that reproduces the interface of the original service as defined in the WSDL specification. Each operation is reproduced with a default response. You can edit the service stub with the stub editor to change the default response or to create conditional responses.

## Editing a service stub

Service stubs are generated with a single default response for each operation in the WSDL specification. You can edit the service stub to change the default responses or to add conditional responses that can simulate the actual service.

To edit the behavior of a service stub:

1. In the test navigator, double-click the stub to open the stub editor.
   Each operation simulated by the stub is represented by an operation element, which contains **Case** elements that describe a condition. Each case contains a response element. Case elements are similar to test verification points and use the same presentation.
2. To change the default response of an operation:

   a. Expand the operation and the **Case : Default** element, and then select the response element.
      The Case : Default element describes the response of the service stub when no other case condition is met.

   b. Edit the **Message** content to specify the XML content returned by the service stub.

3. To add a conditional response case:

    a. Right-click the operation and select **Add > Equals Case**, **Contains Case**, or **Query Case**.

      These conditional case types are similar to the *Equals*, *Contain* and *Query* verification points in service tests.

- Use **Equal Case** to specify a response that is returned by the stub when the entire incoming message content fully matches the specified message content.
- Use **Contains Case** to specify a response that is returned by the service stub when a portion of the incoming message content matches the specified message content.
- Use **Query Case** to specify a response that is returned by the service stub when an XPath query meets the specified criteria.

      You can add as many case elements as necessary to simulate the behavior of the original service. Use the **Up** and **Down** buttons to change the order in which the case conditions are evaluated. Only the first matching condition is executed.

      The default case cannot be removed and is always the last case element in the operation.

    b. Select the response element and edit the **Message** content to specify the XML content returned by the service stub. Use the **Form**, **Tree**, and **Source** views to change the XML content display mode.

4. Select **File > Save** or click the **Save** toolbar button.

**What to do next**

When you have finished editing the service stub, you can deploy the stub to a stub server.

## Deploying service stubs

You deploy and run service stubs on a stub server, which is a small application server dedicated to running service stubs. The client application, or test, addresses the stub server instead of the actual application of the original service.

**Before you begin**

The local stub server runs in the workbench on the local computer. Service stubs can be accessed locally. The local stub server is automatically stopped when you close the workbench.

To use a service stub instead of the original service, you must be able to change the endpoint of the client application or service test to replace the URL of the original application with the URL of the stub server.

1. In the stub editor, click the **Deploy** button.
   Alternatively, you can right-click the stub in the test navigator and select **Deploy On > Local stub server**
   **Result**
   This opens the **Stub Monitor** view.
2. In the **Stub Monitor** view, click ▶**Run**.
   If you make any changes to the service stub, the stub is redeployed to the stub server after saving.
3. To add more service stubs to the stub server, click **Add** and select a service stub from the workspace.

4. Copy the URL of the service stub from the **Stub Monitor** view and paste it into the configuration of the client application.
   You can also directly access the WSDL specification of the service stub, which is a copy of the original WSDL with replaced URL endpoints.

**What to do next**

You can validate that the service stub is responding correctly by using the generic service client to invoke a call.

## Recording service stub activity in a log file

With service stub logging, you can monitor the interactions between an application and the stub server. When the option is enabled, one log file is created for each deployed stub. The log files are presented as a formatted HTML report.

**Before you begin**

You must have created one or several service stubs.

To log service stub activity:

1. Add the following virtual machine (VM) argument to the `eclipse.ini` file: `-DSTUB_LOG_LEVEL=log_level`.

   Use one of the following values for the *log_level* variable:
     - `0`: Disable the log.
     - `1`: Log stub activity without details.
     - `2`: Log stub activity including content of sent and received messages.
     - `3`: Same as level 2 with HTTP headers of received messages.
     - `4`: Same as level 3 with attached files.
   You can also add the following optional arguments:
     - `-DSTUB_LOG_KEEP_PREVIOUS=true`: This option creates a separate log file each time the service stub is redeployed. If the value is not `true` or if the option is not present, the log file is erased if the service stub is redeployed or when the stub server is stopped.
     - `-DSTUB_LOG_SERIALIZE_XML=true`: This option displays the XML content (with log levels 2, 3, and 4) without formatting or indentation. If the value is not `true` or if the option is not present, the XML content is formatted and indented in the log.

   The `eclipse.ini` file is located in the same directory as the `eclipse.exe` launcher binary file that is used to run the product.

   **Example**

   For example, to enable logging with basic content, add the following line to the end of the `eclipse.ini` file: `-DSTUB_LOG_LEVEL=2`.
2. Restart the workbench, and in the **Stub Monitor** window, click the **Run** icon ▶ to restart the stub servers.
3. If the server was launched by a schedule in the performance testing application, then corresponding logs are automatically created in the workspace. If not, complete the following steps to retrieve the log files from the stub server:

⚠️ **Important:** The stub server must be running.

    a. After running your tests, to view the service stub log files, open the **Stub Monitor**, and click the tab for the stub server.

    b. Click the **Synchronize** toolbar button for the selected server.
       **Result**
       An HTML log file is created and displayed for each deployed service stub.

**Result**
The stub log reports are located in a folder named `stubLogs`, which is in the same folder as the corresponding service stub.

## Setting log level for service stubs

While recording a service test, you can set the level of the log details that you want to collect for debugging purposes.

**Before you begin**
You must stop the stub server.

**About this task**
The log level that you set in this way takes precedence over the log level setting that you specify in the eclipse.ini on page 258 file.

1. In the **Stub Monitor** view, in the Service Stubs section, click the **Edit log options** icon 🗿.
2. Select one of the log level options and click **OK**.

**What to do next**
Start the server again for the changes to take effect.

# Sending service requests with the generic service client

The generic service client enables you to send requests to services for which you do not have a convenient client and to view the responses returned by the service.

## Creating transport protocol configurations

Read these topics to configure various transport protocols.

## Creating an HTTP transport configuration

You can create an HTTP transport configuration that describes the transport settings for a service request. Transport and security settings can be associated with any service request.

**Before you begin**

If you are using Secure Sockets Layer (SSL) authentication, ensure that you have valid key files in your workspace.

If you are using SOAP security, ensure that you have configured the environment with the correct libraries and configuration files.

1. Click the **Generic service client**  toolbar button to open the generic service client and click the **Transport** tab.
   **Result**
   This opens the **Transport Configurations** page.
2. On the **Transport Configurations** page, click **Create an HTTP configuration** () to create a new HTTP transport configuration.
3. Type a **Name** for the new transport configuration.
4. Specify the following options for the HTTP transport:

   **HTTP/2**

   >  **Note:** Testing HTTP/2 service is in the Beta mode. For more information, see Preparing to record a HTTP/2 service on page 130.

   To test a service that uses the HTTP/2 protocol, select the **Activate** check box. This check box is automatically selected when you record a service by using a browser. If you use the Generic Service Client component to create a HTTP/2 test, you have to manually select the check box.

   **HTTP/2 client connection timeout**

   Specifies the time limit for the HTTP/2 client to connect to the HTTP/2 server.

   **Time out for the HTTP/2 session creations**

   Specifies the time limit to create the HTTP/2 session. This time starts after the connection is established.

   **Enable HTTP/2 Push**

   The Push functionality of HTTP/2 automatically identifies and passes the related objects or requests to the client when a request is sent to the server. Clear the check box to not use the functionality.

   **Initial session window**

   Specifies the buffer size on the sessions.

   **Initial stream window**

   Specifies the window size for buffer on each stream after the connection is established.

   **HTTP/2 Client Input Buffer Size**

   Specifies the buffer size that is used to read the network traffic.

**Maximum Quantity of Messages that can be queued**

Specifies the maximum number of messages that can be queued for the HTTP/2 client on a thread.

**Maximum Quantity of HTTP/2 thread pool**

Specifies the maximum number of thread pools that will be used by the HTTP/2 client to distribute the workload.

**Minimum Quantity of HTTP/2 thread pool**

Specifies the minimum number of thread pools that will be used by the HTTP/2 client to distribute the workload.

**HTTP/2 client bytebuffer pool size**

Specifies the buffer size to receive the unciphered values.

**Server Name Indication**

**Note:** Not applicable for HTTP/2.

Clear this check box if you do not want to connect to the host computer by using the Server Name Indication protocol. If the host computer is already configured with Server Name Indication protocol, you should keep this check box selected.

**Use HTTP Keep Alive**

Select this option to keep the HTTP connection open after the request. This option is not available if you are using IBM® Rational® AppScan®.

**Use SSL**

Select this option to use an SSL configuration. Click **Configure SSL** to create an SSL configuration or select an existing configuration.

**Platform Authentication**

In this section, specify the type of authentication that is required to access the service. Select **None** if no authentication is required.

**Basic HTTP authentication**

Select this option to specify the **User Name** and **Password** that are used for basic authentication.

**NTLM authentication**

**Note:** Not applicable for HTTP/2.

Select this option to use the Microsoft™ NT LAN Manager (NTLM) authentication protocol. NTLM uses challenge-response authentication. This view lists what

is negotiated (supported by the client and requested of the server) and what is authenticated (the client reply to the challenge from the server).

**Kerberos authentication**

> **Note:** Not applicable for HTTP/2.

Select this option to use the Kerberos authentication protocol between the client and server.

**Connect through proxy server**

> **Note:** Not applicable for HTTP/2.

If the HTTP connection needs to go through a proxy server or a corporate firewall, specify the **Address** and **Port** of the proxy server. If the proxy requires authentication, select either **Basic proxy authentication** or **NTLM proxy authentication**.

**Proxy authentication**

In this section, specify the type of authentication that is required to access the proxy. Select **None** if no authentication is required.

**Basic proxy authentication**

Select this option to specify the **User Name** and **Password** that are used for basic authentication.

**NTLM proxy authentication**

Select this option to use the Microsoft™ NT LAN Manager (NTLM) authentication protocol. NTLM uses challenge-response authentication. This view lists what is negotiated (supported by the client and requested of the server) and what is authenticated (the client reply to the challenge from the server).

**Custom class**

> **Note:** Not applicable for HTTP/2.

Select this option if the communication protocol requires complex, low-level processing with a custom Java™ code to transform incoming or outgoing messages. Click **Browse** to select a Java™ class that uses the corresponding API. This option is not available in IBM® Security AppScan®.

See for more information about SSL authentication.

5. Click **OK** to create the new configuration.

**What to do next**

Once created, you can use your new configuration with any service request that uses the HTTP transport protocol. You can use the **Configurations** list in the generic service client to edit existing configurations or to create duplicate configurations.

## Configuring the workbench for NTLMv2 authentication

NTLMv2 authentication requires access to a third-party library. To record and execute a test that contains NTLMv2 authentication, you must download the library and place it at the right location.

**Before you begin**

Before you can test SOAP-based services that use security algorithms, you must obtain and install a third-party library file.

**About this task**

By default, the HTTP test generation does not enable NTLMv2 authentication, even if it was part of the recording. To automatically enable the correct NTLM version from the recording, set the **Generated NTLM Version** setting to **Guess from recorded data** in the HTTP Test Generation preferences.

To configure the workbench to enable NTLMv2 authentication

1. Download the `jcifs-1.3.19.zip` file from [https://www.jcifs.org/src/](https://www.jcifs.org/src/).
2. Unarchive the zip file and copy the JAR file to the installation directory: `InstallationDirectory\plugins \com.ibm.rational.test.lt.provider_<version>`
3. To automatically enable the correct NTLM version from the recording, In the workbench, click **Window > Preferences > Test > HTTP Test Generation** and set the **Generated NTLM Version** setting to **Guess from recorded data**.

**Results**

When a test was recorded with NTLMv2, the **Generated NTLM Version** setting is selected in the test editor, under **NTLM Authentication**.

## Creating a JMS transport configuration

You can create an JMS transport configuration that describes the transport settings for a service request that uses the Java™ Message Service (JMS) protocol, including JBoss and IBM® WebSphere® JMS. Transport and security settings can be associated with any service request.

**Before you begin**

If you are using SOAP security, ensure that you have configured the environment with the correct libraries and configuration files.

1. Click the **Generic service client**  toolbar button to open the generic service client and click the **Transport** tab.
   **Result**

       This opens the **Transport Configurations** page.

2. On the **Transport Configurations** page, click one of the following buttons:

   **Choose from:**

   - **Create a basic JMS configuration** () to create a new generic JMS transport configuration.
   - **Create a JBoss JMS configuration** () to create a JMS configuration preconfigured for JBoss.
   - **Create a WebSphere JMS configuration** () to create a JMS configuration preconfigured for WebSphere® JMS.

3. Type a **Name** for the new transport configuration and select whether the service is a **queue** or a **topic** destination.

4. Type the address of the JMS end point.

5. Select **Use temporary object** to provide the address of the reception point to the service as a temporary object. If you disable this setting, you must manually specify the reception point address.

6. If the service requires authentication, select **Basic Authentication** and type the user name and password to access the service.

7. If the service requires a custom Java™ Naming and Directory Interface (JNDI) adapter, you can provide your own Java™ class that extends the Apache Axis class. In this case, select Custom Adapter and specify the name of the custom Java™ class. See Extending test execution with custom code on page     for more information about custom code.

8. Specify whether the message type is **Text** or **Binary**.

9. If necessary, click **Add** or **Edit** to specify the **Context factory properties** or **Connector properties** required to access the service.

10. Click **OK** to create the new configuration.

**What to do next**

Once created, you can use your new configuration with any service request that uses the JMS transport protocol. You can use the **Configurations** list in the generic service client to edit existing configurations or to create duplicate configurations.

## Creating a WebSphere® MQ protocol configuration

When you want to send requests to a service that uses WebSphere MQ transport protocol, you can create a protocol configuration to describe the transport settings for a service request.

**Before you begin**

If you are using SOAP security, ensure that you have configured the environment with the correct libraries and configuration files.

**About this task**

By default, messages are sent in bytes. Starting from V10.1.0, you can select message type as Text Message. After you create the protocol configuration, you can change the message format by selecting the **Text Message** check box in the **Message Structure**.

Transport and security settings can be associated with any service request. You can edit the existing configuration or duplicate the default configuration. You must have configured the environment with the correct libraries and configuration files when you use SOAP security.

1. Click the **Generic service client**  toolbar button, and then click the **Transport** tab.
2. From the **Protocol** list, right-click **MQ**, and then click **New MQ protocol configuration**.
3. Enter a name for the new transport configuration in the **Name** field.
4. Enter a name for the queue manager that receives the call in the **Queue Manager Name** field.
5. Enter a name for the queue managed by the queue manager in the **Send Queue Name** field.
6. Select the **Authentication** check box and specify the user name and password to authenticate with the MQ server.
   Alternatively, add or update the login credentials in the **Protocol Configuration** tab of a service test.
7. Select the **Use Local Queue Manager** check box when the WebSphere MQ server is running on the local computer.
8. Perform the following steps if the MQ server is installed on a remote computer:

   a. Clear the **Use Local Queue Manager** check box.

   b. Enter the remote WebSphere MQ server details in the following fields:
      ▪ The IP address or host name in the **Address** field.
      ▪ Listener port number in the **Port** field.
      ▪ Server connection mode channel name in the **Client Channel** field.

9. Select the **Use Temporary Queue for Response** check box if you want the server to create a temporary queue for receiving messages.
10. Perform the following steps to specify the queue that receives the response messages from the queue manager:
      a. Clear the **Use Temporary Queue for Response** check box.
      b. Enter a name for a queue in the **Receive Queue Name** field.
11. **Optional:** Specify the name of the target service in the **Target service** field when you are using the Microsoft .NET framework with SOAP over MQ.
12. **Optional:** Select **Use RFH2 header** when you are using SOAP over MQ. Otherwise, specify the **Message Descriptor** and **Encoding** options for the message header.
13. **Optional:** Click **Configure SSL** to select an existing SSL configuration or to create a new one when the service requires SSL authentication.
14. Click **OK** to create the protocol configuration.

**Results**

You have created a configuration for the WebSphere MQ transport protocol.

**What to do next**

- You can use the protocol configuration for the WebSphere MQ with any service request.
- You can change the message format by selecting the **Text Message** check box in the Message Structure.

---

Related information

# Creating a WebSphere® Java MQ transport configuration

You can create a transport configuration that describes the transport settings for a service request that uses the IBM® WebSphere® Java MQ protocol. Transport and security settings can be associated with any service request.

**About this task**

This topic has instructions to specify the MQ server settings. If you have a single MQ server, you can choose to use the **Default Java MQ protocol configuration** option. If, for a new request, you must point to another MQ server, you can use the instructions in this topic to create a new transport configuration.

1. Click **Generic service client** and click the **Transport** tab.
2. To create a new Java MQ transport configuration, in **Configurations**, select **Java MQ**.
3. In **Create Java MQ protocol configuration**, specify a name for the transport configuration.
4. Complete the following steps in the **Settings** tab:
   a. **Host**: Specify the host name or IP address of the MQ server.
   b. **Port**: Specify the port number that is used on the MQ server.
   c. **Channel**: Name of the MQ communication channel that is used for sending and receiving messages and specified on the server. This field is case-sensitive.
   d. **Queue Manager**: Name of the MQ queue manager as specified on the server.
   e. **Optional:  Use credential**: To access the secure server, specify the login credentials that is needed by the connection.
5. **Optional:  If necessary, complete the following steps in the **SSL** tab:

   **Learn more about the UI elements in the SSL tab:**
     ◦ Select the **Use MQ SSL** check box when the connection to the Queue manager uses SSL.
     ◦ **Peer Name**: Distinguished Name (DN) of the queue manager to be used by SSL. The Distinguished Name is available in the SSL certificate. In MQ, a DN pattern is specified by using the **sslPeerName** variable of **MQEnvironment**. Connections succeed only if Peer Name matches the pattern that is specified.
     ◦ **Cipher Suites**: Select one of the available cipher suites to use for encrypting the transport communications.
     ◦ **Fips Required**: This option specifies whether the requested cipher suite must use FIPS-certified cryptography in WebSphere MQ.
     ◦ **KeyResetCount**: The total number of non-encrypted bytes that can be sent and received within an SSL conversation before the secret key is renegotiated. If left blank or set to zero (default),

the secret key is never renegotiated. This value is ignored if no cipher suite is specified. Valid values are integers 0 - 999,999,999.

- ◦ **SSL Configuration**: Select a SSL setting for the connection or click **Configure SSL** to create a new SSL configuration. See Creating SSL configurations on page 270.

6. **Optional:**  Use the **Options** tab to configure actions such as read, write, and browse on the selected MQ Queues. Click to select the configuration options.

7. **Optional:**  Use the **Advanced** tab to specify the number of queue manager connections for reading messages, temporary destination settings, and to associate a reply with a request.

8. To test the connection, click **Test Transport** and then click **OK**.

   **Result**

   You have created a new transport configuration to point to a MQ server.

**What to do next**

You can now send the Java MQ requests to the configured server. See Sending WebSphere Java MQ endpoint requests on page 279.

## Creating Microsoft™ .NET transport configurations

You can manually create a Microsoft™ .NET transport configuration to describe the transport settings for service requests that use the Windows™ Communication Foundation (WCF) protocol.

**Before you begin**

If you are using SOAP security, ensure that the environment is configured with the correct libraries and configuration files.

Certificates and libraries required by the Microsoft™ client proxy must be installed on the computer, including Microsoft™ .NET libraries.

You must link a modified version of the Microsoft™ client proxy configuration file of the WCF service (by default `client.exe.config`) to the Microsoft™ .NET transport configuration. You must rename the file to `soaclient.exe.config` and edit it as described in the following procedure.

**Tip:** You can create a Microsoft™ .NET transport configuration automatically by importing the Microsoft™ .NET WSDL file. In this case, you must still manually edit the Microsoft™ .NET transport configuration to point to the modified `soaclient.exe.config` file as described in the following procedure. For more information, see Sending service requests with WSDL files on page 272

**About this task**

The product supports testing WCF services that use the following bindings:

- BasicHttpBinding
- WsHttpBinding

- NetMsMqBinding for 1-way calls only
- WSFederationHttpBinding
- WS2007FederationHttpBinding
- NetTcpBinding
- Custom bindings that do not integrate custom extensions in the channel, serialization of the message, transport, and security

**Note:** The following WCF services are not supported:

- Transaction and scopes
- Duplex mode requests, such as callbacks or 2-way services based on the Microsoft™ Message Queuing (MS-MQ) transport

**Only for IBM AppScan users:** To use Generic Service Client with IBM Appscan to test a WCF application, add the following code to the WCF configuration file:

```
<system.diagnostics>    <trace autoflush="true" />
    <sources>      <source name="System.Net"
    maxdatasize="1048576"><listeners><add
  name="System.Net"/></listeners></source>      <source
    name="System.Net.Cache"><listeners><add
    name="System.Net"/></listeners></source>      <source
    name="System.Net.Http"><listeners><add name="System.Net
    "/></listeners></source>      <source
    name="System.Net.Sockets"><listeners><add
    name="System.Net"/></listeners></source>      <source
    name="System.Net.WebSockets"><listeners><add
    name="System.Net"/></listeners></source>    </sources>
    <sharedListeners>      <add
  name="System.Net"
    type="IBM.ServiceModel.Soa.Extension.tools.TrafficTraceListener,
    Soa-Behavior-Library"
  initializeData=""      />
  </sharedListeners>    <switches>      <add name="System.Net"
    value="All"/>      <add name="System.Net.Cache"
  value="All"/>      <add name="System.Net.Http"
  value="All"/>      <add name="System.Net.Sockets"
  value="All"/>      <add name="System.Net.WebSockets"
  value="All"/>    </switches></system.diagnostics>
```

IBM Appscan expects only HTTP requests in WCF. The following HTTP bindings are supported:

- BasicHttpBinding
- Custombinding above standard httpTransport
- WsHttpBinding
- WsFederationHttpBinding
- WS2007FederationHttpBinding

Also, the following patterns are supported:

- Action value ( mandatory )
- Reply Action value ( mandatory )
- Protection level

1. Create a modified `soaclient.exe.config` file by completing the following steps:

   a. Create a copy of `client.exe.config` (or `proxy_client_name.config`) file from the Microsoft™ .NET project and rename the copy to `soaclient.exe.config`.

   b. Edit the `soaclient.exe.config` file to use the version of Microsoft™ .NET that the product supports, as specified on the following line:

   ```
   <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.0"/>
   ```

   c. Edit the `soaclient.exe.config` file so that the endpoints in the configuration file point to the client contract of the product, as specified on the following line:

   ```
   contract="IBM.ServiceModel.Soa.Extension.Stub.IStubTest"
   ```

   d. Import the modified `soaclient.exe.config` file into the workspace.

   **Result**

   After you create the `soaclient.exe.config` file, you can skip the following steps and import the WSDL file to automatically create a Microsoft™ .NET transport configuration based on the information provided by the WSDL. For more information, see Sending service requests with WSDL files on page 272.

2. Click the **Generic service client** toolbar button ( ) to open the generic service client and click the **Transport** tab.

3. On the **Transport Configurations** page, click **Create a Microsoft .NET configuration**.

4. Type a name for the new transport configuration and specify the following options:

   **Location of soaclient.exe.config**

   Specify the location of the `soaclient.exe.config` file. You must create this file manually by copying and editing the `client.exe.config` file from the Microsoft™ .NET service.

   **User authentication**

   If the service requires authentication, select **User Authentication** and type the user name and password to access the service.

   **Endpoint protection**

   By default, the transport configuration uses the endpoint protection level that is described in the `soaclient.exe.config` file. Use this setting to specify a different **Protection level**:
   - **Signature**: Select this option to digitally sign requests.
   - **Encryption and Signature**: Select this option to digitally sign and encrypt requests.

**Advanced properties**

Use this table to list the request and response actions by order of the methods in the WSDL file. Click **Add** to specify the name and value of request and response actions that are required by the service. This table is generated automatically when you import a Microsoft™ .NET WSDL file.

5. Click **OK** to create the transport configuration.

**What to do next**

After you create the configuration, you can use it with any service call that uses the Microsoft™ .NET transport protocol. You can use the **Configurations** list in the generic service client to edit existing configurations or to create duplicate configurations.

## Creating SSL configurations

You can create a Secure Sockets Layer (SSL) configuration that describes the settings for a service request that uses SSL certification mechanisms. SSL configurations can be associated with any service request that uses the HTTP or IBM® WebSphere® MQ transport protocols.

**Before you begin**

If you are using SSL, ensure that you have valid certificate keystore files in your workspace.

If you are using SOAP security, ensure that you have configured the environment with the correct libraries and configuration files. See Configuring the environment for SOAP security on page 123 for more information.

**About this task**

If you have to use different mutual SSL authentications for virtual testers in a test, you can create a dataset that stores all of the trust aliases names. In the test editor, in the **SSL Configuration** tab, you add a SSL configuration and associate it with the dataset. When a schedule is run, the SSL configuration is applied to each virtual tester.

1. Click the **Generic service client**  toolbar push button to open the generic service client, and click the **Transport** tab.
2. Either open an existing HTTP or WebSphere® MQ transport configuration, or create a new one, and then click **Configure SSL**.
3. Click
   **Rename**
   

   to rename the default SSL configuration or **New**  to create one.
4. Specify the following settings for the SSL configuration.

   **Server Authentication**

   This section describes how the client trusts the server.

**Always trust server**

> Select this option if no authentication is required or to ignore server certificates so that all servers are trusted. If you are using single authentication and you want to accept trusted servers only, then disable this option and specify a truststore that contains the trusted server certificates.

**Client truststore**

> When you are using single authentication, the client truststore contains the certificates of all trusted servers. Click **Browse** to specify a KS, JKS, or JCEKS file containing valid certificates of the trusted servers.

**Password**

> If the client truststore file is encrypted, type the password required to access the file.

**Mutual Authentication**

> This section describes how the server trusts the client in addition to server authentication.

**Use client-side certificate**

> If you are using double authentication, select this option to specify a keystore containing the client certificate. This certificate allows the server to authenticate the client.

> **Client certificate keystore**
>
> > Click **Browse** to specify a KS, JKS, or JCEKS file containing a valid certificate that authenticates the client.
>
> **Password**
>
> > If the client truststore file is encrypted, type the password required to access the file.

**Select trust alias for Mutual Authentication**

> Select an alias to be used for the SSL configuration. There could be multiple aliases in a keystore for different security certificates. Choose an appropriate alias for a user. You can also use dataset to store aliases that you can apply to virtual users at run time.

> **Note:** You can copy the contents from an SSL configuration into another SSL configuration by using **Copy** 📋 and **Paste** 📋 in the SSL editor.

5. Click **OK** to create the configuration, and close the SSL editor.

**What to do next**

When the SSL configuration is created, you can use the SSL configuration with any service request that uses SSL certification. You can use the SSL editor to edit existing configurations.

## Sending service requests with WSDL files

You can send requests to services based on SOAP, Java Messaging Service (JMS), WebSphere® MQ, and Microsoft™ .NET that use a Web Service Description Language (WSDL) file to specify the contents of the service request.

**Before you begin**

Ensure that you have a valid WSDL file, which is accessible either on the file system, in the workspace, at a specific URL, or in an IBM® WebSphere® Service Registry and Repository or a Universal Description Discovery and Integration (UDDI) repository.

Ensure that the WSDL files use the correct syntax for the test environment. The generic service client might not work with some WSDL files.

If the service uses Secure Sockets Layer (SSL) authentication, create an SSL configuration before sending the request. For more information, see Creating SSL configurations on page 270.

If the service uses SOAP security for encryption, signature, or other security algorithms, you must first configure the environment with the correct libraries and configuration files, and then create a WSDL security profile. For more information, see Configuring the environment for SOAP security on page 123 and Creating security profiles for WSDL files on page 174.

To import a WSDL file from a secured site that requires mutual authentication, you must have the Keystore file in the workspace.

**About this task**

When you create a call from a WSDL file, the call is configured automatically with any SOAP, JMS, WebSphere® MQ, or Microsoft™ .NET endpoints that are available in the WSDL file. Select the corresponding transport configuration on the **Transport** page of the request.

> **Note:** For the specific requirements related to Microsoft™ .NET support, see Creating Microsoft .NET transport configurations on page 267.

To send a service request based on a WSDL file:

1. Click the **Open the Generic Service Client** toolbar button and select the **Requests** page.
2. Click **Add** and select the method to add a WSDL file or click the corresponding shortcut button on the main page.
   **Choose from:**
   ◦ Click **Add WSDL from Workspace** to add a WSDL file from the local workspace.
   ◦ Click **Add WSDL from File System** to add a WSDL file from the file system.
   ◦ Click **Add WSDL from URL** to download and import an online WSDL from the web.

- Click **Add WSDL from WSRR** to add a WSDL from WebSphere® Service Registry and Repository. Enter the URL of the WebSphere® Service Registry and Repository and click **Connect**. You can click **Search** to browse the contents of the repository.

- Click **Add WSDL from UDDI** to add a WSDL from a Universal Description Discovery and Integration (UDDI) repository. Enter the URL of the UDDI and click **Connect**. You can click **Filter** and **Search** to browse the contents of the repository.

> **Note:** If you are importing the WSDL file from a secured site that requires certificate authentication, click **Import Properties** and, for **Keystore**, select the keystore file that contains the certificate to be provided to the server, and for the **Keystore password**, type the password.

3. Click **OK**.
   **Result**
   The WSDL file is added to the **Request Library**.

4. In the **Request Library**, expand the WSDL file, binding, and operation, and then select the call element.
   **Result**
   The generic service client shows three steps: **Edit Data**, **Invoke** and **View Response**. The details for the call are displayed under the **Edit Data** step.

5. On the **Message** page, use the Form, Tree, or Source views to edit the contents of the request.
   Each view shows a different format of the same data. To add or remove XML elements in the Form or Tree view, click **Schema > Validate and Assist** to comply with an XML Schema Definition (XSD) specified in the schema catalog.

6. On the **Transport** page, specify the transport configuration for the request.
   The transport information from the WSDL file is imported automatically into the transport configuration.

   For Microsoft™ .NET, select the corresponding transport configuration and specify the location of the `soaclient.exe.config` file. You must create this file manually. For details, see Creating Microsoft .NET transport configurations on page 267.

   > **Note:** If you are using IBM® Security AppScan®, only the HTTP and .Net transport protocols are available.

7. On the **Request Stack** page, specify whether to override the security or processing algorithms that are applied to the outgoing request for the WSDL file.
   Click **Show Response Stack** to add a **Response Stack** page to edit the security or processing algorithms for incoming responses.

   > **Note:** These settings apply only to the current request. If you want to edit the request or response stack for all requests that use the current WSDL file, click **Edit WSDL Security** to open the **WSDL Security Editor**.

8. When you are ready to send the service request, click **Invoke**.

**Result**

The generic service client sends the request and displays the message return under the **View Response** step.

**What to do next**

Successful requests are recorded and added to the **Request History** list. If you are using IBM® Rational® Performance Tester or IBM® Rational® Service Tester for SOA Quality, you can create a service test by clicking the **Generate Test Suite** button (⏱️).

## Sending HTTP endpoint requests

You can send requests to services that use an HTTP endpoint.

**Before you begin**

If the service uses Secure Sockets Layer (SSL) authentication, create an SSL configuration before sending the request. For more information, see Creating SSL configurations on page 270.

If the service uses SOAP security for encryption, signature, or other security algorithms, you must first configure the environment with the correct libraries and configuration files, and then create a security profile for the WSDL file. For more information, see Configuring the environment for SOAP security on page 123 and Creating security profiles for WSDL files on page 174

To send a request to an HTTP service:

1. Click the **Open the Generic Service Client** toolbar button 🔷 and select the **Requests** page.
2. Click the **Add** icon ➕ and click a type of request that you want to send or in Request Library, right-click **EndPoints** and select a type of request that you want to send.
3. In the **Configure Protocol** window, select **HTTP** and specify the HTTP transport configuration.
   If necessary, click **New** to create an HTTP transport configuration for the call.

   To send the HTTP/2 requests, in the **Create HTTP Protocol configuration** window, click the **Activate** check box. Before capturing the HTTP/2 traffic, configure the computer. See Preparing to record a test for the HTTP/2 service on page 130 for instructions.

4. Type the URL of the call, the HTTP method and version, and specify any header or cookie properties.
   Click the **Rest mode** check box to split the URL into resource and parameters.
5. Click **Next**.
6. On the **Select Root Element** page, if the service uses a specific XML Schema Definition (XSD), select one from the list or click **Browse** to import the XSD file, and then, select the root element for the request.
   If no XSD is available for the service, select **No Schema**.
7. Click **Finish**.
   **Result**

   The request is added to the **Endpoints** section of the **Request Library**.
8. In the **Request Library**, select the request element.
   **Result**

The generic service client shows three steps: **Edit Request**, **Invoke**, and **View Response**. The details for the request are displayed under the **Edit Request** step.

9. Based on the request selected in Step 2, on the **Message** page, use the **Form**, **Tree**, or **Source** views to edit the contents of the request.

   Each view shows a different format of the same data. To add or remove XML elements in the **Form** or **Tree** view, click **Schema > Validate and Assist** to comply with an XSD specified in the schema catalog.

10. On the **Attachments** page, specify any file attachments to send with the request.

    To add an attachment, click **Add** and follow the wizard to attach a file with the request.

11. On the **Transport** page, if necessary, change the transport configuration to be used by the request.

    To create and edit transport and security configurations, use the **Transport** tab.

12. If you selected SOAP XML request in step 2, on the **Request Stack** page, specify whether you want to override the security or processing algorithms that are applied to the outgoing request for the WSDL file.

    To add a **Response Stack** page to edit the security or processing algorithms for incoming responses, click **Show Response Stack**.

    > 📝 **Note:** These settings apply only to the current request. To edit the request or response stack for all requests that use the current WSDL file, click **Edit WSDL Security** to open the **WSDL Security Editor**.

13. When you are ready, click **Invoke** to send the service request.

    **Result**

    The generic service client sends the request and displays the message return under the **View Response** step.

**What to do next**

Successful requests are recorded and added to the **Request History** list. If you are using IBM® Rational® Performance Tester or IBM® Rational® Service Tester for SOA Quality, you can create a service test by clicking the **Generate Test Suite** button ( ).

## Sending a JMS endpoint request

You can send requests to services that use a Java™ Messaging Service (JMS) endpoint.

**Before you begin**

If the service uses Secure Sockets Layer (SSL) authentication, create an SSL configuration before sending the request. For more information, see Creating SSL configurations on page 270.

To send a request to a JMS service:

1. Click the **Open the Generic Service Client** toolbar button ( ) and select the **Requests** page.
2. Click **Add** ( ) and click a type of request that you want to send or in Request Library, right-click **EndPoints** and select a type of request that you want to send.
3. In the **Configure Protocol** window, select **JMS** and specify the JMS transport configuration.

   If necessary, click New to create an JMS transport configuration for the call.
4. Click **Add** to specify any properties that are to be sent with the call.

5. Click **Next**.

6. On the **Select Root Element** page, if the service uses a specific XML Schema Definition (XSD), select one from the list or click **Browse** to import the XSD file, and then, select the root element for the call.

   If no XSD is available for the service, select **No Schema**.

7. Click **Finish**.

   **Result**

   The request is added to the **Endpoints** section of the **Request Library**.

8. In the **Request Library**, select the request element.

   **Result**

   The generic service client shows three steps: **Edit Request**, **Invoke**, and **View Response**. The details for the request are displayed under the **Edit Request** step.

9. Based on the request selected in Step 2, on the **Message** page, use the **Form**, **Tree**, or **Source** views to edit the contents of the request.

   Each view shows a different format of the same data. To add or remove XML elements in the **Form** or **Tree** view, click **Schema > Validate and Assist** to comply with an XSD specified in the schema catalog.

10. On the **Transport** page, if necessary, change the transport configuration to be used by the request.

    To create and edit transport and security configurations, use the **Transport** tab.

11. If you selected SOAP XML request in step 2, on the **Request Stack** page, specify whether you want to override the security or processing algorithms that are applied to the outgoing request for the WSDL file.

    To add a **Response Stack** page to edit the security or processing algorithms for incoming responses, click **Show Response Stack**.

    > **Note:** These settings apply only to the current request. To edit the request or response stack for all requests that use the current WSDL file, click **Edit WSDL Security** to open the **WSDL Security Editor**.

12. When you are ready, click **Invoke** to send the service request.

    **Result**

    The generic service client sends the request and displays the message return under the **View Response** step.

**What to do next**

Successful requests are recorded and added to the **Request History** list. If you are using IBM® Rational® Performance Tester or IBM® Rational® Service Tester for SOA Quality, you can create a service test by clicking the **Generate Test Suite** button ().

## Sending a WebSphere® MQ endpoint request

You can invoke calls to services that use a WebSphere® MQ endpoint.

**Before you begin**

If the service uses Secure Sockets Layer (SSL) authentication, create an SSL configuration before sending the request. For more information, see .

If the service uses SOAP security for encryption, signature, or other security algorithms, you must first configure the environment with the correct libraries and configuration files, and then create a security profile for the WSDL file. For more information, see Configuring the environment for SOAP security on page 123 and Creating security profiles for WSDL files on page 174.

To send a request to an WebSphere® MQ service:

1. Click the **Open the Generic Service Client** toolbar button (🖼) and select the **Requests** page.
2. Click **Add** (➕) and click a type of request that you want to send or in Request Library, right-click **EndPoints** and select a type of request that you want to send
3. In the **Configure Protocol** window, select **WebSphere MQ** and specify the WebSphere® MQ transport configuration.
   If necessary, click **New** to create an WebSphere® MQ transport configuration for the call. For more information about creating a new WebSphere MQ transport configuration, see Creating a WebSphere MQ transport configuration on page 264.
4. Specify the SOAP action.
   If the service requires that you override the header specified in the WebSphere® MQ transport configuration, select **Override MQ protocol configuration values** and specify the correct details.
5. Click **Next**.
6. On the **Select Root Element** page, if the service uses a specific XML Schema Definition (XSD), select one from the list or click **Browse** to import the XSD file, and then, select the root element for the request.
   If no XSD is available for the service, select **No Schema**.
7. Click **Finish**.
   **Result**
   The request is added to the **Endpoints** section of the **Request Library**.
8. In the **Request Library**, select the request element.
   **Result**
   The generic service client shows three steps: **Edit Request**, **Invoke**, and **View Response**. The details for the request are displayed under the **Edit Request** step.
9. Based on the request selected in Step 2, on the **Message** page, use the **Form**, **Tree**, or **Source** views to edit the contents of the request.
   Each view shows a different format of the same data. To add or remove XML elements in the **Form** or **Tree** view, click **Schema > Validate and Assist** to comply with an XSD specified in the schema catalog.
10. On the **Transport** page, if necessary, change the transport configuration to be used by the request.
    To create and edit transport and security configurations, use the **Transport** tab.
11. If you selected SOAP XML request in step 2, on the **Request Stack** page, specify whether you want to override the security or processing algorithms that are applied to the outgoing request for the WSDL file.
    To add a **Response Stack** page to edit the security or processing algorithms for incoming responses, click **Show Response Stack**.

> **Note:** These settings apply only to the current request. To edit the request or response stack for all requests that use the current WSDL file, click **Edit WSDL Security** to open the **WSDL Security Editor**.

12. When you are ready, click **Invoke** to send the service request.

    **Result**

    The generic service client sends the request and displays the message return under the **View Response** step.

**What to do next**

Successful requests are recorded and added to the **Request History** list. If you are using IBM® Rational® Performance Tester or IBM® Rational® Service Tester for SOA Quality, you can create a service test by clicking the **Generate Test Suite** button ( ).

## Sending OData endpoint batch requests

To test services that use OData protocol, you can send requests in a batch. The request contains HTTP operations such as GET, POST, and PUT to manage data in the service.

**Before you begin**

You must have sent individual requests through Generic Service Client (GSC).

**About this task**

When you send requests in a batch, you can group a set of operations into one HTTP request. You can start a batch request from GSC or from a service test. To initiate a batch request from a service test in the Test editor, select multiple requests to include in a batch, right-click and select **$batch odata requests**.

To initiate a batch request from GSC, complete the following steps:

1. Click the **Open the Generic Service Client** toolbar button  and select the **Requests** page.
2. Click the **Add** icon  and click a type of request that you want to send or in Request Library, right-click **EndPoints** and select **Send a Batch Request**.
3. In the **ODATA batch information** page, select the OData version that your application supports.
4. To set HTTP headers, ensure that the **Set ODATA batch request http headers** radio button is selected.

    If needed, you can change the headers on the next page of the wizard.

5. To group appropriate requests into change sets, select the **ODATA batch with changesets** radio button.
6. In **Selection of calls to batch**, select the requests to include in the batch.

    If you initiated the batch request from the service test, the requests are already selected.

7. Click **Next**.
8. In the **Configure Protocol** window, select **HTTP** and specify the HTTP transport configuration.

    If necessary, click **New** to create an HTTP transport configuration for the call.

To send the HTTP/2 requests, in the **Create HTTP Protocol configuration** window, click the **Activate** check box. Before capturing the HTTP/2 traffic, configure the computer. See Preparing to record a test for the HTTP/2 service on page 130 for instructions.

9. Click **Finish**.

   **Result**

   The request is added to the **Endpoints** section of the **Request Library**.

10. In the **Request Library**, select the request element.

    **Result**

    The generic service client shows three steps: **Edit Request**, **Invoke**, and **View Response**. The details for the request are displayed under the **Edit Request** step.

11. On the **Transport** page, if necessary, change the transport configuration to be used by the request.

    To create and edit transport and security configurations, use the **Transport** tab.

12. When you are ready, click **Invoke** to send the service request.

    **Result**

    The generic service client sends the request and displays the message return under the **View Response** step.

## Sending WebSphere Java MQ endpoint requests

You can send requests to services that use a WebSphere Java MQ endpoint.

**Before you begin**

If the service uses Secure Sockets Layer (SSL) authentication, create an SSL configuration before sending the request. For more information, see Creating SSL configurations on page 270.

To send a request to a Java MQ service:

1. Click **Open the Generic Service Client** (⬛) and select the **Requests** page.
2. Click **Add** (➕) or in Request Library, right-click **EndPoints** and select a type of request to send.
3. In the Configure Protocol window, select **WebSphere Java MQ** and specify the transport configuration. If necessary, create the transport configuration for the call by clicking **New** (see Creating a WebSphere Java MQ transport configuration on page 266).
4. Complete the following information in the **General** tab:

   🖋 **Learn more about the UI elements in the General tab:**

   **Queue**

   Name of the queue as defined on the WebSphere MQ server.

   **Message type**

   The types of messages are these:
   - *Datagram* means that the message does not require a reply.
   - *Request* means that the message requires a reply.

- *Reply* means that the message is a reply to an earlier request message.
- *Report* means that the message is reporting on some expected or unexpected occurrence, usually related to some other message. An example is a request message that contained data that was not valid.

**Message Persistence**

This value indicates whether the message is persistent or not. If the message is persistent, it survives the system failures and restarts of the queue manager. If the message is not persistent, it survives a restart if it is present on a queue having the NPMCLASS(HIGH) attribute. However, even with the NPMCLASS(HIGH) attribute a message does not survive a QMGR class. Nonpersistent messages on queues having the NPMCLASS(NORMAL) attribute are discarded at queue manager restart, even if the message is found on the auxiliary storage during the restart procedure.

**Dynamic Reply**

Select this check box for the WebSphere MQ server to dynamically create a temporary queue as a reply. If this check box is not selected, the message in Reply Queue is used.

**Reply Queue**

This is the name of the message queue to which the application that issued the get request for the message should send the reply and report messages.

**Reply Manager**

This is the name of the queue manager on which the reply-to queue is defined.

**Additional properties**

Specify the additional properties for the queues.

5. **Optional:** If necessary, complete the following information on the **Config** tab:

**Learn more about the UI elements in the Config tab:**

**Message Priority**

This is the priority of the message. The lowest priority is 0.

**Encoding**

This is the numeric encoding of numeric data in the message. This value does not apply to numeric data in the MQMD structure itself.

**Expiry Interval**

This is the period of time, in tenths of a second, after which the message becomes eligible to be discarded if it has not already been removed from the target queue. The expiry interval is set by the application that put the message.

**Character set**

This is the character set identifier of the character data in the application message data.

6. **Optional:** In the **Report** tab, select the report messages to receive.

7. **Optional:** If necessary, complete the following information in the **Context** tab:

**Learn more about the UI elements in the Context tab:**

**Application Identity Data**

This information is defined by the application suite. Use it to provide information about the message or its originator.

**Application Origin Data**

This information is defined by the application suite. Use it to provide additional information about the origin of the message.

**Accounting Token**

This information is needed by the application to appropriately charge for the work that is done as a result of the message.

**User ID**

This is the user identifier of the application that originated the message.

8. **Optional:** In the **Identifiers** tab, for the messages that require binary input, specify the ID in the string format in the second column. The first column is filled automatically in the hexadecimal format.

9. **Optional:** In the **Segmentation** tab, select the segment of the message and click **Next**.

10. This step is not applicable for a Text request. On the Select Root Element page, if the service uses a specific XML Schema Definition (XSD), select one from the list. If the XSD element is not listed, click **Browse** to import the XSD file, and select the root element for the request. If no XSD is available for the service, select **No Schema**.

11. Click **Finish**. The request is added to the **Endpoints** section of the Request Library.

12. In the **Request Library**, select the request element.

**Result**

The generic service client shows three steps: **Edit Request**, **Invoke**, and **View Response**. The details for the request are displayed under the **Edit Request** step.

13. Based on the request selected in Step 2, on the **Message** page, use the **Form**, **Tree**, or **Source** views to edit the contents of the request.

Each view shows a different format of the same data. To add or remove XML elements in the **Form** or **Tree** view, click **Schema > Validate and Assist** to comply with an XSD specified in the schema catalog.

14. On the **Transport** page, if necessary, change the transport configuration to be used by the request.

To create and edit transport and security configurations, use the **Transport** tab.

15. If you selected SOAP XML request in step 2, on the **Request Stack** page, specify whether you want to override the security or processing algorithms that are applied to the outgoing request for the WSDL file.

    To add a **Response Stack** page to edit the security or processing algorithms for incoming responses, click **Show Response Stack**.

    > **Note:** These settings apply only to the current request. To edit the request or response stack for all requests that use the current WSDL file, click **Edit WSDL Security** to open the **WSDL Security Editor**.

16. When you are ready, click **Invoke** to send the service request.

    **Result**

    The generic service client sends the request and displays the message return under the **View Response** step.

**What to do next**

Successful requests are recorded and added to the **Request History** list. If you are using IBM® Rational® Performance Tester or IBM® Rational® Service Tester for SOA Quality, you can create a service test by clicking the **Generate Test Suite** button (⏱).

## Testing all operations in a WSDL file

You can use the generic service client to rapidly send requests to a service using all the operations in a Web Services Description Language (WSDL) file. The calls are generated with default values based on the type of data.

**Before you begin**

Ensure that you have a valid WSDL file. Ensure that the WSDL files use the correct syntax for the test environment. The generic service client might not work with some Web Services Description Language (WSDL) files.

If the service uses Secure Sockets Layer (SSL) authentication, create an SSL configuration before invoking the call. See Creating SSL configurations on page 270 for details.

If the service uses SOAP security for encryption, signature, or other security algorithms, you must first configure the environment with the correct libraries and configuration files, and then create a security profile for the WSDL. See Configuring the environment for SOAP security on page 123 and Creating security profiles for WSDL files on page 174 for details.

Calls will be generated for each operation in the WSDL file using the default values for each type. For example, strings will use the default value `str`. You can change the default values in the **XML Default Values** preferences.

1. Open the generic service client and click the **Requests** tab, and then, click ⌹ **Add a WSDL file**.
2. In the Add WSDL Files window, select an existing WSDL or import a WSDL with one of the following methods:

    **Choose from:**
    - Click **Import from File** to import a WSDL file from the file system.
    - Click **Import from URL** to download and import an online WSDL from the web.

- ◦ Click **Import from WSRR** to import a WSDL from an IBM® WebSphere® Service Registry and Repository (WSRR). Enter the URL of the WSRR and click **Connect**. You can click 🏷️ **Search** to browse the contents of the repository.
- ◦ Click **Import from UDDI** to import a WSDL from a Universal Description Discovery and Integration (UDDI) repository. Enter the URL of the UDDI and click **Connect**. You can click ⇥ **Filter** and 🏷️ **Search** to browse the contents of the repository.

3. Click **OK**.
   **Result**
   The WSDL is added to the **Call Library**.
4. In the Call Library, right-click the WSDL and select **Test WSDL Methods**.
   The call is automatically configured with any SOAP or JMS endpoints that are available in the WSDL.

**What to do next**

Successful calls are recorded and added to the **Request History** list. If you are using IBM® Rational® Performance Tester, you can click the **Generate Test Suite** (🗓️) button to create a service test.

## Viewing message content

The **Raw Transaction Data** view displays the raw XML, text, or binary content of any service request or response that is selected in the generic service client.

**About this task**
The **Raw Transaction Data** view displays plain text, XML, or binary data, depending on the type of the message content.

To view text, XML, or binary message content:

1. In the generic service client, click the **View** menu, and select **Raw Transaction Data**.
   If you are using IBM® Rational® Performance Tester or IBM® Rational® Service Tester for SOA Quality, click
   **Window > Show View > Other > Generic Service Client > Raw Transaction Data**
2. Select a service request or response.
   If you are using Rational® Performance Tester, this view is also linked to the selected request or response in service tests, service stubs, or in the test log.
3. Depending on the nature of the message content, the following actions are available:

   **Text mode**

   When a plain text element is displayed, you can select and copy text. Click **Colorize Text** to enable or disable text colorization for HTML.

   **XML mode**

   When an XML element is displayed, you can select and copy text. Click **Colorize Text** to enable or disable text colorization for XML. Click **Enable XML Pretty Serialization** to improve readability by adding line breaks and indentation to the XML content.

If the XML content is modified by a request or response stack or by the WSDL security editor, the **Stack Contents** pane displays the list of steps in the stack. You can select each step to view the changes to the XML content. You can also select one or two steps and click **Compare Steps** to open a comparison window.

**Binary mode**

When a binary element is displayed, you can switch between **Binary** and **Raw-ASCII** views. Right-click the binary view to perform the following actions:

- ◦ **Select**: Opens the **Select** window, where you can select binary data by string or by specifying the number of characters to select. When a portion of binary data is selected, you can copy it to the clipboard.
- ◦ **Go to Offset**: Opens the **Go to Offset** window, where you can move to bytes at a particular offset.
- ◦ **Find**: Opens the **Find** window, where you can search for and replace binary data in a number of formats.
- ◦ **Encodings**: Select the encoding to use for displaying binary data in the text column.

## Synchronizing a remote WSDL file

For web services that make their Web Services Description Language (WSDL) file available from a URL, you might have to ensure that the WSDL that you work with is always up to date. By synchronizing the WSDL, you ensure that the local copy of the WSDL in your workspace is regularly synchronized with the remote WSDL.

**Before you begin**

Ensure that you have a valid WSDL file. Ensure that the WSDLs use the correct syntax for the test environment. The product might not work with some WSDL files.

WSDL synchronization only works with remote WSDLs that are imported from a URL.

The WSDL synchronization runs either when the workbench is started or after a specified period. If the remote WSDL changes, the local copy of the WSDL is updated. Depending on the changes, a merge is performed and any service requests that use the WSDL are updated. If the changes to the WSDL cannot be automatically applied to the service requests, for example if an operation is removed or renamed or if the XML structure of the service request is changed, the test is marked with a error.

To import a synchronized remote WSDL:

1. Open the generic service client, click the **Requests** tab, and then, click  **Add a WSDL file** .
2. In the **Add WSDL Files** window, click **Import from URL** to download and import a remote WSDL from the web.
3. On the **Import WSDL from URL** page, type the URL of the remote WSDL.
   If you are connecting through a proxy or a corporate firewall, click **Proxy properties** to specify your network settings.
4. In the **Synchronization policy** area, specify whether and when to synchronize WSDLs:

**Choose from:**

- Select **Never** if you do not want the remote WSDL to be updated.
- Select **On session launch** to synchronize the WSDL each time you start the workbench.
- Select **Every** to specify a synchronization period in days.

5. Click **OK**.

**Result**

The WSDL is added to the **Call Library**.

**What to do next**

After the WSDL is imported, you can change the synchronization settings by right-clicking the WSDL in the generic service client **Call Library** or in the test navigator. Then select **WSDL Synchronization**. The **WSDL Synchronization** window also displays the date of the latest synchronization.

Related information

Sending service requests with WSDL files on page 272

Testing all operations in a WSDL file on page 282

## Synchronizing a local WSDL file with GSC

If you edit a local WSDL source file, the Generic Service Client (GSC) should display the changes in the UI. You must keep the GSC up-to-date with the WSDL changes to ensure that you test the latest service request.

**About this task**

When you set GSC to automatically pick the WSDL changes, the GSC calls are fully re-created. This means that when you make some changes to WSDL, there might be some content that you did not change, however, was dependent on the changed content. Therefore, when you use this preference, the whole structure of the GSC calls is re-created.

The Request History view in GSC shows the changes occurred to the WSDL file.

To apply the local WSDL changes in GSC:

1. Click **Window > Preferences > Generic Service Client**.
2. Select the **Apply WSDL changes to GSC** check box.

## Adding static XML headers to a service request

You can add static XML headers to service requests to ensure compliance with WS-Addressing, WS-ReliableMessaging, and WS-Coordination specifications as well as other predefined standards.

**About this task**

Static XML headers are compliant with the web service specifications for service-oriented architecture (SOA). Checks are performed to ensure that the XML headers are valid.

To add a static XML header to a request:

1. Open a service request in the generic service client. The location of the XML header depends on the product that you are using:

   **Choose from:**

   - For IBM® Security AppScan®, click the **Request Stack** tab and in the algorithm stack for the request, click **Add > Static XML Headers**.
   - For IBM® Rational® Performance Tester or other products, click the **Message** tab and click **Form**.

2. On the **Header** bar, click **Add** ( ) to open the menu.

3. Select the web service specification for the request to be comply with, or click **More** to open a detailed list of specifications.

   **Result**

   The XML structure of the header is created.

4. Edit the header as required.

   Some elements require completion or content to be specified. XML elements that are invalid or require attention are marked with a warning or an error symbol.

---

Related information

## Opening file attachments

When a service sends a file attachment with the response, you must import it as a resource to open the attachment.

**Before you begin**

Ensure that you have specified an editor to view the attachment type in. Click **Window > Preferences > General > Editors > File associations**, and specify the editor.

1. Open the message return, and click the **Attachment** tab.

   File attachments are listed with a default name, a MIME type, and a contents ID.

2. Select the line for the attachment that you want to open, and click **Open**.

3. In the **Create Resource** window, type a name for the resource, and select a location where it will be imported, and click **OK**.

   Ensure that the name of the resource includes a file extension that is compatible with the MIME type of the attachment.

**What to do next**

After the attachment has been imported, you can click on **Open** again to open the file in the corresponding editor.

# Chapter 7. Test Execution Specialist Guide

This guide describes tasks that you can perform on schedules, test execution with custom code, and Extending Rational® Service Tester for SOA Quality to support other protocols. This guide is intended for testers or test execution specialists.

## Running schedules

After you have added the user groups, tests, and other items to a schedule, and you are satisfied that it represents a realistic workload, you run the schedule.

## Running a local schedule or test

You can run a test locally or a A schedule, in this context, is used to refer to both VU Schedule and Rate Schedule. on remote locations with a default launch configuration.

**Before you begin**

To play back tests against the applications that require client authentication such as Digital Certificates, Smart Card, or Kerberos, you must provide the appropriate authentication before playing back the test.

- To play back a test with a digital certificate, see Playing back a test with a digital certificate on page 152.
- To play back a test that require smart card authentication, click the **Security** tab in the Test editor, and provide the same smart card certificate alias and PIN that you used for the recording. See the Smart card authentication topic for more information.
- To play back a test that require Kerberos authentication, see Generating tests that use Kerberos.

**About this task**

When you run a schedule or test in this way, IBM® Rational® Performance Tester automatically sets up a simple launch configuration. A test runs on the local computer, with one user. A schedule runs with the user groups or Rate Runner groups and the locations that you have set. However, the execution results have a default name (the same as the schedule or test, with a suffix) and are stored in a default location.

The Rate Schedule can be run only on agent locations. The Rate Schedule can be run on agents that were installed only with PVU-based licenses.

When you run a schedule with multiple agents, an agent might be lost, especially during the long load test run. Losing an agent is not common and occurs during some extreme cases such as when computer's memory is exhausted. When an agent is lost, by default, the schedule is stopped. When the schedule is stopped in this manner, you must fix the reason of agent loss or add more agents before running the schedule. To continue to run the schedule without the lost agent, in the Schedule editor, click the **Advanced** tab and clear the **Loss of an agent halts execution** check box.

Typically, the agents divide the load among themselves. So, running a schedule without the lost agent might give unpredictable results. If you use a segmented dataset and if you run a schedule without the lost agent, the data is not

redistributed among the surviving agents. Also, if the schedule has multiple stages, by default, the load is distributed among the surviving agents at the next stage. But, if the **Replace lost users in current stage** check box is selected, then the load is distributed evenly among the surviving agents in the current stage. If the check box is cleared and a percentage of users or clients are allowed to exit during stage execution, the load is distributed among the surviving agents in the next stage. Loss of an agent in a schedule run is logged in the Performance Report.

To stop a test gracefully without causing incomplete page hits, select the **Active actions are allowed to complete if stop requested** check box at **Window > Preferenes > Test > Test Execution**.

To receive email notification for the status of the run, specify the email properties in **Window > Preferenes > Test > Test Execution**.

1. In the Test Navigator, expand the project until you locate the schedule or test.
2. Right-click the schedule or test, and then click **Run As > Performance Schedule** or **Run As > Test**.

> **Note:** If you run an HTTP schedule on a remote Macintosh computer, the test fails. The cipher suite that is used for recording must be available in Oracle JDK on the Macintosh computer. For example, you can use `TLS_RSA_WITH_AES_128_CBC_SHA` on Macintosh.

**Results**

After you run a test or a schedule, the Performance Test Runs view opens. In this view, you can add comments about the selected result and view the settings that were used to run the schedule. To add comments, in the lower-left panel of the Performance Test Runs view, click **User Comments**. The comments that you enter are displayed on the Summary page of performance reports. To view the settings that were used for a schedule run, click **Schedule Settings**. The Performance Test Runs View Schedule Settings page displays and shows the statistics and test log settings that were used for the run.

> **Note:** When you record a test that includes a file download, the file is not physically saved to disk. However, you can confirm that the file was retrieved from the server by looking in the response of the request that asked for the file. One method to locate the request for large downloaded files is to look for a request with a large response size.

**What to do next**

You can configure a schedule or test. A typical reason for setting up a configuration is to control where the execution results are stored. For more information, see .

## Setting a launch configuration

Instead of using the default launch configuration, you can specify the file name for the execution results, the name of the folder for the execution results, and, for a test, the number of users.

**About this task**

You generally run a A schedule, in this context, is used to refer to both VU Schedule and Rate Schedule. by right-clicking it and selecting **Run > Run VU Schedule**VU Schedule or **Run > Run Rate Schedule**A schedule, in this context, is used to refer to both VU Schedule and Rate Schedule.. However, you should set a launch configuration when:

- You want to specify a name for the execution results, or you want them in a separate folder.
- You plan to run a test outside of a A schedule, in this context, is used to refer to both VU Schedule and Rate Schedule., and you want to run the test with more than one user.
- You want the launch configuration to appear in your toolbar menu.
- You want the launch configuration to be available to other users.

To set a launch configuration:

1. In the Test Navigator, expand the project until you locate the schedule or test.
2. Right-click the schedule or test, and then click **Run As > Run configuration**.
   If the Perspectives page is displayed, keep the defaults.
3. In the **Configurations** area on the left, click **VU Schedule** or A schedule, in this context, is used to refer to both VU Schedule and Rate Schedule., and then click **New**.
   **Result**
   A test configuration, initially named `New_configuration`, is created. Typically, you supply a configuration name that is similar to the schedule name.

   At this point, you can run the schedule if you click **Run**. However, you will not have created a meaningful configuration.

4. Click the **Test Logs** tab and check the default settings. To change the default settings, clear the **Use defaults** check box and type a file name for the execution results. The product appends a time stamp to this name. To overwrite the file each time that you run the configuration, select the **Override existing test log** check box.
5. Click the **Common** tab to inspect or modify your run preferences.
6. In **Save as**, select one of the following options:

| Option | Description |
|---|---|
| **Local** | This launch configuration is stored in your work-space, and it is not visible to other users. |
| **Shared** | Other users have access to the launch configuration; you are asked where to store it. |

7. For **Display in favorites menu**, select one or more of the following options:

| Option | Description |
|---|---|
| **Run** | The configuration is displayed in your **Run** toolbar menu. If you select a toolbar menu at all, this is the logical choice for a schedule or test. |

| Option | Description |
|---|---|
| **Debug** | The launch configuration is displayed in your **Debug** toolbar menu. |
| **Profile** | The configuration is displayed in your **Profile** toolbar menu. |

8. Verify that **Launch in background** is selected. If you do not run the configuration in the background, you cannot do anything in Eclipse until it finishes running the configuration.
9. Click **Apply**, and then click **Run** to run the configured schedule or test, or click **Close** to save the configuration and run it later.

## Running a configured schedule

If you do not use the default launch configuration, you can configure the schedule and then run it.

**Before you begin**

You must configure the schedule before you run it. For more information, see Setting a launch configuration on page 288.

1. In the Test Navigator, expand the project until you locate the schedule.
2. Right-click the schedule, and then click **Run > Run**.
3. In the Configurations area on the left, click **Test Schedule**, and then click the name of the schedule to run.
4. Click **Run**.

**Results**

While the schedule is running, the reports are updated in real time, and you can see the changes.

## Configuring multiple host names for a location

You can run several locations on the same computer by configuring multiple host names for a location. This configuration affects all tests running at that location; all tests will run with the configured port.

To configure multiple host names for a location:

1. Open the hosts file, which maps IP addresses to hosts, with an ASCII editor.
   On Windows™, the hosts file is in `C:\\Windows\system32\drivers\etc\hosts`. On Linux™, the hosts file is in `\etc\hosts`
2. At the end of the hosts file, add your IP mappings. Use one IP address, but map it to two (or more) logical host names.
   **Example**
   For example, you could add map the IP address 123.4.5.6. to two logical hosts, as show in the bottom two lines:
   **Result**

```
# Copyright (c) 1993-1999 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for windows.
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#      102.54.94.97      rhino.acme.com          # source server
#       38.25.63.10      x.acme.com              # x client host

127.0.0.1       localhost
123.4.5.6       remlocationa
123.4.5.6       remlocationb
```

3. Create two deployment locations that have names identical to the names you added in the hosts file:

    a. Open the schedule that contains the user group that you want to run on multiple hosts.

    b. Open the user group, and click the **Location** tab.

    c. Click **Add > Add New**, and enter the location data. Make sure the locations have different directories (in this example, they are tempa and tempb).

    **Example**

    **Result**

| Locations | Options | |
| --- | --- | --- |
| ○ Run this group on the local computer | | |
| ⦿ Run this group on the following locations | | |

| Name | Hostname | Directory |
| --- | --- | --- |
| remlocationa | 123.4.5.6 | D:\\tempa |
| remlocationb | 123.4.5.6 | D:\\tempb |

    d. Click **Finish**.

## Automating tests

You can run a schedule from the command line. You can also set preferences to export results after the run completes from the command line or from the workbench. Together, these features let you run tests and analyze results without opening the workbench. You can even write scripts to process the exported results.

## Creating a command-line config file

Starting from V10.0.2, you can create command line config file from the product, which you can use while running tests from the command-line interface and Maven.

**Before you begin**

You must have performed the following tasks:

- Created test assets in a workspace.
- Installed Maven if you are running tests from the Maven build.

For information about creating tests and installing Maven, see related links.

**About this task**

Previously, you created the config file manually by adding parameters to it for running the tests by using the config file from the command line. Now, you can create a command-line config file from the product by right-clicking the test asset. The required parameters are automatically assigned, and you can specify any optional parameters, while creating the config file. You can use this config file to run the tests from the command-line interface and Maven plug-in that is provided with the product as part of Maven build.

1. In the Test Navigator, browse and select the test.
2. Right-click the test , and then click **Create command line config file**.
3. In the **Create New Config File** window, enter a name for the new configuration file and then click **Next**.
4. Perform the following sub-steps in the **Command Line Arguments** window:

   a. Select the format of the config file from the following options:
      - **Regular** – Use this format to run tests from the command-line interface.
      - **Maven** – Use this format to run tests from the Maven build.

   b. If you want to add more parameters to a config file, specify the values in the fields from the available configuration options.

5. Click **Finish**.

**Results**

The **Config file created** dialog box displays the location of the config file.

**What to do next**

You must complete the following steps:

1. Close the product.
2. Run a test by using the config file either from the command-line interface or from the Maven build.

Related information

## Activating secure storage of dataset passwords for test automation

Starting from 9.2.0.1, you can store the encrypted dataset passwords in the Eclipse secure storage location on the computer. Now, when you run the tests from the command line, the product automatically uses the password and completes the test run. Prior to 9.2.0.1, you could not run the tests from the command line with encrypted datasets.

1. Click **Window > Preferences > Test > Test Execution > Automation Security**.
2. Select the **Activate Secure Storage Support for Encrypted Datasets** check box.
   The password is stored in the Eclipse secure storage. Do not share the computer's login credentials with others.
3. To add the encrypted datasets, click **Add**, select the encrypted dataset, and click **OK**.
   You will be prompted to enter the dataset password that you used when encrypting the dataset.
4. Enter the password and click **OK**.

**Results**

When you run the tests from the command line, the test runs will complete successfully without the need to specify the password. If another user runs the same tests with encrypted datasets, the dataset password must be entered for the tests to run successfully.

## Exporting report counters automatically

You can change the test preferences so that report counters are automatically exported at the end of a run. This option is useful when you run a schedule from the command line because you can automatically export results without opening the workbench.

To automatically export report counters to a CSV file when a test or schedule is complete:

1. Open the Preferences page. Click **Windows > Preferences**.
2. Open the Export Reports page. In the **Preferences** window, expand **Test** and **Performance Test Reports**, and then select **Export Reports**.
3. In the Export Reports window, select the options as follows:

| Option | Description |
| --- | --- |
| **Export reports when run completes from** | Select **Command line** or **Workbench**. If you select **Command line**, you can also select **Print simple CSV reports to command line** to display the exported data on the command line (standard output) as well as export it to the CSV file. The file is displayed after the command line run has completed. |
| **Simple (counters as lines, time ranges as columns)** | To export a simple report in the CSV format, select this check box. |

| Option | Description |
|---|---|
| | **List All Time Ranges** - Select this check box to include data from all the time ranges. The Entire Run time range is included by default.<br><br>**Include per instance counters**: Select this check box to include counters for all the page elements.<br><br>**Export each agent separately** - Select this check box to export data for each agent to a separate CSV file.<br><br>**One file per agent** - Select this check box to export data for each agent (location) to different sections in a single CSV file. |
| **Full (time intervals as lines, counters as columns)** | To export a comprehensive report that includes the result name, node name, and time ranges, select this check box. Typically, you do not include these details unless you are exporting customized reports that include counters from specific test runs.<br><br>**Split output if column exceeds** - Because there will be a lot of data, select this check box to create multiple CSV files if the number of columns exceed the specified limit.<br><br>**Include per instance counters** - Select this check box to include counters for all the page elements.<br><br>**Export each agent separately** - Select this check box to export data for each agent (location) to different sections in the same CSV file.<br><br>**One file per agent** - Select this check box to export data for each agent to a separate CSV file. |
| **HTML report** | Select this check box to export full report data in the HTML format. |
| **Executive Summary report** | Select this check box to export the report in the HTML format. This report summarizes the state of the test or schedule run and display the report on only one HTML page. It can be printed. |

| Option | Description |
|---|---|
| **Select reports to export** | Expand the tree, if necessary, to display the type of report to export. If you select more than one report, each report is exported to a separate CSV file in the Test Runs directory.<br><br>**Show Report Ids** - Select this check box to view the ID of each report. The IDs are used when exporting the specific reports from command line. |

4. Click **Apply**.

**Results**

The CSV file is called `results_file_name.report_name.csv`. It contains metadata about the test run, a blank line, and then lists each counter and its last value. Each counter is on a separate line.

## Running a test from a command line

You can run a test without opening the product by using the command-line interface.

1. Navigate to the directory that contains the `cmdline.bat` and `cmdline.sh` files. On Windows™ operating systems, this directory is typically found as `productInstallationDirectory/cmdline`. For example, `C:\Program Files\IBM\SDP\cmdline`.
2. Issue the following command:
   **Example**

   > 📝 **Note:**
   >> ◦ The workspace is locked after you issue the command. To check the progress of the test during the run, invoke another workspace and open the project through that workspace.
   >> ◦ On Linux operating system, the command must start with `cmdline.sh`.

   If a value contains spaces, enclose the value in quotation marks. To see the online help for this command while you are in the directory that contains the .bat file, type `cmdline -help`.

   The following table explains each option:

| -work-space | Required. The complete path to the Eclipse workspace. |
|---|---|
| **-project** | Required. The path, including the file name of the project relative to the workspace. |
| -eclipse-home | Optional. The complete path to the directory that contains `eclipse.exe`. |

| | |
|---|---|
| **-plugins** | Optional. The complete path to the folder that contains the plugins. Typically, on Windows operating systems, this folder is located at `C:\Program Files\IBM\IBMIMShared\plugins`.<br><br>Required. This option is required only if the folder is at a different location. |
| **-suite** | Optional. However, in a command, it is mandatory to use one of the following options:<br><br>    ◦ -suite<br><br>You must not use the -suite option along with the other options. The path includes the file name of the suite to run relative to the project.<br><br>Starting from V9.2.1.1, you can execute multiple tests simultaneously.<br><br>For example, **-suite** *test1:test2:test3*. |
| **-im-portzip** | Optional. To import the project as test assets with dependencies into your workspace, use the -importzip option. This command is available from V9.2.1.1 and later.<br><br>For example, `C:\User\Desktop\test1.zip` |
| **-varfile** | Optional. You can use this option to specify the complete path to the XML file that contains the variable name and value pairs. |
| **-config-file** | Optional. You can use this option to specify the complete path to a file that contains the parameters for a test run. Each parameter must be on a single line. To create a configuration file, you must use an editor that does not wrap lines. Any parameters, whether required or optional, can be set in the configuration file. The command line parameters override the values in this file.<br><br>✏️ **Notes:**<br><br>    ◦ If you are creating a config file manually, the file must be in the UTF-8 format. You must not use quotation marks in this file even for values that contain spaces.<br><br>    ◦ You can create command line config file from the product, which you can use while running tests or schedules from the command-line interface or Maven. For more information about how to create a command line config file from the product, see related links. |
| **-results** | Optional. You can use this option to specify the name of the results file. The default result file name is the test name with a time stamp appended. You must specify a folder name that is relative to the project to store the test results.<br><br>For example, **-results** *folder/resultname* |

| | |
|---|---|
| **-over-write** | Optional. Determines whether a results file with the same name is overwritten. The default value, `false`, indicates that the new results file is created. If the value is `true`, the file is overwritten and retains the same file name. You must use double quotes `""` for values `true` or `false`. |
| **-quiet** | Optional. Turns off any message output from the launcher and returns to the command shell when the run or the attempt is complete. |
| **-vmargs** | Optional. To specify the Java™ maximum heap size for the Java™ process that controls the command line playback, use the **-vmargs** option with the *-Xmx* argument.<br><br>For example, when you use**-vmargs** *-Xmx4096m*, specify a maximum heap size of 4096m. This method is similar to specifying *-Xmx4096m* in the eclipse.ini file for the workbench when playing back the test from the user interface. |
| **-publish** | Optional. You can use this parameter to publish test results to the IBM® Rational® Test Automation Server. You can use the following options along with the **-publish** parameter:<br>  ◦ *serverURL*#project.name=*projectName*: You can use this option to specify the project name. If the project name is not specified, the name of the current project is used.<br><br>     📝 **Note:** If you provide the server and the project details under **Window > Preferences > Test > Rational Test Automation Server** in the product, and if you use *serverURL*#project.name=*projectName* along with the **-publish** parameter, the server details in the command-line interface takes precedence over the product preferences.<br><br>  ◦ `no`: You can use this option if you do not want to publish test results after the run. This option is useful if the workbench preferences are set to publish the results, but you do not want to publish them.<br><br>  ❗ **Important:** You must provide the offline user token for the server by using the **RTCP_OFFLINE_TOKEN** environment variable before you use the **-publish** parameter in the command-line interface. |
| **-publish_for** | Optional. You can use this option to publish the test results based on the completion status of the tests:<br>  ◦ ALL - This is the default option. You can use this option to publish test results for any text execution verdict.<br>  ◦ PASS - You can use this option to publish test results for the tests that have passed.<br>  ◦ FAIL - You can use this option to publish test results for the tests that have failed.<br>  ◦ ERROR - You can use this option to publish test results for the tests that included errors.<br>  ◦ INCONCLUSIVE - You can use this option to publish test results for the tests that were inconclusive.<br>You can add multiple parameters separated by a comma. |

| | |
|---|---|
| **-export-log** | Optional. You can use this parameter to specify the file directory path to store the exported HTTP test log.<br><br>Starting from V10.0.1, by using the **-exportlog** parameter, you can provide multiple parameter entries when running multiple tests. You must use a colon to separate the parameter entries.<br><br>For example: **-exportlog** *c:/logexport.txt:c:/secondlogexport.txt*<br><br>If there are multiple **-suite** parameter entries with a single **-exportlog** parameter entry, then the **-exportlog** parameter generates the appropriate number of test logs by appending 0, 1, 2, and so on to the **-exportlog** parameter entry name.<br><br>For example: **-suite** *"sampletest1:sampletest2:sampletest3"* **-exportlog** *c:/logexport.txt* The command generates the following test logs:<br><br>    ◦ logexport_0.txt<br>    ◦ logexport_1.txt<br>    ◦ logexport.txt<br><br>The last test log generated has the same name as that of the initial **-exportlog** entry.<br><br>📝 **Note:** If there are multiple **-suite** and **-exportlog** parameter entries, the number of **-suite** entries must match with the number of **-exportlog** entries. Otherwise, the following error message is displayed:<br><br>`Error, number of -suite and -exportlog entries do not match.` |
| **-export-stats** | Optional. You can use this option to export reports in comma-separated values (CSV) format, with the file name derived from the report name. This directory can be relative to the project or a directory on your file system. If the **-exportstatreportlist** option is not specified, the reports specified on the **Export Reports** page of the **Performance Test Report** preferences are exported. |
| **-export-stat-shtml** | Optional. When you want to export web analytic results, you can use this option. The results are exported in the specified directory. You can then analyze the results on a web browser without using the test workbench. |
| **-com-pare** | You can use this argument along with **-exportstatshtml** and **-execsummary** to export the result in compare mode. The value can be paths to the runs and are relative to the workspace. You must separate the paths by a comma. |
| **-export-statre-portlist** | Optional. You can use this option to specify a comma-separated list of report IDs along with **-exportstats** or **-exportstatshtml** to list the reports that you want to export in place of the default reports, or the reports selected under **Preferences**. To view this setting, navigate to **Window > Preferences > Test > Performance Test Reports > Export Reports**. |

| | To copy the report IDs list into your command line, navigate to **Window > Preferences > Test > Performance Test Reports > Export Reports**. Under **Select reports to export**, select the required reports, and click **Copy ID to clipboard**. You can then paste the clipboard content on to your command line editor |
|---|---|
| **-exec-summa-ry** | Optional. You can use this option to export all of the reports for the test run in a printable format, also known as an executive summary, to the local computer. You must specify the path to store the executive summary. |
| **-exec-summa-ryreport** | Optional. You can use this option to export a specific report as an executive summary for the test run to the local computer. You must specify the ID of the report to export.<br><br>For example, to export an HTTP performance report, specify `http`. You must use this option along with **-execsummary**.<br><br>To copy the report IDs list into your command line, navigate to **Window > Preferences > Test > Performance Test Reports > Export Reports**. Under **Select reports to export**, select the required reports, and click **Copy ID to clipboard**. You can then paste the clipboard content on to your command line editor |
| **-user-com-ments** | Optional. You can add text within double quotation mark (*""*) to display it in the **User Comments** row of the report.<br><br>📝 **Note:** You can use the file `CommandLine.exe` to run the command to add comments in a language that might not support Unicode characters on Windows operating system. |
| **-pub-lishre-ports** | Optional. You can use this option to publish test results in Rational® Test Automation Server. The parameters that you can use with it are the following:<br>  ◦ FUNCTIONAL<br>  ◦ MOBILE_WEBUI<br>  ◦ STATS<br>  ◦ TESTLOG<br>For example, **-publishreports** *"STATS, TESTLOG"*<br><br>You must prefix with *"!"* to publish all the reports except the specified one.<br><br>For example, **-publishreports** *"! TESTLOG"*<br><br>All the reports except the TESTLOG report is published to Rational® Test Automation Server after executing the command. |
| **-stdout** | Optional. You can use this option to display the information about the test on the command line. |

| | |
|---|---|
| | After you run a test from the command line, the following outputs are displayed to give you the overall information of the test :<br><br>◦ --VERDICT: The verdict of the test .<br>◦ --REMOTE_RESULT: The URL of the result published to Rational® Test Automation Server.<br>◦ --REMOTE_RESULT_UI: The URL of the result published to Rational® Test Automation Server and can be opened in a browser to analyze the result.<br>◦ --LOCAL_RESULT: The path of the result saved locally.<br><br>For example, **-workspace** *workspace_full_path* **-project** *proj_rel_path* **-publish***publish_url* **-stdout** |
| **-swap-datasets** | Optional. Use this option to replace dataset values during a test . If a test is associated with a dataset, you can replace the dataset at run time while initiating the run from the command line.<br><br>You must ensure that both original and new datasets are in the same workspace and have the same column names. You must also include the path to the dataset when you run the -swap-datasets command.<br><br>For example, **-swapdatsets** */project_name/ds_path/ds_filename.csv:/project_name/ds_-path/new_ds_filename.csv*<br><br>You can swap multiple datasets that are saved in a different project by adding multiple paths to the dataset separated by a semicolon.<br><br>For example, **-swapdatsets** */project_name1/ds_path/ds_filename.csv:/project_name1/ds_-path/new_ds_filename.csv;/project_name2/ds_path/ds_filename.csv:/project_name2/ds_-path/new_ds_filename.csv* |
| **-history** | Use this command when you want to view a record of all events that occurred during a *test* run. However, you must use the command suffixed with any of the following options:<br><br>◦ jaeger: To send test logs to the Jaeger UI during the *test* run.<br>◦ testlog: To send test logs as traditional test logs in during the *test* or *schedule* run.<br>◦ null: To send no test logs either to the Jaeger UI or during the *test* run.<br><br>For example:<br><br>```
-workspace  workspace_full_path  -project  proj_rel_path
-suite  suite_rel_path  -stdout  -history  comma delimited list of modes
```<br><br>```
-workspace  C:/Users/IBM/rationalsdp/test_ws  -project  Project1
-suite  test1.testsuite  -stdout  -history  jaeger
```<br><br>✎ **Note:** You can add multiple options separated by a comma to send test logs during the *test* run to and the Jaeger UI.<br><br>For example: |

> 🖊 **-workspace** *C:/Users/IBM/rationalsdp/test_ws* **-project** *Project1*
> **-suite** *test1.testsuite* **-stdout** **-history** *jaeger, testlog*
>
> For more information about how to view test logs in the Jaeger UI and , see related links.

To stop the test run, you can open another command prompt window and use one of the following options with the cmdline option:

| Command | Description |
|---|---|
| **-stoprun** | Optional. Stops the test run after the specified number of seconds. The block is executed, and the test log is transferred before stopping the run. You must use the **-workspace** command and specify the location of the workspace. |
| **-abandonrun** | Optional. Stops the test run immediately. You must use the **-workspace** command and specify the location of the workspace. |

> 🖊 **Note:** Messages are displayed to indicate when the test is launched and when it is completed unless you include the -quiet option.

**Exemple**

## Examples of the commands for running tests from the command line

You can run tests from the command line either by using a configuration file or by directly specifying the path of the test in the command. Each command-line option must be followed by an appropriate value.

The contents of a sample configuration file, `config_file1` are as follows:

```
workspace=D:\My Workspace
eclipsehome=C:\Program Files\IBM\SDP
plugins=C:\Program Files\IBM\IBMIMShared\plugins
project=myProject
suite=mytestsuite
```

To run tests from the command line by using the sample config file `config_file1` you must use the following command:

cmdline −configfile <config file path>

For example:

cmdline −configfile E:\Workspace1\Project1\Tests\config_file1.txt

To run the tests from the command line without using a configuration file, you must specify the path of the tests along with the command as follows:

cmdline <path of the test>

For example:

cmdline -workspace "D:\My Workspace" -eclipsehome "C:\Program Files\IBM\SDP" -plugins "C:\Program Files\IBM\IBMIMShared\plugins" -project myProject -suite mytestsuite

The -workspace command-line option is followed by a value that contains a space. If the value contains space, then you must enclose the value, D:\My Workspace within quotes. Otherwise, you can provide the value without quotes.

**What to do next**

After you run the test , you may want to export the results for further analysis. For more information, see <span style="color:blue">Exporting report counters automatically on page</span> <span style="color:blue"></span> .

## Controlling cache sizes

If you use an infinite loop and the number of cached responses in a test increases exponentially, you can set a limit to cache for a user group in the schedule.

**About this task**

When the cache limit is reached, the least-recently accessed cached entry is released to accommodate a new entry. Also, when a test follows another test in the schedule, you can clear the cache before a test starts.

1. To clear the cache before a test starts, from the Test Navigator, open a test.
2. Click the **HTTP Options** tab and select the **Clear page cache when the test starts** check box.
3. To set a limit to the number of cache entries, in the Test Navigator, navigate to a schedule and double-click it to open it.
4. Click the user group for which you want set the cache limit.
5. Click the **Options** tab and then click **Edit Options**.
6. Select the **Set cache size limit** check box and, in the **Maximum cache size** field, type a numeric value. This value indicates the number of entries allowed for a user.
7. Click **OK** and save the schedule.

## Increasing memory allocation

The virtual users that access your web server require memory to prepare requests, send requests, and receive responses. Because the amount of memory is not automatically set on remote computers, you might receive an out-of-memory error. To correct this situation, increase the memory allocation for that computer.

**About this task**

If you receive an out-of-memory error when you run a test or schedule, override the default amount of memory that is allocated for that computer. To do this, set the RPT_VMARGS property, which overrides RPT_DEFAULT_MEMORY_SIZE. After the first successful execution, IBM® Rational® Performance Tester

automatically sets value for RPT_DEFAULT_MEMORY_SIZE, which represents the maximum heap that will be specified by Rational® Performance Tester in subsequent executions.

**Note:** Ensure there is at least one successful execution after all locations are created so RPT_DEFAULT_MEMORY_SIZE exists.

**Tip:**

If you see out-of-memory issues, it is a good practice to first check the `javacore*` file. You can also look at the results and verify that the server is responding correctly because many times errors can lead to excessive resource consumption. You can also monitor memory usage with Task Manager or other tools at varying user load levels such as 10, 50, 100, 500 or 1000 users and use that data to make an estimate of the memory needs per virtual user and then project memory requirements for larger user loads. In some cases the best solution is to add another agent.

Rational® Performance Tester sets heap size for RPT_DEFAULT_MEMORY_SIZE based on the bit-type of the JRE:

- For 32-bit Java Runtime Environment (JREs), Rational® Performance Tester sets 70% of the size of physical memory to RPT_DEFAULT_MEMORY_SIZE. Typically, the maximum limit is set to 1200m.
- For 64-bit JREs, some workloads might perform better with a lesser heap size than 70% of physical memory up to a maximum of 12000m.

To increase the memory allocation on a remote computer:

1. In the Test Navigator (from your local computer), expand the project until you find the deployment location that you want to change.

   Deployment locations are represented by the  icon.
2. Right-click the deployment location, and then click **Open**.
3. Under **Property Groups**, click the **General Properties** link, and then click **Add**.
4. In the New Property window:

   a. In the **Property Name** field, type `RPT_VMARGS`.

   b. In the **Operator** field, confirm that the operator is `=`.

   c. In the **Property Value** field, type `-Xmxnnnnm`, where *nnnn* is the amount of memory, in megabytes, and then click **OK**.
      **Example**

   If you need to set multiple `RPT_VMARGS` values for a location, place them in the same property entry and separate them with a space. Do not use multiple property entries to set multiple `RPT_VMARGS` values for a location.
   **Result**

The following **New Property** window sets maximum heap to 1200 megabytes:



> **ⓘ** **Tip:** It is a good practice is to monitor memory usage with Task Manager or other tools at varying user load levels such as 10, 50, 100, 500 or 1000 users and use that data to make an estimate of the memory needs per virtual user and then project memory requirements for larger user loads. In some cases the best solution is to add another agent.

**What to do next**

If you have increased the available memory and you still receive out-of-memory errors, add more remote computers for your user groups. For information about how to do this, see Running a user group at a remote location.

# Extending test execution with custom code

You can extend how you run your tests by writing custom Java™ code and calling the code from the test. You can also specify that results from the tests that are affected by your custom code be included in reports.

## Creating custom Java™ code

Custom code uses references in the test as input and returns modified values to the test. Use the ICustomCode2 interface to create custom code and the ITestExecutionServices interface to extend test execution. These interfaces are contained in the `com.ibm.rational.test.lt.kernel.services` package.

**About this task**

> **✐** **Note:** When you use the ITestExecutionServices interface in your custom code to report test results, the results for the custom code are displayed in the test log. If you log custom verification point verdicts, these are reflected in the overall schedule verdict.

Custom code input values can be located in references or field references. You can also pass a text string as an argument to custom code. References that are used as input to custom code must be included in the same test as the custom code. In the test, the reference must precede the code that it affects. Verify that the test contains the references that are required for customized inputs to your code. For details about creating references and field references, see Creating a reference or field reference on page 246.

If your custom code uses external JAR files, you might need to change the Java™ build path. In some cases, you can avoid changing the build path manually by running the test before adding your custom code to it. The first time a test runs, classes and libraries that are required for compilation are added to the build path. For example, you can import Test and Performance Tools Platform (TPTP) classes that are required to create custom events in the test log if the test, to which you have added your custom code, has run previously. However, if the test has never been run, import errors occur because the classes are not named in the build path for the project until the test has run.

If your code uses external resources, for example, an SQL database or a product that manages customer relationships, you must configure the custom code to work on every computer on which your test runs.

Custom code is saved in the `src` folder of the project that contains the test that calls the code. By default, custom code is located in a package named `test` in the `src` folder.

You can reuse a custom code package for tests that are located in multiple projects. The projects must be in one workspace. To reuse custom code across projects, use the project name before the custom code package. For



example, .

The following example shows the standard Navigator view of two custom code classes. (The Test Navigator does not display Java™ source files.)



When you add the `ReplaceCC.java` and `VerifyYUserID.java` custom code classes to the test and return a value to the test, **Substitute** lists these two classes.

The test package also contains the generated Java™ code for tests in the project.

You can put custom code in a different package (for example, `custom`). Separate custom code from generated code, especially if you use a source-control system.

To add custom code:

1. Open the test, and select a test element.
2. Click **Add** or **Insert**, and select **Custom Code**.
   **Add** appends the custom code to the bottom of the selected test element. **Insert** adds the custom code above the selected test element.

> **Note:** After you add or insert custom code, the Problems view displays an error stating that the new custom code element has no Java™ file. This error message remains until you click **View Code** or **Generate Code**, to remind you that the custom code test element is not yet associated with any Java™ code.

3. Inspect the **Class name** field, and complete one of these steps:
    ◦ If the code to call already exists, change the class name to match its name. Click **View Code** to open the code in the Java™ editor.
    ◦ If the code does not exist, change the class name to describe the purpose of the code. Click **Generate Code** to generate a template class for logging results and to open it in the Java™ editor. If a class with this name exists, you are warned that it will be overwritten.
4. In the **Arguments** field, click **Add**.
5. In the **Custom Code** window, select all inputs that your code requires.
    The **Custom Code** window lists all values in the test that can be used as inputs to your code (references or field references in the test that precede the code).
6. Click **OK**.
    **Result**
    The window closes, the selected references are added to the **Arguments** field.
7. To add text strings as inputs to your custom code, click **Text**, and then type the text string to use.
8. In the test, after your custom code, locate a value that your code returns to the test.
9. Highlight the value.
10. Right-click the highlighted value, click **Substitute**, and select the class name of your custom code.
    **Result**
    The custom code classes that you have added are listed. After you have made your selection, the value to be returned to the test is highlighted in orange, and the **Used by** table is updated with this information.

**What to do next**

Custom code is not displayed in the **Test Navigator** view. To view custom code, open the **Package Explorer** view and use the Java™ tools to identify the custom code that you added.

## Test execution services interfaces and classes

You use the test execution services interfaces and classes to customize how you run tests. These interfaces and classes are located in the com.ibm.rational.test.lt.kernel package. Each interface and class is described briefly in this topic and in detail in the Javadoc information.

The Javadoc for the test execution services interfaces and classes can be accessed from the product by clicking **Help > Help Contents**IBM Rational Performance Tester **API Reference**.

| Interface | Description |
| --- | --- |
| ICustom-Code2 | Defines customized Java™ code for test execution services. Use this interface to create all custom code. |
| ITestEx-ecution-Services | Provides information for adding custom test execution features to tests. Replaces the IKLog interface. All the methods that were available in IKLog are contained in ITestExecutionServices, along with several newly exposed objects and interfaces. This interface is the primary interface for execution services. ITes-tExecutionServices contains the following interfaces: IDataArea, IARM, ILoopControl, IPDLogManager, IS-tatisticsManager2, ITestLogManager, ITime, and ITransaction. |
| IDataArea | Defines methods for storing and accessing objects in data areas. A data area is a container that holds objects. The elements of a data area are similar to program variables and are scoped to the owning con-tainer. To use objects specific to a protocol, you should use objects provided by that protocol that are stored in the protocol-specific data area. |
| IARM | Provides information about defining ARM (Application Response Measurement) specifications. You use this interface if your virtual users are being sampled for ARM processing. |
| ILoop-Control | Provides control over loops in a test or schedule. For example, you can use this interface to break loops at specific points in a test. The loop that is affected is the nearest containing loop found in either the test or the schedule. |
| IPDLog-Manager | Provides logging information such as problem severity, location levels, and error messages. |
| IStatistic-sManag-er | Provides access to performance counters in the ICustomCode2 interface (used for defining custom code). Performance counters are stored in a hierarchy of counters. Periodically, all the counter values in the hierarchy are reported to the testing workbench and collected into test run results, where they are available for use in reports and graphs. Each counter in the hierarchy has a type (defined in class `Stat-Type`). The operations that are available on a counter depend on the counter's type. |
| ITestLog-Manager | Logs messages and verification points to the test log. Use this interface for handling error conditions, anomalies in expected data or other abstract conditions that need to be reported to users, or for compar-isons or verifications whose outcome is reported to the test log. ITestLogManager can also convey infor-mational or status messages after the completion of a test. |
| ITime | Defines basic time services, such as the current system time in milliseconds (adjusted so that all sys-tems are synchronized with the schedule controller), the time the test begins, and the elapsed time from the beginning of the test. |
| ITransac-tion | Provides support for transactions. A collection of named transactions is maintained for each virtual user. Transactions created in custom code can be started and stopped wherever custom code can be used. These transactions can span several tests. Performance counters are kept for custom code transactions and appear in reports. An example of how you could use ITransaction is to create transactions for one virtual user but not another, to help verify responses from tests. |

| Interface | Description |
| --- | --- |
| IEngineInfo | Provides information about the testing execution engine; for example, the number of virtual users running in this engine, the number of virtual users that have completed, the local directory in which test assets are deployed, and the host name of the computer on which the engine runs. |
| ITestInfo | Provides information about the test that is running; for example, the test name and information about the current problem determination log level for this test. |
| IVirtualUserInfo | Provides information about virtual users; for example, the virtual user's name, problem determination log level, TestLog level, globally unique ID, and user group name. |
| IScalar | Provides methods for simple integer performance counters. It is used for counters of `SCALAR` and `STATIC` types. Use this interface to decrement and increment counters. |
| IStat | Defines observational performance counters. It defines the method for submitting a data point to performance counters of type `RATE`, `AVERAGE`, and `RANGE`. |
| IStatistics | Retrieves the performance counter tree associated with the current statistics processor. Stops the delivery of performance counters. Changes the priority of the statistics delivery thread. |
| IStatTree | Provides methods that can retrieve child counters, create the XML fragments that define counters, and set the description field of counters. |
| IText | Contains text-based performance counters. Performance counters that do not fit any of the other counter types can be created as type `TEXT`. `TEXT` counters are not assigned definitions, but they are collected in the test results. |

| Class | Description |
| --- | --- |
| `DataAreaLockException` | Throws an exception whenever an attempt is made to modify a locked DataArea key. |
| `OutOfScopeException` | Indicates that an object created by ITestExecutionServices has been referenced outside of its intended scope. |
| `Transaction-` | Throws an exception when a transaction is misused. The following conditions lead to a `TransactionException` exception: attempting to start a transaction that has already been started, attempting to stop a transaction that has not been started, and getting the start time or the elapsed time of a transaction that has not been |

| Class | Description |
|---|---|
| `Ex-cep-tion` | started. Any operation (except abort()) on a transaction that has been aborted will throw a `TransactionExcep-tion` exception. |
| `Stat-Type` | Provides a list of valid performance counter types. The performance counter types are: `AVERAGE`, `iAVERAGE`, `iRANGE`, `iRATE`, `iSCALAR`, `iSTATIC`, `iSTRUCTURE`, `iTEXT`, `RANGE`, `RATE`, `SCALAR`, `STATIC`, `STRUCTURE`, and `TEXT`. |

## Reducing the performance impact of custom code

If custom code runs inside a page, it can affect that page's response time.

HTTP pages are containers of HTTP requests. On a given HTTP page, requests run in parallel across all of the connections between the agent computer and the system under test.

*Page response time* is the interval between *page start* and *page end*, which are defined as follows: Page start is the first timestamp associated with the client-server interaction. This interaction is either the first byte sent or the first connect of the first HTTP request. Page end is the last timestamp associated with the client-server interaction. This interaction is the last byte received of the last HTTP request to complete. Because of parallelism, the last HTTP request to complete might not be the last one listed for the page.

Typically, you should not insert custom code inside a page. While custom code that runs for only a few milliseconds should have little effect on page response time, the best practice is to place custom code outside a page. Custom code placed outside a page has no effect on page response time, and its execution time can overlap with think time delays.

Do not use custom code for data correlation if you can instead use the data correlation features built into the product. The built-in data correlation code takes advantage of requests running in parallel, whereas custom code actions do not begin until all earlier actions are completed.

You might need to place custom code inside a page to correlate a string from the response of a request inside that page to another request inside the same page. Even in this case, if you split the page into two pages, you can use the built-in data correlation features instead of custom code.

If you still want to run tests with custom code inside HTTP pages, use the Page Element report to evaluate performance. The Page Element report shows the response time and throughput for individual HTTP requests. Custom code does not affect the response time measurement of individual HTTP requests.

Related information

[Performance testing tips on page](#)

## Custom code examples

Custom code enables you to perform such tasks as managing loops, retrieving virtual user information, running external programs from tests, and customizing data correlation.

## Controlling loops

This example demonstrates extending test execution by using custom code to control loops. It provides sample code that shows how you can manipulate the behavior of loops within a test to better analyze and verify test results.

This example uses a recording of a stock purchase transaction using the Trade application. The concepts shown here can be used in tests of other applications.

The test begins with a recording of a stock purchase transaction, using dataset substitution for the login IDs. The pages are wrapped in a five-iteration loop, as shown in the following figure:



Notice that among the various pages of the test, three items of custom code exist (indicated by the green circles with "C"s in them). This example explores these items of custom code.

The Javadoc for the test execution services interfaces and classes can be accessed from the product by clicking **Help > Help Contents**IBM Rational Performance Tester **API Reference**.

The first piece of custom code, `InitializeBuyTest`, is mentioned here:

```
package customcode;
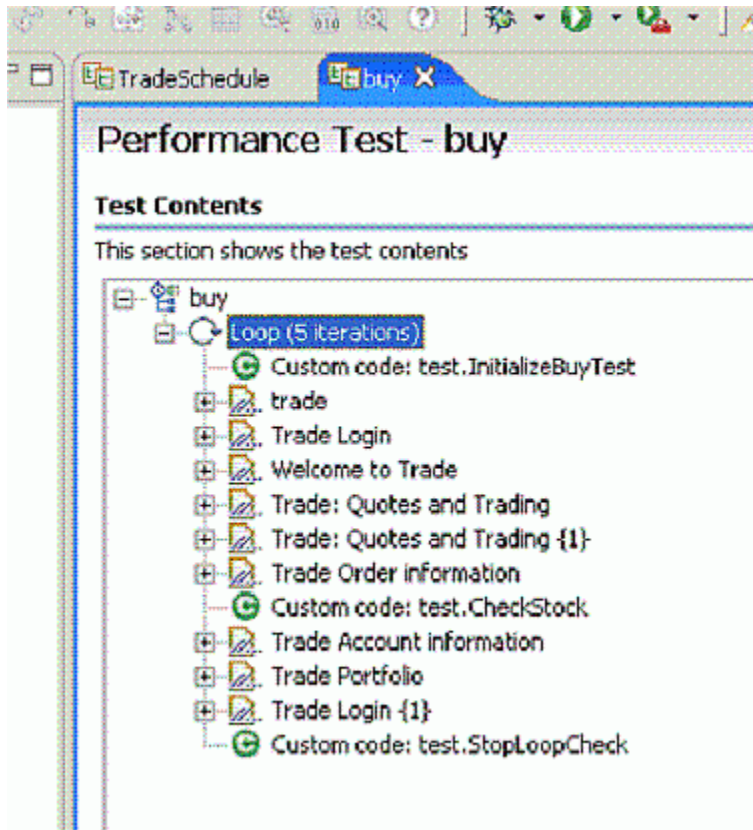
import java.util.Random;

import com.ibm.rational.test.lt.kernel.IDataArea;
import com.ibm.rational.test.lt.kernel.services.ITestExecutionServices;
```

```
import com.ibm.rational.test.lt.kernel.services.IVirtualUserInfo;

/**
 * @author unknown
 */
public class InitializeBuyTest implements
  com.ibm.rational.test.lt.kernel.custom.ICustomCode2 {

 /**
  * Instances of this will be created using the no-arg constructor.
  */
 public InitializeBuyTest() {
 }

 /**
  * For description of ICustomCode2 and ITestExecutionServices interfaces,
  * see the Javadoc.. */
 public String exec(ITestExecutionServices tes, String[] args) {
  // Get the test's data area and set a flag indicating that nothing
  // has failed yet. This flag will be used later to break out
  // of the schedule loop as soon as a failure is encountered.
  IDataArea dataArea = tes.findDataArea(IDataArea.TEST);
  dataArea.put("failedYet", "false");

  // Get the virtual users's data area
  IDataArea vda = tes.findDataArea(IDataArea.VIRTUALUSER);

  // Randomly select a stock to purchase from the set of s:0 to s:499.
     IVirtualUserInfo vuInfo = (IVirtualUserInfo) vda.get(IVirtualUserInfo.KEY);
     Random rand = vuInfo.getRandom();
  String stock = "s:" + Integer.toString(rand.nextInt(499));

  // Persist the name of the stock in the virtual user's data area.
  vda.put("myStock", stock);

  return stock;
 }
```

This custom code is located in the method `exec()`.

First, the data area for the test is acquired to store a flag value, in this case a string of text, to be used later to stop the test loop when an error is discovered. Data stored in this way can be persisted across tests.

Then a randomly generated stock string is created. The value is stored as the variable *stock*, and is passed back as the return value for the method. This return value is used as a substitute in a request later, as shown in the following figure:

The highlighted item uses a substitution (`s%3A716`), which is the value returned by the `InitializeBuyTest` custom code item. We are using custom code to drive the direction of our test.

The next lines of code in `InitializeBuyTest` use the Virtual User data area to store the name of the stock for later reference. Again, data stored in this way can persist across tests.

The second piece of custom code is called `CheckStock`. Its contents are as follows (listing only the `exec()` method this time):

```
public String exec(ITestExecutionServices tes, String[] args) {

  // Get the actual and requested stock purchased.
  String actualStock = args[0].replaceAll("<B>", "");
   actualStock = actualStock.substring(0, actualStock.indexOf("<"));
  String requestedStock = args[1];

   // Set the log level to ALL.
   IDataArea dataArea = tes.findDataArea(IDataArea.TEST);
   ITestInfo testInfo = (ITestInfo)dataArea.get(ITestInfo.KEY);
   testInfo.setTestLogLevel(ITestLogManager.ALL);

   // If the log level is set to ALL, report the actual and requested stock
  // purchased.
   ITestLogManager testLogManager = tes.getTestLogManager();
  if (testLogManager.wouldReport(ITestLogManager.ALL)) {
    testLogManager.reportMessage("Actual stock purchased: "
      + actualStock + ". Requested stock: " + requestedStock
      + ".");
  }

  // If the actual and requested stock don't match, submit a FAIL verdict.
   if (testLogManager.wouldReport(ITestLogManager.ALL)) {
   if (!actualStock.equalsIgnoreCase(requestedStock)) {
    testLogManager.reportVerdict(
      "Actual and requested purchase stock do not match.",
       VerdictEvent.VERDICT_FAIL);

    // Use the test's data area to record the fact that an error has
   // occurred.
    dataArea.put("failedYet", "true");
   }
```

```
    }
      return null;
    }
```

This code begins by extracting two arguments that have been passed to the code. A part of the response in the original recording is highlighted and used as a reference, as shown in the following figure.



Some string manipulation is needed to acquire the text of interest; in this case, the name of the stock that was actually purchased. This newly created reference is then passed into `CheckStock` as an argument, as shown in the following figure:



Note that the return value of `InitializeBuyTest` is passed in as an argument as well.

The `CheckStock` custom code item uses these values to verify that the randomly chosen stock generated by `InitializeBuyTest` is actually purchased during the execution of the test.

`CheckStock` then sets the test log level, reports the actual and requested stock purchase, and raises a FAIL verdict if they do not match. `CheckStock` also stores a `true` value associated with the tag `failedYet` in the test's data area.

The third piece of custom code (`exec()` method only) is mentioned here:

```
public String exec(ITestExecutionServices tes, String[] args) {

  // Get the test log manager.
```

```
  ITestLogManager testLogManager = tes.getTestLogManager();

  // Get the test's data area and get a flag indicating to
  // see if anything has failed yet. If so, stop the loop.
  IDataArea dataArea = tes.findDataArea(IDataArea.TEST);
  String failedYet = (String) dataArea.get("failedYet");

  // Break out of the loop if an error has been encountered.
  if (failedYet.equalsIgnoreCase("true")) {
   tes.getLoopControl().breakLoop();

   if (testLogManager.wouldReport(ITestLogManager.ALL)) {
    testLogManager.reportMessage("Loop stopped.");
   }
  }

  return null;
 }
```

This code uses the test's data area to determine the user-defined value associated with the tag `failedYet`. If `failedYet` is `true`, `StopLoopCheck` breaks out of the test loop.

## Retrieving the IP address of a virtual user

This example shows how to retrieve the local IP address of a virtual user. Retrieving IP addresses is particularly useful when virtual users are using IP aliases.

The following custom code retrieves the IP address that was assigned to a virtual user:

```
import java.net.InetAddress;
import com.ibm.rational.test.lt.kernel.IDataArea;
import com.ibm.rational.test.lt.kernel.services.ITestLogManager;
import com.ibm.rational.test.lt.kernel.services.IVirtualUserInfo;

public String exec(ITestExecutionServices tes, String[] args) {
 IVirtualUserInfo vui = (IVirtualUserInfo)
 tes.findDataArea(IDataArea.VIRTUALUSER).get(IVirtualUserInfo.KEY);
 ITestLogManager tlm = tes.getTestLogManager();

 if (vui != null) {
  String localAddr = null;
  InetAddress ipAddr = vui.getIPAddress();
  if (ipAddr != null)
   localAddr = ipAddr.toString();
  tlm.reportMessage("IPAlias address is " + (localAddr != null ? localAddr : "not set"));
  return localAddr;
 }
else
  return ("Virtual User Info not found");
}
```

## Printing input arguments to a file

The PrintArgs class prints its input arguments to the file C:\arguments.out. This class could be used, for example, to print a response returned by the server.

**Exemple**

```
package customcode;

import com.ibm.rational.test.lt.kernel.services.ITestExecutionServices;

import java.io.*;

/**
 * The PrintArgs class prints its input arguments to the file
 * C:\arguments.out.  This example could be used to print a response
 * returned by the server.
 */

/**
 * @author IBM Custom Code Samples
 */

public class PrintArgs implements
        com.ibm.rational.test.lt.kernel.custom.ICustomCode2 {

    /**
     * Instances of this will be created using the no-arg constructor.
     */
    public PrintArgs() {
    }

    public String exec(ITestExecutionServices tes, String[] args) {
        try {
            FileWriter outFile = new FileWriter("C:\\arguments.out");
            for (int i = 0; i < args.length; i++)
                outFile.write("Argument " + i + " is: " + args[i] + "\n");
            outFile.close();
        } catch (IOException e) {
            tes.getTestLogManager().reportMessage(
                                    "Unable to write to C:\\arguments.out");
        }
        return null;
    }
}
```

## Counting the number of times that code is executed

The CountAllIterations class counts the number of times code is executed by all virtual users. The CountUserIterations class counts the number of times code is executed by an individual virtual user.

**Exemple**

The CountAllIterations class counts the number of times it is executed by all virtual users running in a particular JVM and returns this count as a string.

```
package customcode;

import com.ibm.rational.test.lt.kernel.services.ITestExecutionServices;

/**
```

```
 * The CountAllIterations class counts the number of times it is executed
 * by all virtual users running in a particular JVM and returns this count
 * as a string.  If all virtual users on an agent are running in the same
 * JVM (as would typically be the case), this class will count the number of
 * times it is run on the agent.
 */


/**
 * @author IBM Custom Code Samples
 */

public class CountAllIterations implements
        com.ibm.rational.test.lt.kernel.custom.ICustomCode2 {
    private static int numJVMLoops = 0;

    /**
     * Instances of this will be created using the no-arg constructor.
     */
    public CountAllIterations() {
    }

    public String exec(ITestExecutionServices tes, String[] args) {
        return Integer.toString(++numJVMLoops);
    }
}
```

**Exemple**

The CountUserIterations class counts the number of times code is executed by an individual virtual user.

```
package customcode;

import com.ibm.rational.test.lt.kernel.services.ITestExecutionServices;
import com.ibm.rational.test.lt.kernel.IDataArea;

/**
 * The CountUserIterations class counts the number of times it is executed
 * by an individual virtual user and returns this count as a string.
 */

/**
 * @author IBM Custom Code Samples
 */

public class CountUserIterations implements
        com.ibm.rational.test.lt.kernel.custom.ICustomCode2 {

    /**
     * Instances of this will be created using the no-arg constructor.
     */
    public CountUserIterations() {
    }

    public String exec(ITestExecutionServices tes, String[] args) {
        IDataArea userDataArea = tes.findDataArea(IDataArea.VIRTUALUSER);
        final String KEY = "NumberIterationsPerUser";
```

```
        Number numPerUser = (Number)userDataArea.get(KEY);
        if (numPerUser == null) {
            numPerUser = new Number();
            userDataArea.put(KEY, numPerUser);
        }

        numPerUser.value++;
        return Integer.toString(numPerUser.value);
    }

    private class Number {
        public int value = 0;
    }
}
```

## Setting and clearing cookies for a virtual user

The SetCookieFixedValue class sets a Cookie for a virtual user, and the ClearCookies class clears all cookies for a
virtual user.

**Exemple**

The SetCookieFixedValue class sets a Cookie, defined in the newCookie variable, for a virtual user just as if the server
had returned a Set-Cookie.

```
package customcode;

import com.ibm.rational.test.lt.kernel.services.ITestExecutionServices;
import com.ibm.rational.test.lt.execution.http.cookie.IHTTPVirtualUserInfo;
import com.ibm.rational.test.lt.kernel.IDataArea;

import java.text.ParseException;

/**
 * The SetCookieFixedValue class sets a Cookie, defined in the newCookie
 * variable, for a virtual user just as if the server had returned a Set-Cookie.
 */

/**
 * @author IBM Custom Code Samples
 */

public class SetCookieFixedValue implements
                com.ibm.rational.test.lt.kernel.custom.ICustomCode2 {

    /**
     * Instances of this will be created using the no-arg constructor.
     */
    public SetCookieFixedValue() {
    }

    public String exec(ITestExecutionServices tes, String[] args) {
        String newCookie = "MyCookie=CookieValue;path=/;domain=.ibm.com";
        IDataArea dataArea = tes.findDataArea(IDataArea.VIRTUALUSER);
        IHTTPVirtualUserInfo httpInfo =
                (IHTTPVirtualUserInfo)dataArea.get(IHTTPVirtualUserInfo.KEY);
```

```
        try {
            httpInfo.getCookieCache().setCookie(newCookie);
        } catch (ParseException e) {
            tes.getTestLogManager().reportMessage("Unable to parse Cookie " +
                                                        newCookie);
        }

        return null;
    }
}
```

The ClearCookies class clears all Cookies for a virtual user. For information on how cookies are treated in tests and schedules, seeHow loops affect the state of virtual users.

```
package customcode;

import com.ibm.rational.test.lt.kernel.services.ITestExecutionServices;
import com.ibm.rational.test.lt.execution.http.util.CookieCacheUtil;


/**
 * The ClearCookies class clears all Cookies for a virtual user.
 */

/**
 * @author IBM Custom Code Samples
 */

public class ClearCookies implements
        com.ibm.rational.test.lt.kernel.custom.ICustomCode2 {

    /**
     * Instances of this will be created using the no-arg constructor.
     */
    public ClearCookies() {
    }

    public String exec(ITestExecutionServices tes, String[] args) {
        CookieCacheUtil.clearCookieCache(tes);
        return null;
    }
}
```

## Determining where a test is running

The ComputerSpecific class determines where a test is running

**Exemple**

```
package customcode;

import com.ibm.rational.test.lt.kernel.services.ITestExecutionServices;

import java.net.InetAddress;
import java.net.UnknownHostException;


/**
```

```
 * The ComputerSpecific class determined the hostname on which the test is
 * running, prints the hostname and IP address as a message in the test log,
 * and returns different strings based on the hostname.
 */

/**
 * @author IBM Custom Code Samples
 */

public class ComputerSpecific implements
        com.ibm.rational.test.lt.kernel.custom.ICustomCode2 {

    /**
     * Instances of this will be created using the no-arg constructor.
     */
    public ComputerSpecific() {
    }

    public String exec(ITestExecutionServices tes, String[] args) {
        String hostName = "Unknown";
        String hostAddress = "Unknown";

        try {
            hostName = InetAddress.getLocalHost().getHostName();
            hostAddress = InetAddress.getLocalHost().getHostAddress();
        } catch (UnknownHostException e) {
            tes.getTestLogManager().reportMessage(
                                    "Not able to obtain host information");
            return null;
        }
        tes.getTestLogManager().reportMessage("The hostname is " + hostName +
                                    "; IP address is " + hostAddress);
        if (hostName.equals("host-1234"))
            return "Special";
        else
            return "Normal";
    }
}
```

## Determining where a test is running

The ComputerSpecific class determines where a test is running

**Exemple**

```
package customcode;

import com.ibm.rational.test.lt.kernel.services.ITestExecutionServices;

import java.net.InetAddress;
import java.net.UnknownHostException;

/**
 * The ComputerSpecific class determined the hostname on which the test is
 * running, prints the hostname and IP address as a message in the test log,
 * and returns different strings based on the hostname.
 */
```

```
/**
 * @author IBM Custom Code Samples
 */

public class ComputerSpecific implements
        com.ibm.rational.test.lt.kernel.custom.ICustomCode2 {

    /**
     * Instances of this will be created using the no-arg constructor.
     */
    public ComputerSpecific() {
    }

    public String exec(ITestExecutionServices tes, String[] args) {
        String hostName = "Unknown";
        String hostAddress = "Unknown";

        try {
            hostName = InetAddress.getLocalHost().getHostName();
            hostAddress = InetAddress.getLocalHost().getHostAddress();
        } catch (UnknownHostException e) {
            tes.getTestLogManager().reportMessage(
                                      "Not able to obtain host information");
            return null;
        }
        tes.getTestLogManager().reportMessage("The hostname is " + hostName +
                                        "; IP address is " + hostAddress);
        if (hostName.equals("host-1234"))
            return "Special";
        else
            return "Normal";
    }
}
```

## Extracting a string or token from its input argument

The ParseResponse class extracts a string from its input argument. The ExtractToken class extracts a particular token (string) from its input argument. Both classes can be useful for handling certain types of dynamic data correlation.

**Exemple**

The ParseResponse class extracts a string from its input argument, using a regular expression for pattern matching.

```
package customcode;

import com.ibm.rational.test.lt.kernel.services.ITestExecutionServices;

import java.util.regex.*;

/**
 * The ParseResponse class demonstrates using Custom Code to extract a
 * string from its input argument using a regular expression for pattern
 * matching.
 *
 * In this sample, the args[0] input string is assumed to be the full
```

```
response   from a previous request.   This response contains the day's
headlines in a format such as:
 *
 *     <a class=f href=r/d2>In the News</a><small class=m> <span id=nw>
 *     </span></small></h2>
 *     <div class=ct>
 *     &#149; <a href=s/213231>Cooler weather moving into eastern
U.S.</a>   *   <br>&#149; <a href=s/262502>Digital camera shipments
up</a><br>   *
 * Given the above response, the extracted string would be:
 *        Cooler weather moving into eastern U.S.
 */


/**
 * @author IBM Custom Code Samples
 */

public class ParseResponse implements
        com.ibm.rational.test.lt.kernel.custom.ICustomCode2 {

   /**
    * Instances of this will be created using the no-arg constructor.
    */
   public ParseResponse() {}

   public String exec(ITestExecutionServices tes, String[] args) {
       String HeadlineStr = "No Headline Available";
       String RegExpStr = ".*In the News[^;]*;[^;]*;[^;]*;<a
href=([^>]*)>([^<]*)<";          Pattern pattern =
Pattern.compile(RegExpStr, Pattern.DOTALL);          Matcher matcher =
pattern.matcher(args[0]);
       if (matcher.lookingAt())
           HeadlineStr = matcher.group(2);
       else
           tes.getTestLogManager().reportMessage("Input does not match
pattern.");
       return HeadlineStr;
   }
```

The ExtractToken class extracts a particular string from its input argument.

```
package customcode;

import com.ibm.rational.test.lt.kernel.services.ITestExecutionServices;

/**
 * The ExtractToken class demonstrates using Custom Code to extract a particular
 * token (string) from its input argument.  This can be useful for handling
 * certain types of dynamic data correlation.
 *
 * In this sample, the args[0] input string is assumed to be comma-delimited
 * and the token of interest is the next-to-last token.  For example, if
 * args[0] is:
 *    javascript:parent.selectItem('1010','[Negative]1010','1010','','IBM',
 *        '30181','Rational','1','null','1','1','6fd8e261','RPT')
 * the class will return the string 6fd8e261.
 */
```

```
/**
 * @author IBM Custom Code Samples
 */

public class ExtractToken implements
        com.ibm.rational.test.lt.kernel.custom.ICustomCode2 {

    public ExtractToken() {
    }

    public String exec(ITestExecutionServices tes, String[] args) {
        String ArgStr;
        String NextToLastStr;
        String[] Tokens = args[0].split(",");

        if (Tokens.length > 2) {
            ArgStr = Tokens[Tokens.length - 2];        // Extract next-to-last token

            // Remove enclosing ''
            NextToLastStr = ArgStr.substring(1, ArgStr.length() - 1);
        } else {
            tes.getTestLogManager().reportMessage("Could not extract value");
            NextToLastStr = null;
        }
        return NextToLastStr;
    }
}
```

## Retrieving the maximum JVM heap size

The JVM_Info class retrieves the maximum heap size of the JVM.

**Exemple**

```
package customcode;

import com.ibm.rational.test.lt.kernel.services.ITestExecutionServices;

import java.net.*;

/**
 * The JVM_Info class retrieves the maximum heap size of the JVM.
 * It writes a message in the test log with the hostname where the
 * JVM is running and the JVM's maximum heap size in megabytes.
 */

/**
 * @author IBM Custom Code Samples
 */

public class JVM_Info implements
        com.ibm.rational.test.lt.kernel.custom.ICustomCode2 {

    public JVM_Info() {
    }

    public String exec(ITestExecutionServices tes, String[] args) {
```

```
        Runtime rt = Runtime.getRuntime();
        long maxMB = rt.maxMemory()/(1024*1024); // maxMemory() size is in bytes
        String hostName = "Unknown";

        try {
            hostName = InetAddress.getLocalHost().getHostName();
        } catch (UnknownHostException e1) {
            tes.getTestLogManager().reportMessage("Can't get hostname");
            return null;
        }

        tes.getTestLogManager().reportMessage("JVM maximum heap size for host "
                                        + hostName + " is " + maxMB + " MB");
        return null;
    }
}
```

## Running an external program from a test

The ExecTest class runs a program, defined in the execName variable, on the system where the test is running.

**Exemple**

```
package customcode;

import com.ibm.rational.test.lt.kernel.services.ITestExecutionServices;
import com.ibm.rational.test.lt.kernel.services.ITestLogManager;
import org.eclipse.hyades.test.common.event.VerdictEvent;

import java.io.IOException;

/**
 * The ExecTest class runs a program, defined in the execName variable,
 * on the system where the test is running.
 * The test verdict is set to PASS if the program return code is 0.
 * The test verdict is set to FAIL if the program doesn't execute or
 * if the program return code is non-zero
 * In this sample, the program is perl.exe.
 */

/**
 * @author IBM Custom Code Samples
 */

public class ExecTest implements
        com.ibm.rational.test.lt.kernel.custom.ICustomCode2 {

    /**
     * Instances of this will be created using the no-arg constructor.
     */
    public ExecTest() {
    }

    public String exec(ITestExecutionServices tes, String[] args) {
        ITestLogManager logger = tes.getTestLogManager();
        int rtnval = 1;
        Process p = null;
```

```
        String execName = "C:/Windows/System32/perl.exe C:/Perl/true.pl";

        Runtime rt = Runtime.getRuntime();
        // Execute test
        try {
            p = rt.exec(execName);
        } catch (IOException e) {
            logger.reportMessage("Unable to run = " + execName);
            logger.reportVerdict("Execution of " + execName + " failed",
                                            VerdictEvent.VERDICT_FAIL);
            return null;
        }

        // Wait for the test to complete
        try {
            rtnval = p.waitFor();
            logger.reportMessage("Process return value is " +
                                        String.valueOf(rtnval));
        } catch (InterruptedException e1) {
            logger.reportMessage("Unable to wait for " + execName);
            logger.reportVerdict("WaitFor on " + execName + " failed",
                                            VerdictEvent.VERDICT_FAIL);
            return null;
        }

        // Check the test return code and set the test verdict appropriately
        if (rtnval != 0)
        {
            logger.reportVerdict("Execution failed", VerdictEvent.VERDICT_FAIL);
        } else {
            logger.reportVerdict("Execution passed", VerdictEvent.VERDICT_PASS);
        }

        return null;
    }
}
```

## Adding custom counters to reports

When you want to monitor the specific requirement, you can add custom counters to performance report by using the custom code. After running tests, the results from the custom counters are automatically aggregated in the same way that the default performance testing counters.

Starting from V10.1.0, you can view and monitor the counter information generated by the custom code on a graph when the custom code starts in the test run.

After running tests, you can view the custom counter in the report. You can also view the custom counter information on a different page by creating a custom report. For more information about customizing the report, see related links.

You can add the following custom code in your test to create a custom counter in a report.

```
package test;

import org.eclipse.hyades.test.common.event.VerdictEvent;
```

```
import com.ibm.rational.test.lt.kernel.services.ITestExecutionServices;
import com.ibm.rational.test.lt.kernel.services.stats.CountAggregationLevel;
import com.ibm.rational.test.lt.kernel.services.stats.CounterUnits;
import com.ibm.rational.test.lt.kernel.services.stats.ICounterFolder;
import com.ibm.rational.test.lt.kernel.services.stats.ICounterRegistry;
import com.ibm.rational.test.lt.kernel.services.stats.IStatisticsManager2;
import com.ibm.rational.test.lt.kernel.services.stats.IValueCounter;
import com.ibm.rational.test.lt.kernel.services.stats.ValueAggregationLevel;

import database.DatabaseAccess;
import database.TransactionResult;

public class DatabaseStats implements com.ibm.rational.test.lt.kernel.custom.ICustomCode2 {

 private static boolean registerDone;

 /**
  * This method declares the counters that will be produced during execution.
  * Declaring counters is optional, but it allows to customize some of their
  * attributes, such as the label and unit, and what level of statistical information
  * will be available in reports.
  */
 private static synchronized void registerCounters(ICounterRegistry registry) {
  if (registerDone) return;
  registry.path("Database", "Transaction", "Attempts")
   .count()
   .aggregationLevel(CountAggregationLevel.RATE_RANGE)
   .label("Started Transactions")
   .unit("transactions")
   .register();

  registry.path("Database", "Transaction", "Commits")
   .verificationPoint()
   .label("Transaction Commits VP")
   .register();

  registry.path("Database", "Transaction", "Response Time", "Network")
   .value()
   .aggregationLevel(ValueAggregationLevel.RANGE)
   .unit(CounterUnits.MILLISECONDS)
   .register();

  registry.path("Database", "Transaction", "Response Time", "Commit")
   .value()
   .aggregationLevel(ValueAggregationLevel.DISTRIBUTION)
   .unit(CounterUnits.MILLISECONDS)
   .register();

  registry.path("Database", "Error")
   .text()
   .label("Database Error Message")
   .register();
  registerDone = true;
 }

 private DatabaseAccess database = DatabaseAccess.INSTANCE;
```

```java
/**
 * This custom code adds a record in database. It produces a couple of counters,
 * such as the database transaction attempts, successes/failures, and response time.
 */
public String exec(ITestExecutionServices tes, String[] args) {
 String product = args.length > 0 ? args[0] : "Default";
 IStatisticsManager2 mgr = tes.getStatisticsManager2();
 registerCounters(mgr.registry());

 database.startTransaction();
 mgr.getCountCounter("Database", "Transaction", "Attempts").increment();

 database.executeQuery("INSERT INTO TABLE Purchases VALUES('" + product + "', 1000)");
 TransactionResult result = database.commit();

 mgr.getVerificationPointCounter("Database", "Transaction", "Commits")
  .increment(result.isSuccess() ? VerdictEvent.VERDICT_PASS : VerdictEvent.VERDICT_FAIL);
 if (!result.isSuccess()) {
  mgr.getTextCounter("Database", "Error").addMeasurement(result.getErrorMessage());
 }

 ICounterFolder times = mgr.getFolder("Database", "Transaction", "Response Time");
 times.getValueCounter("Network").addMeasurement(result.getNetworkTime());
 times.getValueCounter("Commit").addMeasurement(result.getCommitTime());

 IValueCounter value = tes.getStatisticsManager2().getValueCounter("MyStats", "Value");
 value.addMeasurement(System.nanoTime() % 2000);

 return null;
 }

}
```

Related information

## Using transactions and statistics

You can use custom code to start transactions, gather additional statistics during a transaction, and stop a transaction.

The following code shows how to start a transaction. Transactions that are generated by test execution services automatically create and manage statistics.

```java
package customcode;

import com.ibm.rational.test.lt.kernel.services.ITestExecutionServices;
import com.ibm.rational.test.lt.kernel.services.ITransaction;

/**
 * @author IBM Custom Code Samples
 */
```

```
public class BeginTransaction implements
  com.ibm.rational.test.lt.kernel.custom.ICustomCode2 {

 /**
  * Instances of this will be created using the no-arg constructor.
  */
 public BeginTransaction() {
 }

 /**
  * For Javadoc information on the ICustomCode2 and ITestExecutionServices interfaces,
  * see the 'Test execution services interfaces and classes' help topic.
  */
 public String exec(ITestExecutionServices tes, String[] args) {
  // the name of the transaction could have been passed in via data correlation mechanism.
  ITransaction foo = tes.getTransaction("foo");
  foo.start();
  return null;
 }
}
```

The following code shows how to gather additional statistics during a transaction.

```
package customcode;

import com.ibm.rational.test.lt.kernel.ITime;
import com.ibm.rational.test.lt.kernel.services.ITestExecutionServices;
import com.ibm.rational.test.lt.kernel.statistics.IScalar;
import com.ibm.rational.test.lt.kernel.statistics.IStat;
import com.ibm.rational.test.lt.kernel.statistics.IStatTree;
import com.ibm.rational.test.lt.kernel.statistics.impl.StatType;


/**
 * @author IBM Custom Code Samples
 */
public class BodyTransaction implements
  com.ibm.rational.test.lt.kernel.custom.ICustomCode2 {

 /**
  * Instances of this will be created using the no-arg constructor.
  */
 public BodyTransaction() {
 }

 /**
  * For Javadoc information on the ICustomCode2 and ITestExecutionServices interfaces,
  * see the 'Test execution services interfaces and classes' help topic.
  */
 public String exec(ITestExecutionServices tes, String[] args) {
  IStatTree tranStat;
  IStatTree timeStat;
  IStatTree countStat;

  IStat timeDataStat = null;  // counter for the time RANGE
  IScalar countDataStat = null; // counter for the count SCALAR

  ITime timer = tes.getTime();
```

```
        IStatTree rootStat = tes.getStatisticsManager2().getStatTree();
        if (rootStat != null) {
         // these counters set up the hierarchy
         tranStat = rootStat.getStat("Transactions", StatType.STRUCTURE);
         timeStat = tranStat.getStat("Body Time", StatType.STRUCTURE);
         countStat = tranStat.getStat("Bocy Count", StatType.STRUCTURE);

         // the name of the counters could have been passed in via data correlation mechanism
         timeDataStat = (IStat) timeStat.getStat("foo", StatType.RANGE);
         countDataStat = (IScalar) countStat.getStat("foo", StatType.SCALAR);
        }

        // get the start time
        long startTime = timer.timeInTest();

        // do the work
        // whatever that work might be

        // get the end time
        long endTime = timer.timeInTest();

        // update timeDataStat with the elapsed time
        if (timeDataStat != null)
         timeDataStat.submitDataPoint(endTime - startTime);

        // update the countDataStat
        if (countDataStat != null)
         countDataStat.increment();

  return null;
 }

}
```

The following code shows how to stop a transaction.

```
package customcode;

import com.ibm.rational.test.lt.kernel.services.ITestExecutionServices;
import com.ibm.rational.test.lt.kernel.services.ITransaction;

/**
 * @author IBM Custom Code Samples
 */
public class EndTransaction implements
  com.ibm.rational.test.lt.kernel.custom.ICustomCode2 {

 /**
  * Instances of this will be created using the no-arg constructor.
  */
 public EndTransaction() {
 }

 /**
  * For Javadoc information on the ICustomCode2 and ITestExecutionServices interfaces,
  * see the 'Test execution services interfaces and classes' help topic.
```

```
  */
 public String exec(ITestExecutionServices tes, String[] args) {
  // the name of the transaction could have been passed in via data correlation mechanism.
  ITransaction foo = tes.getTransaction("foo");
  foo.stop();
  return null;
 }

}
```

## Reporting custom verification point failures

You can use custom code to report a custom verification point failure.

The following code shows how to report a custom verification point failure.

```
package customcode;

import org.eclipse.hyades.test.common.event.VerdictEvent;
import org.eclipse.hyades.test.common.runner.model.util.Verdict;

import com.ibm.rational.test.lt.execution.core.IVerificationPoint;
import com.ibm.rational.test.lt.kernel.services.ITestExecutionServices;

/**
 * @author IBM Custom Code Samples
 */
public class Class implements
  com.ibm.rational.test.lt.kernel.custom.ICustomCode2 {

 /**
  * Instances of this will be created using the no-arg constructor.
  */
 public Class() {
 }

 /**
  * For javadoc of ICustomCode2 and ITestExecutionServices interfaces, select 'Help Contents' in the
  * Help menu and select 'Extending
                     Rational® Performance
                     Tester functionality' -> 'Extending test execution with custom code'
  */
 public String exec(ITestExecutionServices tes, String[] args) {
  tes.getTestLogManager().reportVerificationPoint("CustomVP", VerdictEvent.VERDICT_FAIL);
  return null;
 }

}
```

## Debugging custom code

This example demonstrates debugging custom code by adding a breakpoint. It provides sample code to add a breakpoint. This way of debugging custom code is applicable only for a schedule.

1. Start IBM® Rational® Performance Tester and create a performance test project `MyProject`.

2. Create an HTTP test, **MyTest**, by recording a visit to `http://<hostname>:7080/`.

> **Note:** Before accessing the URL, ensure that Rational® Performance Tester is running. The URL returns an HTTP 404 error, which is expected.

**Result**



3. Expand the first request and click the response element.

4. In the Test Element Details section, right-click in the **Content** field and click **Create Field Reference**.

5. Type the reference name and click **OK**.

6. Click the first page, and then click **Add > Custom Code**.

7. In the **Arguments** section of Test Element Details, click **Add**.

8. Expand the data source for the search results page, select the reference name that you created in step 5, and click **Select**.

9. Click **Generate Code**.

   **Result**

   A new tab with the generated code is displayed.

10. Insert the following the code into the `exec()` method:

```
ITestLogManager history = tes.getTestLogManager();
if (args.length > 0) {
    if (args[0].indexOf("Invester Relations") != -1) {
        history.reportMessage("First page failed.  Bail loop!");
        tes.getLoopControl().continueLoop();
    }
}
```

> ⚠️ **Important:**
> - Fix the double quotation marks, if any, so they are straight and the compiler no longer gives warning.
> - To resolve complier warnings related to importing a class, press **Ctrl + Shift + O**.

The code will look like this:

```java
public String exec(ITestExecutionServices tes, String[] args) {
    ITestLogManager history = tes.getTestLogManager();
    if (args.length > 0) {
        if (args[0].indexOf("Invester Relations") != -1) {
            history.reportMessage("First page failed.  Bail loop!");
            tes.getLoopControl().continueLoop();
        }
    }
    return null;
}
```

11. To set a breakpoint, click anywhere on the `args[0].indexOf` line. Move the pointer to the left-most portion of the text editor window and double-click with the pointer horizontally on the same line. A blue button is displayed in this left-most portion of the window indicating the breakpoint is set.
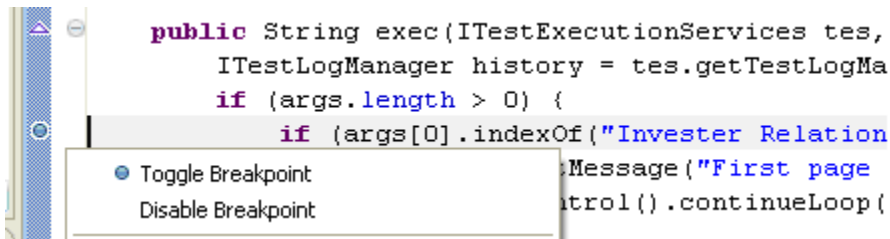


12. Save the custom code and then the test.
13. Create a new schedule, `Schtest`.

   a. In `Schtest`, set the number of users to run to 1.

   b. Click **User Group 1** and click **Add > Test**. Select the `MyTest` test and click **OK**.

   c. Click **User Group 1** and click the **Run this group on the following locations** button.

   d. Click **Add > Add New**.

   e. In the **New Location** window, type the following information:
      i. In **Host name**, type `localhost`.
      ii. In **Name**, type `debuglocation`.
      iii. In **Deployment directory**, type `C:\mydeploy`.
      iv. Click **Finish**.

   f. Save the schedule.

14. In the Test Navigator, right-click **debuglocation** and click **Open**.

15. Click the **General Properties** tab and click **Add**.

16. In the **Property name** field, type `RPT_VMARGS` and in the **Property value** field, add the following values each separated by a space.

```
-Xdebug
-Xnoagent
-Djava.compiler=NONE
-Xrunjdwp:transport=dt_socket,server=y,suspend=y,address=8000
```



17. Save the location.

18. Attach the debugger to the schedule execution process.

    a. Run the schedule.

       Because the schedule is using **debuglocation**, it will pause at the beginning to let you attach the debugger to the execute process.

    b. Click **Window > Open Perspective > Other > Debug**.

    c. Click **Run > Debug Configurations**.

    d. In the **Debug Configurations** window, right-click **Remote Java Application** and click **New**.

    e. Click **Debug**.

       A list of running threads are displayed in the Debug window and the schedule execution pauses at the debug breakpoint.

    f. If you are doing it for the first time, you might need to provide the source location to see the custom Java code. You do this by taking the following steps:

        i. Click **Edit Source Lookup Path** and click **Add**.

        ii. Click **Workspace Folder > OK**.

iii. Now, expand MyProject, select the src folder, and click **OK**. The schedule run stops at the specified breakpoint.
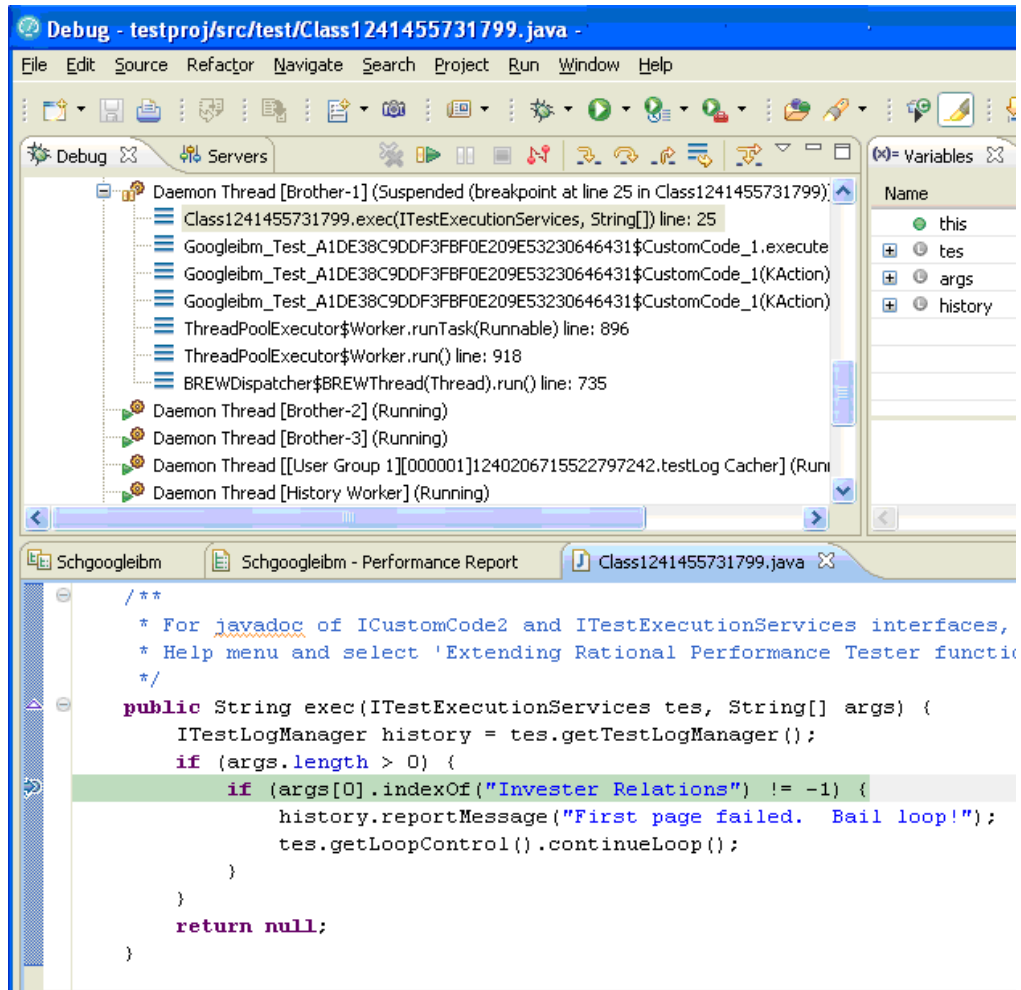


# Migrating custom code from previous versions

You can run scripts that contain custom code from previous releases and edit tests to make new calls to old or new custom code classes.

**About this task**

You can perform the following tasks without any additional steps:

- Run a script that contains custom code that was created in a previous release.
- Edit a test to make a new call to an old custom code class.
- Add new custom code to a test that contains old custom code.

To edit a class in existing custom code so that it can call new TestExecutionServices methods, type cast the *IKlog* argument in the old custom code to the *ITestExecutionServices* interface.

When you migrate the custom code from the previous versions, you must use getStatisticsManager2() as getStatisticsManager() API is deprecated from V10.1.0.

# Chapter 8. Test Manager Guide

This guide describes how to keep track of the performance of the application by evaluating the test results. This guide is intended for test managers.

## Evaluating results in web analytic reports

After the test run, evaluate the results in the web analytic reports. Web analytics collect data using new technologies thereby providing better user experience.

## Comparing results among runs

To analyze the difference between two or more reports, you can compare them. For example, to analyze the performance of the application at different time slots or different milestone builds, you can compare two runs.

**About this task**

You can compare the test runs from the Test Navigator view or from the web analytics report itself. To compare test runs from the Test Navigator view, select the runs, right-click, and click **Compare Results**.

You can compare the test runs that are in the same project or in the different projects. When comparing multiple runs, you cannot compare multiple time-ranges or stages.

To compare runs from the web analytics report:

1. Open the run or report to serve as the basis for comparison.

2. Click the name of the run next to the **Menu** option  schedule [1/22/16, 11:54 AM] ⌄
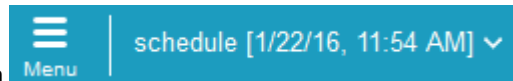
3. Click **Add** ➕ and navigate to the run to compare with.
   Multiple runs are displayed in the report.

4. **Optional:** To add, remove, or move the position of the runs, click **Manage** [Manage...].

---

Related information

## Comparing schedule stages

When you are running a A schedule, in this context, is used to refer to both VU Schedule and Rate Schedule. that contains stages, time ranges are automatically created for each stage. You can view a report that compares these stages, and you also can set preferences to display the report automatically at the end of a staged run.

**About this task**

In addition to comparing stages, you can add time ranges and compare them. To view the compare report automatically at the completion of a run, click **Window > Preferences > Test > Performance Test Reports**, and select **Launch Compare report when staged run completes**.

1. Open the run that consists of stages.
   By default, reports are displayed for the entire run.
2. Click the **Entire Run** menu [Entire Run ∨] and select the stages to compare.
   Both the running and completed stages show up in the list.
3. To add a new time range, click **Add** [+] in the **Entire run** menu.
4. In the **Time Range** dialog box, specify a name, start time, and end time of the run and click **Apply**.

> ✎ **Note:** When you compare stages in a run, you cannot compare data from various geographies at the same time.

## Comparing results from various regions or agent locations

When you run a schedule that includes agents from different regions, use the Web Analytic reports to compare the performance data from these regions.

**Before you begin**

Run a schedule with on premise agents on different geographic regions. See Running schedules on page 287.

**About this task**

When comparing agents, open the Location asset and in the **General Properties** tab, add an `RPT_GEO` property with any value. This value is then displayed in the **All Geos** menu of the report.

To compare performance data:

1. From the Test Navigator view, open the schedule run that includes remote agents. The name of the run corresponds with the name of the schedule and has a timestamp.
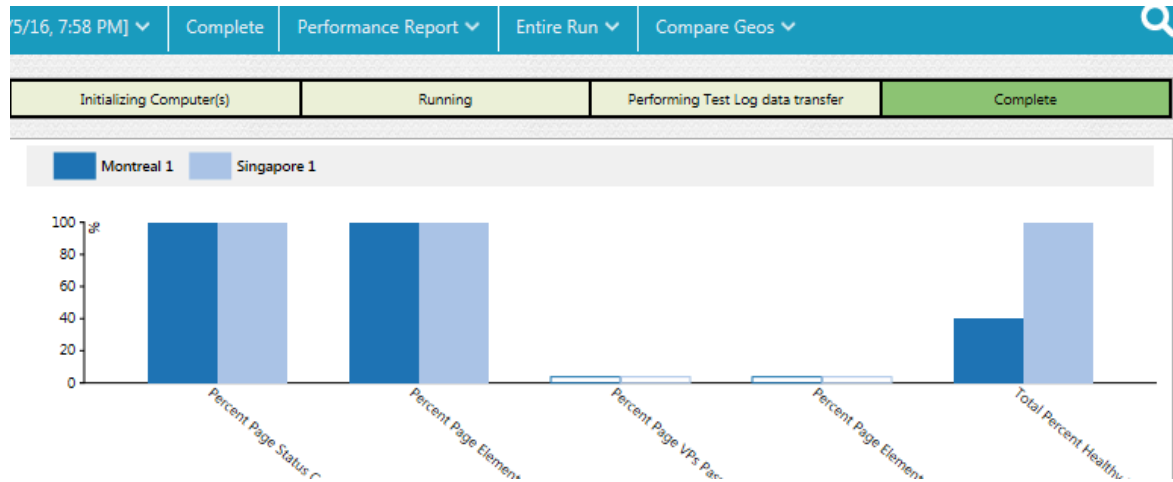2. On the report toolbar, click **All Geos**.
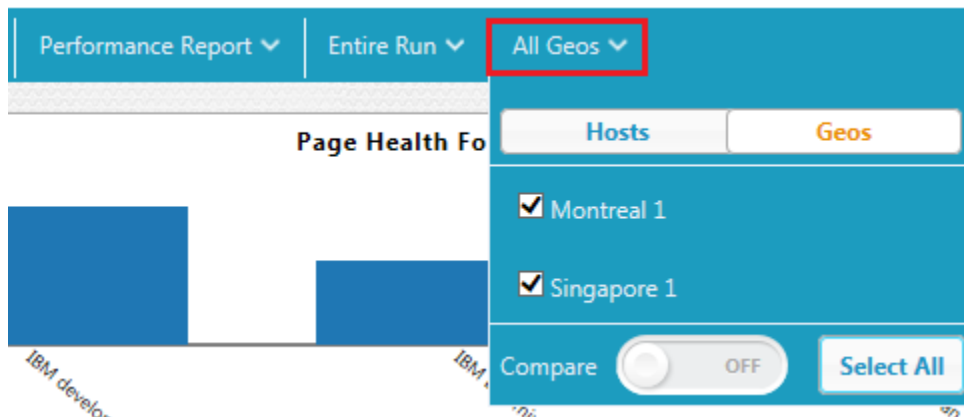
3. Select the regions that you want to compare and click **Compare**.

> 📝 **Note:** When you compare data from various regions, you cannot compare stages in a run at the same time.

The report shows the data in the compare mode.



## Generating functional test reports

You can generate functional test reports of your tests, which summarize the pass or fail verdicts of elements in the test log. Functional reports are generated from the test run as HTML files that use predefined report designs.

**Before you begin**

Before you can generate a functional report, you must successfully run a test or schedule and produce a test run.

The following report types are available:

- Extensible Stylesheet Language Transformation (XSLT) reports: These reports are faster to generate, but do not contain graphs.
- Business Intelligence and Reporting Tools (BIRT) report: These reports contain graphs but are slower to generate. You can customize and create your own BIRT report designs in the Report Design perspective of the workbench. BIRT report generation is not supported when the workbench is running in a VMWare Windows™ image.

> 📝 **Note:** If you use your own XSLT style sheets, verify that the style sheets contain this line: `<xsl:param name="languagePack" select="'default'"/>`

1. In the **Test Navigator**, select a test run or runs.
   You can use the Ctrl key to select multiple test runs or schedule runs. You cannot generate a functional report that contains more than 5000 calls or objects.

2. In the **Test Runs** view, right-click the test runs and select **Generate Functional Test Reports**.
   **Result**
   This opens the **Generate HTML Functional Test Report** wizard.
3. Select the location in the workspace where you want to generate the functional report, and type the **Functional report base name**. A time stamp and the type of report is appended to this base name when the report is generated.
   If you want to keep the temporary XML file that is created to generate the report for debugging purposes, select **Keep intermediate XML data**.
4. Click **Next**.
5. Select a predefined report designs or click **Add** to add a custom BIRT report design or an XSLT style sheet.
   **Choose from:**
     ◦ **Common Functional Test Report**: This produces a generic functional test report for all test protocols.
     ◦ **SAP Functional Report**: This produces a functional test report for SAP tests.
     ◦ **Services - Failed events**: This produces a functional test report for web service tests. The report contains only failed events. Events with other verdicts are not shown in the report.
     ◦ **Services - Failed tests**: This produces a functional test report for web service tests. The report contains only failed tests. Tests with other verdicts or other event types are not shown in the report.
     ◦ **Services - Full**: This produces a functional test report for web service tests. The report contains detailed information on all events.
     ◦ **Services - Summary**: This produces a brief summary functional test report for web service tests.
     ◦ **Services - Truncated**: This produces a functional test report for web service tests. The report contains detailed information on all events, but truncates XML contents after 500 characters.
   One functional report is generated for each selected report design. Report designs marked with **(xslt)** use XSLT style sheets and are more suitable for larger reports.
6. Click **Finish**.

**Results**

The functional reports are generated as HTML files in the specified location in the workspace.

## Publishing test results to the server

The test results indicates the quality of application under test. Different stakeholders to the application might want to check the quality of application but do not have the workbench or desktop client installed. As a desktop client user, you can publish the test result to IBM® Rational® Test Automation Server so that others can view it from the web browser.

**Before you begin**

• Accessed Rational® Test Automation Server.

• Created an offline user token to connect to Rational® Test Automation Server from  Rational® Performance Tester. For more information, refer to Managing access to server.

• Created or joined a project in Rational® Test Automation Server.

- Configured the firewall so that Rational® Test Automation Server enables connection on the port 443.

- Upgraded legacy reports to the Web Analytics report format.

> 📝 **Note:** You can right-click the report and select **Upgrade** to upgrade the legacy report to the Web Analytics report.

**About this task**

You can publish both performance and functional reports. You can set the publish parameters once in the **Preference** page so that you do not have to do it after every run or you can set the parameters every time for the specific result that you want to publish. Based on the parameters, the test result is published after the test run.

If you select **Prompt** from the drop-down list for the **Publish result after execution** option, after each test run, the **Publish Result** dialog box is displayed to publish reports instantly to Rational® Test Automation Server. You can modify the following options before publishing the results:

- If you want to publish reports to other than the default server added in the **Preferences** window, you can change the URL of Rational® Test Automation Server.

  > 📝 **Note:** If you change the server URL, you must enter an offline token to enable the publishing of test results and reports.

- By default, the **Result Name** field populates the name of the selected test result. You can provide a different name that you want to use.

- To identify specific test results, you can enter a tag or comment in the **Labels** field to associate it with the test result.

- You can change the project name if you want to publish reports to a different project.

  > 📝 **Note:** If there are no projects or if you are not a member of any project, you must create a project or become a member of a project on the server.

1. Click **Window > Preferences > Test >** Rational® Test Automation Server.
2. Specify the URL of the server and click **Test Connection**.
3. Enter the offline user token that you created on the server and click **OK**.
4. **Optional:** Click **Manage Offline Tokens** to view and remove the tokens that are associated with the desktop client, and click **Apply and Close**.

   For example, if there is one instance of the desktop client for multiple testers to publish reports, each tester must remove the token created by other testers and add a new token.

5. Click the **Results** page from the navigation to apply settings for publishing reports.

6. Clear the **Use default** Rational Test Automation Server **URL** check box if the URL of the server is different than that is specified at **Window > Preferences > Test >** Rational Test Automation Server.

   The format of the URL is `https://fully-qualified-domain-name:443`.

7. In **Publish result after execution** field, select when to publish test result.
   In the initial stage when you are debugging a test, you might not want to publish the test result. Select one of the following options based on the requirement:
   ◦ Select **Never** to never publish the test results to the server.
   ◦ Select **Prompt** to prompt you to publish the test results after every test run.

   > 📝 **Notes:**
   > ▪ A command line run will always publish the test results even if the workbench is set to **Prompt**.
   > ▪ After each test run, the **Publish Result** dialog box is displayed to publish reports instantly to Rational® Test Automation Server.

   ◦ Select **Always** to publish the test results after every test execution.

8. In **Publish to project** field, select a project that you are a member of on the server. The test result is published to that project.

   You cannot create a new project from the desktop client. If there are no projects or if you are not a member of any project, you must create a project or become a member of a project on the server.

9. In **Reports**, select the reports that you want to publish to the server.
10. Click **Apply and Close**.

**Results**

Test results are automatically published to the Rational® Test Automation Server, depending on the parameters that you have set.

**What to do next**

You can log in to Rational® Test Automation Server and analyze the test results. For more information, refer to Test results and reports overview.

---

Related information

Publishing specific results to the server

## Customizing reports

You can customize reports to specifically investigate a performance problem in more detail than what is provided in the default reports.

## Creating custom reports

If the default reports do not address your needs, you can create your own reports.

**About this task**

Before you create a custom report, determine the ways in which the custom report will be different from or similar to the system-supplied reports. You can use a default report as a template, modify the counters, and save it with a different name. You can create a copy of pages or charts in a report that are based out of existing pages or charts. To copy the pages or charts, go to the Edit view and click the **Duplicate** icon.

You can also create a report from scratch and add the required counters. Counters for test protocol are explained in the topics in the Reports and counters section.

> **Note:** If a counter has a lot of data, the graph is not represented properly. to ensure that the graph is displayed properly, you must filter out some of the data.

1. From the report, click **Menu** and click **New**.
2. In **Create a new report** dialog box, specify a name and description about the new report and click **Create**.
3. To change the page title, click the default page title and specify a different name.
4. Click **Click to insert a row** and specify the number of columns to add the views.
   Each view represents a bar chart, line chart, or pie chart.
5. Select a view. To add counters to the view, click **Settings** ⚙ .
6. On the View Settings page, select a counter and add its details.
7. Click **Apply** and from the **Menu**, click **Save** to save the report.
8. To add more views to the report, repeat steps 4 through 7 again.

## Viewing trending reports

To view the trend of response time for an application over a period of time, open the trend report for a run. In addition to the response time, you can view the trend for the loops, transactions, and performance requirements for the application.

**About this task**

The trend report can help you determine the response times of the application at different milestones. For instance, you can run the performance test for sprint or milestone builds and tag them. When generating the trend report, you can specify conditions such as results that are less than 60 days old and include 'milestone' tag.
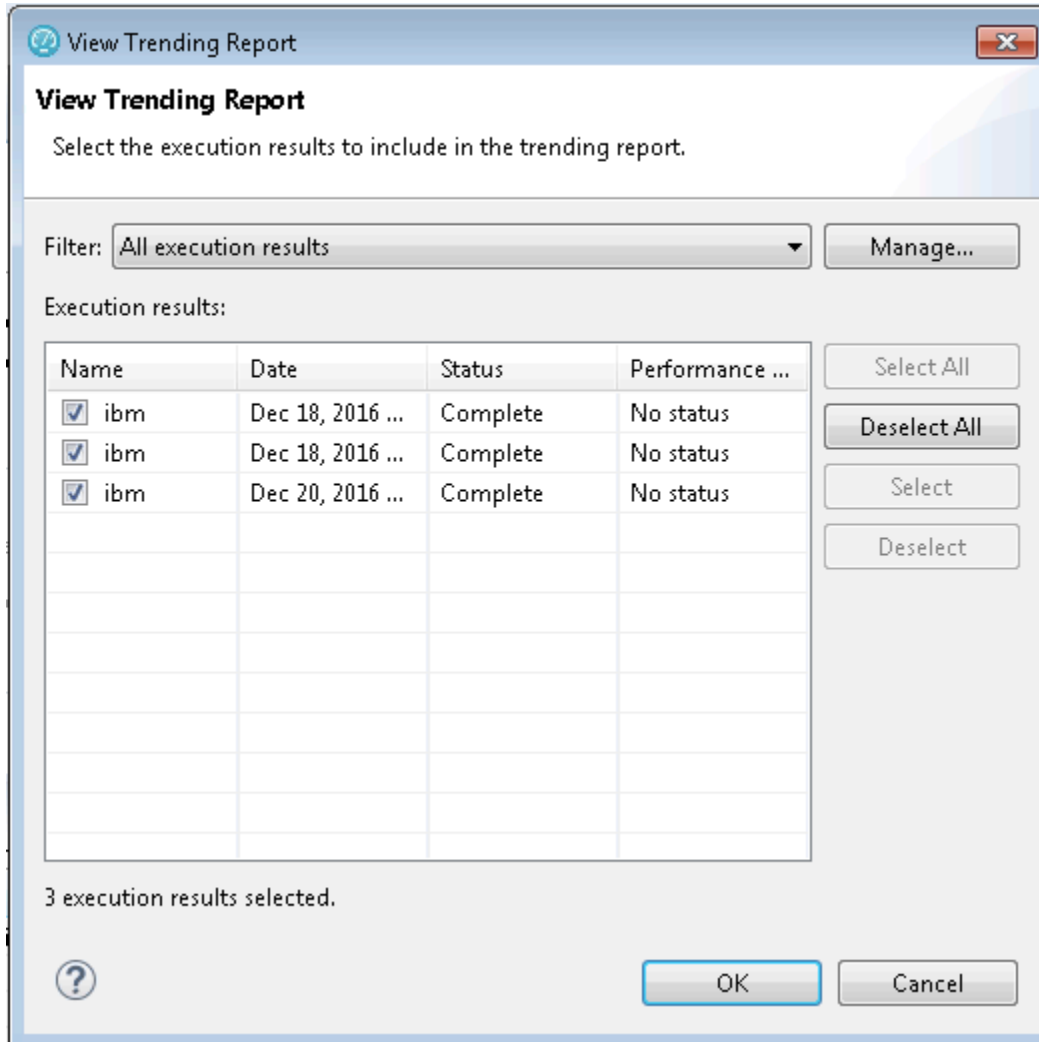
You cannot save a trend report. So, if you close the report, you have to generate it again.

To view the trend report:

1. In the Test Navigator view, select the run for which to open the trend report.
2. Right-click the run and click **View Trend Report**.
3. To view the trend that is based on certain criteria, in **Filter**, select a filter criteria.
   If there is no customized filtering criteria, create one by clicking **Manage** and then **Add**.
4. To save the criteria, click **Save**, specify a name to the filter, and click **OK**.
   The results in the execution results table are filtered out according to the specified criteria.
5. Click **OK**.
   **Result**
   The trend report is generated.

## Filtering results

By filtering the results that are displayed in a report, you can remove unnecessary data and focus on the data that is significant to you. If you save the changes, the report will contain these updates the next time that you open it.

**About this task**

If there is a lot of data in a bar or line chart that causes some of the data to not display by default, you can use the horizontal scroll bar on the X-axis to view the rest of the data. There is also a filter icon to filter the data based on the text on labels.

The table at the bottom of the chart also includes a lot of information. So, now, the tabular data is presented in pages. There is also the filter icon to filter the data in the table.

1. To filter the data in a report, open that report.

2. Click the **Menu** icon Menu and click **Edit**.
3. On the graph or table, click the **Settings** icon .
4. On the View Settings page, in the Filter section, select one of the following options and specify the values:

| Option | Description |
|---|---|
| **Filter by count** | Display the specified number of items. For example, if you select this option and then type 15, the report will show the 15 items with the highest values (**Show highest values**) or the 15 items with the lowest values (**Show lowest values**). |
| **Filter by value** | Display items based on a comparison with the specified value. For example, if you select this option and then type 15, the report will show all of the items that are higher than 15 (**Show counters above filter values**) or lower than 15 (**Show counters below filter value**). |
| **Filter by name** | Display items that match the specified name. If you are filtering a table, the label is usually a page, and is listed in the left column. If you are filtering a graph, the label is a legend in the graph. |

See the changes in the Preview section.
5. Click the **Apply** button.

If you are filtering a table by count or value, the **Primary counter for table filtering** lists the counters in the heading of the table that you are filtering. Select a primary counter. If a value is removed by the primary filter, all other data occupying the same row as the removed value is also removed, thus maintaining the integrity of the results.

**What to do next**

To remove a filter, follow the same steps but at step 4 modify or delete the value.

## Customizing the appearance of graphs in a report

To display the data in a table, bar chart, or line chart in a manner that caters to your test requirements, use the controls that are available in the View Options of a report.

1. In the Test Navigator, expand the project until you locate the run.
   Each run begins with the name of the schedule or test, and ends with the date of the run in brackets.
2. Double-click the run.
   **Result**
   The default report opens.
3. Click the **Menu** icon Menu and click the **Edit** icon.
4. Click the **Settings** icon ✿ for the graph or table to modify.
5. The controls that are available in the View Options section depend on the graph type: bar chart, line chart, or table. For each graph type, only the applicable controls are displayed. You can adjust the following controls:

| Option | Description |
| --- | --- |
| Adapt Y Scale | To compute minimum and maximum limit on the Y axis, select the check box. (all charts) |
| Title | Specify a title to the graph. |
| Show title | To hide the title, clear the check box. |
| X Axis Main items | Select the item to view on the X Axis. |
| Stacked Items | Select the item such as Pages or Time Ranges to view them in stack instead of separate bars. |
| Adapt Y scale on zoomed data | To adjust the Y scale according to zoomed data, select the check box. (line charts) |
| Show time ranges | To display the time range in the background of the chart, select the check box. |
| Line smoothing | To apply corners, clear the check box. |

| Option | Description |
|---|---|
| Orientation | To view bar charts horizontally or vertically, select an orientation. |
| Labels display policy | To hide the labels in a bar chart, select **Hidden**. To be able to accommodate labels within the frame of a bar chart, select **Adaptative**. If you select **Fixed**, long labels might not be visible. |
| Time line visibility | To view the time line of the chart in partial or full view, select **Small** or **Full** options. Drag the time line to create a new time range. If those options are specified, you can drag and create a new time range on the chart itself. If you select **None**, the time line is not visible and you cannot create a new time range on the chart. |

6. After making the changes, click **Apply** and from the Menu click **Save**.

   To apply the changes to other reports, you can export the report definition and import it back. See Exporting report metadata on page 351.

## Changing the report displayed during a run

Use this page to select the default report that opens during a run. Typically, you select **Determine default report based on protocols in test**, which determines the protocols that you are testing and automatically opens the appropriate protocol-specific reports.

1. Open the Default Report Preferences page. Click **Window > Preferences > Test > Performance Test Reports > Default Report**.
2. In the Default Report window, select **Determine default report based on protocols in test** or a specific default report to display a customized report or if the default reports do not meet your needs. Note, however, that you will have to change this setting when you record other protocols.
3. Click **Apply**, and then click **OK**.

## Modifying counters in a graph

To gather additional information for diagnosing performance problems, you can modify the counters that are displayed in a graph.

**About this task**

Counters are specific in-built queries that gather statistical information from the recorded test. The information can be the number of page hits, response time, and user load. By default, each report has pre-defined counters. You can add or remove the counters from the graphs in the report.

1. From the Test Navigator, double-click the report to modify the counters.

2. To add or remove a counter from a report, click the **Menu** icon .

3. Click the **Edit** icon .

4. To modify counters on a specific graph, click **Settings** icon .

5. On the **View Settings** page, select the **Counters** tab and perform the following sub-steps to add, remove, or move the counters in a graph:

   a. To add a counter, click the **Plus** button and select the counters from the drop-down list.

   b. To remove the selected counter, click the **Remove** button .

   c. To move a counter, use the **up-down** control buttons .

   The Preview section displays the result of the actions.

6. **Optional:** For a selected counter, you can change the component of the counter. Based on the counter selection, the **Component** field shows the options available for that counter.

7. **Optional:** You can change the **Cumulated** value for the selected counter if you want to show the cumulation values on a graph. Select one of the following options based on the requirement:
   **Choose from:**

   ◦ Select **No** to display the value of the last interval on the current time range.

   ◦ Select **From beginning of time range** to display the cumulation of all values of the current time range.

   ◦ Select **From beginning of run** to display the cumulation of all values from the beginning of the run to the end of the current time range.

   **Notes:**
   ◦ For line charts, the default value is **No**.
   ◦ For bar chart, pie chart, and tables, the default value is **From beginning of time range**.
   ◦ The fields **Label**, **Path**, and **Unit** are non-editable.

8. Click **Apply**.

9. Click **Save** from the menu.

10. **Optional:** To create another report with these changes, click **Save As**.

11. To exit the edit mode, click the **Edit** icon .

**Results**

You have updated the counter information for the specific report.

## Correcting time offset

Response time breakdown and resource monitoring data is time stamped using the system clock of the host computer. If there are differences between the system clocks of the host computers that you include in a test, then

response time breakdown and resource monitoring data are skewed in reports. The best practice is to synchronize the system clocks on all computers that you include in a test. When this is not possible, you can correct the time offset of each host computer after a test run. Typically, correct the time offset on all computers to match the system clock of the workbench computer.

After you run tests with resource monitoring or response time breakdown enabled, follow these steps to correct the time offset:

1. In the **Test Runs** view, right-click the host where you want to correct the time offset; then click **Correct Time Offset**.
2. Select a **Shift Direction** of positive or negative. A positive shift moves the response time breakdown and resource monitoring data on the selected host to the right. A negative shift moves the response time breakdown and resource monitoring data on the selected host to the left.
3. Type the hours, minutes, or seconds of the time offset you want to use, and click **OK**.

**Results**

The response time breakdown and resource monitoring data on the selected host displays with a corrected time offset.

## Export test results

You can export the test result in different formats to share it with different stakeholders.

## Creating an executive summary from the workbench

To create a printable report that summarizes the findings of the performance test run on a single view, create an executive summary. You can export the data of the test run as an executive summary from a single report or from multiple reports such as Performance Report, Mobile and Web UI Statistical Report, Transaction Report, and Loop Report. You can then open the summary in a word-processing program to further format and annotate the data.

**About this task**

You export the executive summary to a local or a shared directory. You can export a test run from the Web Analytics report, from the test workbench, and from the command line.

When you use the workbench approach to create an executive summary, you can choose to create the summary for multiple runs and multiple report types at the same time. When you use the Web Analytics reports or the command line, you create executive summary for a particular run and a report at a time.

To create an executive summary from the workbench:

1. Click **File > Export > Test > Executive Summary**. You can also right-click the runs to create executive summaries for from the Test Navigator view and click **Export > Test > Executive Summary**. Each run would have one executive summary.
2. In **Export Directory**, specify the folder path to save the executive summary and click **Next**.

3. Select the runs to create the executive summary for. To create an executive summary for comparing two runs, select the **Generate a compare report** check box and select the main run to compare the report with and click **Next**.

4. Select a report to export and click **Finish**.

**What to do next**

A folder with the name of the run is created on the specified folder. To view the executive summary, open the `index.html` file.

## Creating an executive summary from the Web Analytics report

To create a printable report that summarizes the findings of the performance test run on a single view, create an executive summary. You can choose to view the executive summary on a web browser or save it on a computer.

**About this task**

To generate an executive summary for a particular report such as Transaction report or Performance report, open that report and then follow the steps in this topic. To generate an executive summary for multiple reports or test runs at the same time, see Creating Executive Summary from Workbench on page 347.

To create an executive summary from the Web Analytics report:

1. Open the test run to create executive summary for. The test run opens in a web browser.
2. From the dropdown, open the report for which to create executive summary.

3. Click the **Menu** icon **Menu**, click the **Share** icon , and click **Executive Summary**.
4. To view the executive summary of the report in another browser tab, click **View on another tab or page of the browser**. To save the executive summary, click **Save as an HTML file on the local computer**.
5. Click **Generate**.

## Exporting reports to HTML format

When you export a test run and share it, people can analyze test data without using the test workbench. You can also email the test run or post it on a web server. The exported run can be displayed and printed from any browser. A test run contains multiple reports. You can choose to export any or all of the reports.

**About this task**

You can export a single run to a local directory or multiple runs in the compare mode to a directory. In addition to exporting a test run from Web Analytics, you can export it from the test workbench itself and from command line.

To export from the workbench, select a single run or multiple runs and click **Export > Test > Performance Test Run Statistics as HTML application** . To generate a single report comparing multiple runs, in the Export wizard, select the **Generate a compare report** check box and select a base run from the dropdown. To generate one report for each run, do not select the check box.

To export from Web Analytics:

1. Open the test run to export.
   The test run opens in an external or internal web browser.

2. Click the **Menu** icon , click the **Share** icon , and click **Export Session to HTML**.
3. Select the type of report to export and click **Export**.
4. When you export from the workbench, specify a path to the folder to save the exported report.
   Your current project is the default save location. You can create a folder outside of the project to store exported reports.

   When you export from an external browser, the report is compressed and saved to the default download location of the browser.

**What to do next**

You can now share the test run with others. You can also export the test run from command line.

Related information

Running a test from a command line on page 295

## Exporting results to a CSV file

To further analyze test results, you can export all statistics or specific statistics captured during a run to a CSV file.

**About this task**

You can export a single run to a local directory or multiple runs in the compare mode to a directory. You can export the runs from Web Analytics report, workbench, and command line. To export from the workbench, select a single run or multiple runs and click **Export > Test > Performance Test Run Statistics as CSV File**. To export data of specific time ranges, on a subsequent page select a time range.

To export the run from command line, see the parameters in the Running a test from a command line on page 295 topic.

1. Open the test run to export.

2. Click the **Menu** icon , click the **Share** icon , and click **Export Session to CSV**.
3. Select the encoding system for the export.
4. Complete either one of the following steps:
   **Choose from:**

◦ To export only the last value of each counter from the results or to export data of specific time ranges, select **Simple**.

> 📝 **Note:** When you export data of specific time ranges, for example, 5 Users or 15 Users, a separate column is created in the CSV file for each time range.

- ▪ To create multiple CSV files if the number of columns exceed the specified value, select the **Split output if column exceeds** check box and specify a value.

◦ To export all of the data for the run, select **Full**.

To include description about the name of the run, node name, and time range for each counter, select the **Include per instance counters**.

◦ To export data of each location (agent) in a separate section in the CSV file, select the **Export each agent separately**

To export data of each location (agent) to separate CSV files, select the **One file per agent** check box.

5. Click **Export**. If you export from the workbench, the report is saved in the specified folder. If you export from an external browser, the report is downloaded in a compressed format to the default download location of the browser.

**What to do next**

You can now analyze and share the report with people who are not using the workbench.

---

Related information

## Sharing URL of test run

When you share the URL of the test run with other people, they can view and analyze the test results on a browser on their computer if the test workbench is running on your computer at that time.

To share the URL of the test run:

1. Open the test run to share.

2. Click the **Menu** icon  and click the **Share** icon  and select **Share Execution Result URL**.
   A unique URL is created for the test run.

3. Copy the URL and click **Close**.

**What to do next**

You can now share the URL of the test run with anybody.

## Exporting report metadata

To share report metadata with another test workbench user, export the report definition. Use this option to share customized report formats with other users. The recipient imports the metadata with Eclipse's **Import** option and views the report from the Test Navigator or in the list of reports in the web report.

To export report metadata:

1. Click **File > Export**.
2. In the **Export** window, expand the **Test** folder, select **Report Definitions**, and click **Next**.
3. In **Save to File**, select the file that will contain the report. This file is created if it does not exist.
4. In **Select Report**, select the report to export, and then click **Finish**.
   The file is saved in the `.report` format.

**What to do next**

To apply another report definition to your reports, import that report metadata by clicking **File > Import > Report Definition**, and browse to the `.report` file.

## Viewing response time breakdown

You can do detailed analysis of the response time to find bottlenecks in the HTTP traffic of the application.

## Viewing page element responses

You can view the response times for individual page elements in reports, to determine which elements are the slowest.

**About this task**

Page element response times do not include client delay or connection time. Because page elements can be returned in parallel from the server under test, the page response time is not necessarily the sum of the page element response times.

1. Open the web analytics reports.
2. On the Page Performance report, click a page (represented by a bar) and click **Page Element Responses**.
   The Page Element Responses report displays response time for all of the elements of the page.
3. To return to the original report, click the Page Performance link in the breadcrumb.

## Viewing page response time contributions

You can view the response time contributions for individual pages to determine how much time was actually taken by the page to load and the time taken for the connection to go through and the delay on the client side of each page.

**Before you begin**

Because page elements can be returned in parallel from the server under test, the page response time is not necessarily the sum of the page element response times. Client delay and connection time also contribute to page response time. The page response time can be greater than the sum of the page element response times if, for

example, a lengthy connection time adds a delay. Connection time includes the time required for Domain Name Services (DNS) lookups. Conversely, the page response time can be less than the sum of the page element response times if multiple page elements are returned in parallel.

1. Open the web analytics report.
2. On the Page Performance report, click a page (represented by a bar) and click **Page Response Time Contributions**.

   The Page Response Time Contributions report shows the average response time taken for Connection Time, Client Delay Time, and Page Element Response Time.
3. To return to the original report, click the Page Performance link in the breadcrumb.

## Viewing response time breakdown data

In addition to viewing page element and page response time, you can do further analysis to view the response time data for each method.

**Before you begin**

You must have instrumented the servers. See Instrumenting local servers.

1. Open the web analytics reports.
2. On the Page Performance report, click a page (represented by a bar) and click **Display Response Time Breakdown**.
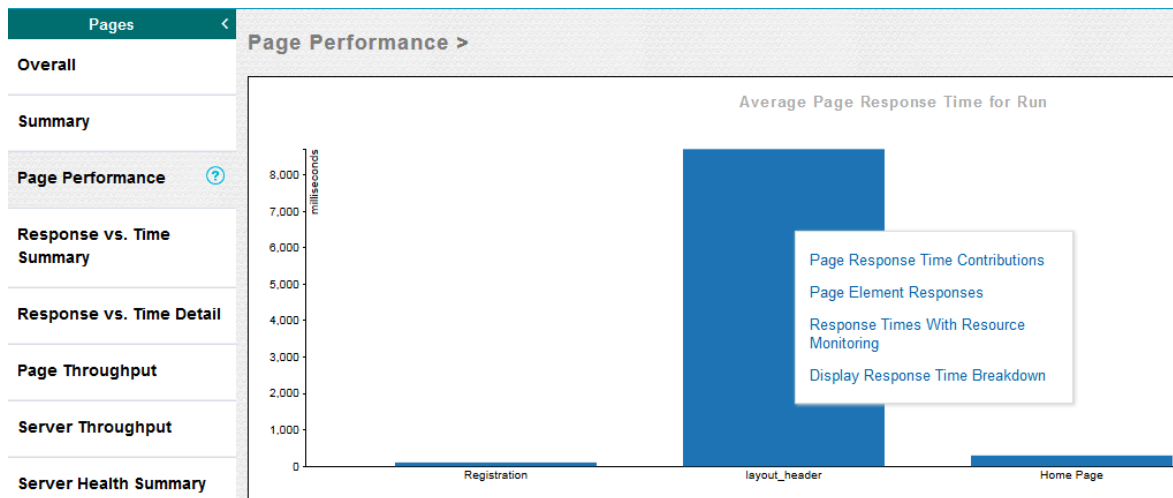


   You can also directly access the response time breakdown data from the Page Element Responses report.
3. In the workbench, the **Page Element Selection** dialog is open.
4. Select a page element to view response time breakdown for and click **Finish**.

## Comparing service test response contents

After running a service test with equal or contain verifications points, you can use the **Raw transaction data** view to compare the actual XML or text contents of the message returns with the expected results of the verification points in the test log.

**Before you begin**

Before comparing results you must have run a service test with equal or contain verification points.

To compare the actual and expected results of a service message return:

1. After running the test, open the test log.
2. Make sure that the **Raw Transaction Data** view is open. If necessary, click **Window > Show View > Raw Transaction Data** to open the view.
3. Select an equal or contain verification point.
   **Result**

   The **Raw Transaction Data** view displays a comparison between the expected data from the verification point and the actual data that is received during the run.

---

Related information

# Logs overview

uses logs to store different types of information, which you can use to determine the reason for a test failure.

has the following logs:

**Test logs**

The test log contains a historical record of events that occurred during a test run or a schedule run, as well as the status of each verification point. The test log sets a verdict for each run as follows:

- **Pass** indicates that all verification points matched or received the expected response and all the test steps successfully completed.
- **Fail** indicates that at least one verification point did not match the expected response or that the expected response was not received or a Web UI step did not run successfully.
- **Error** indicates one of the following results: a primary request was not successfully sent to the server, no response was received from the server for a primary request, or the primary request response was incomplete or could not be parsed.
- The verdict is set to **Inconclusive** only if you provide custom code that defines a verdict of **Inconclusive**.

The verdict is rolled up from the child elements to the test level. For example, if a user group contains 25 virtual users, and five virtual users have failed verdicts, that user group has only one failed verdict, not five.

The test log file is stored in binary format with a `.executiondlr` file name extension in the project directory of your workspace. You can also view the test log in the user interface.

For more information about viewing test logs, see .

**Problem determination logs**

You can set the level of information that is saved in the problem determination log during a run. By default, only warnings and severe errors are logged. Typically, you change this log level only when requested to do so by the Support person.

The problem determination logs contain internal information about the playback engine. These logs are particularly useful for debugging problems such as Kerberos authentication, SSL negotiation, and resource constraints on an agent. The log files are named `CommonBaseEvents00.log` and are located in the deployment directory. For example, if you play back a schedule on an agent and set `C:\Agent` as the deployment directory, the problem determination log files are in a directory similar to `C:\Agent\deployment_root\<UserName>` `\A1E14699848784C00D2DEB73763646462\CommonBaseEvents00.log`. If a large amount of log information is generated, multiple `CommonBaseEvents` files are created.

For more information about setting problem determination level, see Setting the problem determination level.

**Agent logs**

Look in `%TEMP%` directory for the majordomo.log file. This file contains information about the attempts to contact the workbench including information about any failures and the reason for the failures.

On the Microsoft™ Windows operating system, the `%TEMP%` directory is typically at `%USERPROFILE%\AppData\Local` `\Temp`.

If the majordomo service is configured to log in as Local System Account, then the `%TEMP%` directory is at `%SystemRoot%\TEMP`, typically `C:\Windows\TEMP`.

**Error logs**

If an error message is displayed when you run tests, try looking up the error message in the *Performance testing error messages* section of the online help. Only the most common error messages are listed. If no error message is displayed when you encounter a problem, open the error log by clicking **Window > Show View > Error Log**. If the workbench shuts down while running tests, restart the workbench and examine the error log. By default, warning and error messages are logged. You can increase the default logging level by clicking **Window > Preferences > Logging**. The log file is stored in the `.metadata` directory of your workspace. To avoid excessive logging, the Logging Level should be adjusted for individual Logger Names in the Loggers tab. For example, to get more information about a problem connecting with IBM® Rational® Quality Manager, increase the Logging Level for com.ibm.rational.test.lt.rqm.adapter Logger Name. For the licensing issue, adjust the level for com.ibm.rational.test.lt.licening Logger Name. When you no longer need the extra logging, use the **Restore Default** button in the Logging Preferences to reset all the levels to their recommended defaults.

## Viewing test logs

To see a record of all the events that occurred during a test run or a schedule run, as well as the status of each verification point, open the test log for that run. You can also compare an event from the test log with the request or response in the test to view the differences between the recording and the playback of the test.

**About this task**

The test log file is stored in binary format with a `.executiondlr` file name extension in the project directory of your workspace. You can also view the test log in the user interface.

1. In the Test Navigator view, right-click the executed test; then click **Display Test Log**.
2. On the **Overview** tab, view the verdict summary for the executed test. To see the potential data correlation errors in a separate view, click **Display Potential Data Correlation Errors**.
3. On the **Events** tab, view the errors, failures, and passes for each event in the test.

   ◦ To navigate to the verdict type, click the **Select the verdict type** ✔ ▼ icon.
4. On the **Data Correlation** tab, see all the references and substitutions that occurred during a test execution, as well as the data correlation errors. By default, you view both references and substituters. To view only substituters, click the **Show References** 🔳 icon. To view the correlation data for each virtual user that was executed, click the **Merge Users** 👥 icon. This icon is enabled only for a schedule. In the **Data Correlation** section, when you click an event, you can see the correlation data in either the Content View or the Table View.

**What to do next**

From the test log, you can submit, search, and open defects in a defect tracking system. For details on configuring the test log preferences and working with defects, see Associating defects with a test log.

## Viewing reports after a run

Reports are generated and displayed automatically after a run. Each test result begins with the name of the schedule or test, and ends with the timestamp of the run in brackets.

**About this task**

In version 8.5.1 or later, for a service test report, you can choose not to generate a report automatically after a run by clicking **Window > Preferences > Test > Performance Test Reports > Service Test Reports** and clearing the **Functional Test Report generation after Test execution**.

1. In the Test Navigator, expand the project until you locate the run.
2. Do either of the following:

◦ To view the default report, double-click the run. To change the default report, Open the Default Report Preferences page. Click **Window > Preferences > Test > Performance Test Reports > Default Report**

◦ To view another report, right-click the test run, click **Display Report**, and then select the report to display.

> 📝 **Note:** You can also view reports remotely from a web browser. For information about viewing reports remotely, see .

## Accessing reports remotely

Before executing a schedule or test, you can enable an option so that you can access reports remotely from a web browser. When you make changes to a report, the changes are saved to the workspace where the workbench is running.

1. On the Rational® Performance Tester workbench, click **Windows > Preferences > Test > Performance Test Reports > Web Reports**.

2. To enable remote access to reports, select the **Allow remote access from a web browser** check box.

3. To enable the remote control of schedule execution tasks, select the **Allow control of schedule execution from the web browser** check box.

4. **Optional:** By default, the non-secure port number for web reports is 8080. If this port number is used by another service, you can type another port number.

5. **Optional:** To provide security for web reports, select the **Security is required to access reports** check box.

   a. By default, the secure port number for web reports is 8443. If this port number is used by another service, you can type another port number.

   b. Select the **User authentication is required to access reports** check box and specify the login credentials.
   You must use the same login credentials to access reports remotely.

6. Click **OK**.

7. To access web analytics reports remotely, open a web browser and type
   `http://host_name:8080/analytics/web/index.html`. To access a secured report, type
   `https://host_name:8080/analytics/web/index.html` and specify the login credentials if you have set it.

   To access old web reports remotely, on another computer, open a web browser and type
   `http://host_name:8080/RPTWeb/WebAnalytics/`. To access a secured report, type
   `https://host_name:8080/RPTWeb/WebAnalytics/` and specify the login credentials if you have set it.

   The host name is the Rational® Performance Tester workbench computer name and the port number is as specified in **Windows > Preferences > Test > Performance Test Reports > Web Reports**.

## Exporting test logs

To process data from a performance test in another application or to use search tools to locate text in a test log, export the test log to a text file.

1. In the Test Navigator, right-click the run, and select **Export Test Log**.
    a. **Optional:** To export only a portion of the test log, open the test log by right-clicking the test run and then selecting **Display Test Log**. Right-click the elements to export, and then select **Export Log Element**.

   **Result**

   The **Export Test Log** window opens.
2. In the Export Test Log window, specify a location for saving the file, and then select options as follows:

| Option | Description |
|---|---|
| Export format | Select default encoding or Unicode encoding. |
| Include event time stamps | Select to include event time stamps. |
| Include detailed protocol data | Select to include detailed protocol data. This option is available only for HTTP test runs. |
| Include response content | Select to include response content. This option is available only for HTTP test runs. |
| Include known binary data | Select to export binary data. This option is available only for HTTP test runs. |

3. Click **Finish**.

   **Result**

   The test log is exported to a text file.

## Exporting event log

To view all the events that occurred during the run of a test from another file, you can export this data from the Event Log panel, to an XML, CSV, or text file.

**Before you begin**

You must run a test to view data in the Event Log panel.

1. On the Event Log panel toolbar click the **View Menu** arrow icon ▽ and select **Export Event Log**.
2. In the Save dialog box, specify the location and format in which you want to save the events.

## Exporting event console output

To view errors and other events of a test run from another file, you can export this data from the Execution Event Console view to an XML, CSV, or text file.

**Before you begin**

- Ensure that the Execution Event Console view is open by clicking **Window > Show View > Execution Event Console**.
- Ensure that the test is run and the Execution Event Console view contains data.

1. From the Execution Event Console view toolbar, click the **View Menu** arrow icon ▽ and select **Export**.
2. In the Save dialog box, specify the location and format in which you want to save the events.

## Viewing resource monitoring data

You can analyze the performance of the computer resources, application server, or database servers by viewing the resource monitoring data in web analytics reports.

## Adding resource counters to reports

To view performance data of resource counters that are not shown in the report by default, you can add the resource counters.

**Before you begin**

You must have enabled capturing of resource monitoring data in the schedule. See Enable resource monitoring.

1. Open the Performance Report, and from the list of Pages, click **Resources**.
2. Click the **Menu** icon [Menu] and then click the **Edit** icon .
3. To update the graph, click the **Settings** icon ⚙ on the graph. There would be multiple Settings icons for a report.
4. On the View Settings page, click **Counters**, and click the **Add** icon + .
5. From the dropdown, select **Resource Monitoring** and then from the **Component** dropdown, select a unit of measurement such as Min, Max, or Average.
6. Click **Apply**, click **Save**, and then click the **Edit** icon.

**Results**

The changes are reflected in the resource monitoring graph.

## Filtering resource counters

To view the performance data that is more important for you, filter the resource monitoring counters. You can filter resource counters the same way you filter other results displayed in reports.

**Before you begin**

To view the Resources page in the Performance Report, you must have enabled capturing of resource monitoring data in the schedule. See Enable resource monitoring.

1. Open the Performance Report, and from the list of Pages, click **Resources**.

2. Click the **Menu** icon  and then click the **Edit** icon .

3. To update the graph, click the **Settings** icon � on the graph. There would be multiple Settings icons for a report.

4. On the View Settings page, click **Filters**.

5. From the dropdown, click one of the following filter types:

| Option | Description |
|---|---|
| Filter by count | Display the specified number of items. For example, if you select this option and then type `15`, the report will show the 15 items with the highest values (**Show highest values**) or the 15 items with the lowest values (**Show lowest values**). |
| Filter by value | Display items based on a comparison with the specified value. For example, if you select this option and then type `15`, the report will show all of the items that are higher than 15 (**Show counters above filter value**) or lower than 15 (**Show counters below filter value**). |
| Filter by label | Display items that match the specified label. If you are filtering a table, the label is typically a page, and is listed in the left column. If you are filtering a graph, the label is a legend in the graph. |

6. From the **Primary counter for table filtering** dropdown, click a primary resource counter.

   Typically, this is the **Resource Monitoring** counter.

7. Click **Apply**, click **Save**, and then click the **Edit** icon.

**Results**

The changes are reflected in the resource monitoring graph.

# Chapter 9. Troubleshooting Guide

This guide describes how to analyze and resolve some of the common problems that you might encounter while you work with  Rational® Service Tester for SOA Quality.

## Troubleshooting performance testing

This topic provides information about how to troubleshoot several problems with IBM® Rational® Service Tester for SOA Quality.

If you run tests and encounter problems, make sure that you have followed all the Performance testing tips.

If an error message is displayed when you run tests, try looking up the error message in the *Performance testing error messages* section of the online help. Only the most common error messages are listed. If no error message is displayed when you encounter a problem, open the error log by clicking **Window > Show View > Error Log**. If the workbench shuts down while running tests, restart the workbench and examine the error log. By default, warning and error messages are logged. You can increase the default logging level by clicking **Window > Preferences > Logging**. The log file is stored in the `.metadata` directory of your workspace. To avoid excessive logging, the Logging Level should be adjusted for individual Logger Names in the Loggers tab. For example, to get more information about a problem connecting with IBM® Rational® Quality Manager, increase the Logging Level for com.ibm.rational.test.lt.rqm.adapter Logger Name. For the licensing issue, adjust the level for com.ibm.rational.test.lt.licening Logger Name. When you no longer need the extra logging, use the **Restore Default** button in the Logging Preferences to reset all the levels to their recommended defaults.

In addition to the online help, you can find workarounds or solutions to problems in the Rational® Performance Testing forum on developerWorks®, and in the Support Knowledge Base technotes for  Rational® Performance Tester.

You might encounter some of these problems while performance testing:

### Connectivity problems between workbench and agent computers

If the workbench stops or locks up when you attempt to start running tests, it is important to confirm that all the agent computers are running. Perform the following steps to confirm your installation is properly configured:

- Confirm that there is sufficient disk space available on the workbench computer and the agent computers.

- Restart the workbench computer.

- Verify the network connectivity between the workbench computer and agent computers. To confirm the hostname in majordomo.config file can be DNS resolved on the agent machine, use a shell ping to the workbench hostname. If the ping results fail use the IP address of the workbench instead.

- Confirm the server port number on the test workbench computer. Click **Window > Preferences > Server**. This is the port number that should be specified in majordomo.config file on the agent machines.

- Restart the agent computers and verify the Majordomo process is running.

- On the agent machines, set the optional debug flag in the majordomo.config file. Set the value equal to true; the default value is false. You do not have to restart the agent. Within about ten seconds it should automatically pick up the changes to majordomo.config.

  Look in `%TEMP%` directory for the majordomo.log file. This file contains information about the attempts to contact the workbench including information about any failures and the reason for the failures.

  On the Windows operating system, the `%TEMP%` directory is typically at `%USERPROFILE%\AppData\Local\Temp`.

  If the majordomo service is configured to log in as Local System Account, then the `%TEMP%` directory is at `%SystemRoot%\TEMP`, typically `C:\Windows\TEMP`.

- You can check the agent status on the workbench computer by clicking the  icon. For the Agent Controller, you can attempt to share files between the workbench computer and agent computers. Click **Window > Preferences > Agent Controller > Hosts**, and then add the agent computers as hosts, and click **Test Connection** to test connectivity to the instances of the Agent Controller that are running on the agent computers.

## Recording configuration problems

### No HTTP traffic is captured while recording

See Recording reliable HTTP tests for instructions on configuring your web browser. If you are attempting to use Internet Explorer to record tests from a secure website, see Configuring Internet Explorer for recording from a secure web site. Disable firewalls on the workbench computer and the agent computers.

### No traffic is captured while recording

Ensure that the recorder type that you select matches the protocol in use by the system under test. For example, do not attempt to use the HTTP recorder if the system under test uses the Citrix protocol.

### No test is generated after recording

When the test generator cannot create a test from the recorded traffic, typically an error message is displayed or written to the error log. Try looking up the error message in the *Performance testing error messages* section of the online help. Error messages might also be documented in technotes in the Support Knowledge Base at http://www.ibm.com/software/awdtools/tester/performance/support/.

### Recorder controls are not available

If you use a workspace from a different version of the product, the recorder controls might not be available. Instead, the recorder controls from the other version of the product are displayed. Click **Window > Reset Perspective** to reset the **Performance Test** or **Service Test** perspective. Alternately, click **File > New > Other** to select the wizard to use.

## Problems running large tests or long-run tests

If a test runs but ends with errors, check that the workbench computer and agent computers meet the hardware and software requirements that are detailed in the installation guide. Pay close attention to the memory and disk space requirements. See Increasing memory allocation on page 302 for more information on how to set the maximum heap size to avoid out-of-memory errors. Monitor processor and memory usage on the workbench and agent computers and watch for excessive processor use or excessive memory use by javaw.exe or java.exe processes. If error messages pertain to processes stopping unexpectedly, see this support article: http://www.ibm.com/support/docview.wss?uid=swg21395486.

Run tests with fewer virtual users that use the default schedule settings to determine whether the behavior is linked to the number of users. Examine the test log for error messages that the system under test generates. Run tests with a single virtual user and make sure that the system under test is not generating errors, before you attempt to run tests with a large number of users. If you encounter problems, restart the workbench and agent computers before attempting to run tests again.

If the workbench shuts down while running tests, search for file names that begin with `javacore`. The name of `javacore` files includes the date, time, and process ID. If you find a `javacore` file with a date, time, and process ID matching the workbench, open the file in a text editor. You can find the reason for failure at the beginning of the `javacore` file.

## Data correlation errors

If you can record tests successfully, but the expected behavior is not triggered in your application when you run tests, you might need to perform manual data correlation. Typically when additional data correlation is needed, the test log includes messages similar to this message:`Unable to extract the value`. To troubleshoot data correlation problems, try running tests using only one virtual user running on the workbench computer, and compare the playback to the recorded test to determine which responses from the system under test are unexpected. See Debugging HTTP tests to learn how to use the test log and the **Protocol Data** view to troubleshoot HTTP tests. To learn more about data correlation, see Correlating response and request data on page 237.

## Common errors integrating with IBM® Rational® Quality Manager

All modes of the adapter use the Eclipse error log. You can view the log by opening the workbench and clicking **Window > Show View > Error Log**. By default, warning and error messages are logged. You can turn on more detailed logging for the adapter by clicking **Window > Preferences > Logging**. The log component for the adapter is named `com.ibm.rational.test.lt.rqm.adapter`.

If you are running the adapter as a Windows™ service or from the command line, you can view the `adapter.log` file without opening the test workbench.

| Problem | Solution or cause |
|---|---|
| Where do you look for errors or warnings? | In the workbench, click **Window > Show View > Error Log**. |

| Problem | Solution or cause |
|---|---|
| You do not see the adapter available for selection. | <ul><li>Verify that the Engineering Test Management server address that is provided to the adapter is correct. Provide the correct address.</li><li>Check the provided login and password. Provide the correct password.</li></ul> |
| The adapter continuously fails to connect to Engineering Test Management. | Make sure that the server is running. If necessary, restart the server or check network connectivity. |
| The adapter is displayed as red in the selection dialog box. | <ul><li>The adapter is not communicating with the server.</li><li>The adapter might already be in use.</li></ul> |
| You attempt to import a script from the adapter but no scripts are found. | <ul><li>Make sure the project path that is entered in Engineering Test Management is a project under the workspace that is associated with the running adapter. You have to enter only the project name. This is less error prone than typing the complete project path, but either forms are acceptable.</li><li>If running from the command line or as a service, be certain the WORKSPACE_DIR environment variable that is set in the `adapter.config` file is the same path as seen in the select workspace dialog box when running the test workbench. Be careful not to set the path to a project folder under the workspace directory.</li><li>Make sure that you are not using a workspace that contains a project that was copied from a shared location. A workspace that contains projects from shared locations cannot be used for projects that are not shared.</li></ul> |
| The adapter is running from the command line or as a service, and tests continue to fail. | Run the adapter in GUI mode so that you can see what happens when the test workbench runs the test script. |
| Adapter Windows™ services does not start. A error message states that the service failed to start in a timely fashion. | Ensure that the computer has .NET 2.0 or later. This platform can be installed from the Windows™ Update Site or manually. For more information on installing .NET, see http://support.microsoft.com/kb/923100. |
| When testing shared assets, the execution fails with and an `IOException` message is displayed. | The most likely cause is that the Engineering Test Management to UNC shared location is not set up correctly. |

| Problem | Solution or cause |
|---|---|
| | • From Engineering Test Management, ensure that you can access the UNC shared directory without being prompted for a password. You might have to map a drive on Windows™ for the Engineering Test Management system to log into the UNC share.<br>• Ensure that you have defined the shared resource in Engineering Test Management under **Admin > System Properties > Resources**.<br>• Ensure that the test-script points to a shared location that still exists. If you have associated a Engineering Test Management test script with a shared location that has changed (for example if the IP address has been reassigned) you might need to reassociate every test script<br>• Ensure that the UNC shared directory that is specified in Engineering Test Management points to a project. |
| When testing shared assets, the execution fails with a low level model error. | Ensure that the adapter has the required protocol extensions installed. The test assets located on the shared location can only be run on an adapter workspace that supports those protocols. |
| Service tests that were created in a previous version of the product cannot be run. | Upgrade every SOA asset to the latest version. |
| The adapter cannot connect to the server, and one of the following error messages is displayed:<br>• `Communications error with server`<br>• `Error occurred while registering the adapter` | • When using Engineering Test Management 3.0 or later, the server URL that is configured for the adapter must exactly match the public URI of the Engineering Test Management server. The server public URI is available on the Engineering Test Management administration page. By default the administration page is at https://*server-name*:9443/qm/admin.<br>• The adapter user must be a member of the Engineering Test Management project area. Open the project area administration page on the Engineering Test Management server to determine whether the adapter user is a member of the project area. For Engineering Test Management 3.0 and later, the adapter user must be a member in the test team member role, not the test team contributor role. This error can also occur if you have modified these roles from their defaults. |

# Performance testing error messages

Find more information about the error messages.

## PRXE**0101**W

%1 terminating due to exception: %2

## PRXE**4943**W

Transaction [%1] has been aborted.

## PRXE**4951**I

User group [%1] was not found.

## RMSE**0003**W

RMSE0003W There are currently no selected counters for the source named *{0}*.

**Explanation:** The source has no counters selected.

**System action:** Execution of the schedule will continue but the information related to this source won't be collected.

**User response:** Consider selecting at least one counter from the Resource Monitoring Service web console.

## RMSE**0004**W

RMSE0004W
The
source
named
*{0}*
is
no
longer
available.

**Explanation:** This source has been removed from the Service web console after it was added to this schedule.

**System action:** Execution of the schedule will continue but the information related to this source won't be collected.

**User response:** Consider adding it back, then edit the schedule to update the sources to be monitored during its execution.

## RMSE**0005**W

RMSE0005W
The
source
named
*{0}*
is
reporting
the
error
message
*{1}*.

**Explanation:** Look at the reported error.

**System action:** Execution of the schedule will continue but the information related to this source won't be collected.

**User response:** Consider fixing it from the Resource Monitoring Service web console.

## RMSE**0006**W

The
server
does
not
support
resource
monitoring
labels.

**Explanation:** The server does not support resource monitoring labels.

**System action:** Execution of the schedule will continue but the resource monitoring counters won't be collected.

**User response:** Consider using a service that supports this feature.

## RPAC**0001**W

The
JAR
%1
referenced
in
preferences
could
not
be
found.
Preferences
on
the
cloud
workbench
will
be
cleared.

**Explanation:** The Resource Monitoring preferences list a JAR file that is required for an instrumented application server type. This JAR file must be mapped to a new location and transferred to the cloud workbench. But this transaction failed, because the file could not be found locally.

**System action:** Execution in the cloud will continue but the instrumented application server types that require the listed JAR file might fail.

**User response:** Open the child preference page under Test -> Performance Resource Monitoring. Ensure that the listed files exist and can be found in a valid location.

---

## RPHD**1032**E

Error occurred while instructing __PT_ACRONYM__ engine to enable real-time protocol data for user: %1. It's possible that no data will be seen for this user in the Protocol Data view.

**Explanation:** There was a general error when starting real-time browsing in the Protocol Data View.

**System action:** The Protocol Data View will not be updated in real-time during this run. This does not affect test execution or post-run usage of the view.

**User response:** Ensure there is a stable connection with the Performance Test Agent and System Under Test. If problem persists, contact support.

---

## RPHD**1034**E

Error occurred while instructing __PT_ACRONYM__ engine to disable real-time protocol data for user: %1. It's possible that data for this user will continue to be displayed in the Protocol Data view.

**Explanation:** = There was a general error when ending real-time browsing in the Protocol Data View.

**System action:** None.

**User response:** If the Protocol Data View no longer updates for additional runs or when the test editor selection is changed, closing the view and reopening it may help.

---

## RPHE0001E

example
of
translatable
error
message
%1

---

## RPHE0010W

Unknown
authentication
scheme
'%1'
discovered
in
HTTP
401
response,
ignoring.

---

## RPHE0011W

Unrecognized
authentication
header
'%1'
discovered
in
HTTP
401
response,
ignoring.

## RPHE0012W

No
authentication
headers
found
in
HTTP
401
response,
ignoring.

## RPHE0013W

The
server
requested
NTLM
authentication
but
no
NTLM
authentication
context
was
supplied
with
this
request.
Authentication
is
not
possible.

## RPHE0014W

NTLM authentication failed for this request. Verify that the NTLM authentication context values for this request are correct.

## RPHE0100W

Host name '%1' can not be resolved.

**Explanation:** A connection could not be established with the host. This can occur if the testing environment changes so that the host name is no longer correct. This can also occur when running a test on a different computer, such as an agent computer, from the workbench computer that was used for recording, if the new computer cannot resolve the host name.

**User response:** If the host name is incorrect due to a change in the testing environment, update the host name in the test. Otherwise, try to resolve the host name using the command nslookup <hostname>. Run nslookup on the agent computer if the error is happening on the agent computer. If nslookup is also unable to resolve the name, contact your network administrator. If nslookup resolves the host name, but the test continues to fail, try changing the host name to a fully-qualified host name. Alternatively, edit the hosts file.

## RPHE0101W

Encountered error while updating dynamic cookie cache while interpreting 'Set-Cookie' header with value '%1' sent from web-server '%2' retrieving URI '%3'. Explanation message: '%4'. Cache not updated to include this cookie value.

## RPHE0102W

Unexpected challenge(HTTP status code=401) received during HTTP playback to web-server '%1' retrieving URI '%2'. This behavior differs from the behavior recorded during test creation. For authentication to playback correctly a challenge must be recorded during test creation.

## RPHE0103W

Authentication failed during HTTP playback to web-server '%1' retrieving URI '%2'. Probable cause: username '%3' and/or password '%4' incorrect.

## RPHE0104W

Exception occurred during attempt to write request to web-server '%1' getting url '%2'. Explanation: %3

## RPHE0105W

General un-handled exception occurred during socket I/O read from web-server '%1' retrieving URI '%2'. Explanation message: '%3'.

**Explanation:** This error occurs when the server abruptly closes the connection to the virtual user. Servers might close connections if the virtual user is detected as a security risk due to a invalid cookie, failed SSL negotiation, or an improperly formatted request.

**User response:** Compare the request that was sent at run time (in the test log) to the one that is in the test. To determine if differences between the requests are valid, record the test again and compare the two requests.

## RPHE0106W

A read time-out occurred during a socket I/O read from web-server '%1' retrieving URI '%2'. Since this URI is the primary request for the current page all secondary requests will be skipped and the next page will be attempted. Current time-out value

**Explanation:** The server did not return the response data before the timeout interval elapsed. If the server is under heavy load, the behavior can be caused by bottlenecks on the server or the agent computers. This error can also occur if an incorrect request is sent and the server is unable to respond.

**User response:** If the server is under heavy load, examine the server and agent computers to find and fix bottlenecks. Increase the timeout value. To stop tests or virtual users when this error occurs, enable error handling in the test and configure the server timeout error condition. If the server is not under heavy load, examine the request to ensure that it is valid and accurate.

## RPHE0107W

A read time-out occurred during a socket I/O read from web-server '%1' retrieving URI '%2'. This secondary request will be skipped. Current time-out value of '%3' milliseconds should be increased if long delays are expected on this request.

**Explanation:** The server did not return the response data before the timeout interval elapsed. If the server is under heavy load, the behavior can be caused by bottlenecks on the server or the agent computers. This error can also occur if an incorrect request is sent and the server is unable to respond.

**User response:** If the server is under heavy load, examine the server and agent computers to find and fix bottlenecks. Increase the timeout value. To stop tests or virtual users when this error occurs, enable error handling in the test and configure the server timeout error condition. If the server is not under heavy load, examine the request to ensure that it is valid and accurate.

## RPHE0108W

A connect time-out occurred during a socket I/O connect to web-server '%1' attempting to retrieve URI '%2'. Since this URI is the primary request for the current page all secondary requests will be skipped and the next page will be attempted.

**Explanation:** This error can occur if the server or agent computer is under heavy load. This error can also occur if the server or host computer is not configured with enough connections, or if the agent computer is not configured with enough sockets.

**User response:** Examine the server and agent computers to find and fix bottlenecks. To stop tests or virtual users when this error occurs, enable and configure error handling in the test.

---

## RPHE0109W

A connect time-out occurred during a socket I/O connect to web-server '%1' attempting to retrieve URI '%2'. This secondary request will be skipped.

**Explanation:** This error can occur if the server or agent computer is under heavy load. This error can also occur if the server or host computer is not configured with enough connections, or if the agent computer is not configured with enough sockets.

**User response:** Examine the server and agent computers to find and fix bottlenecks. To stop tests or virtual users when this error occurs, enable and configure error handling in the test.

## RPHE0110W

Unexpected challenge(HTTP status code=407) received while accessing HTTP proxy '%1' retrieving URI '%2'. This behavior differs from the behavior recorded during test creation. For authentication to playback correctly a challenge must be recorded during test creation.

**Explanation:** When the test was recorded, no basic authentication was required on the proxy server. When the test is run, the proxy server is requesting basic authentication information that is not in the test.

**User response:** Record the test again to capture basic authentication information. Play back the new test, or add the basic authentication information to the request in the original test.

---

## RPHE0111W

Authentication failed accessing proxy-server '%1' retrieving URI '%2'. Probable cause: username '%3' and/ or password '%4' incorrect.

**Explanation:** Basic authentication failed when connecting to the proxy server. This can occur if an incorrect user name or password is supplied.

**User response:** Ensure that user name and password are correct.

## RPHE0112W

An error occurred during decoding of content received from web-server '%1' attempting to retrieve URI '%2'. Explanation message: '%3'.

## RPHE0113E

Error encountered during the process of URI substitution for host= %1 and URI = %2 . Data correlation supplied a malformed URI= %3 . Explanation: %4. If you attempted to perform a custom data substitution on this URI ensure it has proper URI syntax. If you did

## RPHE0113W

An error occurred during encoding of an annotated execution history event property. Explanation message: '%1'.

## RPHE0114E

An error was encountered during transform of response data. %1

**Explanation:** The response data was not in a format that the data transformer could interpret. This can occur when an error is returned from the server instead of valid response data.

**User response:** Examine the response data for errors.

## RPHE0114W

Exception occurred during attempt to write request to proxy-server '%1' getting URL '%2' on host '%3'. Explanation: %4.

## RPHE0115E

An error was encountered during un-transformation of request data. %1

**Explanation:** The transformed request data could not be converted into the format required by the server. This can occur because of a faulty data substitution. This can also occur if you manually edit the request data and invalidate the transformed data format.

**User response:** Correct the faulty substitution or the invalid data formatting.

## RPHE0115W

Unable to successfully establish a connection to web-server '%1' retrieving URI '%2'. Web-server closing the connection after connection was just established.

## RPHE0117W

Unexpected exception occurred during connection close to web-server '%1' retrieving URI '%2'. Explanation: %3.

## RPHE0118W

HTTP parsing error encountered while retrieving URI '%1' from web-server '%2'. If this URI is the primary request for the current page all secondary requests will be skipped and the next page will be attempted.

## RPHE0119E

IP aliasing is enabled but no IP address was found for virtual user %1. Verify correct network interface name(s) are specified.

## RPHE**0120**E

Exception occurred during attempt to connect to proxy-server '%1' getting URL '%2' on host '%3'. Explanation: %4.

## RPHE**0121**E

Unable to authenticate with the proxy-server. Possible solution: re-record test due to possible proxy-server '%1' authentication changes.

## RPHE0122W

Web-server '%1' unexpectedly closed the connection while in the process of retrieving URI '%2'. The response body MAY be incomplete due to a missing "chunk". If missing chunk was last (zero length) chunk, data is complete.

## RPHE0123W

Infinite redirection loop detected getting URL '%1'. If this is expected and understood increase RPT_VMARGS rptMaxRedirection parameter. Redirected history %2

## RPHE0124W

Unexpected server redirection occurred getting URL '%1'. We were redirected to the same URI which issued this request. Redirected history %2

## RPIB0007E

%1

## RPKG0090E

Exception thrown while creating connection variables

**Explanation:** Exception thrown while creating connection variables

**System action:** Can not create the connection variable

**User response:** None required

## RPKG0100E

Exception
thrown
by
the
launch
configuration
core

**Explanation:** Exception thrown by the launch configuration

**System action:** None required

**User response:** None required

## RPKG0101E

Exception
thrown
during
an
update
to
a
launch
configuration

**Explanation:** Exception thrown during an update to a launch configuration

**System action:** None required

**User response:** None required

## RPKG0110E

The
data
source
type
%1
is
not
expected

**Explanation:** The data source type %1 is not expected

**System action:** None required

**User response:** None required

---

## RPSF0172E

__PT_ACRONYM__/
SAP:
Unable
to
start
SAP
GUI,
please
check
SAP
GUI
installation.

**Explanation:** SAP GUI can't be reached.

**System action:** The test is stopped.

**User response:** Install SAP GUI with scripting options as recommended by SAP.

---

## RPTA0001I

Setting
the
log
verbosity
left
me
with
%1
users

## RPTA**0002**E

A
Test
cannot
be
launched
on
the
specified
Driver

## RPTA**0003**E

%1

## RPTA**0004**E

A Test could not be launched on Driver: %1. The Test Execution Framework was not able to deliver an Executor. This is an internal error, please contact support. For more information, see the Troubleshooting section of the online help.

**Explanation:** The Test and Performance Tools Platform (TPTP) infrastructure did not produce an executor for the test. This error message might display if firewalls are active on the local computer or the agent computer.

**User response:** Disable firewalls on both the local computer and the agent computer. If you do not want to disable firewalls, you can instead enable a firewall-aware connection. For more information on enabling a firewall-aware connection, see Running with a workbench behind a firewall. On the local computer, check the properties of the location that represents the agent computer. This error can occur if the deployment root directory is not specified correctly in the location that represents the agent computer. Check the Error Log for further information on the error. To open the Error Log, click Window > Show View > Error Log. Restart the Agent Controller. Restart the application.

## RPTA0009E

A Test could not be launched on Driver: %1 due to an internal error. Please see Problem Determination Log. For more information, see the Troubleshooting section of the online help.

**Explanation:** An exception was thrown during an attempt to obtain the operating system attribute of the location asset.

**User response:** Check the Error Log for further information on the error. To open the Error Log, click Window > Show View > Error Log. Open the location asset representing the agent computer in the Test Navigator, and verify that all information and properties are correct. Delete the location asset representing the agent computer in the Test Navigator, and create a new location asset. You might need to delete the location and create a new one, if the location asset representing the agent computer asset is corrupted.

---

## RPTA0010E

An error has been encountered while launching a Test on Driver: %1. Please see Problem Determination Log. For more information, see the Troubleshooting section of the online help.

**Explanation:** An exception was thrown while starting the test. The exception did not contain an error message.

**User response:** Check the Error Log for further information on the error. To open the Error Log, click Window > Show View > Error Log. Restart the Agent Controller. Restart the application.

---

## RPTA0011E

An error has been encountered while launching a Test on Driver: %1. An Executor was not returned and neither was an error message. This is an internal error, please contact support.

**Explanation:** The Test and Performance Tools Platform (TPTP) infrastructure produced neither an executor for this test nor error messages.

**User response:** Check the Error Log for further information on the error. To open the Error Log, click Window > Show View > Error Log. Restart the Agent Controller. Restart the application.

## RPTA**0012**E

An error has been encountered while launching a Test on Driver: %1. There are no Data Processors present. This is an internal error, please contact support.

## RPTA0013E

An error has been encountered while launching a Test on Driver: %1. Data Processors have not been configured correctly. This is an internal error, please contact support. For more information, see the Troubleshooting section of the online help.

**Explanation:** The test application was unable to configure the Data Processor for either the test log or the statistics portion of the test infrastructure.

**User response:** Check the Error Log for further information on the error. To open the Error Log, click Window > Show View > Error Log. Restart the Agent Controller. Restart the application.

---

## RPTA0014E

A
Test
could
not
be
launched
on
Driver:
%1.
The
Test
Execution
Framework
encountered
an
Exception.
This
is
an
internal
error,
please
contact
support.

## RPTA**0015**E

An error was encountered while launching a Test on %1. \nPlease examine your Deploy Directory: %2, the error could be caused by one of the following: \n \n1. The Deploy Directory path must be absolute (start with Drive Letter or "/"). \n2. The Deploy Directory

## RPTA**0016**E

An error has been encountered while launching the test. A required dataset %1 is missing or invalid in your project.

## RPTA**0017**E

An error has been encountered while launching the test. A required dataset %1 has been replaced. One or more test(s) are referencing a different version of the dataset.

## RPTA**0018**E

ready

## RPTA**0019**E

not ready on port

## RPTA**0020**E

Check
Agents
Failed

## RPTA**0021**E

%1
deployment
directory
%2
format
not
compatible
for
operating
system
%3.
For
more
information,
see
the
Troubleshooting
section
of
the
online
help.

**Explanation:** The deployment directory that is specified in the location asset representing the agent computer is incorrect for the operating system that is specified in the location asset.

**User response:** Open the location representing the agent computer in the Test Navigator, and edit the deployment directory or the operating system.

## RPTA**0022E**

Timed
out
after
%1
seconds
waiting
for
the
license
server.
Check
network
connectivity
to
the
license
server
and
ensure
the
license
server
is
running.
For
more
information,
see
the
Troubleshooting
section
of
the
online
help.

**Explanation:** The test application was unable to connect to the license server in the allotted time.

**System action:** The test run stops.

**User response:** Run the __RLKA_NAME__ to check for connectivity to the license server or to point to a different server.

## RPTA**0023**E

Virtual users have exited prior to stage completion. At the end of stage %1 there were %2 users running when %3 were expected. A common reason for this is a schedule which has assigned an insufficient amount of work (for one or more User Groups),

**Explanation:** During schedule execution, at the end of the current stage, the actual number of users running did not match the expected number of users. For example, if the current stage specifies that 100 users should run for 1 hour and only 90 users are running at the end of the hour, this message is displayed.

**User response:** Check the Error Log for further information on the error. To open the Error Log, click Window > Show View > Error Log. Typically, this message is displayed when virtual users did not have enough work to do for the duration of the stage. For schedules that contain more than one stage, verify that the workload under each user group is contained inside an infinite loop. Use infinite loops because the stage duration is controlled by the time when users stop. If virtual users have sufficient workload, look in the test log for more information about why virtual users stopped. The virtual users that stopped might have encountered errors. By default, this message is displayed when the number of expected users does not match the number of actual users running at the end of a stage. You can change this setting to specify the percentage of users that may stop during a stage without being considered an error. To change the error condition, create the -DrptStopTolerance property in the eclipse.ini file in the installation directory. For example, -DrptStopTolerance=80 specifies that 80% of the users may stop unexpectedly during stage execution without being considered an error.

---

## RPTA**0024**E

Exception encountered adding or removing users.

**Explanation:** This error message is displayed when a dataset reference between a test and a dataset is broken. Whenever a dataset is used, a reference is created in the test. The reference is a link that points to the physical dataset file in the test project. This link can break if the test is copied or imported into another project without copying or importing the associated dataset file. This link can also break if the dataset file is deleted.

**System action:** None.

**User response:** Do not copy or import individual test assets. Instead, copy or import entire projects. If you have already copied or imported individual test assets, copy the dataset from the previous project or create a new dataset that contains the same information. Open the test with the broken reference and link the dataset to the test. \nDo not delete dataset files.

## RPTA**0025**E

The
schedule
has
no
user
group.

**RPTA0026E**

The RPT_VMARGS option rptPre811PageResponseTimes is specified on at least one location and is missing from at least one other location. Please ensure that either all locations include this option or none do. See "adjusted page response time for increased accuracy" in the help for more information.

## RPTA**0025**I

Run
Completed
(%1)

## RPTA**0026**I

Run
Terminated
(%1)

## RPTA**0027**I

%1:
%2

## RPTA**0031**E

Location
template
file
%1
is
not
found
(referenced
from
location
file
%2)

**Explanation:** A location template file referenced by a location file is missing or inaccessible.

**User response:** Create a location template file with the given name. If the location template file exists but is in a closed project, open the project.

## RPTA**0032**I

Found location template [%1] for [%2] (instances found: %3)

## RPTA**0033**I

%1 remote location(s) associated with location template [%2]

## RPTA**0034**E

Cannot change stage duration if Until Finished specified

## RPTA0035E

Duration time specified is less than what has already elapsed

## RPTA0036E

Schedule must be in the Running state to change stage duration

## RPTA0037E

Agent %1 not ready, time of last contact: %2

**Explanation:** The specified agent is not in contact with the workbench. The schedule cannot run until all agents that are used in the schedule are actively connected.

**System action:** Install and configure the Rational® Performance Tester load generation agent on the agent computer.

**User response:** Ensure that the specified agent has a load generation agent installed and is properly configured to this workbench. Restart the schedule. See the online help for information about how to install and configure the load generation agent.

---

## RPTA**0038**E

No
successful
contact

---

## RPTA**0039**E

Unknown
host
'%1'

**Explanation:** The specified agent name is not resolving in the Domain Name System (DNS).

**User response:** Ensure that the agent name is spelled correctly in the location.

---

## RPTA**0040**E

Unable
to
complete
deployment
to
agents
because
of
an
unexpected
error
in
the
publish
phase.
%1

**Explanation:** A deployment error occurred that is likely a low-level I/O error or an unrecoverable internal error.

**User response:** Check the exception messages for possible causes such as a lack of hard-disk space.

## RPTA**0041**E

%1

**Explanation:** The specified agent is not in contact with the workbench. The schedule cannot run until all agents that are used in the schedule are actively connected.

**System action:** Install and configure the __PT_RR_SHORTNAME__ load generation agent on the agent computer.

**User response:** Ensure that the specified agent has a __PT_RR_SHORTNAME__ load generation agent installed and is properly configured to this workbench. Restart the schedule. See the online help for information about how to install and configure the load generation agent.

## RPTA**0042**E

Agent
version
%1
incompatible
on
host
%2.
Minimum
agent
version
%3
required.

**Explanation:** The version of the __PT_AGENT_ACRONYM__ is not compatible with a feature in the schedule.

**System action:** The schedule cannot be launched so schedule execution ends.

**User response:** Upgrade the __PT_AGENT_ACRONYM__ on the machine specified to match the workbench version.

## RPTA**0043**E

Error
encountered

**Explanation:** An unexpected error occurred.

**User response:** Look for more details about the error in the message posted.

## RPTA0100W

Failed
to
delete
file
%1

## RPTA0518E

An
error
has
been
encountered
while
launching
the
test.
A
required
dataset
%1
is
missing
or
invalid
in
your
project.

**Explanation:** A test contains a link to a dataset that cannot be found or that is corrupted. This can happen when a project is not imported completely, or when a file is deleted.

**System action:** The test run does not start.

**User response:** Open the test. On the Common Options page, fix the broken link so that it points to a valid dataset file or delete the link.

## RPTA1050E

Rational® Service Tester for SOA Quality is licensed to only support single user execution. Please adjust the number of Users to 1 in the schedule and rerun. Contact __VENDOR_NAME__ regarding the use of __PT_RR_SHORTNAME__ for your load testing needs.

## RPTC**0003**E

Wrong
type
of
project
'%1'.

## RPTC**0004**E

Unable
to
access
test
variable
initialization
file.
Make
sure
the
specified
file
path
is
accessible:
%1

## RPTC**0005**E

Error while processing XML file containing variable initializations. Make sure the file contains valid XML of the expected format: %1

## RPTC**0006**E

Error while gather test variable initializations. No variable initializations will be honored for this run.

## RPTC0007E

Error processing license request for feature '%1'. This feature will not be available.

**Explanation:** The workbench could not find the ibmrpt_pvu license. Either the license does not exist with the license server or activation kit or the workbench was unable to acquire it. As a result, the capabilities that this license enables are not be available.

**User response:** To enable the capabilities for the license, ensure that the ibmrpt_pvu license is available for the workbench.

## RPTC0008I

Setting Variable [name='%1', value='%2', source='%3', user group='%4', location='%5']

## RPTC**00020**E

Unexpected
I/O
error
while
communicating
with
workbench
%1

**Explanation:** During test-log transfer a network error occurred on the agent communicating to the workbench.

**System action:** The agent re-attempts to communicate with the workbench.

**User response:** If the problem persists, inspect error and take corrective action.

## RPTC**1001**W

The
file
path
specified
for
the
Zip
Utility
is
invalid.

## RPTC**1002**W

Could
not
get
the
classpath
for
project
'%1'.

## RPTC**1009**I

Undefined

## RPTC**1011**I

%1: Request delivered

## RPTC**1012**I

%1: successfully added %2 to the configuration file

## RPTC**1013**I

%1: successfully removed %2 from the configuration file

## RPTC**1014**I

%1: %2 is already in the configuration file

## RPTC**1015**l

%1: Request timed out

## RPTC**1016**l

%1: Agent not ready

## RPTC**1017**l

%1: Agent not known

## RPTC**1018**l

%1: Unknown host exception

## RPTC**1019**l

%1: %2

## RPTC**1020**l

License type: %1

**Explanation:** Lists the brand of licensing being used (either HCL or IBM).

**System action:** License checkouts will attempt to acquire a license of the corresponding type.

**User response:** No action required.

## RPTC**1021**I

License
valid:
%1

**Explanation:** Indicates whether a valid license was successfully acquired (true/false).

**System action:** If true, the functionality associated with the acquired license will be enabled.

**User response:** If false, check your license configuration.

## RPTC**1030**E

Unable
to
replace
dataset
'%1'
with
'%2':
%3.

**Explanation:** An error occurred attempting to replace datasets.

**System action:** Execution will complete with error.

**User response:** Refer to the error message for more details, change the command line options related to replacing datasets.

## RPTC**1031**E

The
dataset
'%1'
doesn't
exist.

**Explanation:** Unable to locate the specified dataset referenced in the dataset command line option.

**System action:** Command line execution will be cancelled.

**User response:** Change the command line options related to replacing datasets.

## RPTC1032E

The
dataset
'%1'
is
incompatible
with
existing
dataset
'%2'.

**Explanation:** The specified replacement dataset does not have compatible columns, type, etc.

**System action:** Execution will complete with error.

**User response:** Ensure the dataset has the same columns of the dataset it is replacing.

## RPTE**0005**W

Unable
to
attach
requirements
report
into
RQM
result,
because
the
default
requirements
report
has
been
deleted.
You
can
recreate
the
default
reports
by
click
restore
defaults
button
on
the
Default
Reports
preference
page.

**Explanation:** When a test run started by __QM_NAME__ completes, the default report is attached to the __QM_NAME__ execution results. This error occurs when the report selected as the default report on the Default Report preferences page does not exist.

**System action:** No report is attached to the __QM_NAME__ execution results.

**User response:** Click Window > Preferences > Test > Performance Test Reports > Default Report to open the Default Report preferences page. Check that the selected report exists. Click Restore Defaults to reset the default reports.

---

## RPTE0011W

Unexpected error while releasing system resources for test log export. This may cause an increased memory footprint, until Rational® Performance Tester is restarted.

**Explanation:** Test log export has completed (possibly with errors described earlier in the workspace log), but when releasing assets used during the export operation, there was an unexpected error.

**System action:** Memory allocated to this operation may not have been freed. Previous errors are likely to be present explaining the root cause.

**User response:** It is advisable to restart the application to free memory allocated during this operation. The exported test log file may be available but there may be errors.

## RPTE**0147**E

The password saved for an encrypted column in dataset "%1" was invalid. Set a new password in the Automation Security preference page.

**Explanation:** The value saved in the Automation Security preference page for the specified dataset was not correct. It will be ignored.

**System action:** The password in the preference is ignored. If running from the workbench, it will prompt for a password before execution. Otherwise, execution will fail.

**User response:** Update the password in the Test - Test Execution - Automation Security preference page.

## RPTH0130I

No sample time closely matches request at time= %1

## RPTH049E

A statistical adapter is missing reference to the target result.

**Explanation:** This is an internal error when loading results files. It could indicate that the result is corrupted, or it could only be a timing issue.

**System action:** The result cannot be opened.

**User response:** Close all reports and restart the workbench. If the result still does not open, kill any CPU-intensive processes running in the background.

## RPTI**0069**E

Local
on
premise
agent
%1
not
in
contact
with
this
workbench.

## RPTI**0070**E

See
Error
Log
for
more
details.

## RPTI**0071**I

There
was
an
error
while
updating
the
workspace
after
downloading
remote
files.

## RPTI0072E

Modify majordomo.config on %1 and configure it to poll this workbench.

## RPTI0072I

Remote Launch Status: %1

## RPTI0073E

Project is NULL

## RPTI0074E

Exception occurred while creating and unzipping project: %1

## RPTI0075E

Error running schedule. Could not find schedule %1 in project %2.

## RPTI0110I

Provision time (MM:SS): %1

## RPTI0111I

Launch time (MM:SS): %1

## RPTI0112I

Execution time (MM:SS): %1

## RPTI0113I

Results transfer time (MM:SS): %1

## RPTI0141E

```
--------------------------------
\nError
Dialog
\n
%1:
%2\nConsult
workspace
error
log
({workspace}/.metadata/
.log)
for
further
information.
\n--------------------------------
\n
```

**Explanation:** This message is displayed to the command-line output when an error occurs during execution. It displays details about the error and directs the user where to find additional information.

**System action:** None.

**User response:** This message occurs as a generic way to display errors during command-line execution. Consult the workspace log for further details including additional error messages.

## RPTI**0142**E

The
Usage
Metrics
version
%1
required
by
the
licensed
component
%2
is
not
available.

**Explanation:** The license that you are using requires Usage Metrics reporting for a later version of the product.

**System action:** The execution will not start.

**User response:** Update the product to a newer version, or obtain a license that is applicable to the current version of the product.

## RPTI**0143**E

The
licensing
system
failed
to
return
Usage
Metrics
enablement
for
component
%1.

**Explanation:** An error occurred while determining if the license requires Usage Metrics reporting.

**System action:** The execution will not start.

**User response:** Verify that the license is not meant for a newer version of the product. Otherwise, contact support.

## RPTI**0144**W

No RTCP instance is available to report Usage Metrics. No Usage Metrics will be reported for this execution.

**Explanation:** The license enables Usage Metrics reporting, but either the preference for the Usage Metrics server is not set, or it is set but the server is not active or reachable.

**System action:** The execution will be done normally, but the Usage Metrics will not be logged. This is allowed by the license you are using.

**User response:** If you have set up __QUALITY_SERVER__, go to Preferences > Test > __QUALITY_SERVER__, and fill in the server details for Usage Metrics reporting. Verify that the server can be reached from this machine by going to http://servername:7828 in a browser on the local machine.

## RPTI0145E

No RTCP instance is available to report Usage Metrics. Per license policy, execution cannot happen unless a RTCP is defined and running.

**Explanation:** The license requires Usage Metrics reporting, but either the preference for the Usage Metrics server is not set, or it is set but the server is not active or reachable.

**System action:** The execution will not start.

**User response:** Install __QUALITY_SERVER__ (if not done already), then go to Preferences > Test > __QUALITY_SERVER__, and fill in the server details for Usage Metrics reporting. Verify that the server can be reached from this machine by going to http://servername:7828 in a browser on the local machine.

## RPTI0146E

TPTP
Datapools
and
Datasets
cannot
coexist
in
the
same
test.
Test
run
aborted.

**Explanation:** A legacy datapool and a new dataset were both detected in the same test.

**System action:** Test execution will be aborted and will not be successful until the test contains only one of the two asset types (dataset or datapool).

**User response:** With the latest version of this product, convert the datapool to a dataset, then open the test containing the legacy datapool in the test editor and save it. Then, restart test execution.

## RPTJ0063E

An
IOException
was
encountered
while
creating
the
Annotation
File
on
Driver:
%1

## RPTJ**0075**E

An IOException was encountered while creating the Execution Log File on Driver: %1 :: %2

## RPTJ**1002**E

Driver %1 returned an unrecognized response: %2. The last command sent was: %3

## RPTJ**1003**E

While
waiting
for
an
acknowledgement
from
the
Driver,
an
unrecognized
response
was
received.

## RPTJ**1004**E

The
workbench
was
waiting
for
an
Acknowledgement
from
the
__VENDOR_NAME__
Agent
Controller
on
Driver
%1
and
none
was
received.

**Explanation:** A required response from an agent was not received.

**System action:** Execution ends because the required acknowledgement from the agent was not received.

**User response:** Monitor resource usage on the agent. Add additional agents if memory or CPU usage is high on a any agent.

---

## RPTJ**1005**E

Error
while
processing
a
message
from
the
__VENDOR_NAME__
Agent
Controller.

**Explanation:** An unexpected error occurred while handling a command from a load generating agent.

**System action:** Execution ends because of an unexpected error while communicating with an agent.

**User response:** Check the workbench Error Log for more information.

## RPTJ**1006**E

Execution failure. No status received from location %1 in %2 seconds. Workbench memory usage at %3 percent of the configured JVM heap. Possible location or workbench overload. For more information, see the Troubleshooting section of the online help.

**Explanation:** The workbench cannot communicate with the agent computer.

**User response:** Try running the schedule again, using default values for all parameters and running at reduced user load levels. It is possible one agent computer is overloaded. If you can run successfully with the default values, make changes to the schedule settings or user load incrementally to determine the cause of failure. Increase the statistics interval to 60 seconds and try running the schedule again. Check the error log for messages that might indicate the cause of the failure. Click Window > Show View > Error Log to open the error log.

## RPTJ**1007**E

The Driver: %1 has encountered a communication error. Please refer to Problem Determination Log for more details. For more information, see the Troubleshooting section of the online help.

**Explanation:** The agent computer that the message specifies encountered a problem when trying to run a command sent from the workbench.

**User response:** Check the error log for messages from the agent computer that the error message specifies. Click Window > Show View > Error Log. Check the test log for any failures from virtual users. This message might be displayed when you add or remove users manually or by means of schedule stages.

**RPTJ1008E**

The Driver: %1 has become unresponsive, possibly due to an out-of-memory condition. At last notification this Driver was using %2 percent of its allocated memory. Please refer to the "Increasing memory allocation" Help topic for information on how to increase memory allocation. For more

**Explanation:** The workbench cannot communicate with the agent computer. The agent computer might have a memory allocation problem.

**User response:** Try running the schedule again, using the default values for Test Log and Problem Determination log levels. Follow the instructions in Increasing memory allocation. Set the memory allocation to the size of physical memory minus 256 megabytes, up to a limit of 1500 megabytes. For example, on an agent computer with one gigabyte of physical memory, set the memory allocation to 756 megabytes.

---

## RPTJ**1009**E

The Driver: %1 is running %2, however the user selected %3 as the Drivers operating system.

---

## RPTJ**1010**E

Error while transferring file on Driver: %1. Transfer FROM: %3 TO: %2

## RPTJ1011E

The '%1' Protocol/ Feature is not supported on the %2 platform, so the Test %3 can't be executed on location %4. For more information, see the Troubleshooting section of the online help.

**Explanation:** The test includes a feature or protocol that is not supported on one of the agent computers where it is scheduled to run.

**User response:** Edit the schedule and associate the user groups that include the problem test with agent computers that support the feature or protocol.

## RPTJ**1012**E

The operating system (%1) for location %2 is not recognized. Please use an operating system that matches or begins with the name of one of the recognized platforms: %3

## RPTJ1013E

No valid license key for %1 Protocol/ Feature found. The Test %3 cannot be executed. For more information, see the Troubleshooting section of the online help.

**Explanation:** The test includes a feature or protocol that requires a license for the number of virtual users that are included in the run.

**System action:** The test run stops.

**User response:** Run the __BRAND_NAME__ License Key Administrator and check for available license keys for the feature or protocol and number of users that you want. To learn more about license keys, see the installation guide. \nAdd the required license key or point to a server that has the required license key.

## RPTJ**1014**E

Execution on the %1 Platform requires a license and no valid license key (%2) was found to enable it, so the Test %3 can't be executed on location %4. For more information, see the Troubleshooting section of the online help.

**Explanation:** One of the agent computers that is specified for the test run requires a license, but no license key was available for that platform.

**User response:** Run the Rational License Key Administrator and check for available license keys for the platform that you want. To learn more about license keys, see the installation guide. Add the required license key or point to a server that has the required license key or run the test on a different platform.

## RPTJ**1015**E

The specified operating system (%1) for location %2 is inconsistent with the actual platform (%3) running at that location. Please update the operating system to match and then try again. For more information, see the Troubleshooting section of the online help.

**Explanation:** The operating system that is specified in the agent computer asset does not match the operating system that is running on the computer at the specified address.

**User response:** 1. Open the schedule in the test editor. 2. Select the user group that runs on the location mentioned in the error message. 3. In the Schedule Element Details, click the Locations tab, and then select the location mentioned in the error message. 4. Click Edit. 5. Select the appropriate value from the Operating system list. 6. Click OK.

---

## RPTJ1016E

After deploying File: %2 to Driver: %1, %3 Byte(s) where found on the socket. Please refer to the Problem Determination Log for more details.

## RPTJ**1017**E

An IOException ocurred while deploying File: %2 to Driver: %1. Please refer to the Problem Determination Log for more details.

## RPTJ**1018**E

A SocketException ocurred while deploying File: %2 to Driver: %1. Please refer to the Problem Determination Log for more details.

## RPTJ**1019**E

An
UnsupportedEncodingException
ocurred
while
deploying
File:
%2
to
Driver:
%1
Please
refer
to
the
Problem
Determination
Log
for
more
details.

## RPTJ1020E

An IOException ocurred while deploying File: %2 to Driver: %1. \nA possible cause is that the __VENDOR_NAME__ Agent Controller was started by a non-root user. \nThe Agent Controller needs to be started by the root user.

**Explanation:** Deployment of test assets to an agent failed.

**System action:** Execution ends because required test assets could not be copied to an agent.

**User response:** Ensure that the Majordomo process is started by the root user.

---

## RPTJ1021E

An InactiveAgentException has occurred while deploying to Driver: %1. Please refer to the Problem Determination Log for more details. For more information, see the Troubleshooting section of the online help.

**Explanation:** The Test and Performance Tools Platform (TPTP) infrastructure threw an InactiveAgentException when the TPTP infrastructure attempted to communicate with the Agent Controller.

**User response:** Check the Error Log for further information on the error. To open the Error Log, click Window > Show View > Error Log. Restart the Agent Controller on the agent computer.

## RPTJ**0121**I

Send
RATEGENERATORS
to:
%1,
string
'%2'

## RPTJ**1022**E

The
workbench
received
notification
that
the
execution
process
on
Driver
%1
has
terminated.

**Explanation:** The process running on the agent computer ended unexpectedly.

**User response:** Ensure that there is at least one successful test run, possibly with fewer virtual users, so that the maximum memory value for the agent is set correctly. Check the javacore* file on the agent computer or the logs in the deployment directory for further information on the process failure.

## RPTJ**1023**E

Communication with Driver %1 has been lost, possibly due to an out-of-memory condition. At last notification this Driver was using %2 percent of its allocated memory. Please refer to the "Increasing memory allocation" Help topic for information on how to increase memory allocation.

For

## RPTJ**1024**E

Error
during
initialization
of
annotation
transfer
progress
listener.

## RPTJ**1025**I

Run
Completed
(%1)

## RPTJ**1026**I

Run
Terminated
(%1)

## RPTJ**1030**E

Non-
fatal
internal
exception
occurred
during
code
generation
optimization.
Code
generation
will
not
use
meta-
cache.

## RPTJ**1040**E

The license required for the '%1' Protocol/ Feature and %2 virtual users could not be checked out, so the Test %3 cannot be executed. For more information, see the Troubleshooting section of the online help.

**Explanation:** The test application was unable to check out a license for a feature or protocol in the test run.

**User response:** Run the Rational License Key Administrator and check for available license keys for the platform to run the feature or protocol. Add the required license key, point to a server that has the required license key, or run the test on a different platform.

## RPTJ**1041**E

The '%1' Protocol/ Feature is disabled due to a licensing configuration error.

## RPTJ**1042**E

%1 Failure checking out license for '%2' Protocol/ Feature and %3 virtual users. The Test %4 cannot be executed.

## RPTJ**1043**E

%1 The '%2' Protocol/ Feature is not supported on the %3 platform, so the Test %4 can't be executed on location %5.

**RPTJ1044E**

Timed out after %1 seconds while waiting for the license server. Ensure that network connectivity to the license server exists and that the license server is running. For more information, see the Troubleshooting section of the online help.

## RPTJ1100I

A hang has been avoided during execution history receipt with %1 by a forceful load test executor state change

## RPTJ1101E

A session on driver %1 did not release promptly. Please check the agent controller.

**RPTJ1102W**

The testLog event loader thread in the workbench has ended before processing all testLog events from %1. The testLog may be incomplete.

## RPTJ**1103**W

The
test
executor
for
%1
has
been
artificially
set
to
HISTORY_COMPLETE
because
the
testLog
event
loader
thread
is
not
longer
running.

**Explanation:** A monitoring process indicates that the test log loader stopped prematurely. This is not a definite indication of a problem.

**User response:** Check that the expected events exist at the end of the test log. If so, no further action is necessary.

## RPTJ**1104**E

Remote
debug
never
received
event
%1,
process
exit
value
%2

**Explanation:** Expected remote debug event was not received

**User response:** Check the Error Log for remote process failure reason

## RPTJ**1141**E

Temporary
dataset
file
%1
not
created.

**Explanation:** Temporary dataset file can't be created on the system.

**System action:** Original dataset is used.

**User response:** Check corresponding file properties on the system.

## RPTJ**1142**E

Temporary
dataset
data
are
not
generated:
%1

**Explanation:** Error reached during data generation.

**System action:** Original dataset is used.

**User response:** Check corresponding connection information.

## RPTJ**1200**W

Failed
to
delete
file
%1

## RPTJ**1220**E

An InactiveAgentException has occurred attempting to send [%1] to driver %2

## RPTJ**1221**E

The CommandHandler for %1 has encountered an exception while processing %2

## RPTJ**1240**E

Driver
%1
has
reported
a
NOK.
The
last
command
sent
to
that
driver
was:
%2

**Explanation:** A schedule command sent from the workbench to the agent computer could not be run by the agent computer.

**User response:** Run the schedule using the default settings. Look for unusual assignments of numbers of virtual users to agent computers at stage transitions.

## RPTJ**1241**E

Driver %1 has reported a NOK with the message: %2. The last command sent to that driver was: %3

## RPTJ**1242**E

Driver %1 has reported a %2 status

## RPTJ**1244**E

The AgentCommandListener for %1 has encountered an exception while processing %2

## RPTJ**1245**E

Driver %1 has reported that it is no longer receiving messages from the workbench. The previous message received from this driver, %2 milliseconds ago, was %3. At present no commands have been sent to this driver.

**Explanation:** Schedule commands sent from the workbench to the agent computer were not received by the agent computer.

**User response:** Ensure that there is at least one successful test run, possibly with fewer virtual users, so that the maximum memory value for the agent is set correctly. Use more agent computers to run the schedule.

---

## RPTJ**1261**E

The
ResponseHandler
for
%1
has
encountered
an
exception
while
processing
%2

---

## RPTJ**1270**E

Failure
attempting
to
launch
test
execution.

---

## RPTJ**1271**E

The
process
executing
the
test
has
ended
unexpectedly.

**Explanation:** The process that runs tests could not start, or it stopped before the test run ended.

**System action:** The test run stops.

**User response:** Check the core files or the logs for further information on the process failure. If you are using Java Virtual Machine (JVM) arguments, check the argument syntax and try running tests without the arguments. Run the test inside a schedule.

## RPTJ**1280**E

The communication path for returning test results from %1 has not been established. Check network connectivity between that machine and the workbench including any firewalls.

## RPTJ**1400**I

%1% %2/%3 files %4/%5 bytes deployed

## RPTK**0000**I

%1

## RPTK1001E

__PT_ACRONYM__ has detected the presence of an invalid Virtual Tester license key. If you have recently upgraded __PT_ACRONYM__, note that this is a new check performed by release 7.0.1 or later, and instructions for replacing invalid Virtual Tester license keys should have already been sent to

**Explanation:** Invalid Virtual Tester license key(s).

**System action:** System will not execute schedule run(s) that require a Virtual Tester license if one is not available.

**User response:** You must replace all invalid Virtual Tester license keys. If you need further assistance, please contact your sales representative or Technical Support.

---

## RPTK1016E

The
specified
license
server's
version
level
is
not
compatible
with
this
version
of
__PT_ACRONYM__.

**Explanation:** The specified license server's version level is not compatible with this version.

**System action:** Incompatible version.

**User response:** Check the license server's version.

---

## RPTK1019E

Unable
to
verify
system
time.

**Explanation:** The system time has been tampered with since the last successful license check.

**System action:** Future license checks will automatically fail.

**User response:** Contact Technical Support.

## RPTK**1020**E

Unable
to
locate
license
directory.

**Explanation:** Unable to locate license directory.

**System action:** Stop execution.

**User response:** Please verify that the license directory exists.

## RPTK**1021**E

License
has
expired.

**Explanation:** An expired license was found.

**System action:** Request a license key from user.

**User response:** Enter a new license key.

## RPTK**1022**E

Invalid
license
file.

**Explanation:** A valid license was not found.

**System action:** Request a license key from user.

**User response:** Enter a valid license key.

## RPTK1023E

Unable
to
find
a
license
supporting
%1
virtual
users.

**Explanation:** The currently installed license key(s) do not support enough VUs for this operation.

**System action:** Request a license key from user.

**User response:** Enter another license key to enable more VUs.

## RPTL0001W

Unable
to
retrieve
data
from
the
test.

## RPTL0002W

Failed
to
store
test
data
into
annotations.

## RPTL**0003**W

Failed
to
attach
the
annotation
to
the
test.

## RPTL**0004**W

Unable
to
open
test
annotation
to
read
data.

**Explanation:** The test appears to be corrupted.

**System action:** Attempts to open the test fail.

**User response:** Make sure your disk has enough space. If it does, try recreating the test from the recording.

## RPTL**0005**W

Failed
to
create
a
temporary
file
to
save
test
data.

## RPTL**0009**I

Test %1 is of an older version %2.

## RPTL**0010**E

Error creating metadata cache.

## RPTL**0011**E

Error reading metadata cache for %1.

## RPTR**0000**W

%1

## RPTR**0001**W

Failed to add annotation to execution history for file %1

## RPTR**0002**W

Unexpected
error
in
data
validity
check
of
LoadTimeEObjectConsumer

## RPTR**0003**W

Failed
to
add
properties
to
parent
id
%1

## RPTR**0004**W

Failed
to
delete
temp
file
%1

## RPTR**2001**E

Unexpected
exception
in
container
complete
loader.
Heap
growth
likely.

## RPTR2003W

Execution

Variables

-

Output

## RPTS1000E

Unable
to
start
the
agent
communication
service
because
of
an
error:
%1.
RPT
will
not
be
able
to
execute
schedules.

**Explanation:** The agent communication service could not start. This service is a lightweight web server that agents use to communicate with the workbench and to serve web reports. Typically, this error occurs when a server process on the workbench computer is listening on the same port that Rational® Performance Tester requires. This error can also occur when two instances of Rational® Performance Tester run on the same workbench.

**User response:** If multiple instances of the Rational® Performance Tester workbench are running on the same computer, close all but one instance. These instances include Rational® Performance Tester workbenches that are running on multiple user desktop systems. If the error message RPTS1002E_PORTS_CONSUMED is also displayed in the error log, see the message for that error. After the error is resolved, restart Rational® Performance Tester.

## RPTS1002E

RPT is unable to execute a schedule because one of the ports(%1) it uses to communicate with the agents has been taken by another RPT(or other server) process. Ensure only one RPT instance is running.

**Explanation:** Typically, this error occurs when a server process on the workbench computer is listening on the same port that Rational® Performance Tester requires. This error can also occur when two instances of Rational® Performance Tester run on the same workbench.

**User response:** Identify and stop the other process or service on the workbench that is using the ports that Rational® Performance Tester requires. Restart Rational® Performance Tester. You can also change the ports that Rational® Performance Tester uses by configuring the workbench and all agent computers. To change the ports, click Window > Preferences > Test > Server, and click Preferences > Test > Performance Test Reports > Web Reports.

## RPTS**1510**E

Unable
to
stop
the
agent
communication
service
because
of
an
error:
%1

## RPTS**1001**I

Agent
communication
service
listening
on
ports(%1)

**Explanation:** The agent communication service requires these local server ports to communicate with agents.

**User response:** This message is for informational purposes only.

## RPTX**0001**E

The combination of transformer and feature you have selected is invalid. Transformer (%1) was not expecting data type (%2).

## RPTX0002E

The combination of feature and transformer you have selected is invalid. Feature (%1) was not expecting data type (%2) to be returned by transformer (%3).

## RPTX0003E

Transformer (%1) has experienced a fatal error. Additional information (%2).

## RPTX0004E

Feature (%1) has experienced a fatal error. Additional information (%2).

## RPTX0005E

No class can be found for the specified transformer id (%1). Please check to make sure you have installed this transformer.

## RPTX**0006**E

Class definition missing. Please add jar that contains definition of (%1) to the classpath of the test project.

**Explanation:** Some requests or responses contain data that is encoded for Google Web Toolkit (GWT). To decode the data, Rational® Performance Tester requires access to the class definition.

**User response:** Add the JAR file that contains the class definitions to the classpath of the test project.

## RPTX**0007**E

The transformation raised a GWT serialization exception: %1

**Explanation:** The Google Web Toolkit (GWT) transformation could not be applied because of the indicated reason.

**User response:** Verify that the test elements containing the GWT encoded or decoded data are correct

## RPTX**0008**E

The
Silverlight
decoder
raised
an
exception:
%1

**Explanation:** The Microsoft Silverlight decoder did not work because of the indicated reason.

**User response:** Verify that the test elements containing the Silverlight encoded data are correct

## RPTX**0009**E

The
Silverlight
encoder
raised
an
exception:
%1

**Explanation:** The Microsoft Silverlight encoder did not work because of the indicated reason.

**User response:** Verify that the elements containing the Silverlight decoded data are correct

## RPTX**0010**E

The
GraniteDS
transformer
made
an
error
when
encoding
or
decoding:
%1

**Explanation:** The GraniteDS encoder did not work because of the indicated reason.

**User response:** Verify that the elements containing the GraniteDS encoded or decoded data are correct.

## RPXD0022W

The
time
to
extract
references
seems
excessive.
It
was
%1
milliseconds.

**Explanation:** It is taking a long time to extract data from your response for your references.

**System action:** None.

**User response:** Examine each of the regular expressions for your references. Make sure they don't have .* with no qualifiers or other poorly formed constructs. When you write the regular expression in the test you can click verify to get an idea of how long it is taking to execute.

## RPXE0061I

Loop
iteration
started
late
by
%1
millseconds

**Explanation:** A scheduled loop iteration started execution later than expected given the specified rate.

**System action:** Execution continues along with attempt to catch up in order to maintain desired rate.

**User response:** Add additional users or agents to increase capacity in order to maintain desired rate.

## RPTX1010I

Start of RPT project resolve. Repository=<%1>, Bootstrap=<%2>

## RPTX1011I

Attempting to resolve asset=<%1>

## RPTX1012I

End of RPT project resolve. No detected errors

## RPTX1017I

Downloaded asset %1 from remote repository, local asset created.

## RPTX1018I

Using
local
cached
version
of
asset
%1.

## RPTX1019I

RPT
testsuite=<
%1>
found
the
following
dependencies=<
%2>

## RPTX1081E

Exception
occurred
while
uploading
Mobile
report.

**Explanation:** A low-level exception occurred uploading the mobile report. It is unexpected.

**System action:** The RQM report will fail to upload.

**User response:** If possible take corrective action, otherwise contact support.

## RPTX**1082**E

An
error
occurred
when
generating
the
HTML/
zip
report.

**Explanation:** The HTML generator for the Execution Report has failed.

**System action:** No execution report uploaded into RQM results

**User response:** Ensure that the temporary directory is accessible on your file system.

## RPTX**2001**E

Adapter
unable
to
start
test
because
__PT_ACRONYM__
is
already
executing
a
test.

**Explanation:** The adapter received a request to start a test while another test on the adapter is in-progress.

**System action:** The adapter ignores the request to launch another test.

**User response:** Wait for the test which is currently executing on the adapter to complete, then re-initiate the launch.

**RPTX2002E**

Error encountered parsing RQM adapter preferences: %1. Please enter proper credentials in the Eclipse Quality Adapter preference page (Windows->Preferences).

**RPTX2003E**

Project <%1> could not be found during RQM import.

## RPTX**2004**E

Test
log
is
unavailable,
no
test
results
returned
to
RQM.

## RPTX**2005**E

Statistics
log
is
unavailable,
no
statistic
results
returned
to
RQM:
%1

## RPTX**2006**W

Display
unavailable,
no
__PT_ACRONYM__
HTML
reports
will
be
attached
to
RQM
execution
results.

**Explanation:** The adapter requires access to a virtual display to generate HTML reports. The adapter was unable to successfully create a display so HTML reports may be unavailable.

**System action:** HTML reports are not generated at the end of execution.

**User response:** If HTML reports are required, start the adapter with display access. Refer to documentation on how to start the adapter with a display.

---

## RPTX**2007**I

Start
RQM
Execution
Request
Project=
%1
Name=
%2

---

## RPTX**2008**I

Start
RQM
Import
Request
Project=
%1

---

## RPTX**2009**I

End
RQM
Execution
Request

---

## RPTX**2010**I

End
RQM
Import
Request

## RPTX2011E

Unable to interpret RQM configuration file %1. If file was hand edited make sure parameters are the correct format. If you are unable to get this file into the correct format, please erase and re-configure.

## RPTX**2012**E

Invalid
RQM
connection
parameter:
%1.
Adapter
was
not
launched.

## RPTX**2013**E

Adapter
was
stopped
while
a
test
was
executing.
The
results
of
this
test
may
be
unreliable.

## RPTX**2014**E

Adapter was stopped while preparing to run an RQM script. There are no results for the attempted test script run.

## RPTX**2015**E

Testsuite '%1' or project '%2' does not exist. Ensure workspace started by adapter contains project and testsuite.

## RPTX2016I

%1

## RPTX2017E

%1
Reason:
%2

## RPTX2018W

%1

## RPTX2019I

The
RQM
Adapter
has
been
disconnected.

## RPTX2020I

The
RQM
Adapter
has
stopped.

## RPTX2021E

Unexpected
error
occurred
while
executing
RQM
test
script.

## RPTX2022E

Unexpected error occurred while processing an import request from RQM.

## RPTX2023W

Error occurred while update the run status back to the RQM server. This may cause the RQM test progress page to contain inaccurate data.

## RPTX**2024**E

Unable to attach the following file to the RQM results. This may cause the attached HTML report not to render correctly. File name: %1

## RPTX**2025**E

Error occurred while registering the adapter: %1.

## RPTX2026E

Error occurred setting the default adapter name. Please set the name in the Eclipse Quality Adapter preference page (Windows->Preferences).

## RPTX2027W

Multiple test runs were detected when the stop request was received from RQM.

## RPTX2029W

Was unable to perform stop request from RQM. Likely the run was already shutting down when the request came in.

## RPTX2030I

Request to stop the test is being delayed until the appropriate run state is reached.

## RPTX2031I

A request to stop the currently running test has been received by RQM.

## RPTX2032I

Successfully issue a stop command to the running test. Please wait for the test to end.

## RPTX**2033**E

Error
attempting
to
stop
a
test.

## RPTX**2034**E

Unable
to
create
directory
%1
no
further
information.
Ensure
user
has
permission
to
create
directory
in
that
location.

## RPTX**2035**E

Error occurred while attempting to automatically update pre-8.0 asset %1 for RQM execution.

## RPTX**2036**E

RQM remote resource access is not supported for pre-8.0 SOA assets. Please update your entire SOA project to 8.0 or greater before sharing.

## RPTX**2037**E

Launch
was
aborted:
%1

## RPTX**2050**E

Unable
to
download
remote
asset
%1
into
local
workspace.
Remote
repository
%2.
Ensure
RQM
system
has
connectivity
to
the
remote
repository
and
the
file
exists.

## RPTX2051E

Unable
to
browse
%1
in
remote
repository
%2.
Ensure
RQM
system
has
connectivity
to
the
remote
repository
and
the
directory
exists.

## RPTX2055E

Error
occurred
reading
the
adapter
connection
file.

RPTX**2056**E

Error
occurred
saving
the
adapter
connection
file.

RPTX**2057**E

Unable
to
complete
import
operation
because
the
specified
path
is
not
in
the
adapters
workspace.
Try
specifying
the
only
the
project
name.

## RPTX2058E

The specified script <%1> is not in the workspace currently being used by the adapter. You can only execute scripts which are in the adapters workspace.

## RPTX₂₀₆₀E

The script path specified by RQM does not seem to be valid. Please ensure the RQM test script has a script path which contains the project and script name.

## RPTX2061W

Run verdict is inconclusive because no performance requirements exist in the last user stage for the associated VU Schedule.

## RPTX2062W

Run verdict is inconclusive because there are zero performance requirements in the last user stage of the associated VU Schedule.

## RPTX2063W

No time range was generated for the user stage of the associated VU Schedule. Performance requirements reported to RQM will be based on the default time range.

**RPTX2070E**

Error occurred while setting the RQM project area. Make sure a valid project area is specified on the Quality Manager Adapter preference page. The adapter is attempting to connect to RQM using the default project area.

## RPTX**2071**E

Error occurred while retrieving list of project areas. Please verify Quality Manager connection information. See error log for more details.

## RPTX**2072**E

Error occurred calling for the web analytics dash-board link.

**Explanation:**  Rational® Performance Tester could not open the external URL for the dashboard that references Rational Quality Manager records. This error occurs when a problem exists with the classpath for the result analysis.

**User response:** Ensure that Rational Quality Manager is version 4.0 or later and  Rational® Performance Tester is version 8.3 or later. If the error log contains startup errors, resolve the errors and check whether the problem is resolved.

## RPTX2073E

Error
occurred
while
translating
RQM
server
execution
variables
to
__PT_ACRONYM__.

**Explanation:** An unexpected error occurred while setting up execution variables.

**System action:** Execution variables are unavailable during execution.

**User response:** Contact support if this error persists.

## RPTX2074E

Error connecting RTW adapter and successful connecting RPT adapter. This suggest RQM does not support RTW script type introduced in 4.0.3. If RPT adapter is not required it can get disabled by adding -DrtwStartAdapter=false in eclipse.ini.

**Explanation:** Error connecting RTW adapter and successful connecting RPT adapter. This suggest RQM does not support RTW script type introduced in 4.0.3. If RTW adapter is not required it can get disabled by adding -DrtwStartAdapter=false in eclipse.ini.

**User response:** Use a RQM system supporting RTW script type. Add -DrtwStartAdapter=false in eclipse.ini to disable RTW script type. To import and execute RTW assets under the RPT adapter script type also add -DrptAvoidRQMImportFiltering in eclipse.ini

---

### RPTX2075E

Unable to interpret expression <%1> from RQM control file <%2>. Ignoring. Reason <%3>

**Explanation:** The RQM control file is of an invalid format.

**System action:** The control file instruction will be ignored.

**User response:** Change the file so it follows the specified format supplied by support.

## RPTX2077E

Unable to browse to <%1>. Make sure it exist on the shared location.

**Explanation:** The project referenced does not exist on the shared location.

**System action:** The RQM execution will stop.

**User response:** Ensure all required projects exist on the shared location.

## RPTX2079W

Errors attempting to load available SmartCard aliases. See documentation on how to configure your system to use SmartCard to authenticate to Quality Manager.

**Explanation:** An error occurred while attempting to load SmartCard aliases.

**System action:** User is unable to configure SmartCard using the preference UI.

**User response:** Ensure com.ibm.security.capi.IBMCAC is listed as provider 1 in file *{install}*\\SDP\\jdk\\jre\\lib\
\security\\java.security. See documentation on manual steps required to configure adapter for SmartCard usage.
Contact support if issues persist.

## RPWF0011E

Error occurred while completing test generation

## RPWF**0012**E

Error
occurred
while
processing
a
packet
at
test
generation

## RPWF**0021**E

WSDL
Exception
raised
while
processing
WSDL
source

## RPWF**0032**E

Error
while
generating
test
from
Axis
recording

## RPWF**0051**E

Error
occurred
while
setting
classpath
entry
for
recorder

## RPWF**0052**E

I/O exception occurred while resolving keystore or truststore path

## RPWF**0056**E

Error occurred while launching web services HTTP proxy

## RPWF**0066**E

Error occurred while launching axis client recorder agent

## RPWF**0071**E

Exception thrown while creating a wizard page control

## RPWF**0072**E

Exception thrown while parsing URL: %1

## RPWF**0074**E

Exception thrown while finishing the axis recording wizard

## RPWF**0075**E

Exception thrown while looking for an available port

## RPWF**0076**W

Exception
thrown
while
adding
SOA
Tester
certificate
to
the
trustore
%1

## RPWF**0081**W

A
proxy
authorization
%1
is
used
without
any
proxy

## RPWF**0082**W

No
free
name
can
be
found;
reusing
%1

## RPWF**0083**E

Resource
file
%1
not
found
in
workspace
%2

## RPWF**0084**E

Workspace
location
cannot
be
determined

## RPWF**0085**E

Cannot
retrieve
the
operation
name
from
the
envelope
%1

## RPWF**0101**E

Core
exception
thrown
using
org.eclipse.debug.core
plugin

## RPWF**0102**E

Exception thrown during launch configuration update

## RPWF**0103**E

Exception thrown while resolving a bundle entry path

## RPWF**0104**E

Exception thrown while identifying localhost IP address

## RPWF**0111**E

Exception thrown while creating a substitution: %1

## RPWF0112E

Exception thrown while creating a reference: %1

## RPWF0121W

Unknown format. Skipping the test generation for: %1

## RPWF0122W

Skipped call: %1

## RPWF0123W

Skipped request: It could be that provided password was not ok

## RPWF**0124**W

Attachments
not
generated.

## RPWF**0130**W

Could
not
find
project
for
URI:
%1

## RPWF**0131**W

Loading
XSD
Schema
failed:
%1

## RPWF**0132**E

Error
while
generating
test
from
Generic
Service
Client:
Can't
show
wizard

## RPWF0140E

An error has occurred: %1

## RPWH0007W

Unhandled Security Algorithm '%1'

## RPWH0009W

Unable to serialize data

## RPWH0010W

Unable to deserialize data

## RPWH0012E

Unable to open editor for '%1'

## RPWH0014E

Parse Error in '%1'

## RPWH**0015**E

Unable
to
create
resource
'%1'

## RPWH**0016**E

Failed
to
export
source
text
'%1'

## RPWH**0017**E

A
connection
error
occurred
on
'%1',
please
check
the
URL
or
the
network
configuration

**Explanation:** A connection error occurred.

**System action:** URL can not be reached, action is aborted.

**User response:** Check the URL or the network configuration.

## RPWS**0001**E

Exception
raised
during
data
harvest
execution

**Explanation:** Reference can't be performed.

**System action:** Reference is not performed: get empty data.

**User response:** Check the corresponding reference.

## RPWS**0002**E

Exception
raised
during
data
substitution
execution

**Explanation:** Substitution can't be performed.

**System action:** Substitution is not performed: write recorded data.

**User response:** Check the corresponding substitution.

## RPWS**0003**E

Exception
raised
on
harvest
data
management

**Explanation:** Reference can't be performed.

**System action:** Reference is not performed: get empty data.

**User response:** Check the corresponding reference.

## RPWS**0004**E

Exception
raised
on
substitution
data
management

**Explanation:** Substitution can't be performed.

**System action:** Substitution is not performed: write recorded data.

**User response:** Check the corresponding substitution.

## RPWS**0005**E

Exception
raised
during
WebSocket
read
action

**Explanation:** Read action can't be performed.

**System action:** No data are receive.

**User response:** Check the application side, may be the server closes the connection.

## RPWS**0006**E

Exception
raised
during
WebSocket
write
action

**Explanation:** Write action can't be performed.

**System action:** No data are sent.

**User response:** Check the application side, may be the server closes the connection.

## RPWS0007E

Unable
to
get
WebSocket
connection

**Explanation:** WebSocket connection information is wrong.

**System action:** No data will be receive or sent on this connection.

**User response:** Check the WebSocket connection, may be the test is corrupted.

## RPWS0008E

Unable
to
read
from
a
closed
connection

**Explanation:** WebSocket connection is closed.

**System action:** No data will be sent on this connection.

**User response:** Check why the WebSocket server closed connection.

## RPWY0002E

An
exception
occurred
in
%1

**Explanation:** An exception was detected.

**System action:** Current action is aborted.

**User response:** Check the cause of the exception.

## RPWY**0003**I

Information:

%1

(%2)

## RPWY**0004**W

Warning:

%1

(%2)

## RPWY**0005**E

An
error
occurred
while
importing
external
schema
%1

## RPWY**0006**E

Unable
to
correlate
automatically

## RPWY**0007**E

An
exception
%1
occurred
in
%2

**Explanation:** An exception was detected.

**System action:** Current action is aborted.

**User response:** Check the cause of the exception.

## RPWZI0002E

Exception raised during WebSocket connection creation.

**Explanation:** The workbench could not create a connection for WebSocket elements in split test.

**System action:** No connection are created by the workbench.

**User response:** The user need to create manually the connection, or to get the upgraded HTTP request in the split selection.

## RPXD0001E

Unknown Segment Offset/ Length for Segmented Dataset: %1

## RPXD0002E

Bad Dataset Mode: %1

## RPXD0003E

Dataset not initialized: %1

## RPXD**0004**E

End
of
non-
wrapped
dataset
reached:
%1

## RPXD**0005**E

Dataset
with
multiple
Equivalence
Classes
cannot
be
segmented

## RPXD**0006**E

segmented
DatapoolMap
null:
%1

## RPXD**0007**F

No
registered
data
correlation
handler
for
this
IKAction

## RPXD0017W

Pattern matching failed for: regex (%1) str (%2)

## RPXD0018E

Skipping substitution, reference value was null. original string: (%1) offset: (%2)

## RPXD0019E

Data Correlation: Failed Substitution \nReference[%1]\nSubstitution[%2]\n \nDetails: \n \nA failed reference occurred in a prior request. Since the reference named [%3] was null, we were unable to substitute a new value for the substituter named [%4], original string [%5] at offset [%6] and this request may

**Explanation:** A reference for an expected data substitution is null.

**System action:** None.

**User response:** To find the failed reference, open the test and go to the substitution site. Right-click the substitution site and select Go To > Reference. When troubleshooting failed references, start with the first error message. The first failed reference can cause subsequent failed references. Search the test log "for unable to extract" to find the first error message.\n\nExamine the request that generated the response. The request contains a value that might need to be correlated. For example, the request might contain a username that must be unique to play back the test successfully. In that case, use a dataset to provide a list of unique username values. You might need to manually correlate a value by using the Test Data Sources view. Values that typically are correlated include timestamps, dates, ids, and other alphanumeric strings.\nIf you no longer need the data correlation mentioned in the message, remove that data correlation from the test.

## RPXD**0020**E

Data Correlation: Failed Extraction \nReference[%1]\n \nDetails: \n \nWe were unable to extract the value for the reference named [%2], with the regular expression [%3]. This could mean a later request will fail. Please compare the response in the test log to the corresponding response in the

**Explanation:** The response received during playback is different from the response received when the test was recorded. The data correlation code was unable to use the regular expression expected value.

**System action:** None.

**User response:** Examine the request that generated the response. The request contains a value that might need to be correlated. For example, the request might contain a username that must be unique to play back the test successfully. In that case, use a dataset to provide a list of unique username values. You might need to manually correlate a value by using the Test Data Sources view. Values that typically are correlated include timestamps, dates, ids, and other alphanumeric strings. \nIf you no longer need the data correlation mentioned in the message, remove that data correlation from the test.

---

## RPXD0021E

Dataset %1 is accessed using different dataset modes by different tests.

---

## RPXD0021W

Setting variable %1 to value %2.

---

## RPXE0001W

%1

## RPXE0010W

Engine
shutdown
problem
joining
workers

## RPXE0011W

Failed
to
report
exception

## RPXE0012W

Schedule
failed
to
load

## RPXE0013W

Unable
to
create
test

## RPXE0014W

Setting
log
level
to
%1

## RPXE0015W

Attempt
to
add
object
to
Schedule
which
is
not
a
UserGroup

## RPXE0016W

Virtual
User
%1
experienced
error
%2

## RPXE0017W

Connect
timeout
for
action
%1
(%2)
user
%3

## RPXE**0018**W

Read
timeout
for
action
%1
(%2)
user
%3

## RPXE**0019**W

Connect
exception
for
action
%1
(%2)
user
%3

## RPXE**0021**W

Read
exception
for
action
%1
(%2)
user
%3

## RPXE**0023**W

Iterating
over
keys
exception

## RPXE**0024**W

CancelledKeyException

## RPXE0025W

NullPointerException

## RPXE0027W

UserGroup
exception

## RPXE0028W

User
Group
%1
does
not
implement
createTesterWorkload()

## RPXE0029W

Worker
caught
throwable

## RPXE0030W

Connection
leak,
I/O
state
%1

## RPXE0031W

Exception
finishing
connection
for
action
%1
(%2)
user
%3

## RPXE**0033**W

Finish
read
get
buffer
interrupted
for
action
%1
(%2)
user
%3

## RPXE**0035**W

Finish
read
exception
for
action
%1
(%2)
user
%3

## RPXE**0036**W

Engine
thread
startup
exception

## RPXE**0037**W

Engine
request
to
report
exception

## RPXE0038W

Exception creating cache file, cacheFileName: %1, extension: %2, dir: %3

## RPXE0039W

User %1 experienced exception %2

## RPXE0040W

User %1 caught exception trying to report severe error.

## RPXE0041W

Engine hard stop after %1 second timeout

## RPXE**0042**I

%1 received request to stop

## RPXE**0043**I

Forced stop of action %1

## RPXE**0044**W

No IP address was found for the local host

## RPXE**0045**W

Ignoring invalid network interface %1

## RPXE**0046**W

Could
not
find
any
usable
network
interfaces

## RPXE**0047**E

SyncPointSubsystem
Unknown
sync
point:
%1

## RPXE**0048**W

%1
STOPUSERS
users=
%2
stagger=
%3
timelimit=
%4
active
users=
%5

## RPXE0049W

%1 had %2 non-sampled users asked to stop active users= %3

## RPXE0050W

%1 had %2 sampled users asked to stop

## RPXE0051W

%1 after wait for compliance active users= %2 target= %3

## RPXE0052W

%1
abandon
user
%2

## RPXE0053W

%1
abandoned
%2
users

## RPXE0054W

%1
end
stop
%2
users
SUCCESS
active
users=
%3

## RPXE0055W

%1
end
stop
%2
users
FAIL
active
users=
%3

## RPXE**0056**W

%1 occurred in %2. Message: %3

## RPXE**0057**E

Exception while reading test variable initialization file: %1

## RPXE**0058**E

Exception while initializing virtual users test variables.

## RPXE**0059**E

Unable to get Kerberos ticket from KDC for server %1.

## RPXE0060E

Failed
to
load
test
from
'%1'
due
to
exception:
%2

**Explanation:** While trying to find and load class files required to execute the test a problem was encountered.

**User response:** See exception description for failure reason.

## RPXE0100W

%1
terminated
due
to
exception:
%2

## RPXE0102W

IKAction:
%1
(%2)
caught
Exception
in
preFinish()
for
%3
(%4)

## RPXE0103W

IKAction:
%1
(%2)
caught
Exception
in
postFinish()
for
%3
(%4)

## RPXE0104W

KernelChannel
connect(),
exception
while
trying
to
bind
to
local
address
%1:
%2

## RPXE2501E

An error occurred while attempting to handshake with the server using protocol %1 and cipher suite %2. This type of failure is often related to a mismatch between the requested protocol or cipher suite and the ones the server is expecting or may be related

**Explanation:** An SSL connection between a client and server is set up by a handshake, the goals of which are: To satisfy the client that it is talking to the right server (and optionally visa versa). Also, for the parties to have agreed on a cipher suite, which includes which encryption algorithm they will use to exchange data. These goals were not achieved.

**System action:** Execution ends because a secure connection cannot be established with the server.

**User response:** If the server requires a client digital certificate work with the server administrator to obtain one. If the server requires strong ciphers work with customer support to obtain the required and restricted ciphers.

---

### RPXE**2550**E

The
digital
certificate
RCS
file
'%1'
was
not
found
or
was
corrupt:
%2

---

### RPXE**2552**I

digital
certificate
alias

---

### RPXE**2900**E

The
server
rejected
the
client's
digital
certificate.

## RPXE2901W

The server closed the connection abruptly. This is probably due to an overloaded server or to a problem negotiating a digital certificate or cipher suite. Check the web server's SSL error log for more details.

## RPXE**4000**W

Schedule
or
Test
not
found.
May
not
have
compiled.
-
%1

## RPXE**4001**E

Runner
Exception
occurred

## RPXE**4002**E

Communications
Error:
Invalid
Logging
Level

## RPXE**4003**E

Communications
Error:
Invalid
TestLog
Level
for
%1
events

## RPXE**4004**E

Communications
Error:
Invalid
Statistics
Level
or
Interval

## RPXE**4005**E

Runner
Exception
occurred
-
See
problem
determination
log

## RPXE**4006**E

Communications
Error:
Invalid
Dataset
information

## RPXE**4007**E

Communications
Error:
No
communication
from
the
workbench
in
%1
milliseconds.
For
more
information,
see
the
Troubleshooting
section
of
the
online
help.

## RPXE**4008**E

Attempt
to
change
statistic
interval
ignored.

## RPXE**4008**l

Think: requested time %1 milliseconds, actual time %2 milliseconds

## RPXE**4009**l

Delay: requested time %1 milliseconds, actual time %2 milliseconds

## RPXE**4010**I

Schedule completed. See Performance Report, Verification Points Report, and/ or Percentile Report to further evaluate the results of this run according to your success criteria.

## RPXE**4011**E

Communications Error: Invalid Stop timeout

## RPXE**4013**I

Additional events from %1

## RPXE**4014**E

Communications
Error:
Invalid
RunStagger
information
for
%1
(pairCount)

## RPXE**4015**E

Communications
Error:
Invalid
RunStagger
information
for
%1
(pair
%2)

## RPXE**4016**E

Failed
to
start
users
for
user
group:
%1.

RPXE**4017**l

Additional execution history events from %1 are available, but they have been stored separately upon user request. See file %2. Refer to the most current version of the product release notes for information on how to access and view them.

## RPXE**4018**E

Failed
to
write
message
to
workbench
[%1]

## RPXE**4019**E

Failed
to
remove
users
for
user
group:
%1.

## RPXE**4020**E

Failed
to
add
users
for
userGroup:
%1
numUsers=[%2]
startId=[%3]

## RPXE**4021**E

Failed
to
add
users
because
the
runner
is
not
in
a
runnable
state.

## RPXE**4022**E

failed
to
add
desired
number
of
users

## RPXE**4023**E

failed
to
reach
target
number
of
users
ramping
down

## RPXE**4024**E

not
runnable
or
command
failed

## RPXE**4025**E

failed
to
set
the
DataView
state
of
user
%1[%2]
to
%3.

## RPXE**4026**E

DataView
command
%1
is
not
yet
implemented.

## RPXE**4027**E

DataView
command
%1
is
not
recognized.

RPXE**4028**E

MessageEventFilter command parsing error in token %1[%2] of command [%3]

RPXE**4029**E

The testLog message event filter specified by [%1] cannot be constructed. This filter element will be ignored.

## RPXE**4050**I

Operating
System
Info:
name
[%1]
architecture
[%2]
version
[%3]

## RPXE**4100**W

Cannot
open
execution
history
cache
file
[%1],
execution
history
will
not
be
cached

## RPXE**4101**E

Error
closing
execution
history
cache
file
[%1]

## RPXE**4102**E

Error
reading
%1
bytes
from
execution
history
cache
file
[%1]

## RPXE**4103**E

Error
writing
%1
bytes
to
execution
history
cache
file
[%2]

## RPXE**4104**E

Error
opening
execution
history
cache
file
[%1]
for
reading

## RPXE**4105**E

Error
testing
execution
history
cache
file
[%1]
for
available
input

## RPXE**4106**E

Unexpected
EOF
reading
%1
bytes
from
execution
history
cache
file
[%2]

## RPXE**4107**E

Exception
processing
execution
history
event

## RPXE**4108**E

Error
reading
execution
history
cache
file
[%1]

## RPXE**4109**E

Error
writing
to
TestLog
cache
file
[%1]

## RPXE**4110**E

Error
closing
TestLog
cache
file
[%1]

## RPXE4111W

Cannot open testLog cache file [%1] for random access writing, the testLog may contain bad data.

## RPXE4112W

Error removing testLog event from cache file [%1]. Writing %2 bytes at offset %3.

## RPXE**4120**E

Error
writing
to
TestLog
[%1]

## RPXE**4150**E

Error
opening
execution
history
annotation
file
[%1]

## RPXE**4151**E

Error
writing
%1
bytes
to
execution
history
annotation
file
[%2]

## RPXE**4152**E

Error
flushing/
closing
history
annotation
file
[%1]

## RPXE4153E

Error deleting history annotation file [%1]

## RPXE4200W

Warning: Statistics delivery thread running behind statistics interval by %1 milliseconds

## RPXE4201W

Warning: Statistics delivery thread over slept by %1 milliseconds

## RPXE**4202**E

Error:
Statistics
delivery
thread
over
slept
by
%1
milliseconds

## RPXE**4203**E

Error:
Statistics
collection
time
too
long:
%1
bytes
%2
milliseconds

## RPXE**4204**W

Warning:
Statistics
collection
time
too
long:
%1
bytes
%2
milliseconds

## RPXE**4205**E

Error:
Statistics
write
time
too
long:
%1
bytes
%2
milliseconds

## RPXE**4208**E

Error:
Could
not
create
agent
measurements
file
%1.

**Explanation:** It is not possible to create a file on the file system.

**System action:** Unable to create a file. The agent measurements will not be available.

**User response:** You do not have the permissions on your file system or it is full.

## RPXE**4209**I

Error:
Statistics
collection
thread
was
interrupted

**Explanation:** An error occurred which caused the interruption of the statistics collection.

**System action:** Statistics may be incomplete.

**User response:** Run the test again.

## RPXE**4210**E

Error:
A
severe
error
occurred
when
processing
statistics.

**Explanation:** An exception occurred on the agent while processing statistics and/or sending them to the server.

**System action:** Statistics will be incomplete.

**User response:** Contact support.

## RPXE**4211**E

Error:
A
severe
error
occurred
when
sending
statistics.

**Explanation:** An exception occurred on the agent while sending statistics to the server.

**System action:** Statistics will be incomplete.

**User response:** Contact support.

## RPXE**4212**E

Error:
A
severe
error
occurred
when
closing
statistics.

**Explanation:** An exception occurred on the agent while completing the statistics processing.

**System action:** Statistics may be incomplete.

**User response:** Contact support.

---

### RPXE**4213**E

Statistics
sub-
system
error:
%1

**Explanation:** A severe error occurred during writing to the agent measurements file.

**System action:** The agent detailed measurements will not be available.

**User response:** Start a test execution again.

---

### RPXE**4214**W

Statistics
sub-
system
warning:
%1

**Explanation:** Warning message to the user during writing to the agent measurements file.

**System action:** The agent detailed measurements may be affected by a problem.

**User response:** Fix the problem given by the message or contact support.

---

### RPXE**4215**E

Statistical
counter
descriptors
file
not
found:
%1.

**Explanation:** Unable to find the counter descriptors file in the deployment directories.

**System action:** Statistics will not be available.

**User response:** Start the test again, contact support if the problem persists.

---

## RPXE**4215**I

Statistics
sub-
system
message:
%1

**Explanation:** This message is displayed in debug mode.

**System action:** No action.

**User response:** You can report this message to the support.

---

## RPXE**4216**E

Problem
in
statistical
counter
descriptors
file:
%1.

**Explanation:** The counter descriptors file has a problem.

**System action:** Statistics will not be available.

**User response:** Start the test again, contact support if the problem persists.

## RPXE**4217**E

Submitted
value
%1
is
out
of
range
of
allowed
values
for
the
counter
type
%2.

**Explanation:** The value is out of the limits of the counter.

**System action:** Measurements and statistics for the specified counter will not be available.

**User response:** If you are a protocol developer, fix the problem. Otherwise, contact support.

## RPXE**4218**E

In
order
to
use
this
method,
the
runtime
type
of
the
counter
must
be
either
STATIC
or
RATE.

**Explanation:** A protocol is using a legacy API to change the value of a counter.

**System action:** Measurements and statistics for the specified counter will be inaccurate.

**User response:** If you are a protocol developer, use a runtime counter type to STATIC or RATE. Otherwise, contact support.

## RPXE**4219**E

Mismatch
between
runtime
type
%1
and
static
counter
type
%2.

**Explanation:** The type of the counter in runtime and in the statistic definition do not match.

**System action:** The runtime type will be applied.

**User response:** If you are a protocol developer, change the declared counter type, or the runtime type. Otherwise, contact support.

---

## RPXE**4220E**

No
static
declaration
found
for
counter
%1.

**Explanation:** Unable to find a definition for the counter.

**System action:** The counter values will be ignored.

**User response:** Add a definition for the counter, or use an undeclared counter.

---

## RPXE**4221E**

Attempt
to
create
an
undeclared
counter
%1
(type
%2)
over
a
declared
counter
of
a
different
type
(%3).

**Explanation:** An attempt to create an undeclared counter was made, but a counter declaration with another type already exists.

**System action:** The undeclared counter values will be ignored.

**User response:** Use another path for the undeclared counter that does not conflict with the existing declared counter.

---

## RPXE**4900**I

Test
execution
completed
with
no
reported
problems

---

## RPXE**4901**I

%1
ERROR
verdicts
reported

---

## RPXE**4902**I

%1
FAIL
verdicts
reported

---

## RPXE**4903**I

%1
INCONCLUSIVE
verdicts
reported

---

## RPXE**4904**I

All
reported
verdicts
PASSed

## RPXE**4905**I

%1
ERROR
verdict
reported

## RPXE**4906**I

%1
FAIL
verdict
reported

## RPXE**4907**I

%1
INCONCLUSIVE
verdict
reported

## RPXE**4908**I

%1
FAIL
verdict
roll-
up

## RPXE**4909**I

%1
ERROR
verdict
roll-
up

## RPXE**4910**I

%1
INCONCLUSIVE
verdict
roll-
up

## RPXE**4911**I

%1
PASS
verdict
roll-
up

## RPXE**4912**I

%1
ERROR
verdicts
reported
from
driver
%2

## RPXE**4913**I

%1
FAIL
verdicts
reported
from
driver
%2

## RPXE**4914**I

%1
INCONCLUSIVE
verdicts
reported
from
driver
%2

## RPXE**4915**l

%1
ERROR
verdict
reported
from
driver
%2

## RPXE**4916**l

%1
FAIL
verdict
reported
from
driver
%2

## RPXE**4917**l

%1
INCONCLUSIVE
verdict
reported
from
driver
%2

## RPXE**4918**l

duration

**RPXE4920I**

%1 was successfully invoked. This does not indicate the pass/ fail verdict of the test itself, only that the invocation of the test was successful. Expand to inspect verdicts.

## RPXE4921I

%1 was invoked. This does not indicate the pass/fail verdict of the test itself, only that the invocation of the test was successful. No verdicts will be reported from the test.

RPXE**4930**I

The %1 testLog level was pushed from %2%3 to %4%5.

RPXE**4931**I

The %1 testLog level was popped from %2%3 to %4%5.

RPXE**4932**I

The %1 testLog level was changed from %2%3 to %4%5.

## RPXE**4940**I

Transaction [%1] started %2 milliseconds after start of test run.

## RPXE**4941**I

Transaction [%1] stopped %2 milliseconds after start of test run. Elapsed time: %3 milliseconds.

## RPXE**4942**I

Transaction [%1] aborted.

## RPXE**4944**W

Transaction [%1] is already started.

## RPXE**4945**W

Transaction
[%1]
has
not
been
started.

## RPXE**4948**W

Execution
Variables
-
Input

## RPXE**4950**I

Null
user
group
name.

## RPXE**4952**E

Unable
to
find
target
loop
named
'%1'.
Error
handler
did
not
complete
properly.

**Explanation:** The loop name specified in the loop handler does not exist.

**System action:** The user will not follow the loop error handler and will continue execution at the next action.

**User response:** Change the loop handler to point to an existing loop.

## RPXE**5301**E

Error
encountered
while
loading
Native
Library:
%1

## RPXE**5305**E

A
required
customer-
supplied
file
was
not
found.
Please
check
the
"external_files"
folder
and
your
installation
instructions
for:
%1

## RPXE**5330**E

Unable
to
apply
dataset
swap:
%1

**Explanation:** An error occurred attempting to parse the data set swap command-line option.

**System action:** The data set swap will not occur.

**User response:** See the command-line usage to ensure the command syntax is correct.

---

## RPXE**5500**W

Unable to apply Open Tracing context. The root Jaeger span will be unparented. %1

**Explanation:** An error occurred when attempting to create an Open Tracing span context from the properties starting with OPENTRACING_CTX_.

**System action:** Jaeger logging will still occur but the root span will be linked to a parent span.

**User response:** Make sure the content of properties starting with OPENTRACING_CTX_ is correct.

## RPXE**5501**W

Transaction
times
for
this
run
do
not
include
failing
transactions,
according
to
workbench
Test
Execution
preferences.

**Explanation:** A failing transaction will not be added to stats. This will only be logged once per transaction, but multiple instances may have failed.

**System action:** Execution will continue as normal. This is not an error condition.

**User response:** If this behavior is not desired, uncheck preferences at Test > Test Execution.

## RRIT0001E

Environment variable INTEGRATION_TESTER_AGENT_HOME not set to Rational® Integration Tester Agent installation location, or does not contain expected RunTests(.exe) program.

**Explanation:** The test execution cannot find the Rational® Integration Tester Agent.

**System action:** None.

**User response:** Set the environment variable INTEGRATION_TESTER_AGENT_HOME to point the root installation directory of the Rational® Integration Tester agent. This must be done on each location used in a schedule.

## RRIT0002E

Error unzipping __IT_PRODUCT_NAME__ project.

**Explanation:** The __IT_PRODUCT_NAME__ project cannot be deployed.

**System action:** None.

**User response:** Verify that there is enough disk space on the executing location.

## RRIT0003E

__IT_PRODUCT_NAME__
library
not
found.

**Explanation:** The library required to communicate with the __IT_PRODUCT_NAME__ agent is missing in the installation.

**System action:** None.

**User response:** Contact your support.

## RRIT0004E

Error
processing
messages
received
form
the
__IT_PRODUCT_NAME__
client.

**Explanation:** A communication error has occurred with the __IT_PRODUCT_NAME__ agent.

**System action:** None.

**User response:** Try again, contact your support if problem persist.

## RRIT0005E

Some
tag
values
are
missing.

**Explanation:** A value cannot be assigned to a tag defined in an __IT_PRODUCT_NAME__ test during execution.

**System action:** None.

**User response:** Verify that each tag of each __IT_PRODUCT_NAME__ test maps to a variable in the schedule or compound test.

## RRITUI1002W

Open
__IT_PRODUCT_NAME__
resources
has
been
disabled
in
Test
>
__IT_PRODUCT_ACRONYM__
Integration
preferences

**Explanation:** User as disabled __IT_PRODUCT_NAME__ resource but want to open this kind of resource.

**System action:** None.

**User response:** Open Test > __IT_PRODUCT_ACRONYM__ Integration preference and enable open resources by checking __IT_PRODUCT_NAME__ is installed on this machine .

## DCRC0001E

Missing
message
for
log
entry
'*{0}*'
in
class:
*{1}*

## DCRC0002E

Cannot
get
Log
key
'*{0}*':
SecurityException
raised

## DCRC**0003**E

Cannot
initialize
Log
key
'*{0}*'

## DCRC**0008**W

Warning:
field
'*{0}*'
is
not
defined
in
class:
*{1}*

## DCRC**0009**W

Warning:
cannot
get
check
message
versus
log
key
mapping
for
'*{0}*'
of
class
*{1},*
SecurityException
raised

## DCRC**0010**E

Unexpected
exception,
please
check
Error
Log
view:
*{0}*

## DCUI**0001**E

unexpected
exception

**Explanation:** An exception that could not be handled occurs during processing.

**User response:** Close rule editor and report exception to product support.

## DCUI**0003**E

Error
getting
persistent
property
'*{0}*'

## DCUI**0004**E

Error
setting
persistent
property
'*{0}*'

## DCUI**0006**E

Cannot
reload
resource
'*{0}*'

## DCUI0007W

Failed
to
encode
model
to
clipboard.

## DCUI0008W

Failed
to
decode
model
from
clipboard.

## DCUI0009E

None
of
the
attribute
providers
own
attribute
id
'*{0}*'.

**Explanation:** Rule file refer to an unknown rule attribute id. File may be edited on a system having more protocol extension rather than current one.

**User response:** Rule file should not be edited on this product installation.

## DCUI0010E

Missing
IRuleUIProvider
extension
point
for
'*{0}*'

**Explanation:** Rule file refer to a rule that is unknown on this product installation.

**User response:** Rule editor is able to display that rule on the tree but not able to edit it contents.

---

## DCUI**0011**E

Missing
IConditionUIProvider
extension
point
for
'*{0}*'

**Explanation:** Rule file refer to a rule condition that is unknown on this product installation.

**User response:** Rule editor is able to display that rule condition on the tree but not able to edit it contents.

---

## DCUI**0012**E

Cannot
save
editor
'*{0}*'

---

## DCUI**0013**E

Missing
IRulePassUIProvider
extension
point
for
'*{0}*'

**Explanation:** Rule file refer to a rule pass that is unknown on this product installation.

**User response:** Rule editor is able to display that rule pass on the tree but not able to edit it contents.

## DCUI**0014**E

Missing IRuleArgumentUIProvider extension point for '*{0}*'

## DCUI**0015**E

Missing IRuleArgumentContainerUIProvider extension point for '*{0}*'

## DCUI**0016**E

Try rule failed

## DCUI**0017**E

Try rule failed: '*{0}*'

## DCUI**0998**E

Cannot load file '*{0}*'

# Chapter 10. Reference Guide

This guide describes, additional topics to gain more knowledge about Rational® Service Tester for SOA Quality.

## Accessibility features

Accessibility features help users who have physical disabilities, such as visual and, hearing impairment, or limited mobility, to use the software products successfully.

Accessibility features are product dependent and might include one or more of the following aspects:

- Keyboard-only operation

- Screen reader usage

- Color and typeface preferences

**Note:** The accessibility features mentioned here apply to the Windows operating system. Some of these features might also work on Linux, but are not officially supported.

### Accessibility compliance

To understand the accessibility compliance status for Rational® Performance Tester Rational® Service Tester for SOA Quality, refer to Accessibility Conformance Reports.

For more information about IBM and accessibility, refer to IBM Accessibility.

The product documentation is published by using Oxygen XML WebHelp Responsive. To understand the accessibility compliance status for Oxygen XML WebHelp Responsive, refer to WebHelp Responsive VPAT Accessibility Conformance Report.

### Accessing UI elements

Rational® Performance Tester Rational® Service Tester for SOA Quality supports navigation in the UI by using different methods such as a mouse, keyboard, or touchpad.

You can use the keyboard keys such as **Tab**, arrow keys such as **UP**, **DOWN**, **LEFT**, and **RIGHT** to navigate to the different pages in the **Navigation** pane or to the different action labels in the right pane on the UI.

## Keyboard shortcuts for performance and service testing

The keyboard shortcuts for performance and service testing are available when you record or edit a test or a schedule.

| Key combination | Description |
| --- | --- |
| Ctrl+S | Save the test or schedule. |

| Key combination | Description |
| --- | --- |
| Alt+Shift+T, G | Generate a test from the selected recording (`.recmodel`) file. |
| Alt+Shift+T, R | Create a report (the test must be selected in the Test Navigator). |
| Alt+Shift+T, T | Test connection (a location must be selected in the Test Navigator). |
| Alt+Shift+X, B | Run the test (a test must be selected in the Test Navigator). |
| Alt+Shift+X, C | Run the schedule (a schedule must be selected in the Test Navigator). |
| Del | Delete the selection |
| Ctrl+Del | Delete the selection |
| Insert | Insert a new element (same as the **Insert** push button). |
| Ctrl+Insert | Add a new element (same as the **Add** push button). |
| Ctrl+Up Arrow | Move the element up. |
| Ctrl+Down Arrow | Move the element down. |
| Ctrl+Alt+<, Ctrl+Alt+> | Resize the test editor and schedule editor windows. The new size is retained when you reopen the window. |
| Ctrl+Shift+F1 | During HTTP recording, insert a comment. |
| Ctrl+Shift+F2 | During HTTP recording, insert a screen capture. |
| Ctrl+Shift+F3 | During HTTP recording, insert a synchronization point. |
| Ctrl+Shift+F4 | During HTTP recording, start a transaction. |
| Ctrl+Shift+F5 | During HTTP recording, end a transaction. |
| Ctrl+Shift+F6 | During HTTP recording, insert a split point. |
| Ctrl+Shift+F7 | During HTTP recording, set the name of the current page. |

The following keyboard shortcuts are available when you record Citrix performance tests:

| Key combination | Description |
| --- | --- |
| Tab or Shift+Tab | Cycle the focus through the UI elements |
| Arrows | Select a push button |
| Space | Click a push button or toggle between selections |

When you record Citrix performance tests and you work in image synchronization mode, you can use these keys:

| Key | Description |
| --- | --- |
| Space | Set the origin of selection area |
| Arrows | Move the cursor |

| Key | Description |
|---|---|
| Enter | Select the image synchronization area and set the synchronization area (press twice) |
| Esc | Cancel the selection |

# General reference for performance testing

See these performance testing topics for general reference.

## Data correlation rules

You can customize how data is correlated by using data correlation rules.

### Rules that create elements

**Create a built in data source**

Inserts a built-in data source in the test.

**Create a custom code**

Inserts a custom code element in the test.

**Create a dataset column**

Creates a dataset column that can be used by substitution sites.

**Create a reference**

Creates a reference in data that matches a specified regular expression.

**Create a substitution**

Creates a substitution site in data that matches a specified regular expression.

**Create a variable assignment**

Inserts a variable assignment in the test.

**Create a variable declaration**

Creates a variable that can be used by substitution sites.

### Rules that change elements

**Encode a substitution**

Specifies whether substitution fields are encoded or decoded.

**Rename a data source site**

Changes the name of a data source that matches a regular expression.

**Rename a substitution site**

Changes the name of a substitution site that matches a regular expression.

**Replace reference regular expression**

Changes the regular expressions that are used by references.

**Unlink a substitution**

Removes the links between substitution sites and references and other data sources.

## Rules that find elements

### Find a reference

Returns a reference. Add child conditions to specify the reference to find.

### Find a substitution

Returns a substitution site. Add child conditions to specify the substitution site to find.

### Find a variable

Returns a variable. Add child conditions to specify the variable to find.

## Rules that remove elements

### Remove a built in data source

Deletes data sources from the test. Add child conditions to specify the data sources to delete. If you do not add child conditions, this rule deletes all data sources in the test.

### Remove a custom code

Deletes custom code elements from the test. Add child conditions to specify the custom code elements to delete. If you do not add child conditions, this rule deletes all custom code elements in the test.

### Remove a reference

Deletes references from the test. Add child conditions to specify the references to delete. If you do not add child conditions, this rule deletes all references in the test.

### Remove a substitution

Deletes substitution sites from the test. Add child conditions to specify the substitution sites to delete. If you do not add child conditions, this rule deletes all substitution sites in the test.

### Remove a variable assignment

Deletes variable assignments from the test. Add child conditions to specify the variable assignments to delete. If you do not add child conditions, this rule deletes all variable assignments in the test.

### Remove a variable declaration

Deletes variables from the test. Add child conditions to specify the variables to delete. If you do not add child conditions, this rule deletes all variables in the test.

## Error conditions

Error conditions include verification point failures, connection failures, server timeouts, custom code alerts, custom code exceptions, and problems with data correlation. You can specify an action to take when the error condition occurs. The Errors report displays the error conditions and error behavior that occurred in a test or schedule.

### Page Title Verification Point Failure [HTTP]

The returned title for the primary request for an HTTP page does not match the expected title. The default value of the expected page title is what is returned between the <title></title> tags during recording. See Specifying the expected page title for more information.

### Response Code Verification Failure [HTTP]

The returned response code does not match the expected response code. You can specify an exact match or a relaxed match. See Specifying the expected response code for more information.

### Response Size Verification Failure [HTTP]

The number of bytes returned does not match the expected number of bytes. You can control how closely the returned response size must match the recorded response size. See Specifying the expected response size for more information.

### Content Verification Point Failure

The received data does not match the expected data. Content verification point controls are protocol-specific.

### Connection Failure

The workbench or agent computers cannot connect to the server under test.

### Authentication Failure

An attempt to log in to the server under test failed.

### End of Dataset reached

The last row of the dataset is reached. See Dataset overview on page       for more information.

### Reference Extraction Failure

The response received during playback is different from the response received when the test was recorded. Data correlation failed because the regular expression that is associated with the reference did not match the expected value.

### Substitution Failure

A reference for an expected data substitution is a null reference.

### Server Timeout

The server under test does not respond before the timeout interval elapses.

### Custom Verification Point Failure

A custom verification point did not return a Pass status after performing a verification written in Java™ code. See Reporting custom verification point failures on page 329 for more information.

**Custom Code Alert**

Custom code reported an **RPTCondition.CustomCodeAlert** condition. The following code reports a custom code alert:

```
tes.getTestLogManager().reportErrorCondition(RPTCondition.CustomCodeAlert);
```

See the ITestLogManager Javadoc for more information.

The Javadoc for the test execution services interfaces and classes can be accessed from the product by clicking **Help > Help Contents**IBM Rational Performance Tester **API Reference**.

**Custom Code Exception**

The custom code in a test has an exception. By default, Rational® Performance Tester exits the user whenever there is an exception in custom code. For information on setting different actions, see Error-handling behavior on page

Related information

Specifying error-handling behavior

# Resource monitoring data sources

Resource monitoring data can be captured or imported from a number of sources.

**IBM® Tivoli® Monitoring**

IBM Tivoli® Monitoring monitors and manages system and network applications on a variety of platforms and keeps track of the availability and performance of all parts of your enterprise network. IBM® Tivoli® Monitoring provides reports that you can use to track trends and troubleshoot problems.

Not all IBM® Tivoli® Monitoring agents are supported. Over 100 IBM® Tivoli® Monitoring agents are available from IBM® and non-IBM vendors. The following IBM® Tivoli® Monitoring agents are supported for resource monitoring data collection:

- Operating system agents
    - Monitoring Agent for Linux™ OS
    - Monitoring Agent for UNIX™ OS
    - Monitoring Agent for Windows™ OS
    - Monitoring Agent for z/OS®
- Application agents
    - Monitoring Agent for Citrix
    - Monitoring Agent for IBM® DB2®
    - Monitoring Agent for IBM® Tivoli® Composite Application Manager for WebSphere®
    - Monitoring Agent for IBM® WebSphere® Application Server
    - Monitoring Agent for IBM® WebSphere® MQ

- Monitoring Agent for Oracle Database
- Monitoring Agent for SNMP-MIB2 (only)

### IBM® DB2® Monitoring

IBM DB2® collects information from the database manager, its databases, and any connected applications. The snapshot monitor captures the state of database activity at a particular point in time.

### IBM® WebSphere® Performance Monitoring Infrastructure

IBM WebSphere® Application Server collects performance data and provides interfaces so that external applications can monitor that performance data. To help identify performance problems and help tune an environment that runs web applications, data is collected through the Performance Monitoring Infrastructure (PMI). The Performance Monitoring Infrastructure is the underlying framework in WebSphere® Application Server that gathers performance data from various runtime resources, such as Java™ Virtual Machine (JVM) and Thread Pools, and application components, such as servlets and Enterprise JavaBeans™ (EJB) components.

### Java™ Management Extensions

Java Management Extensions (JMX) can monitor performance characteristics of application servers and applications that are run on application servers. The following application servers support JMX monitoring:

- Apache HTTP Server
- Apache Tomcat
- JBoss Application Server
- Oracle WebLogic Server
- SAP NetWeaver

Java™ Virtual Machines also support JMX monitoring.

### Oracle Database Metrics

Oracle Database collects metrics that are related to database health and workload.

### UNIX™ rstatd

With the rstatd daemon, users can collect performance statistics remotely from networked UNIX™ (or Linux™) computers. The rstatd daemon collects statistics that are related to network, virtual memory, interrupt, disk, and processor usage.

### Simple Network Management Protocol (SNMP) agents

The Simple Network Management Protocol (SNMP) is typically used to monitor network health, performance, and hardware. SNMP agents are software components that are installed on managed devices and collect management information.

### Windows™ Performance Monitor

Windows Performance Monitor (PerfMon) collects data from performance objects. The Microsoft™ Windows™ operating system provides performance objects for the major hardware components: memory, processors, and so on. Each performance object provides specific performance counters. For example, the `Memory` object provides a `Pages/`

`sec` counter that tracks the rate of memory paging. Other programs on the computer, including Internet Information Services (IIS) and Microsoft™ SQL Server, can install their own performance objects. For example, a mail server program might install a mail performance object. The specific counters depend on the version of the Windows™ operating system and on the additional programs that are installed on the computer.

## Response time breakdown data sources

Response time breakdown data can be imported from a number of sources.

### IBM® Tivoli® Composite Application Manager for Application Diagnostics

IBM Tivoli® Composite Application Manager for Application Diagnostics enables users to view the health of web applications and servers, then drill down to diagnostic information for specific application requests to identify the root cause of problems.

### IBM® Tivoli® Composite Application Manager for Response Time Tracking

IBM Tivoli® Composite Application Manager for Response Time Tracking measures the level of service that the application delivers to users. It does this by monitoring the availability and response time that users experience at the client desktop. It works with a wide range of web-based, e-business, and Microsoft™ Windows™ applications that run in many different environments.

### IBM® Tivoli® Composite Application Manager for WebSphere®

IBM Tivoli® Composite Application Manager for WebSphere® provides immediate problem determination, availability monitoring, and performance analysis for enterprise WebSphere® applications running on Windows™, UNIX™, OS/400®, and z/OS® environments. IBM® Tivoli® Composite Application Manager for WebSphere® monitors heterogeneous environments consisting of both mainframes and distributed systems.

### IBM® Tivoli® Monitoring for Transaction Performance

IBM Tivoli® Monitoring for Transaction Performance is a centrally managed suite of software components that monitor the availability and performance of web-based services and Windows™ applications.

---

Related information

IBM Tivoli Composite Application Manager for Applications

# UI preferences

Read the UI preferences topics.

## VU Schedule editor reference

Most VU Schedule editor settings apply either to the entire schedule or to individual user groups.

## Schedule properties

When you open a schedule, you can set its properties.

### User Load page

Right-click in the table, and select **Add** to add a stage. To modify a stage, select the row, and then click**Edit** or click the user icon in the first column.

**Users**

Enter the total number of users to be active in the stage (not the number of users to add or subtract to those currently running).

**Run for specified period of time**

Enter the length of time (and the time units) for the stage to run. When the specified number of users is achieved, the users will run for up to this time. When the time expires, the users continue to run if they are required for the next stage; otherwise, they are stopped gracefully.

Click **Show Advanced** to set further options to prepare the system under test before the users actually enter the stage:

**Change Rate**

Enter a number to set a delay between adding or removing each user, rather than adding them or subtracting them all at once. Staggering users avoids overloading the system, which can cause connection timeouts. The **User Load Preview** shows this delay in black.

**Settle Time**

A system under test might react to a sudden change in user population. With a defined settle time, which starts when the target number of users is reached, the system under test can settle into a steady state so that it can accurately reflect the user population. The **User Load Preview** shows this time in black.

**Time limit for a user to respond to a stop request**

Optionally enter a value. When a virtual user is asked to stop, it completes its current action (such as an HTTP request) and then finishes. If a virtual user has not finished within the specified time limit, the user is forced to finish.

**User Load Preview**

Previews the user population stages over time. The red line segments indicate that the total number of users has been achieved for the state.

### Think Time page

**Use the recorded think time**

Select to play back a test at the same rate that it was recorded. This option has no effect on the think time.

**Specify a fixed think time**

Each user's think time is exactly the same value: the value that you specify. Although this does not emulate users accurately, it is useful if you want to play a test back quickly.

**Increase/decrease the think time by a percentage**

Type a percentage in **Think time scale**. Each user's think time is multiplied by that percentage. A value of 100 causes no change in think times. A value of 200 doubles the think times, so the schedule plays back half as fast as it was recorded. A value of 50 reduces the think times by half, so the schedule plays back twice as fast. A value of 0 indicates no delays.

**Vary the think time by a random percentage**

Each user's think time is randomly generated within the upper and lower bounds of the percentages that you supply. The percentage is based on the recorded think time. For example, if you enter 10 in **Lower limit** and enter 90 in **Upper limit**, the think times will be between 10 percent and 90 percent of the original recorded think time. The random time is uniformly distributed within this range.

**Maximum think time**

Setting a maximum think time is useful with tests that emulate actual think times. By setting a maximum, you do not have to search for and edit each long think time within a test. Numerous factors can generate long think times, for example, you might be interrupted while recording. To restore the original think times, clear this check box.

## Resource Monitoring page

**Enable resource monitoring**

Select to activate resource monitoring. The available data sources are captured from these sources:

- Apache HTTP Server Managed Beans
- Apache Tomcat Managed Beans
- IBM® Tivoli® monitoring agents
- IBM® DB2® snapshot monitors
- The IBM® WebSphere® Performance Monitoring Infrastructure
- JBoss Application Server Managed Beans
- Java™ Virtual Machine Managed Beans
- Oracle Database
- Oracle WebLogic Server Managed Beans
- SAP NetWeaver Managed Beans
- the UNIX™ rstatd monitor
- Simple Network Management Protocol (SNMP) agents
- Windows™ Performance Monitor

Resource monitoring data can provide a more complete view of a system to aid in problem determination.

**Ignore invalid resources when executing the schedule**

Select this setting to suppress any error messages that invalid resources cause, such as unreachable hosts or invalid host names. If you select this option, you must view logs to see error messages.

### Statistics page

#### Statistics log level

These options are listed in order of the increasing amount of data that they collect for the test log.

##### None

Collects minimal statistical data. Use this option to run a schedule quickly for testing purposes.

##### Schedule Actions

Reports the number of active and completed users in the run.

##### Primary Test Actions

For HTTP tests, this option reports page-related actions (attempts, hits, and verification points). For SAP tests, this option reports information on SAP screens.

##### Secondary Test Actions

For HTTP tests, this option reports information that is related to page elements. This option does not apply to SAP tests.

##### All

Provides statistics for all actions.

#### Statistics sample interval

Sets the sampling interval for reports. When you run a schedule, the reports show such information as response time during a specific interval, the frequency of requests being transferred during an interval, and the average response trend during an interval. You set this sampling interval here.

> **Note:** In Rational® Performance Tester V9.1.1 and later, during a test or schedule run, the **Elapsed Time** value is updated every 5 seconds irrespective of a value set in the **Statistics sample interval** field. You can view the **Elapsed Time** value that is changing in the **Run Summary** section on the **Performance Summary** tab of the report.

#### Only store All Hosts statistics

Select this option unless you are running a performance test over different WANs, and you are interested in seeing the data from each remote computer.

### Variable Initialization

Use this page to initialize variables at the schedule level. When you initialize variables at the schedule level, all the user groups in the schedule use the variable initial values, except those for which a specific value is defined.

#### Add

Add a variable and initialize a value. The **Used by** column displays the test name that uses the corresponding variable. A warning icon is displayed for a variable that overrides the value specified at

the schedule level or user group level and uses the value defined at the test level with the visibility set to **This test only**. Hover the cursor over the warning icon to view the tests that override the variable initial values.

### Export

Export the variables defined at the schedule level to a file.

### Use variable initial values file

Select this check box to use the variable values from a file. Click **Browse** to select an existing file or click **New** to create a file.

## Performance Requirements page

### Enable Requirements

Select to enable the use of performance and functional requirements for this schedule.

### Name

Specifies the name of this set of requirements. This name is used in the Requirements report. By default, the name is `Performance Schedule -schedule_name`.

### Use Defaults

Click to reset **Name** to the default value.

### Requirement

All the requirements are displayed in the table. Shaded requirements are not defined for this schedule. To define a requirement, set an **Operator** and **Value**.

Expand the **Custom** section, and then double-click the row to add the counter information generated by using the custom code to the requirement.

### Operator

Click this field to display a list of mathematical operators. Select an operator for this requirement.

### Value

Click this field to set a value for the requirement.

### Standard

Select to mark the requirement as standard. If a standard requirement is not met, the schedule run will have a verdict of fail, and this verdict will roll up to the entire run, like a verification point failure. Clear to make the requirement *supplemental*. In general, supplemental requirements are those that are tracked internally. A supplemental requirement cannot cause a run to fail, and its results are restricted to one page of the Requirements report.

### Hide Undefined Requirements

Select to see only the requirements that you have defined. This hides the shaded rows.

**Clear**

Select one or more requirements and click to remove the definition. The requirement is still available and can be redefined.

## Test Log page

The default setting, to log all errors and warnings and primary test actions, fits most purposes. However, you can log any type of information, from no information to all information from all users, although neither is typical.

- To see only errors and warnings, set the first two **What to Log** check boxes to **All**; then clear the third check box, **And also show all other types**, to avoid logging successful events.
- To check the structure of a schedule, when you are not interested in the test execution results, set all three **What to Log** check boxes to **Schedule Actions**.

  Both choices and the default setting limit the size of the test log and reduce the total time to run the schedule by significantly reducing the test log transfer time at the end of a test.

If you are debugging a test, you might set all three **What to Log** fields to **All** or **Action Details**. These settings produce large test logs, especially if your tests are long or you are running a large number of users. Large test logs, in turn, increase the test log transfer time, and might even cause your computer to run out of disk space.

To reduce transfer times and the likelihood of running out of disk space, sample information from a very small subset of users; smaller even than the default of 5 users per user group. A fixed sampling rate samples the same number of virtual users from each group. A percentage sampling rate samples a percentage of virtual users from each group, but guarantees that at least one user is sampled from a group.

## Response Time Breakdown page

### Enable collection of response time data

Select to activate response time breakdown collection. This data shows you the response time breakdown for each page element.

### Detail level

Select **Low** or **Medium** to limit the amount of collected data.

### Only sample information from a subset of users

If you set the detail level to **High** or **Medium**, set a sampling rate to prevent the log from getting too large.

### Fixed number of users

The number that you select is sampled from each user group. Unless you have specific reasons to collect data from multiple users, select **Fixed number of users**, and specify one user per user group.

### Percentage of users

The percentage that you select is sampled from each user group, but at least one user is sampled from each user group.

### Problem Determination page

#### Problem determination log level

In general, change the problem determination level only when asked to by IBM® Software Support. However, under certain conditions, you might want to change the problem determination level. For example, if problems occur when a run reaches a certain number of users, you might increase the level to **Config**, which is the most detailed level to use without consulting IBM® Software Support.

#### Only sample information from a subset of users

Select this option to set a sampling rate.

#### Fixed number of users

Specify the number of users to sample from each user group.

#### Percentage of users

The percentage that you select is sampled from each user group, but at least one user is sampled from each group.

## User group properties

When you open a user group, you can set these properties.

#### Group size

Specifies either an absolute number of users, or a percentage of users, which you control dynamically.

### Locations

#### Run this group on the local computer

Indicates that the user group should be run on your computer.

#### Run this group at the following locations

Indicates that the user group should be run on one or more remote computers, at the indicated locations. Typically, you run a user group at a remote location if you are running a large number of virtual users.

### Options

Use the Options page to override the think time behavior of your schedule for a specific user group and to specify protocol specific options.

#### Override think time option

Select this check box to specify a think time behavior for the current user group.

#### Use the recorded think time

Select to play back a test at the same rate that it was recorded. This option has no effect on the think time.

**Specify a fixed think time**

Each user's think time is exactly the same value: the value that you specify. Although this does not emulate users accurately, it is useful if you want to play a test back quickly.

**Increase/decrease the think time by a percentage**

Type a percentage in the **Think time scale**. Each user's think time is multiplied by that percentage. A value of 100 causes no change in think times. A value of 200 doubles the think times, so the schedule plays back half as fast as it was recorded. A value of 50 reduces the think times by half, so the schedule plays back twice as fast. A value of 0 indicates no delays.

**Vary the think time by a random percentage**

Each user's think time is randomly generated within the upper and lower bounds of the percentages that you supply. The percentage is based on the recorded think time. For example, if you select a **Lower limit** of 10 and an **Upper limit** of 90, the think times will be between 10 percent and 90 percent of the original recorded think time. The random time is uniformly distributed within this range.

**Limit think times to a maximum value**

Setting a maximum think time is useful with tests that emulate actual think times. By setting a maximum, you do not have to search for and edit each long think time within a test, if, for example, you are interrupted during recording. No think time used will be greater than the maximum limit you set, even if you have chosen to vary the think time by a percentage that would exceed this maximum. To restore the original think times, clear this box.

**Protocol-specific options**

Click **Edit options** to set protocol-specific options for all tests in the user group. These settings override the protocol-specific options of the schedule.

## Variable Initialization

Use this page to initialize variables at the user group level. When you initialize variables at the user group level, all the tests in the user group use the variables. If the same variable is defined at the schedule level, precedence is given to the variable at the user group level.

**Add**

Add a variable and initialize a value. The **Used by** column displays the test name that uses the corresponding variable. A warning icon is displayed for a variable that override the value specified at the schedule level or user group level and uses the value defined at the test level with the visibility set to **This test only**. Hover the cursor over the warning icon to view the tests that overrides the variable initial values.

**Export**

Export the variables defined at the user group level to a file.

**Use variable initial values file**

> Select this check box to use the variable values from a file. Click **Browse** to select an existing file or click **New** to create a file.

## Generic service client references

Read these reference topics for Generic service client.

## Generic service client call details

In the generic service client, service calls contain the content and the transport information for the call. The contents are made of the SOAP envelope. The transport information refers to the information that is required to send and receive and answer depending on the selected protocol.

### Message

This page shows the XML content of the request and provides access to data correlation. The same content is presented in three different ways.

**Form**

> This view provides a simplified view of the message that focuses on editing the values of the XML content. Use the **Schema** menu to enable assistance with editing XML content so that the XML is valid and complies with the XSD specification.

> In the **Form** view, add the XML headers that are required for standard web service calls. On the **Header** bar, click **Add** (➕) to create the default XML header structure for WS-Addressing, WS-ReliableMessaging or WS-Coordination requests, or click **More** for other standards. You can enable or disable XML header elements and specify the correct values for each XML element. Checks are performed to ensure that the XML content is valid.

> > 📝 **Note:** To add XML headers to calls in IBM® Security AppScan®, add a **Static XML Headers** algorithm on the **Request Stack** tab of the request.

**Tree**

> This view provides a hierarchical view of the XML structure of the message, including elements, namespaces, and the associated values. You can use **Add**, **Insert**, **Remove**, **Up**, and **Down** to edit the XML elements and namespaces in the tree.

> Use **Skip if Empty** column to select the empty XML elements that you want to skip. This column is visible only if you selected the **Display the 'Skip if Empty' column in XML tree viewer** check box in **Window > Preferences > Test > Test editor > Service test**.

> Click **Filter** to hide or show namespace, attribute, or text nodes, depending on your requirements.

Click **Allow only valid modifications** to enable smart editing, based on a specified XML schema document (XSD). To specify a set of XSD documents for the workbench, in the test navigator, right-click the project and select **Properties** and **Schema Catalog**. Disable **Allow only valid modifications** if you do not have an XSD or if you want to bypass the schema.

You can right-click an XML element to convert it to an XML fragment. This enables you to perform data correlation (use datasets and create references) on the entire XML fragment instead of only on the value.

**Source**

This view displays the source XML content of the message or plain text content. To format XML content, click **Format XML text**. To wrap XML content into a single line, click **Pack XML text to single line**. Similar controls are available for JSON content.

> ⚠️ **Important:** In the Source view, do not edit the tags that start with `SoaTag`. If you delete or change these tags, any references and substitutions in the test will be broken. You cannot recreate these tags after you delete them.

## Attachments

This page lists the MIME or DIME attachments that are attached to the request. The contents of this view conform to the Multipurpose Internet Mail Extensions (MIME) or Direct Internet Message Encapsulation (DIME) specification. You can use this page to add workbench resources as MIME or DIME attachments and change properties.

The **Content ID** is the identifier that the request uses to refer to the attachments. The method for using this identifier depends on your server requirements.

**MIME or DIME**

Select whether the attachment conforms to the Multipurpose Internet Mail Extensions (MIME) or Direct Internet Message Encapsulation (DIME) specification

**Use MTOM transmission mechanism**

By default, the request uses SOAP Messages with Attachments (SwA) to handle attachments. Select this option to handle attachments with the SOAP Message Transmission Optimization Mechanism (MTOM).

## Transport

This page covers the transport settings used to send the request. The transport protocol settings apply to a transport configuration, which can be either HTTP, Java™ Message Service (JMS), WebSphere® MQ, or Microsoft .NET. You can create several configurations for each protocol so that you can easily switch protocols or variants of protocols.

**Note:** If you are using IBM® Security AppScan®,only the HTTP transport protocol is available.

### HTTP

Select **HTTP** to use the HTTP transport for the request. At the request level, you can update a URL or SOAP action and the reference to the global configuration of a test.

#### Protocol configuration

Click **Change** to specify a predefined transport configuration or to create a configuration. HTTP transport configurations contain proxy and authentication settings that can be reused.

#### URL

Specify the URL end point of the service request.

#### Rest mode

Use this check box to split the REST URL so that it is easy to understand the different parts of REST URL. When you use this option, the main section of URL is placed in the URL field, the resource part is placed in the **Resource** field, and the parameters are placed in the **Parameters** field. Use the **Add** button to manually add more parameters.

#### Method

Specify the HTTP method to be used to invoke the service request.

#### Headers

Specify the names and values of any custom HTTP headers that are required by the service. Click **Add**, **Edit** or **Remove** to modify the headers list.

#### Cookies

Specify the names and values of any cookies that are required by the service. Click **Add**, **Edit** or **Remove** to modify the cookies list.

### JMS

Select **JMS** to use the Java™ Messaging Service transport for the request. This page enables you to add string properties that are attached to the request for a JMS configuration. These will be sent as message properties through JMS.

#### Protocol configuration

Click **Change** to specify a predefined transport configuration or to create a configuration. JMS transport configurations contain generic end point, reception point, and adapter settings that can be reused.

**Properties**

Specify the names and values of any string properties that are required by the request for the current JMS transport configuration. These are sent as message properties through JMS. Click **Add**, **Edit** or **Remove** to modify the properties list.

**WebSphere® MQ**

Select **MQ** to use the IBM® WebSphere® MQ transport for the request. This page enables you to specify the SOAP action and override the settings for the WebSphere® MQ configuration selected at the test level.

**Protocol configuration**

Click **Change** to specify a predefined transport configuration or to create a configuration. WebSphere® MQ transport configurations contain generic queue, header, and SSL settings that can be reused.

**SOAP Action**

Specifies the SOAP action to be used to invoke the WebSphere® MQ request.

**Override MQ protocol configuration values**

Select this option to configure the fields of the WebSphere® MQ message. You can replace a subset of an MQ message descriptor with a custom format for use with other server types, specifically when using an XML message request.

**Customize message header**

Select this option to specify custom headers for the transport for the SOAP over MQ feature that is provided by WebSphere® MQ. This feature uses a predetermined MQ message format (RFH2), therefore, when selected, other **Message Descriptor** options are disabled.

**Message descriptor**

These settings replace the message descriptor and header settings of the MQ protocol configuration. Refer to WebSphere® MQ documentation for information about message descriptors.

**Microsoft™ .NET**

Select **Microsoft .NET** to use the Microsoft .NET Framework transport for requests based on Windows™ Communication Foundation (WCF). This page enables you to override the settings for the Microsoft™ .NET configuration selected at the test level.

**Item**

Click **Add** to specify the name and value of the WCF actions that are required by the service. This table is automatically generated when you import a Microsoft .NET WSDL file. Refer to the Microsoft™ .NET WCF documentation for more information.

## Request Stack

Use this page to specify the stack that applies security and addressing parameters and algorithms to service requests before they are sent. Stacks are a set of algorithms that are executed in a given order. Use the WSDL security editor to define a stack for each WSDL. The stack will be applied to all requests that use the WSDL.

### Override stack

By default, you edit a stack which attached to a specific WSDL file in the WSDL security editor. Select this option to specify a different security algorithm stack only for the current service request.

### Show response stack

The **Request Stack** page contains algorithms that are applied only to outgoing service requests. Select **Show response stack** to add a **Response Stack** page. The **Response Stack** page allows you to edit security and addressing parameters and algorithms that are applied to incoming responses.

### Security Algorithm Details

Click **Add**, **Insert**, or **Remove** to add or remove security algorithms in the stack. Click **Up** and **Down** to change the order of a selected algorithm in the security stack. The following security algorithms can be added to the security stack:

#### Static XML Headers

Use this algorithm to add the XML headers that are required for web service standard calls. On the **Header** bar, click **Add** (➕) to create the default XML header structure for WS-Addressing, WS-ReliableMessaging or WS-Coordination requests, or click **More** for other standards.

You can enable or disable XML elements in the Header section and specify the correct values for each XML element. Checks are performed to ensure that the XML headers are valid.

> 📝 **Note:** The Static XML Headers algorithm is available only in IBM® Security AppScan®. To add static XML headers to calls in other products, expand the **Headers** section on the **Message** tab of the request.

#### Time Stamp

The time stamp security algorithm adds time stamp information to the XML document in the response. For details on security algorithms, refer to the web service security specification.

##### Actor / Role name

Specify the name of the recipient of the algorithm header element, if required.

**Must understand**

Select whether it is mandatory that the algorithm header is processed by the recipient, if required. The recipient is either the Actor name or the server.

**Expiration delay**

Specify the delay after which the time stamp expires.

**Millisecond precision**

Select this option to produce a time stamp that uses millisecond precision instead of the default (1/100th second).

**User name token**

The user name token security algorithm adds a user name token to the XML document in the message. For details on security algorithms, refer to the web service security specification.

**Actor / Role name**

Specify the name of the recipient of the algorithm header element, if required.

**Must understand**

Select whether it is mandatory that the algorithm header is processed by the recipient, if required. The recipient is either the Actor name or the server.

**Name**

Type the name of the user.

**Password**

Type the password of the user.

**Password type**

Specify the password type for the security algorithm as defined in the Web Services Security UsernameToken profile.

**Use nonce**

Select this check box to add the Nonce element to the User Name Token XML code. In most cases, the Nonce ID is required.

**Use created**

Select this check box to add current timestamp to the Created XML element in the User Name Token XML.

**XML Encryption**

The XML encryption security algorithm specifies how the XML document is encrypted. For details on security algorithms, refer to the web service security specification.

**Actor / Role name**

Specify the name of the recipient of the algorithm header element, if required.

**Must understand**

Select whether it is mandatory that the algorithm header is processed by the recipient, if required. The recipient is either the Actor name or the server.

**Identifier type**

Select the type of key identifier to be used for the encryption. The following key identifiers are available, as defined in the Web Services Security (WSS) specification X509 profile and the OASIS WSS 1.1 specification:

- ISSUER_SERIAL
- BST_DIRECT_REFERENCE
- X509_KEY_IDENTIFIER
- SKI_KEY_IDENTIFIER
- EMBEDDED_KEYNAME
- THUMBPRINT_IDENTIFIER
- ENCRYPTED_KEY_SHA1_IDENTIFIER

**User XPath part selection**

This enables you to specify an XPath query that describes parts of the XML document that can be subjects of the algorithm. By default, the body is the subject.

**Key**

Select the key used for the encryption. The details of each key vary.

- **x509 key**: This specifies the name and password of the x509 key and the keystore where it is located.
- **Raw key**: This specifies the name and the byte value of your SecretKey in hexadecimal.
- **Encrypted key**: This specifies a reference to an encrypted key that was previously defined in the security stack. Click **Insert a new encrypted key** to create a new encrypted key definition block.

**Encoding Algorithm Name**

Specify the encryption method to be used as defined in the XML Encryption Syntax and Processing specification.

**Key Encoding Algorithm**

Specify the standard algorithm for encoding the key as defined in the XML Encryption Syntax and Processing specification.

## XML Signature

The XML signature security algorithm specifies how the XML document is signed. For details on security algorithms, refer to the web service security specification.

**Actor / Role name**

Specify the name of the recipient of the algorithm header element, if required.

**Must understand**

Select whether it is mandatory that the algorithm header is processed by the recipient, if required. The recipient is either the Actor name or the server.

**Security token**

Select the type of key identifier to be used for the signature. The following key identifiers are available, as defined in the the Web Service Security (WSS) specification X509 profile and OASIS WSS 1.1 specification:

- ISSUER_SERIAL
- BST_DIRECT_REFERENCE
- X509_KEY_IDENTIFIER
- SKI_KEY_IDENTIFIER
- KEY_VALUE
- USER_NAME_TOKEN
- CUSTOM_SYMM_SIGNATURE

In addition, the following identifiers are available when the signature is based on a UsernameToken profile:

- USER_NAME_TOKEN
- CUSTOM_SYMM_SIGNATURE

**User XPath part selection**

Specify an XPath query that describes parts of the XML document that can be the subjects of the algorithm. By default, the body is the subject. Click the **XPath Helper** button to build the Xpath expression.

**Key**

Select the key used for the encryption. The details of each key vary.

- **x509 key**: This key specifies the name and password of the x509 key and the keystore where it is located.
- **User name token key**: This specifies a user name and password for the signature.
- **Encrypted key**: This specifies a reference to an encrypted key that was previously defined in the security stack. Click **Insert a new encrypted key** to create a new encrypted key definition block.

**Signature algorithm name**

Specify the signature method algorithm as described in the XML Signature Syntax and Processing specification.

**Canonicalization**

Specify the canonicalization method to be used as described in the XML Signature Syntax and Processing specification.

**Digest algorithm method**

Specify which digest method to be used based on the algorithm method used on the server side.

**Inclusive namespaces**

Specify whether the canonicalization is exclusive as described in the Exclusive XML Canonicalization specification.

**Encrypted Key**

This block defines an encrypted key that can be used in an XML signature or XML encryption block. The encrypted key block must be before a block that uses the encrypted key.

**Actor / Role name**

Specify the name of the recipient of the algorithm header element, if required.

**Must understand**

Select whether it is mandatory that the algorithm header is processed by the recipient, if required. The recipient is either the Actor name or the server.

**Key name**

Specify the name of the encrypted key.

**Identifier type**

Select the type of key identifier to be used for the key. The following key identifiers are available, as defined in the the Web Service Security (WSS) specification X509 profile and OASIS WSS 1.1 specification:

- ISSUER_SERIAL
- BST_DIRECT_REFERENCE
- X509_KEY_IDENTIFIER
- THUMBPRINT_IDENTIFIER
- SKI_KEY_IDENTIFIER

**Key size**

Specify the size of the key in bits.

**Key encoding algorithm name**

Specify the algorithm to be used for encoding the key.

**Keystore**

Select a keystore or click **Edit Security** to define a new keystore or to manage the existing keystores.

**Name**

Select a key contained in the specified keystore.

**Password**

Type the password for the selected key name.

**Custom Security Algorithm**

If you want to use a Java™ class as a custom security algorithm, then use this stack element to apply the custom algorithm to the service.

**Java™ Project**

If you have not implemented a custom Java™ class, select **Java Project**, type a name for the new project, and click **Generate** to create a new Java™ class with the default structure for custom security implementations.

> ✏️ **Note:** If you are using IBM® Security AppScan®, this field is not available.

**Implementation class**

Specify the name of the class that implements the custom security algorithm. Click **Browse Class** to select an existing Java™ class from the workspace.

**Properties**

Use this table to send any specific properties and associated values to the custom security algorithm.

**WS-Addressing Algorithm**

Use this block if your service uses either WS-Addressing 2004/08 or the WS-Addressing 1.0 Core standard.

### Namespace

Specify the namespace for either WS-Addressing 2004/08 or WS-Addressing 1.0 Core.

### Action if request uses WS-Addressing

Select the action to complete if WS-Addressing is already in the request.

### Replace anonymous address in Reply-to with:

Select this option to generate the specified address in the Reply-to header instead of an anonymous address.

### Remove WS-Addressing from response

Select this option to strip any WS-Addressing headers from the response.

**WS-Policy Algorithm**

Use this block if your service requires a security policy file compliant with the WS-Policy specification.

### Use policy included in WSDL (WS-PolicyAttachment)

Select this option to use the security policy configuration that is attached to the WSDL as in the WS-PolicyAttachment specification.

### Policy

If you are not using the WS-PolicyAttachment specification, specify the XML policy file. Click **Browse** to add a policy file from the workspace or to import a policy file.

### Signature configuration

Select this option to specify a keystore for any signature that is specified in the policy. Click **Edit Security** to add a keystore from the workspace or to import a keystore.

### Encryption configuration

Select this option to specify a keystore for any encryption that is specified in the policy. Click **Edit Security** to add a keystore from the workspace or to import a keystore.

### Decryption configuration

Select this option to specify a keystore for any decryption that is specified in the policy. Click **Edit Security** to add a keystore from the workspace or to import a keystore.

**Retrieve token from security token server (WS-Trust and WS-SecureConversation)**

Select this option, and click **Configure** to specify a Security Token Server
(STS) to use with the policy.

**Additional properties**

Use this table to specify settings for the advanced properties or specific
implementations of the WS-Security specification. Click **Add** to add a
property name and to set a value.

## Response Stack

Use this page to specify the stack that applies security and addressing parameters to responses after they are
received. Stacks are a set of algorithms that are executed in a given order. Use the WSDL security editor to define a
stack for each WSDL. The stack will be applied to all requests that use the WSDL.

**Override stack**

By default, you edit the security algorithm stack attached to a specific WSDL file in the WSDL Security
Editor. Select this option to specify a different security algorithm stack only for the current response.

**Show response stack**

Clear the **Show response stack** option to hide the **Response Stack** page.

**Security Algorithm Details**

Click **Add**, **Insert**, or **Remove** to add or remove security algorithms in the stack. Click **Up** and **Down** to
change the order of a selected algorithm in the security stack. The following security algorithms can be
added to the security stack:

**XML Encryption**

The XML encryption security algorithm specifies how the XML document is encrypted. For
details on security algorithms, refer to the web service security specification.

**Actor / Role name**

Specify the name of the recipient of the algorithm header element, if
required.

**Must understand**

Select whether it is mandatory that the algorithm header is processed by the
recipient, if required. The recipient is either the Actor name or the server.

**Identifier type**

Select the type of key identifier to be used for the encryption. The following
key identifiers are available, as defined in the Web Services Security (WSS)
specification X509 profile and the OASIS WSS 1.1 specification:

- ISSUER_SERIAL
- BST_DIRECT_REFERENCE

- X509_KEY_IDENTIFIER
- SKI_KEY_IDENTIFIER
- EMBEDDED_KEYNAME
- THUMBPRINT_IDENTIFIER
- ENCRYPTED_KEY_SHA1_IDENTIFIER

**User XPath part selection**

This enables you to specify an XPath query that describes parts of the XML document that can be subjects of the algorithm. By default, the body is the subject.

**Key**

Select the key used for the encryption. The details of each key vary.

- **x509 key**: This specifies the name and password of the x509 key and the keystore where it is located.
- **Raw key**: This specifies the name and the byte value of your SecretKey in hexadecimal.
- **Encrypted key**: This specifies a reference to an encrypted key that was previously defined in the security stack. Click **Insert a new encrypted key** to create a new encrypted key definition block.

**Encoding Algorithm Name**

Specify the encryption method to be used as defined in the XML Encryption Syntax and Processing specification.

**Key Encoding Algorithm**

Specify the standard algorithm for encoding the key as defined in the XML Encryption Syntax and Processing specification.

**Encrypted Key**

This block defines an encrypted key that can be used in an XML signature or XML encryption block. The encrypted key block must be before a block that uses the encrypted key.

**Actor / Role name**

Specify the name of the recipient of the algorithm header element, if required.

**Must understand**

Select whether it is mandatory that the algorithm header is processed by the recipient, if required. The recipient is either the Actor name or the server.

**Key name**

Specify the name of the encrypted key.

**Identifier type**

Select the type of key identifier to be used for the key. The following key identifiers are available, as defined in the the Web Service Security (WSS) specification X509 profile and OASIS WSS 1.1 specification:

- ISSUER_SERIAL
- BST_DIRECT_REFERENCE
- X509_KEY_IDENTIFIER
- THUMBPRINT_IDENTIFIER
- SKI_KEY_IDENTIFIER

**Key size**

Specify the size of the key in bits.

**Key encoding algorithm name**

Specify the algorithm to be used for encoding the key.

**Keystore**

Select a keystore or click **Edit Security** to define a new keystore or to manage the existing keystores.

**Name**

Select a key contained in the specified keystore.

**Password**

Type the password for the selected key name.

**Custom Security Algorithm**

If you want to use a Java™ class as a custom security algorithm, then use this stack element to apply the custom algorithm to the service.

**Java™ Project**

If you have not implemented a custom Java™ class, select **Java Project**, type a name for the new project, and click **Generate** to create a new Java™ class with the default structure for custom security implementations.

**Note:** If you are using IBM® Security AppScan®, this field is not available.

**Implementation class**

Specify the name of the class that implements the custom security algorithm. Click **Browse Class** to select an existing Java™ class from the workspace.

**Properties**

Use this table to send any specific properties and associated values to the custom security algorithm.

## Generic service client binary call details

In the generic service client, binary calls are specialized calls for sending binary messages. The transport information refers to the information that is required to send and receive and answer depending on the selected protocol.

**Update node name automatically**

Select this option to automatically rename the request in the Test Contents view.

**Do not wait for response**

Select this option to skip directly to the next request in the test after the current request is sent.

**Time Out (ms)**

This is the timeout value in milliseconds. If no response is received after the specified time, an error is produced.

**Think Time (ms)**

This specifies the programmatically calculated time delay that is observed for each user when this test is run with multiple virtual users. Think time is a statistical emulation of the amount of time actual users spend reading or thinking before performing an action.

**Update Response**

Click this button to invoke the request with the current settings and to use the response to create a binary response element or to update the existing response element.

**Message**

**Source**

This page presents the binary contents of the request and provides access to data correlation. The same contents are presented in Binary and Raw ASCII views.

**Attachments**

This page lists the MIME or DIME attachments that are attached to the request. The contents of this view conform to the Multipurpose Internet Mail Extensions (MIME) or Direct Internet Message

Encapsulation (DIME) specification. You can use this page to add workbench resources as MIME or DIME attachments and change properties.

**Transport**

This page covers the transport protocol used to send the request. The transport protocol can be either HTTP, Java™ Message Service (JMS), or WebSphere® MQ. You can create several configurations for each protocol so that you can easily switch protocols or variants of protocols.

> **Note:** If you are using IBM® Security AppScan®,only the HTTP transport protocol is available.

## Attachments

This page lists the MIME or DIME attachments that are attached to the request. The contents of this view conform to the Multipurpose Internet Mail Extensions (MIME) or Direct Internet Message Encapsulation (DIME) specification. You can use this page to add workbench resources as MIME or DIME attachments and change properties.

The **Content ID** is the identifier that the request uses to refer to the attachments. The method for using this identifier depends on your server requirements.

**MIME or DIME**

Select whether the attachment conforms to the Multipurpose Internet Mail Extensions (MIME) or Direct Internet Message Encapsulation (DIME) specification

**Use MTOM transmission mechanism**

By default, the request uses SOAP Messages with Attachments (SwA) to handle attachments. Select this option to handle attachments with the SOAP Message Transmission Optimization Mechanism (MTOM).

## Generic service client text call details

In the generic service client, text calls are specialized calls for sending text messages. The transport information refers to the information that is required to send and receive and answer depending on the selected protocol.

**Update node name automatically**

Select this option to automatically rename the request in the Test Contents view.

**Do not wait for response**

Select this option to skip directly to the next request in the test after the current request is sent.

**Time Out (ms)**

This is the timeout value in milliseconds. If no response is received after the specified time, an error is produced.

**Think Time (ms)**

This specifies the programmatically calculated time delay that is observed for each user when this test is run with multiple virtual users. Think time is a statistical emulation of the amount of time actual users spend reading or thinking before performing an action.

**Update Response**

Click this button to invoke the request with the current settings and to use the response to create a binary response element or to update the existing response element.

## Message

### Source

This page presents the binary contents of the request and provides access to data correlation. The same contents are presented in Binary and Raw ASCII views.

### Attachments

This page lists the MIME or DIME attachments that are attached to the request. The contents of this view conform to the Multipurpose Internet Mail Extensions (MIME) or Direct Internet Message Encapsulation (DIME) specification. You can use this page to add workbench resources as MIME or DIME attachments and change properties.

### Transport

This page covers the transport protocol used to send the request. The transport protocol can be either HTTP, Java™ Message Service (JMS), or WebSphere® MQ. You can create several configurations for each protocol so that you can easily switch protocols or variants of protocols.

> **Note:** If you are using IBM® Security AppScan®,only the HTTP transport protocol is available.

## Attachments

This page lists the MIME or DIME attachments that are attached to the request. The contents of this view conform to the Multipurpose Internet Mail Extensions (MIME) or Direct Internet Message Encapsulation (DIME) specification. You can use this page to add workbench resources as MIME or DIME attachments and change properties.

The **Content ID** is the identifier that the request uses to refer to the attachments. The method for using this identifier depends on your server requirements.

### MIME or DIME

Select whether the attachment conforms to the Multipurpose Internet Mail Extensions (MIME) or Direct Internet Message Encapsulation (DIME) specification

**Use MTOM transmission mechanism**

By default, the request uses SOAP Messages with Attachments (SwA) to handle attachments. Select this option to handle attachments with the SOAP Message Transmission Optimization Mechanism (MTOM).

## Transport

This page covers the transport settings used to send the request. The transport protocol settings apply to a transport configuration, which can be either HTTP, Java™ Message Service (JMS), WebSphere® MQ, or Microsoft .NET. You can create several configurations for each protocol so that you can easily switch protocols or variants of protocols.

**Note:** If you are using IBM® Security AppScan®,only the HTTP transport protocol is available.

**HTTP**

Select **HTTP** to use the HTTP transport for the request. At the request level, you can update a URL or SOAP action and the reference to the global configuration of a test.

**Protocol configuration**

Click **Change** to specify a predefined transport configuration or to create a configuration. HTTP transport configurations contain proxy and authentication settings that can be reused.

**URL**

Specify the URL end point of the service request.

**Rest mode**

Use this check box to split the REST URL so that it is easy to understand the different parts of REST URL. When you use this option, the main section of URL is placed in the URL field, the resource part is placed in the **Resource** field, and the parameters are placed in the **Parameters** field. Use the **Add** button to manually add more parameters.

**Method**

Specify the HTTP method to be used to invoke the service request.

**Headers**

Specify the names and values of any custom HTTP headers that are required by the service. Click **Add**, **Edit** or **Remove** to modify the headers list.

**Cookies**

Specify the names and values of any cookies that are required by the service. Click **Add**, **Edit** or **Remove** to modify the cookies list.

**JMS**

Select **JMS** to use the Java™ Messaging Service transport for the request. This page enables you to add string properties that are attached to the request for a JMS configuration. These will be sent as message properties through JMS.

**Protocol configuration**

Click **Change** to specify a predefined transport configuration or to create a configuration. JMS transport configurations contain generic end point, reception point, and adapter settings that can be reused.

**Properties**

Specify the names and values of any string properties that are required by the request for the current JMS transport configuration. These are sent as message properties through JMS. Click **Add**, **Edit** or **Remove** to modify the properties list.

**WebSphere® MQ**

Select **MQ** to use the IBM® WebSphere® MQ transport for the request. This page enables you to specify the SOAP action and override the settings for the WebSphere® MQ configuration selected at the test level.

**Protocol configuration**

Click **Change** to specify a predefined transport configuration or to create a configuration. WebSphere® MQ transport configurations contain generic queue, header, and SSL settings that can be reused.

**SOAP Action**

Specifies the SOAP action to be used to invoke the WebSphere® MQ request.

**Override MQ protocol configuration values**

Select this option to configure the fields of the WebSphere® MQ message. You can replace a subset of an MQ message descriptor with a custom format for use with other server types, specifically when using an XML message request.

**Customize message header**

Select this option to specify custom headers for the transport for the SOAP over MQ feature that is provided by WebSphere® MQ. This feature uses a predetermined MQ message format (RFH2), therefore, when selected, other **Message Descriptor** options are disabled.

**Message descriptor**

These settings replace the message descriptor and header settings of the MQ protocol configuration. Refer to WebSphere® MQ documentation for information about message descriptors.

**Microsoft™ .NET**

Select **Microsoft .NET** to use the Microsoft .NET Framework transport for requests based on Windows™ Communication Foundation (WCF). This page enables you to override the settings for the Microsoft™ .NET configuration selected at the test level.

**Item**

Click **Add** to specify the name and value of the WCF actions that are required by the service. This table is automatically generated when you import a Microsoft .NET WSDL file. Refer to the Microsoft™ .NET WCF documentation for more information.

## Generic service client message return details

In the generic service client, message returns are generated after a service call is successfully invoked. Message returns display the content returned by the service.

### Message

This page shows the XML content of the request and provides access to data correlation. The same content is presented in three different ways.

**Form**

This view provides a simplified view of the message that focuses on editing the values of the XML content. Use the **Schema** menu to enable assistance with editing XML content so that the XML is valid and complies with the XSD specification.

In the **Form** view, add the XML headers that are required for standard web service calls. On the **Header** bar, click **Add** (➕) to create the default XML header structure for WS-Addressing, WS-ReliableMessaging or WS-Coordination requests, or click **More** for other standards. You can enable or disable XML header elements and specify the correct values for each XML element. Checks are performed to ensure that the XML content is valid.

> 🖊 **Note:** To add XML headers to calls in IBM® Security AppScan®, add a **Static XML Headers** algorithm on the **Request Stack** tab of the request.

**Tree**

This view provides a hierarchical view of the XML structure of the message, including elements, namespaces, and the associated values. You can use **Add**, **Insert**, **Remove**, **Up**, and **Down** to edit the XML elements and namespaces in the tree.

Use **Skip if Empty** column to select the empty XML elements that you want to skip. This column is visible only if you selected the **Display the 'Skip if Empty' column in XML tree viewer** check box in **Window > Preferences > Test > Test editor > Service test**.

Click **Filter** to hide or show namespace, attribute, or text nodes, depending on your requirements.

Click **Allow only valid modifications** to enable smart editing, based on a specified XML schema document (XSD). To specify a set of XSD documents for the workbench, in the test navigator, right-click the project and select **Properties** and **Schema Catalog**. Disable **Allow only valid modifications** if you do not have an XSD or if you want to bypass the schema.

You can right-click an XML element to convert it to an XML fragment. This enables you to perform data correlation (use datasets and create references) on the entire XML fragment instead of only on the value.

**Source**

This view displays the source XML content of the message or plain text content. To format XML content, click **Format XML text**. To wrap XML content into a single line, click **Pack XML text to single line**. Similar controls are available for JSON content.

> ⚠️ **Important:** In the Source view, do not edit the tags that start with `SoaTag`. If you delete or change these tags, any references and substitutions in the test will be broken. You cannot recreate these tags after you delete them.

## Attachments

This page lists the MIME or DIME attachments that are attached to the request. The contents of this view conform to the Multipurpose Internet Mail Extensions (MIME) or Direct Internet Message Encapsulation (DIME) specification. You can use this page to add workbench resources as MIME or DIME attachments and change properties.

The **Content ID** is the identifier that the request uses to refer to the attachments. The method for using this identifier depends on your server requirements.

**MIME or DIME**

Select whether the attachment conforms to the Multipurpose Internet Mail Extensions (MIME) or Direct Internet Message Encapsulation (DIME) specification

**Use MTOM transmission mechanism**

By default, the request uses SOAP Messages with Attachments (SwA) to handle attachments. Select this option to handle attachments with the SOAP Message Transmission Optimization Mechanism (MTOM).

### Response Properties

This page lists the names and values of properties of the response.

# WSDL security editor reference

With the Web Service Description Language (WSDL) security editor you can create and edit security configurations for a WSDL file.

## Keystores

In this page, you can edit the keystores that are used for the WSDL file. The keystore contains the public and private keys that are required for the specified security protocol.

### Defined Keystores

Click Add or Remove to add or remove keystore files from the workbench.

### Keystore Details

This specifies the location and file name of the selected keystore. Click **Browse** to select a different file.

#### Name

This specifies the name of the keystore. This name is used throughout the test instead of the file name.

#### File

Click **Browse** to specify a keystore file containing a valid server certificate. The following formats are supported:

- KS
- JKS
- JCEKS
- PKCS12 (p12 or PFX)
- PEM

#### Password

If the keystore file is encrypted, type the required password.

## Security Stacks

In this page you can edit the security algorithm stacks that the security protocol uses. Security stacks are a set of algorithms that are executed in a given order.

### Security Stacks

Click **Add**, **Remove**, or **Rename** to add, remove, or rename the security stacks that are associated with the WSDL file.

### Security Algorithm Details

Click **Add**, **Insert**, or **Remove** to add or remove security algorithms in the stack. Click **Up** and **Down** to change the order of a selected algorithm in the security stack. The following security algorithms can be added to the security stack:

#### Time Stamp

The time stamp security algorithm adds time stamp information to the XML document in the response. For details on security algorithms, refer to the web service security specification.

**Actor / Role name**

Specify the name of the recipient of the algorithm header element, if required.

**Must understand**

Select whether it is mandatory that the algorithm header is processed by the recipient, if required. The recipient is either the Actor name or the server.

**Expiration delay**

Specify the delay after which the time stamp expires.

**Millisecond precision**

Select this option to produce a time stamp that uses millisecond precision instead of the default (1/100th second).

**User name token**

The user name token security algorithm adds a user name token to the XML document in the message. For details on security algorithms, refer to the web service security specification.

**Actor / Role name**

Specify the name of the recipient of the algorithm header element, if required.

**Must understand**

Select whether it is mandatory that the algorithm header is processed by the recipient, if required. The recipient is either the Actor name or the server.

**Name**

Type the name of the user.

**Password**

Type the password of the user.

**Password type**

Specify the password type for the security algorithm as defined in the Web Services Security UsernameToken profile.

**Use nonce**

Select this check box to add the Nonce element to the User Name Token XML code. In most cases, the Nonce ID is required.

**Use created**

Select this check box to add current timestamp to the Created XML element in the User Name Token XML.

**XML Encryption**

The XML encryption security algorithm specifies how the XML document is encrypted. For details on security algorithms, refer to the web service security specification.

**Actor / Role name**

Specify the name of the recipient of the algorithm header element, if required.

**Must understand**

Select whether it is mandatory that the algorithm header is processed by the recipient, if required. The recipient is either the Actor name or the server.

**Identifier type**

Select the type of key identifier to be used for the encryption. The following key identifiers are available, as defined in the Web Services Security (WSS) specification X509 profile and the OASIS WSS 1.1 specification:

- ISSUER_SERIAL
- BST_DIRECT_REFERENCE
- X509_KEY_IDENTIFIER
- SKI_KEY_IDENTIFIER
- EMBEDDED_KEYNAME
- THUMBPRINT_IDENTIFIER
- ENCRYPTED_KEY_SHA1_IDENTIFIER

**User XPath part selection**

This enables you to specify an XPath query that describes parts of the XML document that can be subjects of the algorithm. By default, the body is the subject.

**Key**

Select the key used for the encryption. The details of each key vary.

- **x509 key**: This specifies the name and password of the x509 key and the keystore where it is located.
- **Raw key**: This specifies the name and the byte value of your SecretKey in hexadecimal.
- **Encrypted key**: This specifies a reference to an encrypted key that was previously defined in the security stack. Click **Insert a new encrypted key** to create a new encrypted key definition block.

**Encoding Algorithm Name**

Specify the encryption method to be used as defined in the XML Encryption Syntax and Processing specification.

**Key Encoding Algorithm**

Specify the standard algorithm for encoding the key as defined in the XML Encryption Syntax and Processing specification.

## XML Signature

The XML signature security algorithm specifies how the XML document is signed. For details on security algorithms, refer to the web service security specification.

**Actor / Role name**

Specify the name of the recipient of the algorithm header element, if required.

**Must understand**

Select whether it is mandatory that the algorithm header is processed by the recipient, if required. The recipient is either the Actor name or the server.

**Security token**

Select the type of key identifier to be used for the signature. The following key identifiers are available, as defined in the the Web Service Security (WSS) specification X509 profile and OASIS WSS 1.1 specification:

- ISSUER_SERIAL
- BST_DIRECT_REFERENCE
- X509_KEY_IDENTIFIER
- SKI_KEY_IDENTIFIER
- KEY_VALUE
- USER_NAME_TOKEN
- CUSTOM_SYMM_SIGNATURE

In addition, the following identifiers are available when the signature is based on a UsernameToken profile:

- USER_NAME_TOKEN
- CUSTOM_SYMM_SIGNATURE

**User XPath part selection**

Specify an XPath query that describes parts of the XML document that can be the subjects of the algorithm. By default, the body is the subject. Click the **XPath Helper** button to build the Xpath expression.

**Key**

Select the key used for the encryption. The details of each key vary.

- **x509 key**: This key specifies the name and password of the x509 key and the keystore where it is located.
- **User name token key**: This specifies a user name and password for the signature.
- **Encrypted key**: This specifies a reference to an encrypted key that was previously defined in the security stack. Click **Insert a new encrypted key** to create a new encrypted key definition block.

**Signature algorithm name**

Specify the signature method algorithm as described in the XML Signature Syntax and Processing specification.

**Canonicalization**

Specify the canonicalization method to be used as described in the XML Signature Syntax and Processing specification.

**Digest algorithm method**

Specify which digest method to be used based on the algorithm method used on the server side.

**Inclusive namespaces**

Specify whether the canonicalization is exclusive as described in the Exclusive XML Canonicalization specification.

**Custom Security Algorithm**

If you want to use a Java™ class as a custom security algorithm, then use this stack element to apply the custom algorithm to the service.

**Java™ Project**

If you have not implemented a custom Java™ class, select **Java Project**, type a name for the new project, and click **Generate** to create a new Java™ class with the default structure for custom security implementations.

> ✏️ **Note:** If you are using IBM® Security AppScan®, this field is not available.

**Implementation class**

Specify the name of the class that implements the custom security algorithm. Click **Browse Class** to select an existing Java™ class from the workspace.

**Properties**

Use this table to send any specific properties and associated values to the custom security algorithm.

# Security Considerations

This document describes the actions that you can take to ensure that your installation is secure, customize your security settings, and set up user access controls.

## Enabling secure communication between multiple applications

The workbench computer that controls the execution of the test communicates with the remote agent computers. The agents apply load for IBM® Rational® Performance Tester. The communication can be secure or nonsecure. By default, the tool use nonsecure communication. Also, if a workbench computer uses a self-signed certificate, it cannot be changed. Agent computers are trusting.

- Configuring port numbers for agents

## Ports, protocols, and services

The Majordomo service running on remote agents must run with administrator or super user credentials, which means that the test execution it supports has full privileges on the test computer where it resides. Product communication uses HTTP and HTTPS. Ports are configurable.

- Configuring port numbers for agents on page

## Customizing your security settings

Datasets can be encrypted and access controlled by password that is difficult, but not impossible, to break.

- Encrypted datasets overview on page

## Privacy policy considerations

This software offering does not use cookies or other technologies to collect personally identifiable information. For additional information on cookies, see the Notices on page dclviii topic.

## Security limitations

Passwords are stored using Eclipse mechanisms that are difficult but not impossible to break.

Workbench and agent communication is encrypted but not absolutely safe from impersonation attack.

# Notices

This document provides information about copyright, trademarks, terms and conditions for the product documentation.

This information was developed for products and services offered in the US.

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, MD-NC119*
*Armonk, NY 10504-1785*
*US*

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

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

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 jurisdictions 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.

Any references in this information to non-IBM® websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM® product and use of those websites is at your own risk.

IBM® may use or distribute any of the information you provide 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 Director of Licensing*
*IBM Corporation*
*North Castle Drive, MD-NC119*
*Armonk, NY 10504-1785*
*US*

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.

The performance data discussed herein is presented as derived under specific operating conditions. Actual results may vary.

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.

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 actual people or business enterprises 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,

## Trademarks

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.

## Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

### Applicability

These terms and conditions are in addition to any terms of use for the IBM® website.

### Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM®.

### Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM®.

### Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM® reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM®, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM® MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

# IBM Online Privacy Statement

IBM Software products, including software as a service solutions, (Software Offerings) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user, or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM's Privacy Policy at http://www.ibm.com/privacy and IBM's Online Privacy Statement at http://www.ibm.com/privacy/details in the section entitled Cookies, Web Beacons and Other Technologies, and the IBM Software Products and Software-as-a-Service Privacy Statement at http://www.ibm.com/software/info/product-privacy.

# Index