# HCL OneTest™ Server

10.1.3 Documentation

March 2021

# Special notice

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

# Contents

# Chapter 1. Release Notes for HCL OneTest™ Server

This document includes information about What's new, installation and upgrade instructions, known issues, and contact information of HCL Customer Support.

### Contents

## Description

HCL OneTest™ Server is a server that includes capabilities such as project and role-based security, Docker-based distribution and installation, and running of test cases. For more information about the server, see HCL OneTest Server overview on page 18.

## What's new

The following sections list the new features, enhancements or other changes made in this release.

- **Introducing Team Spaces on HCL OneTest™ Server**

  HCL OneTest™ Server is now provisioned to display an initial Team Space, which is a logical partition of HCL OneTest™ Server.

  The operations that you managed at the server level can now be performed at the Team Space level. You can now configure licenses or an SMTP server, create projects, register the remote agents, dockers, or intercepts within a Team Space. See Team Space overview on page 91.

- **System modeling**

  - You can create a system model to view the logical representation of any application under test in a Team Space. The system modeling is a Tech Preview feature.
  - You can configure a Team Space repository, which is different from the project repository, to save the system model.
  - You can create components that represent different types of assets and associate the components by defining the relationship between the components. You can also create child components for components. You can use these concepts to build a system model and view the model. You can then publish the system model to the Team Space repository.
  - You can also associate the virtual services in a project repository with components of the system model. From the associated component, you can view and run the virtual services.

  See System modeling on page 180.

- **Working with virtual services**

  You can now view the virtual service resources that are in your project repositories from the **Resources** page and then configure a run of the virtual service. Working with virtual services is a Tech Preview feature.

  After you start a virtual service, the running instances can be viewed from both the **Resources** page and **Instances** page.

  You can perform the following tasks on virtual services:

    - View information about the details, behavior, usage, routing rules, and diagnostic information of the virtual service instance.
    - Modify the behavior or the logging level of a running instance.

  See Management of virtualized services on page 319.

- **Stubs are excluded from Execution and Progress pages**

  Starting from 10.1.3, stubs are not displayed on the **Execution** page to start them nor are they displayed on the **Progress** page after they are started. Stubs must be started from the **Virtualization > Resources** page and stubs that are running can be viewed from the **Virtualization > Instances** page.

  See Management of virtualized services on page 319.

- **Running JUnit tests on HCL OneTest™ Server**

  You can now run JUnit tests that are within a Maven project from HCL OneTest™ Server.

  You must commit the Maven project as a `pom.xml` to a repository. After you add the repository to your project on HCL OneTest™ Server, you can configure a run of the JUnit tests from the **Execution** page.

  After the tests are run, you can view the detailed reports of the tests. See Configuring a JUnit test run on page 293.

- **Agents are selected automatically for an override**

  When you install static agents V10.1.2 and add the agents to your project on HCL OneTest™ Server, HCL OneTest™ Server determines and then displays the capabilities of the agents. Further, HCL OneTest™ Server compares the capabilities of the agents with the required capabilities specified for the agent in the test assets and automatically selects the most suitable agent for an override. You can either proceed with running the test on the agent selected or select an agent from the agents displayed that have the same capabilities. See Adding an agent to a project on page 220.

- **Working with capabilities that you define for static agents**

  When you install static agents V10.1.3 and register them with HCL OneTest™ Server, you can add, view, edit, or delete the capabilities of static agents. See Working with agent capabilities on page 222.

- **Refreshing of the migrated project repositories**

After you migrate a project from a previous version to a later version of HCL OneTest™ Server, the repositories that are configured for the project are refreshed only after the project owner logs in to HCL OneTest™ Server.

- **Generating the test data by using multiple schemas**

  Previously, when you used the JDBC connection to generate the test data by using HCL® OneTest™ Data, then you were able to generate the test data for only one schema. You can now generate the test data for multiple schemas simultaneously by maintaining referential integrity. Multiple schemas that you use to generate the test data now belong to a single JDBC connection.

  You can also view the relationship among the selected multiple schemas on the Schema integrity page.

  See Generating test data by using multiple schemas on page 169.
- **Execution of properties based on priority**

  When you set multiple properties for any item type, then HCL OneTest Data uses an order of priority while executing the properties. See Setting type properties on page 129.
- **Establishing a secure and trusted integration between applications and HCL OneTest Data**

  You can now establish a secure and trusted integration between applications and HCL OneTest™ Data by placing the SSL certificate in the trust store.

## Installing HCL OneTest™ Server

- For instructions about installing the software, see Installation of the server software on page 35.
- For instructions about upgrading the software, see Server software upgrade methods on page 55.

## Known issues

You can find information about the known issues identified in this release of HCL OneTest™ Server.

**Table 1. Download documents and technotes**

| Product | Download document | Knowledge Base |
|---|---|---|
| HCL OneTest™ Server | Release document | HCL support |

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 HCL support

- For technical assistance, contact HCL Customer Support.
- Before you contact HCL support, you must gather the background information that you might need to describe your problem. When you describe a problem to the HCL 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 HCL OneTest™ Server.

### Contents

## Disclaimers

This report is subject to the Terms of Use 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 HCL's current product plans and strategy, which are subject to change by HCL without notice. HCL 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 HCL or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of HCL software.

References in this report to HCL products, programs, or services do not imply that they will be available in all countries in which HCL operates. Product release dates and/or capabilities referenced in this presentation may change at any time at HCL'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. Discrepancies found between reports and other HCL 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 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 HCL OneTest™ Server.

| Hardware | Platform | Requirement | Notes |
|---|---|---|---|
| Disk space | Container | 256 GB | Additional resources are required by the cluster for the execution of test assets. |
| Memory | Container | 32 GB | Additional resources are required by the cluster for the execution of test assets. |
| Processor | Container | 8 CPUs | Additional resources are required by the cluster for the execution of test assets. |
| Network | Container | Gigabit (1000) Ethernet | |
| Other hardware | Container | | Operating system support is specified by the supported platforms: Kubernetes and OpenShift. Refer to their associated documentation for a list of supported operating systems. |

Related information

System Requirements on page 12

# Operating systems and Containers

You can find details about the supported operating systems and containers.

**Contents**

## Bit version support

Different parts of a product might run on the same operating system but support different application bitness.

For example, one part of the product might run only in 32-bit mode, whereas another might support 64-bit tolerate mode.

| Bitness | Description |
|---|---|
| 64-Tolerate | The product or part of the product runs as a 32-bit application in the 64-bit platforms listed as supported. |
| 64-Exploit | The product or part of the product runs as a 64-bit application in the 64-bit platforms listed as supported. |

### Operating systems

| Operating system | Hardware | Bitness | HCL OneTest™ Server | Notes |
|---|---|---|---|---|
| Ubuntu Server 18.04 LTS | x86-64 | 64-Exploit | ✓ | Running K3s v1.19 |
| Ubuntu Server 20.04 LTS | x86-64 | 64-Exploit | ✓ | Running K3s v1.19 |

### Containers

You can find details about the supported containers.

| Container | Runtime | Version | Notes |
|---|---|---|---|
| HCL OneTest™ Server | OpenShift Container Platform | 4.5 | The Container Platform that is supported by OpenShift V4.5.24. |

Related information

System Requirements on page 12

# Host prerequisites

You can find the prerequisites that support the operating capabilities for HCL OneTest™ Server.

### Contents

### Licensing

| License Server | Version | HCL OneTest™ Server |
|---|---|---|
| Flexnet Operations | 2017R4 | ✓ |

### Web browsers

| Browser | Version | HCL OneTest™ Server |
|---|---|---|
| Apple Safari | 13 or later | ✓ |
| Google Chrome | 78 or later | ✓ |
| Microsoft Edge | 80 or later | ✓ |
| Microsoft Edge Chromium | 86 or later | ✓ |
| Mozilla Firefox (including ESR versions) | 68 or later | ✓ |

Related information

System Requirements on page 12

## Supported software

You can find details about the additional software that are supported.

### Contents

### Development tools

| Supported software | Version | HCL OneTest™ Server |
|---|---|---|
| Git | 2.19.0 | ✓ |
| | 2.22.0 | ✓ |

### Runtime environment

| Supported software | Version | HCL OneTest™ Server | Notes |
|---|---|---|---|
| Helm CLI | 3.4 | ✓ | Required locally for Helm. |
| | 3.3.4 | ✓ | |
| Jaeger Operator | 1.20 | ✓ | Required if tests contribute trace information. The version depends on the target platform. |
| OCP CLI (oc) | 4.5 | ✓ | Required locally for OCP. |
| | 4.4 | ✓ | |
| OpenShift Container Platform (OCP) | 4.5 | ✓ | The Container Platform that is supported by OpenShift V4.5.24. |
| | 4.4 | ✓ | The Container Platform that is supported by OpenShift V4.4.3. |
| OpenShift Container Storage (OCS) | 4.5 | ✓ | The storage class must support ReadWriteMany (RWX) so that executions |
| | 4.4 | ✓ | |

| Supported software | Version | HCL OneTest™ Server | Notes |
|---|---|---|---|
| | | | can be performed on all nodes. |

Related information

System Requirements on page 12

# Chapter 3. Getting Started Guide

This guide provides an overview and describes the task flows to get you started with HCL OneTest™ Server. This guide is intended for new users.

## HCL OneTest™ Server overview

HCL OneTest™ Server brings together test data, test environments, and test runs and reports into a single, web-based browser for testers and non-testers.

HCL OneTest™ Server provides the following capabilities:

**Web-based continuous testing platform**

HCL OneTest™ Server is a web-based continuous testing platform built on modern, cloud native technologies that enables test teams to run a breadth of tests that includes API, functional, and performance tests and to benefit from a holistic view of test progress. HCL OneTest™ Server is built on Docker for easy deployment. You can scale your testing on the cloud or remote on premise systems with native Docker, Kubernetes, and IBM Red Hat OpenShift support.

**Role-based access and security**

Security is a key concern for HCL clients and therefore, HCL OneTest™ Server brings a comprehensive, role-based access control scheme to the server with project owners assigning specific member's key permissions (by using roles), for example managing test data or working with secrets such as passwords.

**Running of tests from the server by using Docker containers**

Server-based running of tests is the starting point for HCL OneTest™ Server. For members of a project with the appropriate role, HCL OneTest™ Server enables direct running of tests from the browser by using transient Docker containers.

**Connected agents for existing performance agents**

Agent owners can connect existing performance agents to the server and add them to a project for running schedules and Accelerated Functional Testing (AFT) Suites on the current infrastructure.

**Project overview statistics**

The Overview page for HCL OneTest™ Server offers you a quick, simple view on the state of testing for your projects.

**Project home page**

The home page lists your projects and other projects, which makes it easy to manage different projects within an organization.

**Reporting and the Resource Monitoring Service**

HCL OneTest™ Server provides the home for capabilities that previously were hosted on HCL® Quality Server. Reporting and the Resource Monitoring Service are in HCL OneTest™ Server and provide a more direct relationship with their related projects. These capabilities also benefit from the project level, role-based access controls. You can view unified test results to help you make informed business decisions.

**Test data authoring**

Beyond the concept of a project held in a Git repository for a simple location of tests and related assets, you can do full concurrent editing of test data sets directly from HCL OneTest™ Server. This true multiuser capability enables team members to collaborate more easily as well as try out data changes without impacting the rest of the team. When satisfied with the results, team members can push their changes. You can also fabricate sample data to perform tests by using the data generator supported in HCL OneTest™ Server.

**Integration with DevOps tools**

You can integrate HCL OneTest™ Server with various popular DevOps tools like Jenkins, UrbanCode Deploy, and Microsoft Azure DevOps to get more value from your DevOps pipelines. You can use UrbanCode Deploy to define a deployment process that automatically triggers test cases and have those test insights available directly within UrbanCode Velocity. With the Microsoft Azure DevOps integration, you can also define a DevOps pipeline that includes direct execution of tests by using the HCL OneTest™ Server scalable infrastructure. HCL OneTest™ Server also integrates with Jira for defect tracking and GitHub for software source management and version control.

## Supported versions of assets and resources

You can find information about the versions of assets, resources, agents and Dockers that are supported in HCL OneTest™ Server. You can also find information about the versions of the desktop clients that are supported.

The following table lists the versions of assets or resources created in the desktop clients along with the agents, Dockers, or proxies that are supported in HCL OneTest™ Server:

| Resource | Supported | Not supported |
|---|---|---|
| Assets or resources created in HCL OneTest™ UI | Assets or resources created in HCL OneTest™ UI which are of versions earlier or the same as HCL OneTest™ Server. | Assets or resources created in any later version of HCL OneTest™ UI that is later than the version of HCL OneTest™ Server. |
| Assets or resources created in HCL OneTest™ API | Assets or resources created in HCL OneTest™ API which are of versions earlier or the same as HCL OneTest™ Server | Assets or resources created in any later version of HCL OneTest™ API that is later than the version of HCL OneTest™ Server. |
| Assets or resources created in HCL OneTest™ Performance | Assets or resources created in HCL OneTest™ Performance which are of | Assets or resources created in any later version of HCL OneTest™ Performance that is later than the version of HCL OneTest™ Server. |

| Resource | Supported | Not supported |
|---|---|---|
| | versions earlier or the same as HCL OneTest™ Server | |
| HCL OneTest™ Performance Agents | HCL OneTest™ PerformanceAgents which is the same version as HCL OneTest™ Server | Before you install the agents that are of the current version, see Agents overview on page 217. |
| Dockers on a remote host computer | Image of HCL OneTest™ Server which is the same version as HCL OneTest™ Server that you want to use to run tests, must be installed on the Docker on a remote host computer. | Before you set up Dockers remotely, see Managing Docker hosts on page 224. |
| HTTP proxies | Installation of the proxy component from HCL® Quality Server which is the same version as HCL OneTest™ Server. | Before you use HTTP proxies, see Setting up the HTTP proxy on page 323. |

If there is a mismatch in the versions of the assets, resources, agents, or images used in Dockers with the version of HCL OneTest™ Server that you are using, any of the following events can occur:

- Warnings or Errors are displayed when you add a repository to a project in your team space.
- Warnings or Errors are displayed when you open a project that you migrated from a previous version of HCL OneTest™ Server.
- Warnings or Errors are displayed when you add or manually refresh a repository to which you added assets or resources that were created on a desktop client with a version previous to or later than the version of HCL OneTest™ Server that you want to use.
- Test runs that you start or schedule on an agent or Docker fail to run.
- You cannot view the proxies that are registered with HCL OneTest™ Server on the **Agents and Intercepts** page.

## Tests supported on HCL OneTest™ Server

You can find information about the types of tests that you can configure for a test run in a project on HCL OneTest™ Server. You can also find information about the desktop clients in which you can create the test.

The following table lists the test types that are supported and those tests that are not supported for running on HCL OneTest™ Server:

| Product | Supported test runs | Not supported test runs |
|---|---|---|
| HCL OneTest™ UI | • Accelerated Functional Testing (AFT) Suites<br>• Compound Tests that contain any of the following types of tests:<br>◦ Web UI tests<br>◦ Traditional HTML tests<br>◦ Mobile tests | Traditional non-HTML tests.<br>An independent Web UI test and a mobile test outside of a Compound Test or an AFT Suite. |
| HCL OneTest™ API | API Suites that include the following Test Suites:<br>• Test Suites that contain scenarios with references to local stubs.<br>• Test Suites that contain stubs that virtualize services run in any Kubernetes cluster or in the Kubernetes cluster that hosts HCL OneTest™ Server.<br>Stubs | Test Suites that have tests with subscribe actions that operate in the watch mode.<br>A stand-alone API test case cannot be run independently. An API test case must be part of a Test Suite. |
| HCL OneTest™ Performance | • Compound Tests that contain performance tests<br>• Rate Schedules<br>• VU Schedules | A single test script of any test extension.<br>Tests that belong to 32-bit test extensions and SOA Quality included in VU Schedule, Rate Schedule, and Compound Test. |
| Postman | Postman resources | |
| JMeter | JMeter tests | |
| JUnit | JUnit tests | |

**Note:** You must create compound tests for the same type of tests. For example, you can create a compound test that has different mobile tests that use different mobile testing platforms.

**Restriction:** You cannot merge folders that contain different test projects in the same project directory. For example, you cannot create a project directory that contains a Performance test project and an API Suite project. You must create separate project directories for each test type. For example, you must create the Performance test in the project directory as `/MyPerfProject/` and the API Suite in the project directory as `/MyAPIProject`, before you commit the projects to the remote repository.

Related information

Tests configurations and test runs

## HCL OneTest Data overview

HCL® OneTest™ Data is an automated and customizable tool to generate the test data. HCL® OneTest™ Data is one of the components of HCL OneTest™ Server and can be accessed as Data Fabrication in the HCL OneTest™ Server GUI.

You can generate the test data in various file formats such as Excel files, Native file, Comma-Separated Values (CSV), JavaScript Object Notation (JSON), and Extensible Markup Language (XML).

The test data is the core component of any application testing. Creating test data manually is very time-consuming. With HCL® OneTest™ Data generator, you can generate huge volumes of intelligent data to perform application testing without the risk of data leaks or privacy issues. The generated test data can drive the testing of any application by using HCL OneTest suite.

HCL® OneTest™ Data provides the following capabilities:

- Generates real-time data automatically.
- Provides powerful built-in API to match the data with the generated datatype.
- Supports flexibility to model the data.
- Supports both single and multi-tenancy.

HCL® OneTest™ Data provides the following benefits:

- Protects data privacy.
- Improves the efficiency and accuracy of predefined data and real-time data generation.
- Avoids extraction of real and potentially sensitive information from production system.

You can find more information about the tasks that you must perform to generate the test data by using HCL® OneTest™ Data.

## Data design environment overview

The data design environment is a graphical interface to design all its components and data generation rules to generate test data on user demand. To design the components for data generation, you must log in to HCL OneTest™ Server, and then create a project.

You use various components to structure data in HCL® OneTest™ Data. The components of data design environment are:

- Files
- Connections
- Schemas

The definitions of data stored in the schemas are used to define a map. The map specifies the transformation logic in the form of rules. The map identifies the data definition of a schema, validates the type properties of data by using rules, and compiles the rules to generate the real-time test data.

You can refer to workspace management to manage workspace in HCL® OneTest™ Data.

Related information

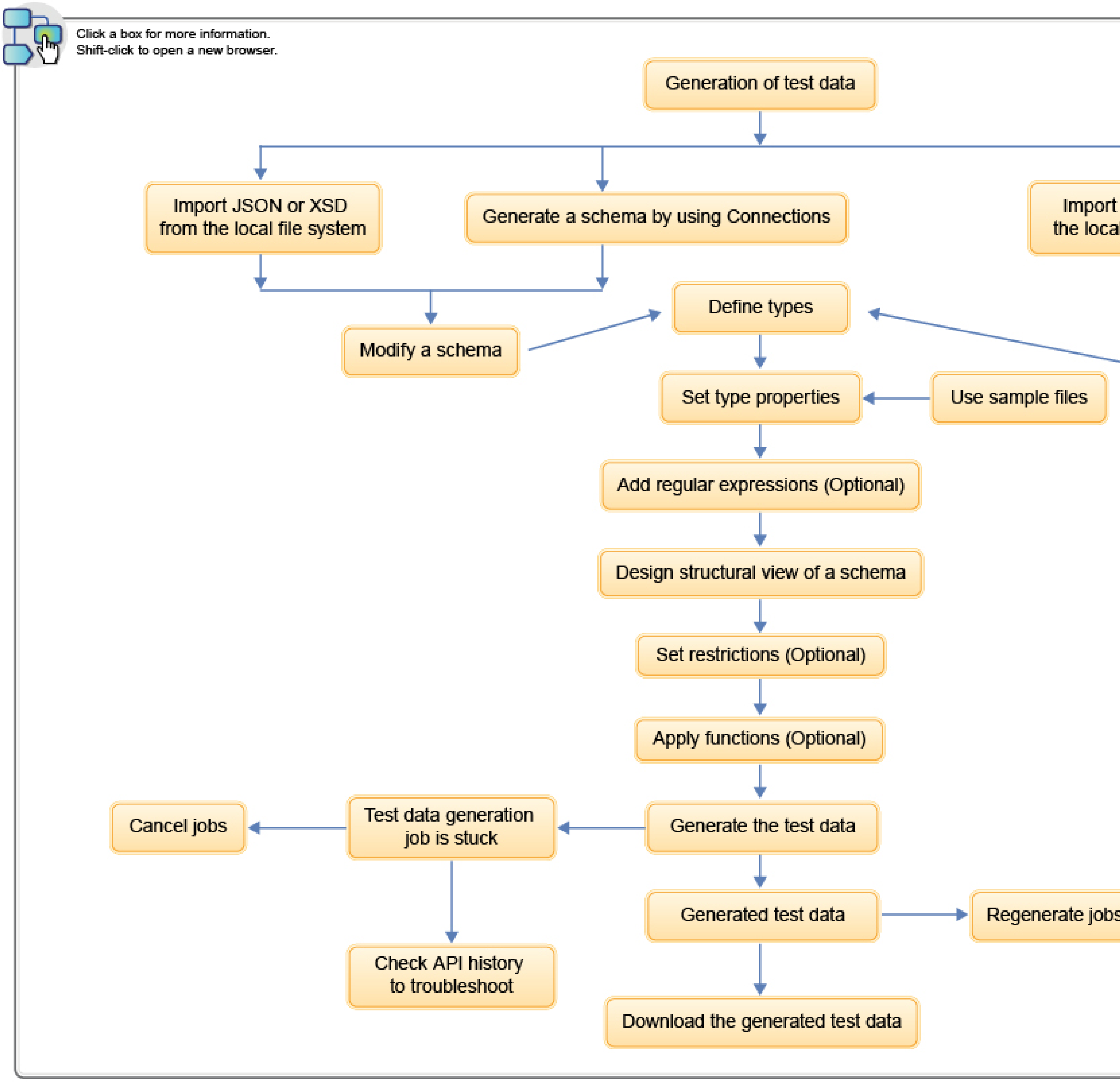
Test assets and a server project on page 496

## Task flows: HCL OneTest Data

You can use the task flow diagram to get started with HCL® OneTest™ Data. After you select your project in HCL OneTest™ Server and navigate to the **Data Fabrication** page, you can complete the tasks in sequence to generate the test data. You can click the links to get more information about the tasks.

### Generating the test data by using HCL OneTest Data

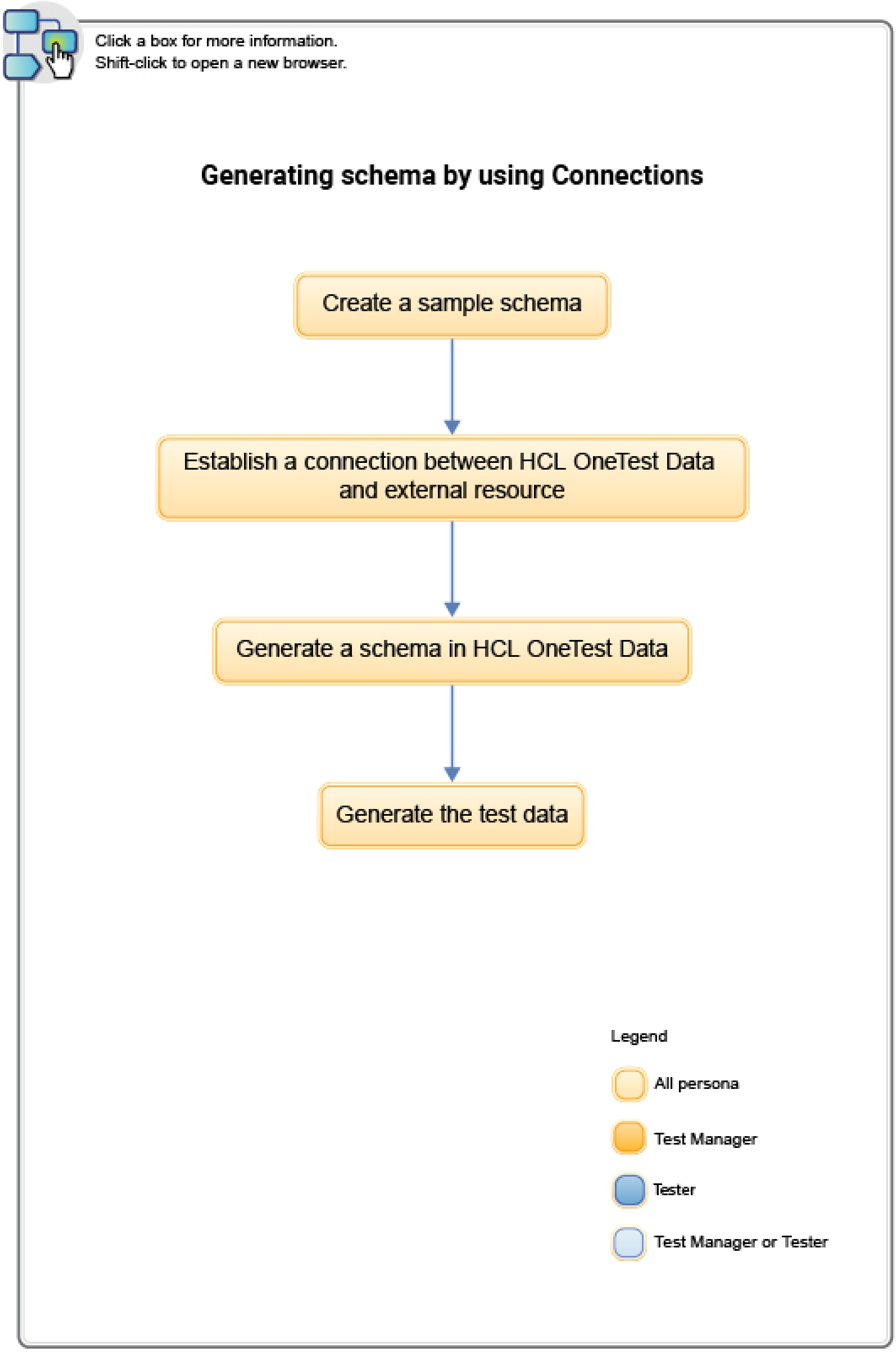
You can use the task flow diagram to help you to generate the test data.


Click a box for more information.
Shift-click to open a new browser.
Generation of test data
Import JSON or XSD from the local file system
Generate a schema by using Connections
Import the local
Modify a schema
Define types
Set type properties
Use sample files
Add regular expressions (Optional)
Design structural view of a schema
Set restrictions (Optional)
Apply functions (Optional)
Cancel jobs
Test data generation job is stuck
Generate the test data
Generated test data
Regenerate jobs
Check API history to troubleshoot
Download the generated test data

1. Generation of test data on page 22
2. Import JSON or XSD from the local file system on page 121
3. Generate a schema by using Connections on page 122
4. Import data from the local file system on page 124
5. Create a schema on page 123
6. Modify a schema on page 128
7. Design a schema on page 128
8. Define types on page 128
9. Set type properties on page 129
10. Use sample files on page 136
11. Add regular expressions on page 134

## Generating schema by using Connections

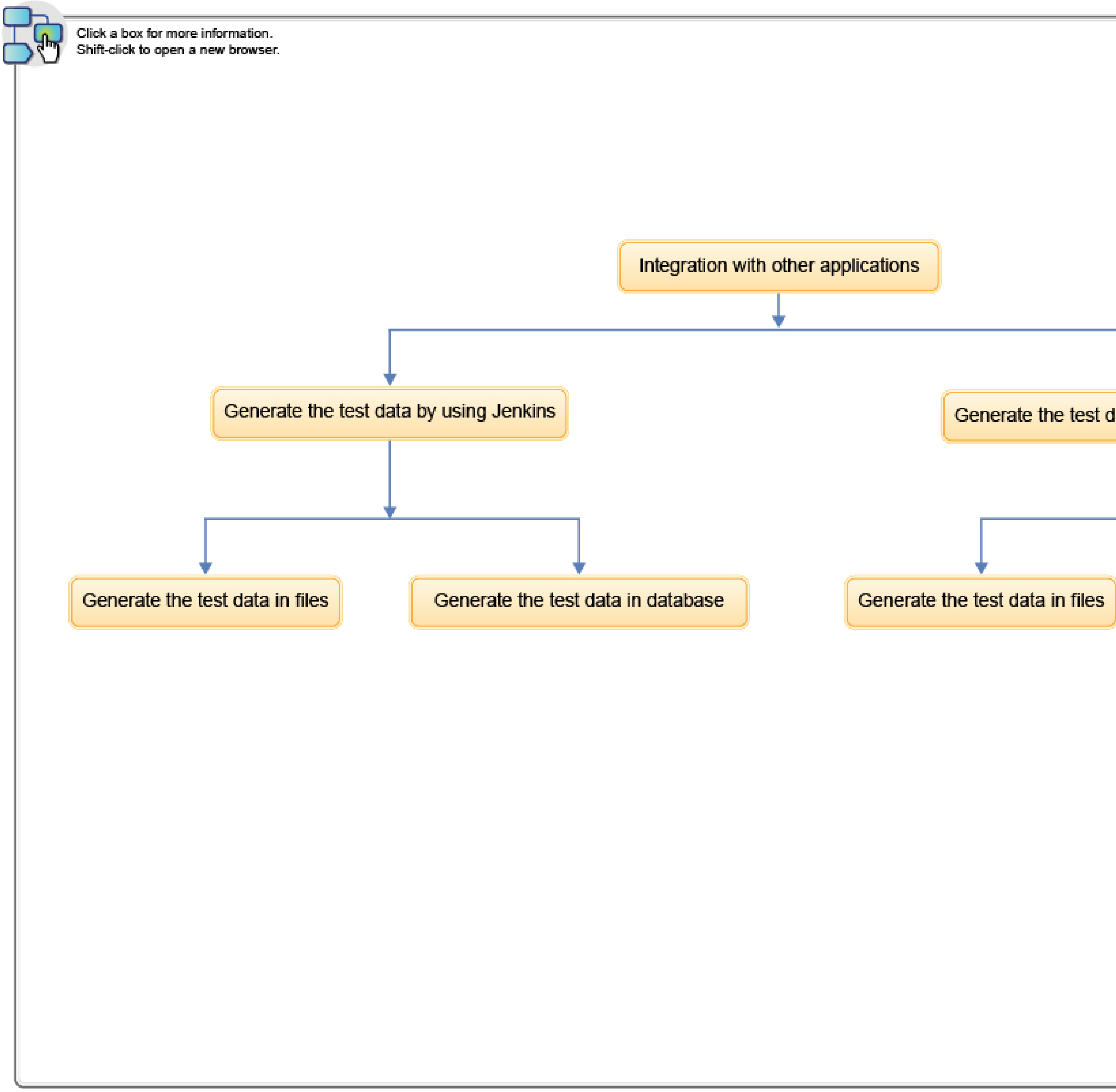
You can use the task flow diagram to generate a schema by using **Connections** in the **Data Fabrication** page.


Click a box for more information.
Shift-click to open a new browser.
Generating schema by using Connections
Create a sample schema
Establish a connection between HCL OneTest Data and external resource
Generate a schema in HCL OneTest Data
Generate the test data
Legend
All persona
Test Manager
Tester
Test Manager or Tester

1. Create a sample schema on page 147
2. Establish a connection between HCL OneTest Data and external resource on page 122
3. Generate a schema in HCL OneTest Data on page 150
4. Generate the test data on page 152

## Integrating HCL OneTest Data with other applications

You can use the task flow to integrate HCL® OneTest™ Data with other applications such as Jenkins and UrbanCode™ Deploy.


Click a box for more information.
Shift-click to open a new browser.
Integration with other applications
Generate the test data by using Jenkins
Generate the test d
Generate the test data in files
Generate the test data in database
Generate the test data in files

1. Integration with other applications on page 460
2. Generating the test data by using Jenkins on page 460
3. Generating the test data by using UrbanCode Deploy on page 466
4. Generating the test data by using Jenkins on page 460
5. Generating the test data by using Jenkins on page 460
6. Generating the test data by using UrbanCode Deploy on page 466
7. Generating the test data by using UrbanCode Deploy on page 466

## Project management

You can create a project to manage the data design environment that is used to generate the test data. If the data design environment that you created is no longer required, then you can archive or delete the project. If you do want to use an archived project, you can unarchive it.

While you manage your project, the following components of HCL® OneTest™ Data are impacted:

- Projects
- Schemas
- Jobs
- Data files
- API history
- Map files
- Files
- Connections
- Actions

If you want to modify the name of your HCL® OneTest™ Data project, then you can rename the project in HCL OneTest™ Server.

**Note:** When you delete any archived project, then that project is deleted based on the configured job settings.

Related information

Configuring job settings on page 176

## Workspace management

You can view and manage all the data design components such as files, connections, or schema of your project on the **Workspace** page.

**Quick Links** is a list of links that enables you to access the frequently used features of HCL® OneTest™ Data. You can export or import the project by using the list of links in **Quick Links**. You can also export or import a selective list of components.

## Importing a project or components

When you have a project or any component of a project into your local file system and you want to reuse it for another project, you can import those artifacts from the local file system.

**Before you begin**

- You must have a project or any component of a project in your local file system.
- The project or component that you want to import must be in a zip format.

1. From the list of links in **Quick Links**, click **Import into workspace**.

   **Result**

   The **Import** dialog box appears.
2. Select a file you want to import into your workspace and click **Import**.

**Results**

The selected file is imported successfully into the workspace.

**Note:** If the project file you selected to import already exists in the workspace, the import process fails.

## Exporting a project or components

When you want to reuse your existing project for another project, you can export either the selected components or an entire project to your local file system.

**Before you begin**

You must have a project.

1. To export the entire project, click **Export current workspace** from the list of links in **Quick Links**.

   **Result**

   The project with all its artifacts is exported to your local file system.
2. To export the selective components from a project, click **Selective export of current workspace**.

   **Result**

   The **Selective Export** dialog box appears.
3. Select schemas, files, or connections from the drop-down list. You can select multiple components to export.
4. Click **OK** to export.

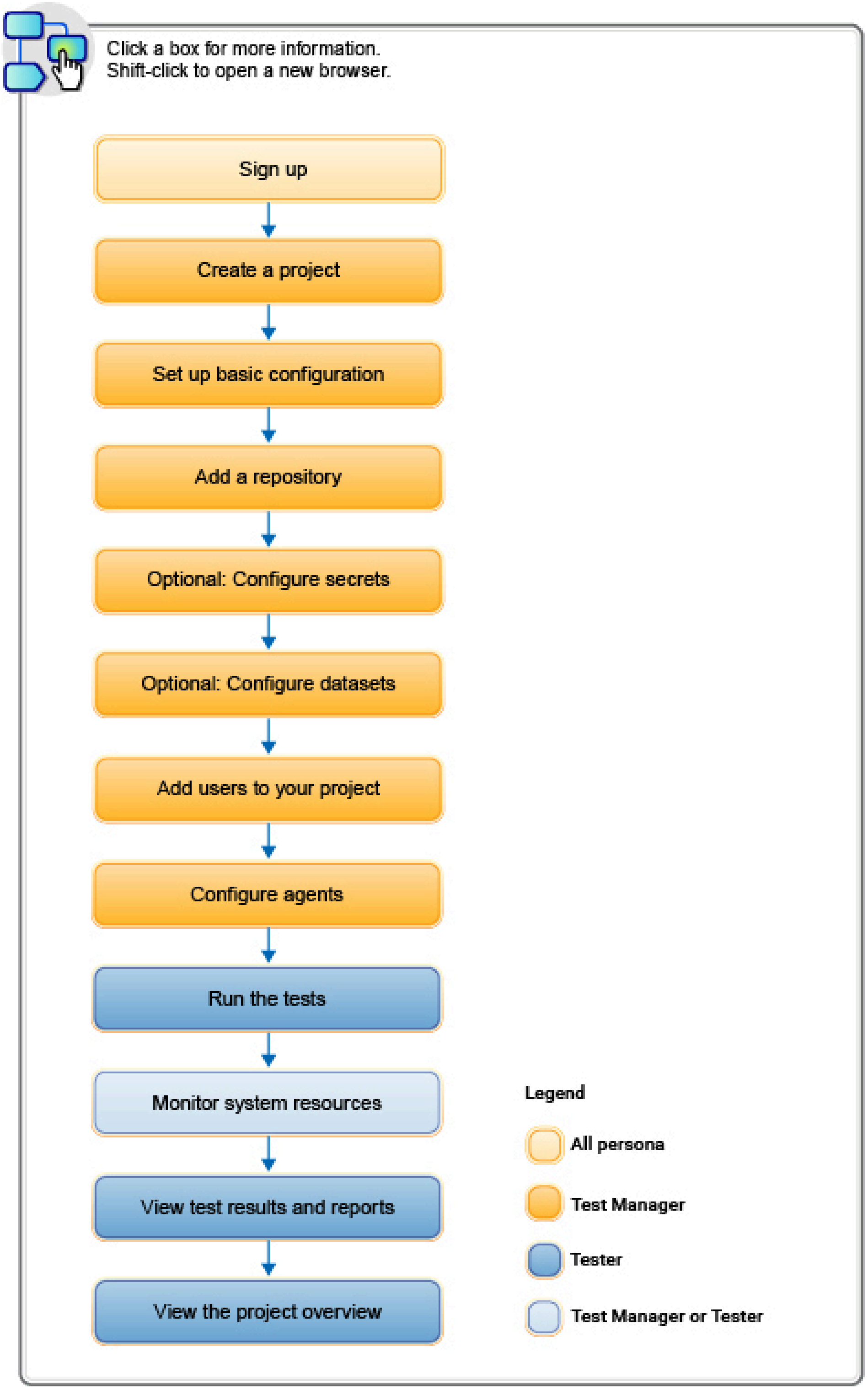
**Results**

The selected file is exported successfully from the workspace.

# Task flow: Test runs and results in HCL OneTest™ Server

You can use the task flow diagram to get started with HCL OneTest™ Server. After you install the software, you can complete the tasks in sequence to run test assets in HCL OneTest™ Server and view and analyze the test results.

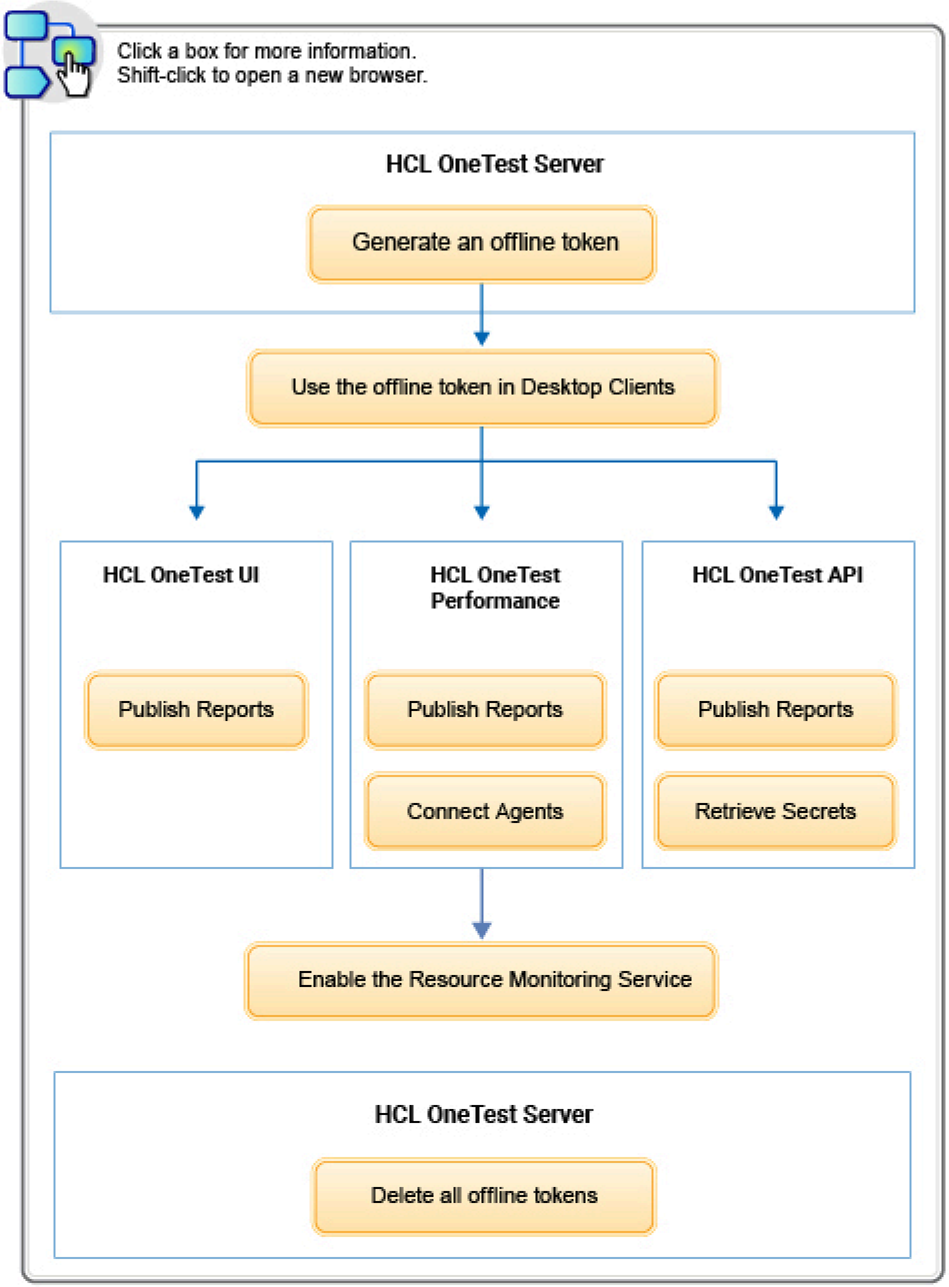
You can click the links to get more information about the tasks. The diagram also provides the tasks to be performed by personas. HCL OneTest™ Server supports user roles.


Click a box for more information.
Shift-click to open a new browser.
Sign up
Create a project
Set up basic configuration
Add a repository
Optional: Configure secrets
Optional: Configure datasets
Add users to your project
Configure agents
Run the tests
Monitor system resources
View test results and reports
View the project overview
Legend
All persona
Test Manager
Tester
Test Manager or Tester

1. Default user administration on page 68
2. Test assets and a server project on page 496
3. Adding a project on page 499
4. Adding repositories to a server project on page 499
5. Protecting API test assets by using secrets on page 512
6. Managing an encrypted dataset on page 518
7. Adding users to a server project on page 501
8. Adding an agent to a project on page 220
9. Tests configurations and test runs

# Task flow: HCL OneTest™ Server access from desktop clients

You can use the task flow diagram to help you access the server from the different desktop clients to retrieve secrets, publish reports, or enable resource monitoring agents. You can perform these tasks after you install and set up the server.


Click a box for more information.
Shift-click to open a new browser.
HCL OneTest Server
Generate an offline token
Use the offline token in Desktop Clients
HCL OneTest UI
Publish Reports
HCL OneTest Performance
Publish Reports
Connect Agents
HCL OneTest API
Publish Reports
Retrieve Secrets
Enable the Resource Monitoring Service
HCL OneTest Server
Delete all offline tokens

1. Managing access to HCL OneTest Server on page 482
2. Managing access to HCL OneTest Server on page 482
3. Test results and reports overview on page 360
4. Test results and reports overview on page 360
5. Test results and reports overview on page 360
6. Retrieving secrets
7. Managing access to HCL OneTest Server on page 482
8. Configuring HCL OneTest Performance Agent
9. Enablement of Resource Monitoring services for a schedule

Related information

Managing access to server projects on page 504

# 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 compliance

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

HCL OneTest™ Server 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.

# Chapter 4. Administrator Guide

This guide describes how to install HCL OneTest™ Server software. After you install the software, you can perform administration tasks such as license configuration, user management, security, memory, and disk usage management, back up and restore user data. This guide is intended for administrators.

## Installation of the server software

To get started with HCL OneTest™ Server, you must first install the server software.

You can install HCL OneTest™ Server on the following platforms:

- Red Hat OpenShift V4.5
- Ubuntu server (using k3s)

### Task flow for installing the sever software on Red Hat OpenShift

You must perform the following tasks to install HCL OneTest™ Server:

1. Set up a Red Hat OpenShift Container Platform.
2. Install the Jaeger operator to trace test logs and Jaeger-based reports when you run tests.
3. Install the OpenShift Container Platform command-line interface (CLI) and the Helm software.
4. Install HCL OneTest™ Server.
5. Configure and manage the license.

### Task flow for installing the sever software on Ubuntu

You must perform the following tasks to install HCL OneTest™ Server:

1. Set up a Ubuntu server.
2. Install the OpenSSH server and the Helm software.
3. Set up the k3s Kubernetes environment by using the provided script.
4. Install HCL OneTest™ Server.
5. Configure and manage the license.

## Installation of the server software on Red Hat OpenShift

You can find information about the tasks that you can perform to install HCL OneTest™ Server software on the Red Hat OpenShift platform.

## Prerequisites for installing the server software on Red Hat OpenShift

You must first complete certain tasks before you install HCL OneTest™ Server on the Red Hat OpenShift platform.

The following sections describe each prerequisite in detail:

- Internet access on page 36
- Harbor repository credentials on page 45
- Red Hat OpenShift platform on page 36
- Mandatory software on page 37
- Service virtualization through Istio on page 37

### Internet access

You must have access to the internet to install HCL OneTest™ Server.

### Harbor repository credentials

- **New customers**:

  You must create a support ticket to get the credentials that are required to access the product binaries from the Harbor repository. For more information, refer to How to create an HCL Support case.

- **Existing customers**:

  You must use your existing Harbor repository credentials to access the product binaries.

### Red Hat OpenShift platform

You must set up the Red Hat OpenShift platform and install the following components in your IT infrastructure:

- Red Hat OpenShift Container Platform V4.5. For more information, refer to the Red Hat OpenShift documentation.
- Dynamic Volume Provisioning that supports ReadWriteOnce (RWO) and ReadWriteMany (RWX) access modes. For more information, refer to the Dynamic provisioning section in the Red Hat OpenShift documentation.
- Jaeger operator to trace test logs and Jaeger-based reports when you run tests. For more information, refer to Installing Jaeger.
- **Optional**: Red Hat Service Mesh V2.0, if Istio recording and service virtualization features are required. For more information, refer to the Installing Red Hat OpenShift Service Mesh section in the Red Hat OpenShift documentation.

For more information about the service virtualization feature, see Management of virtualized services on page 319.

- **Optional**: You can configure the OpenShift software-defined networking (SDN) to provide a unified cluster network that enables the communication between pods across the OpenShift Container Platform cluster. The default installation process includes the NetworkPolicy resources and these resources are acted upon only if you configure the SDN appropriately. For more information, refer to the OpenShift SDN default CNI network provider section in the RedHat OpenShift documentation.

You must have access to the OpenShift cluster with cluster administrator privileges. You can run the following command on the OpenShift command-line interface to gain access:

```
oc login -u kubeadmin -p {password} https://api.{openshift-cluster-dns-name}:6443
```

For more information about hardware and software requirements, see System Requirements on page 12.

### Mandatory software

You must install the following software on Red Hat OpenShift:

- Helm V3.5.2. For more information, refer to the Installing a Helm chart on an OpenShift Container Platform cluster section in the Red Hat OpenShift documentation.
- OpenShift CLI. For more information, refer to the Getting started with the CLI section in the Red Hat OpenShift documentation.

### Service virtualization through Istio

**Disclaimer:** This release contains access to the Virtual services that virtualize the Istio based services feature in HCL OneTest™ Server as a Tech Preview. The Tech Preview is intended for you to view the capabilities for virtualized services offered by HCL OneTest™ Server, and to provide your feedback to the product team. You are permitted to use the information only for evaluation purposes and not for use in a production environment. HCL provides the information without obligation of support and "as is" without warranty of any kind.

To provide your feedback, you must perform the following tasks:

1. Sign up or Log in to the Ideas portal.
2. Click **Add a new idea** to provide your feedback on the Tech Preview features.
3. Select **OneTest TechPreview Programs** in the **Choose a workspace for this idea** field.
4. Provide all the necessary details, and then click **Share Idea**.

The default configuration does not enable service virtualization through Istio. To use the service virtualization feature, you must configure it appropriately. As a Tech Preview feature, you can virtualize the *bookinfo* sample application. You must install this application in the `Bookinfo` namespace. For more information about the *bookinfo* application, refer to the Istio documentation.

For more information about how to enable service virtualization through Istio, see Installing the server software on Red Hat OpenShift on page 41.

## Installing the server software on Red Hat OpenShift

You can install HCL OneTest™ Server on the Red Hat OpenShift server that has the Kubernetes Engine environment to run functional, integration, and performance tests. HCL OneTest™ Server combines test data, test environments, and test runs and reports into a single web-based browser for testers and non-testers.

**Before you begin**

You must have performed the following tasks:

- Completed tasks provided in the Prerequisites section. See Prerequisites for installing the server software on Red Hat OpenShift on page 36.
- Installed the following software:
  - OpenShift CLI
  - Helm
- Copied the **Secret key** from the Harbor repository.

1. Open and log in to the terminal.
2. Create a namespace to install the server software by running the following command:

   ```
   oc new-project test-system
   ```

   **Remember:** `test-system` is the name of the namespace. If you created a namespace by using a different value, then you must use that value in place of the `test-system` in all the instances in this procedure.

3. Add the software registry to Helm to access the server install charts by running the following command:

   ```
   helm repo add hclsoftware https://hclcr.io/chartrepo/ot --username {okta-email-address}
    --password {harbor-cli-secret}
   ```

   **Note:** You must replace `{okta-email-address}` with the user name of the Harbor repository and replace `{harbor-cli-secret}` with the secret key that you copied from the Harbor repository.

If the user name contains any special characters, such as $, you must enclose it within single quotes.

4. Run the following command to get the latest updates from the repository:

   ```
   helm repo update
   ```

5. Create a Secret to pull images that are used by HCL OneTest™ Server by running the following commands:

   ```
   oc create secret docker-registry hclcr.io \
   -n test-system \
   --docker-server=hclcr.io \
   --docker-username={okta-email-address} \
   --docker-password={harbor-cli-secret} \
   --docker-email=example@abc.com
   ```

   **Notes:**

   - You must replace `{okta-email-address}` with the user name of the Harbor repository and replace `{harbor-cli-secret}` with the secret key that you copied from the Harbor repository.

     If the user name contains any special characters, such as $, you must enclose it within single quotes.

   - You can replace `example@abc.com` with the administrator email address, if required.

6. Perform one of the following steps to enable certificates as trusted certificates:

   a. Perform the following steps to add the Certificate Authority (CA) into a Secret:

      i. Run the following command to verify whether an additional CA is required:

      ```
      curl -sw'%{http_code}' -o/dev/null \
      "https://wildcard.$(oc get -n openshift-ingress-operator ingresscontroller default
       -ojsonpath='{.status.domain}')"
      ```

      If the result of the command is `503`, the CA is already trusted. You must continue with 7 on page 40.

      If the result of the command is `000`, then CA must be added into a Secret. You must continue with 6.a.ii on page 39.

      ii. Run the following command to get the default CA in a PEM format:

      ```
      oc get -n openshift-ingress-operator secret router-ca -ojsonpath='{.data.tls\.crt}'
       | base64 --decode > ca.crt
      ```

      iii. Run the following command to validate that the CA used to sign the certificate is same for ingress:

      ```
      curl -sw'%{http_code}' -o/dev/null --cacert ca.crt \
      "https://wildcard.$(oc get -n openshift-ingress-operator ingresscontroller default
       -ojsonpath='{.status.domain}')"
      ```

The result of the command must be `503`.

If the result of the command is `000`, then the certificate configuration has been customized. You must find the certificate of the signer to go to the next step.

iv. Run the following command to create an ingress Secret to store the CA:

```
oc create secret generic -n test-system ingress --from-file=ca.crt=ca.crt
```

b. Perform the following steps to add the Certificate Authority (CA) into a Secret if you are using OpenShift Service Mesh service virtualization, a Tech Preview feature:

i. Run the following command to retrieve the charts:

```
helm pull --untar hclsoftware/hcl-onetest-server --version 4.1013.0
```

ii. Run the following script from the `files` directory:

```
./files/certificate.sh -n istio-system -s istio-ingressgateway-certs
 {openshift-cluster-dns-name}
```

iii. Run the following command to create an ingress Secret:

```
oc create secret generic -n test-system ingress \
  "--from-literal=ca.crt=$(oc get -n istio-system secret istio-ingressgateway-certs
 -ojsonpath='{.data.ca\.crt}' | base64 --decode)"
```

iv. Run the following commands to enable an OpenShift route that the product creates for the Istio gateway:

```
cat <<EOF | oc apply -n istio-system -f - >/dev/null
apiVersion: maistra.io/v1
kind: ServiceMeshMemberRoll
metadata:
  name: default
spec:
  members:
    - test-system
EOF
```

When some components such as static agents or Docker agents want to communicate with HCL OneTest™ Server, the component presents its certificate to the server to verify its identity. HCL OneTest™ Server trusts the component only if it is signed by a recognized and trusted CA. Therefore, you must add the signed CA into a trust by placing it in a Secret to enable certificates as trusted certificates.

7. Perform the following steps to install the server software:

You must go to step 7.c on page 41 if you have completed step 6.b on page 40.

a. Run the following command to get the latest updates from the repository:

```
helm repo update
```

b. Run the following command to retrieve the charts required to install the server software:

```
helm pull --untar hclsoftware/hcl-onetest-server --version 4.1013.0
```

c. Run the following commands to update the runAsUser and fsGroup to match the Security Context Constraints (SCC):

HCL OneTest™ Server is compatible with the restricted Security Context Constraint. You must run this command to ensure that the runAsUser and fsGroup strategies match with the SCC policy.

```
sed -i -e "s/runAsUser: 1001/runAsUser: $(oc get project test-system -oyaml \
  | sed -r -n 's# *openshift.io/sa.scc.uid-range: *([0-9]*)/.*#\1#p')/g;
         s/fsGroup: 1001/fsGroup: $(oc get project test-system -oyaml \
  | sed -r -n 's# *openshift.io/sa.scc.supplemental-groups: *([0-9]*)/.*#\1#p')/g"
 hcl-onetest-server/values-openshift.yaml
```

d. Perform one of the steps described in the following table to install the server software based on your requirement:

| Step description | Step no |
|---|---|
| To install the server software | Go to step 7.d.i on page 41 |
| To install the server software and enable OpenShift Service Mesh service virtualization, a Tech Preview feature | Go to step 7.d.ii on page 41 |
| To install the server software and enable Jaeger for performance and Web UI tests logs | Go to step 7.d.iii on page 42 |

i. Run the following command to install the server software:

```
helm install {my-ots} ./hcl-onetest-server -n test-system \
-f hcl-onetest-server/values-openshift.yaml \
--set global.persistence.rwxStorageClass=rook-ceph-file
--set global.hclOneTestIngressDomain=onetest.{openshift-cluster-dns-name} \
--set global.hclFlexnetURL=https://hclsoftware.compliance.flexnetoperations.com \
--set global.hclFlexnetID={cloud-license-server-id} \
--set global.hclImagePullSecret=hclcr.io \
--set global.hclOneTestPasswordAutoGenSeed={password-seed}
```

ii. Run the following command to install the server software and to enable OpenShift Service Mesh service virtualization, a Tech Preview feature:

```
helm install {my-ots} ./hcl-onetest-server -n test-system \
-f hcl-onetest-server/values-openshift.yaml \
--set global.persistence.rwxStorageClass=rook-ceph-file
--set global.hclOneTestIngressDomain=onetest.{openshift-cluster-dns-name} \
--set global.hclFlexnetURL=https://hclsoftware.compliance.flexnetoperations.com \
--set global.hclFlexnetID={cloud-license-server-id} \
--set global.hclImagePullSecret=hclcr.io \
--set global.hclOneTestPasswordAutoGenSeed={password-seed} \
-f hcl-onetest-server/values-openshift-demo.yaml
```

**Important:** You must run the following command to enable service virtualization in the specific namespace:

Where, {my-ots} is the release name that you provided during the installation of the server software.

```
oc create rolebinding istio-virtualization-enabled -n
 bookinfo --clusterrole={my-ots}-execution-istio-test-system
 --serviceaccount=test-system:{my-ots}-execution
```

**Note:** When you uninstall the chart, the manually created role bindings are not deleted from the namespace. You can run the following command to delete the role bindings:

```
oc delete rolebinding istio-virtualization-enabled -n bookinfo
```

iii. Run the following command to install the server software and to enable Jaeger for performance and Web UI tests logs:

```
helm install {my-ots} ./hcl-onetest-server -n test-system \
-f hcl-onetest-server/values-openshift.yaml \
--set global.persistence.rwxStorageClass=rook-ceph-file
--set global.hclOneTestIngressDomain=onetest.{openshift-cluster-dns-name} \
--set global.hclFlexnetURL=https://hclsoftware.compliance.flexnetoperations.com \
--set global.hclFlexnetID={cloud-license-server-id} \
--set global.hclImagePullSecret=hclcr.io \
--set global.hclOneTestPasswordAutoGenSeed={password-seed} \
--set-string execution.annotations.sidecar\\.jaegertracing\\.io/inject=true \
--set global.jaegerAgent.enabled=true \
--set global.jaegerAgent.internalHostName=localhost \
--set global.jaegerDashboard.enabled=true \
--set global.jaegerDashboard.externalURL={my-jaeger-dashboard-url}
```

You must substitute the value of the variables in the helm install command with the actual value:

- `{my-ots}` with the release name of your choice.

  **Note:** The release name must consist of lower case alphanumeric characters or "-" (hyphen), start with an alphabetic character, and end with an alphanumeric character. For example, *my-org* or *abc-123*.

- `{openshift-cluster-dns-name}` with the ingress DNS name that you selected for the server.

> **Remember:** You must provide the value that consists of lowercase alphanumeric characters, “-”(hyphen) or “.”(period). Also, the value must start and end with an alphanumeric character.

> **Note:** You can run the following command to obtain the default value of `openshift-cluster-dns-name`:
>
> ```
> oc get --namespace=openshift-ingress-operator ingresscontroller/default
>  -ojsonpath='{.status.domain}'
> ```

- `{password-seed}` with a value of your choice.

  > **Important:** This password seed is used to create several default passwords for the server. You must store the password seed securely. When you install the server software by using the backup of the user data, you can reuse the password seed. You can use this seed to restore backup files either on the current or later versions of the server software.

- **Optional**: `{cloud-license-server-id}` with the ID of the license server, if you are setting the license value for the first time.

  > **Important:** If you are upgrading the product from the previous version, you must configure the **OneTest License Server ID** value from the **License Configuration** window when the installation of the server is complete.

- `{my-jaeger-dashboard-url}` with the URL of the Jaeger server.

8. **Optional:** Run the following command to remove a job that is used to initialize the PostgresQL database during the installation of the server software:

   ```
   oc delete job {my-ots}-postgresql-init -n test-system
   ```

9. **Optional:** Perform the following steps to migrate user data into HCL OneTest™ Server, if you upgraded from the previous version:

   a. Run the following script from the `hcl-onetest-server/files` directory to create a directory that contains metadata related to the Persistent Volume Claims and their Persistent Volumes:

   `migrate.sh create-pvcs -n test-system {my-ots}`

   b. Run the following script from the `hcl-onetest-server/files` directory to merge the data into the server:

   `migrate.sh merge-dbs -n test-system {my-ots}`

c. Run the following command to remove the resources that were created during the migration process:

```
migrate.sh delete-temp-resources -n test-system {my-ots}
```

where `{my-ots}` is the release name that you provided during the installation of the server software.

10. Run the following command to verify and test the installed server software:

```
$ helm test {my-ots} -n test-system
```

where `{my-ots}` is the release name that you provided during the installation of the server software.

**Results**

On successful installation of HCL OneTest™ Server, the output displays the URL to access the HCL OneTest™ Server UI.

**What to do next**

You can perform the following tasks:

- Back up the user data that are saved in the Kubernetes clusters to secure your data. See Backing up and restoring the user data on Red Hat OpenShift on page 58.
- Configure the license to use HCL OneTest™ Server. See Configuring named user licenses on page 101.

Related information

Troubleshooting Guide on page 537

## Installation of the server software on Ubuntu

You can find information about the tasks that you can perform to install HCL OneTest™ Server software on the Ubuntu platform.

## Prerequisites for installing the server software on Ubuntu

You must first complete certain tasks before you install HCL OneTest™ Server on the Ubuntu platform.

The following sections describe each prerequisite in detail:

- Internet access on page 45
- Harbor repository credentials on page 45
- Ubuntu server on page 45
- Backup user data on page 45
- Mandatory software on page 46

- k3s environment on page 46
- Service virtualization through Istio on page 46

### Internet access

You must have access to the internet to install HCL OneTest™ Server.

### Harbor repository credentials

- **New customers**:

  You must create a support ticket to get the credentials that are required to access the product binaries from the Harbor repository. For more information, refer to How to create an HCL Support case.

- **Existing customers**:

  You must use your existing Harbor repository credentials to access the product binaries.

### Ubuntu server

You must install an Ubuntu server and ensure that you have a working Domain Name Server (DNS) that resolves the host name into a machine-readable IP address.

For more information about hardware and software requirements, see System Requirements on page 12.

**Note:** Depending on your testing workload, HCL OneTest™ Server might require more resources. You must use the entire disk space and set up Logical Volume Manager (LVM) by using the `ext4` file system. If your organization requires application data to be stored in a separate partition, then you can create a mount point at `/var/lib/rancher/k3s/storage/` with at least 128 GB capacity.

### Backup user data

If you want to migrate user data from a previous version of the product during the installation of the server software, then you must perform one of the following tasks based on the existing version of the server software:

- Back up the data from V10.0.2, Fix Pack 1 or earlier. See Backing up the user data from a previous release.
- Back up the data from V10.1.0 or later. See Backing up and restoring the user data on Ubuntu on page 60.

  **Note:** You must select the existing version of HCL OneTest™ Server from the drop-down list to view the procedure for Backing up and restoring the user data for your version, as the instructions differ for versions 10.1, 10.1.1, and 10.1.2.

## Mandatory software

You must install the following mandatory software:

- Helm V3.5.2. For more information, refer to the Helm documentation.

  **Note:** The helm command must be in one of the directories in your PATH environment variable.

- OpenSSH Server. For more information, refer to the Installation section in the OpenSSH Server documentation.

  **Note:** If you use ssh to log in to the computer, then you can skip this step as OpenSSH is already installed on your computer.

## k3s environment

You must set up the k3s Kubernetes environment along with other configurations such as firewall, Jaeger, and Prometheus server. See Setting up a Kubernetes environment (k3s) on Ubuntu on page 47.

## Service virtualization through Istio

**Disclaimer:** This release contains access to the Virtual services that virtualize the Istio based services feature in HCL OneTest™ Server as a Tech Preview. The Tech Preview is intended for you to view the capabilities for virtualized services offered by HCL OneTest™ Server, and to provide your feedback to the product team. You are permitted to use the information only for evaluation purposes and not for use in a production environment. HCL provides the information without obligation of support and "as is" without warranty of any kind.

To provide your feedback, you must perform the following tasks:

1. Sign up or Log in to the Ideas portal.
2. Click **Add a new idea** to provide your feedback on the Tech Preview features.
3. Select **OneTest TechPreview Programs** in the **Choose a workspace for this idea** field.
4. Provide all the necessary details, and then click **Share Idea**.

The default configuration does not enable service virtualization through Istio. To use the service virtualization feature, you must configure it appropriately. As a Tech Preview feature, you can virtualize the *bookinfo* sample application. You must install this application in the `Bookinfo` namespace. For more information about the *bookinfo* application, refer to the Istio documentation.

For more information about how to enable service virtualization through Istio, see Setting up a Kubernetes environment (k3s) on Ubuntu on page 47.

## Setting up a Kubernetes environment (k3s) on Ubuntu

You can use the script that is provided with HCL OneTest™ Server to set up a k3s Kubernetes environment.

**Before you begin**

You must have completed the following tasks:

- Installed the following software:
    - OpenSSH server
    - Helm
- Ensured that your computer has a Domain Name Server (DNS) resolvable host name to resolve the host name into a machine-readable IP address.
- Copied a **Secret key** from the Harbor repository.

**About this task**

As part of the k3s Kubernetes environment set up, you can enable the following software:

- **Jaeger**: By using this software, you can trace test logs and Jaeger-based reports when you run tests.
- **Prometheus server**: By using this software, you can monitor your system resources by using metrics data that are collected by a Prometheus server.

1. Use an SSH session to log in to the Ubuntu server.
2. Add the software registry to Helm by running the following command:

   ```
   helm repo add hclsoftware https://hclcr.io/chartrepo/ot --username {okta-email-address}
    --password {harbor-cli-secret}
   ```

   **Note:** You must replace `{okta-email-address}` with the user name of the Harbor repository and replace `{harbor-cli-secret}` with the secret key that you copied from the Harbor repository.

   If the user name contains any special characters, such as $, you must enclose it within single quotes.

3. Run the following command to get the latest updates from the repository:

```
helm repo update
```

4. Run the following commands to fetch the scripts that are used to install Kubernetes:

```
helm pull --untar hclsoftware/hcl-onetest-base --version 4.1013.0
chmod +x hcl-onetest-base/*.sh
```

5. Run any of the following commands to install the k3s Kubernetes environment:

   **Note:** The second option allows you to override the name of the Kubernetes domain that is created.

   **Choose from:**

   - Run the following commands to install with the default name that is either based on IP address or fully qualified host name:

     ```
     #Run the following commands if you are on Ubuntu 18.04
     $ cd hcl-onetest-base
     $ sudo ./ubuntu-init.sh

     #Run the following commands if you are on Ubuntu 20.04
     $ cd hcl-onetest-base
     $ sudo HOME=$HOME ./ubuntu-init.sh
     ```

   - Run the following commands to install k3s Kubernetes environment by overriding the default name:

     ```
     $ cd hcl-onetest-base
     $ sudo INGRESS_DOMAIN={onetest.}myorg.com HOME=$HOME ./ubuntu-init.sh
     ```

     where:

     - *{onetest.}* is a sub-domain name that you specified for the server. For example, `testenv`.

       **Note:** The sub-domain must consist of lowercase alphanumeric characters, *"-"*(hyphen) or *"."*(period). Also, the value must start and end with an alphanumeric character.

     - *myorg.com* is the domain name of your organization. For example, `hcl.com`

   You can access the product through a web browser by using any of the following ways:

   - **Fully Qualified Hostname**: When the server is configured, you can use `hostname -f` command to get the fully qualified hostname defined in the DNS to access HCL OneTest™ Server. For example, `{onetest}.myorg.com`

   - **IP address**: You can use the IP address to access HCL OneTest™ Server when you cannot create a specific DNS record for the server. For example, `ip-address.nip.io`

   **Result**

   On completion of the ubuntu-init.sh script, a namespace with the name `test-system` is created to install the server software and the output displays the following information on the command-line interface:

- The INGRESS_DOMAIN that is in use. This is, the URL from where you can access HCL OneTest™ Server. You must use this value for the **global.hclOneTestIngressDomain** parameter in step 4 on page 51 in the server installation topic.
- The DNS information that the Kubernetes cluster uses to resolve names.
- Certificate Authority (CA) that must be import into the browser to prevent certificate errors.

  You can run the following command to get the certificate from the system:

  ```
  kubectl get secret ingress -n test-system -o jsonpath={.data.ca\\.crt} | base64 -d
  ```
- Instructions to confirm whether the Kubernetes environment has started.

  You can refer to the **Results** section for more details on how to verify the Kubernetes environment has started.

6. Perform one of the following options to configure a firewall:
   **Choose from:**

   - Run the following script to configure the firewall that allows traffic on *cni0* and port *443*:

     ```
     #Run the following command if you are on Ubuntu 18.04
     $ sudo ./ubuntu-firewall.sh

     #Run the following command if you are on Ubuntu 20.04
     $ sudo HOME=$HOME ./ubuntu-firewall.sh
     ```

     **Note:** You must consult your network administrator before you run this script and confirm that whether the firewall is compatible with your corporate policy.
   - Update the firewall that allows traffic on *cni0* and port *443*, if your Ubuntu server is already configured with the firewall.
7. **Optional:** Run the following command to enable Istio service virtualization, a Tech Preview feature:

   ```
   #Run the following command if you are on Ubuntu 18.04
   $ sudo ./ubuntu-init.sh --demo

   #Run the following command if you are on Ubuntu 20.04
   $ sudo HOME=$HOME ./ubuntu-init.sh --demo
   ```
8. **Optional:** Run the following command to enable the Jaeger traces for performance and Web UI tests:

   ```
   ./service.sh expose jaeger
   ```

**Note:** If you do not enable Jaeger, HCL OneTest™ Server produces text output in a microservice log file instead of Jaeger traces when you run performance and Web UI test assets.

**Important:** The Jaeger traces are not protected, thus, any information logged into the Jaeger server might be easily accessible by anyone who has or discovers the *<server-url>/jaeger* URL.

9. **Optional:** Run the following command to enable the Prometheus server to monitor your system resources by using metrics data:

   ```
   ./service.sh expose prometheus
   ```

   **Important:** The Prometheus metrics are not protected, thus, any information logged into the Prometheus server might be easily accessible by anyone who has or discovers the *<server-url>/prometheus* URL.

**Results**

You have set up the Kubernetes environment on Ubuntu.

**Note:** You can run the `kubectl get pods -A` command to verify that the Kubernetes environment is working. After a while, the status of the pods must be `Running` or `Complete` state.

**What to do next**

- You must log in to the server host system again after the installation process is completed so that changes to the group membership are applied.
- You can install the server software. See Installing the server software on Ubuntu by using k3s on page 50.

Related information

Jaeger Operator

Prometheus server

Troubleshooting Guide on page 537

## Installing the server software on Ubuntu by using k3s

You can install HCL OneTest™ Server on the Ubuntu server that has a Kubernetes environment to run functional, integration, and performance tests. HCL OneTest™ Server combines test data, test environments, and test runs and reports into a single, web-based browser for testers and non-testers.

**Before you begin**

You must have performed the following tasks:

- Completed tasks provided in the Prerequisites section. See Prerequisites for installing the server software on Ubuntu on page 44.
- Set up k3s Kubernetes environment. See Setting up a Kubernetes environment (k3s) on Ubuntu on page 47.
- Logged in to the server host system again after the you completed the k3s environment setup.
- Copied the following values to be used during the installation process:
    - The **Secret key** from the Harbor repository.
    - The INGRESS_DOMAIN value that displayed on completion of the `ubuntu-init.sh` script.

1. Use an SSH session to log in to the Ubuntu server.
2. Run the following command to get the latest updates from the repository:

   ```
   helm repo update
   ```

3. Create a Secret to pull images that are used by HCL OneTest™ Server by running the following command:

   ```
   kubectl create secret docker-registry hclcr.io \
   -n test-system \
   --docker-server=hclcr.io \
   --docker-username={okta-email-address} \
   --docker-password={harbor-cli-secret} \
   --docker-email=example@abc.com
   ```

   **Notes:**

    - You must replace `{okta-email-address}` with the user name of the Harbor repository and replace `{harbor-cli-secret}` with the secret key that you copied from the Harbor repository.

      If the user name contains any special characters, such as $, you must enclose it within single quotes.
    - You can replace `example@abc.com` with the administrator email address, if required.

4. Perform the following steps to install the server software:

    a. Run the following command to get the latest updates from the repository:

   ```
   helm repo update
   ```

    b. Run the following command to retrieve the charts required to install the server software:

   ```
   helm pull --untar hclsoftware/hcl-onetest-server --version 4.1013.0
   ```

    c. Perform one of the steps described in the following table to install the server software based on your requirement:

| Step description | Step no |
|---|---|
| To install the server software | Go to step 4.c.i on page 52 |
| To install the server software and enable Istio service virtualization, a Tech Preview feature | Go to 4.c.ii on page 52 |
| To install the server software and enable the Jaeger UI<br>**Note:** You must perform this step only if you have not enabled the Jaeger UI to be exposed during the setting up of Kubernetes k3s environment. | Go to step 4.c.iii on page 53 |
| To install the server software and enable performance test runs on the statuc agents | Perform 4.c.iv on page 53 |

i. Run the following command to install the server software:

```
helm install {my-ots} ./hcl-onetest-server -n test-system \
-f hcl-onetest-server/values-k3s.yaml \
--set global.hclOneTestIngressDomain={my-ingress-dns-name} \
--set global.hclFlexnetURL=https://hclsoftware.compliance.flexnetoperations.com \
--set global.hclFlexnetID={cloud-license-server-id} \
--set global.hclImagePullSecret=hclcr.io \
--set global.hclOneTestPasswordAutoGenSeed={password-seed}
```

ii. Run the following command to install the server software and to enable Istio service virtualization, a Tech Preview feature:

```
helm install {my-ots} ./hcl-onetest-server -n test-system \
-f hcl-onetest-server/values-k3s.yaml \
--set global.hclOneTestIngressDomain={my-ingress-dns-name} \
--set global.hclFlexnetURL=https://hclsoftware.compliance.flexnetoperations.com \
--set global.hclFlexnetID={cloud-license-server-id} \
--set global.hclImagePullSecret=hclcr.io \
--set global.hclOneTestPasswordAutoGenSeed={password-seed} \
-f hcl-onetest-server/values-k3s-demo.yaml
```

**Important:** You must run the following command to enable service virtualization in the specific namespace after the server installation is complete:

```
kubectl create rolebinding istio-virtualization-enabled -n
 bookinfo --clusterrole={my-ots}-execution-istio-test-system
 --serviceaccount=test-system:{my-ots}-execution
```

**Note:** When you uninstall the chart, the manually created role bindings are not deleted from the namespace. You can run the following command to delete the role bindings:

```
kubectl delete rolebinding istio-virtualization-enabled -n bookinfo
```

iii. Run the following command to install the server software and to enable the Jaeger UI, if you have not enabled the Jaeger UI to be exposed during the setting up of Kubernetes k3s environment:

```
helm install {my-ots} ./hcl-onetest-server -n test-system \
-f hcl-onetest-server/values-k3s.yaml \
--set global.hclOneTestIngressDomain={my-ingress-dns-name} \
--set global.hclFlexnetURL=https://hclsoftware.compliance.flexnetoperations.com \
--set global.hclFlexnetID={cloud-license-server-id} \
--set global.hclImagePullSecret=hclcr.io \
--set global.hclOneTestPasswordAutoGenSeed={password-seed} \
--set global.jaegerDashboard.externalURL={my-jaeger-dashboard-url} \
--set global.jaegerAgent.internalHostName=localhost
```

iv. Run the following command to install the server software and to enable performance test runs on the static agents:

```
helm install {my-ots} ./hcl-onetest-server -n test-system \
-f hcl-onetest-server/values-k3s.yaml \
--set global.hclOneTestIngressDomain={my-ingress-dns-name} \
--set global.hclFlexnetURL=https://hclsoftware.compliance.flexnetoperations.com \
--set global.hclFlexnetID={cloud-license-server-id} \
--set global.hclImagePullSecret=hclcr.io \
--set global.hclOneTestPasswordAutoGenSeed={password-seed} \
--set networkPolicy.enabled=false
```

You must substitute the value of the following variables with the actual value in the command:

- `{my-ots}` with the release name of your choice.

  **Note:** The release name must consist of lower case alphanumeric characters or "-" (hyphen), start with an alphabetic character, and end with an alphanumeric character. For example, *my-org* or *abc-123*.

- `{my-ingress-dns-name}` with the INGRESS_DOMAIN value that displayed on completion of the `ubuntu-init.sh` script.
- **Optional**: `{cloud-license-server-id}` with the ID of the license server, if you are setting the license value for the first time.

**Important:** If you are upgrading the product from the previous version, you must configure the **OneTest License Server ID** value from the **License Configuration** window when the installation of the server is complete.

- `{password-seed}` with a value of your choice.

  **Important:** This password seed is used to create several default passwords for the server. You must store the password seed securely. When you install the server software by using the backup of the user data, you can reuse the password seed. You can use this seed to restore backup files either on the current or later versions of the server software.

- `{my-jaeger-dashboard-url}` with the URL of the Jaeger server.

5. **Optional:** Run the following command to remove a job that is used to initialize the PostgresQL database during the installation of the server software:

   ```
   kubectl delete job {my-ots}-postgresql-init -n test-system
   ```

6. **Optional:** Perform the following steps to migrate data into HCL OneTest™ Server, if you upgraded the server software from the previous version:

   a. Run the following script from the `hcl-onetest-server/files` directory to create a directory that contains metadata related to the Persistent Volume Claims and their Persistent Volumes:

   ```
   migrate.sh create-pvcs -n test-system {my-ots}
   ```

   b. Run the following script from the `hcl-onetest-base` directory to back up the data:

   ```
   sudo backup.sh create-pvc-links -v ~/migration-pvc-links
   ```

   c. Run the following command to stop the cluster and HCL OneTest™ Server:

   ```
   k3s-killall.sh
   ```

   d. Run the following script from the `hcl-onetest-base` directory to restore the backed-up data:

   ```
   sudo backup.sh restore -v ~/migration-pvc-links <backup-file-name> --release
   ```

   **Note:** You must replace `<backup-file-name>` with the name of the back up file that you saved.

   e. Run the following command to restart Kubernetes and to start HCL OneTest™ Server:

   ```
   sudo systemctl start k3s
   ```

   f. Run the following script from the `hcl-onetest-server/files` directory to merge the data into the server:

   ```
   migrate.sh merge-dbs -n test-system {my-ots}
   ```

g. Run the following command to remove the resources that were created during the migration process:

```
migrate.sh delete-temp-resources -n test-system {my-ots}
```

7. Run the following command to verify and test the installed server software:

```
$ helm test {my-ots} -n test-system
```

where `{my-ots}` is the release name that you provided during the installation of the server software.

### Results

On completion of the installation of server software, the output displays the following information on the command-line interface:

- Instructions to access Keycloak to manage and authenticate users.

  You must use *keycloak* as user name and the password can be retrieved by running the following command:

  ```
  kubectl get secret -n namespace onetest-keycloak-postgresql -o jsonpath="{.data.password}" | base64
   --decode; echo
  ```

  where:

  - *onetest* is a sub-domain name that you select for the server.
  - *namespace* is the name of the namespace that you created.

- The URL to access the HCL OneTest™ Server UI.

### What to do next

You can do the following tasks:

- Back up the user data that are saved in the Kubernetes clusters to secure your data. See Backing up and restoring the user data on Ubuntu on page 60.
- Configure the license to use HCL OneTest™ Server. See Configuring named user licenses on page 101.

Related information

Helm documentation

Troubleshooting Guide on page 537

## Server software upgrade methods

When you want to use the enhanced functionalities of HCL OneTest™ Server, you must upgrade to the latest version of the server software.

## Server software upgrade on Red Hat OpenShift

When you want to use the latest version of the server software on Red Hat OpenShift, you must upgrade HCL OneTest™ Server from the previous version of the software.

Before you upgrade the server software, you must ensure that you have completed the following tasks:

- Installed Helm V3.5.2. For more information, refer to the Installing a Helm chart on an OpenShift Container Platform cluster section in the Red Hat OpenShift documentation.
- Informed users that HCL OneTest™ Server is offline temporarily during the upgrade process.
- Completed all test executions that are running on the existing HCL OneTest™ Server V10.1.0 or later.
- Stopped all stub executions that are running on the existing HCL OneTest™ Server V10.1.0 or later.
- Canceled any scheduled test runs that have a future date or time.

In upgrade (by new install) method, depending on the existing version of the software, you can upgrade the server software by using any of the following options.

To upgrade HCL OneTest™ Server V10.1.0 or later to latest version, you must perform these tasks in sequence as listed in the following table:

| | Task description | More information |
|---|---|---|
| 1 | Uninstall HCL OneTest™ Server V10.1.0 or later. | Uninstalling the server software from Red Hat OpenShift on page 65 |
| 2 | Install the latest version of the server software. | Installing the server software on Red Hat OpenShift on page 38 |

To upgrade HCL OneTest™ Server V10.0.2, Fix Pack 1 or earlier to latest version, you must perform these tasks in sequence as listed in the following table:

| Tasks | Task description | More information |
|---|---|---|
| 1 | Upgrade HCL OneTest™ Server V10.1.2. | Installation of the server software on Red Hat OpenShift |
| 2 | Uninstall HCL OneTest™ Server V10.1.2. | Uninstalling the server software |
| 3 | Install the latest version of the server software. | Installing the server software on Red Hat OpenShift on page 38 |

**Important:**

After you migrate the server data from a previous version to a newer version, the **Project Owner** must log in to HCL OneTest™ Server so that other members of the project can access the test assets in that project.

## Server software upgrade on Ubuntu

When you want to use the latest version of the server software on Ubuntu, you must upgrade Rational® Test Automation Server from the previous version of the software.

Before you upgrade the server software, you must ensure that you have completed the following tasks:

- Installed Helm V3.5.2. For more information, refer to the Helm documentation.
- Informed users that HCL OneTest™ Server is offline temporarily during the upgrade process.
- Completed all test executions that are running on the existing HCL OneTest™ Server V10.1.0 or later.
- Stopped all stub executions that are running on the existing HCL OneTest™ Server V10.1.0 or later.
- Canceled any scheduled test runs that have a future date or time.

In upgrade (by new install) method, depending on the existing version of the software, you can upgrade the server software by using any of the following options.

To upgrade Rational® Test Automation Server V10.1.0 or later to latest version, you must perform these tasks in sequence as listed in the following table:

| Tasks | Task description | More information |
|---|---|---|
| 1 | Back up the user data from Rational® Test Automation Server V10.1.0 or later. | • For V10.1.0, refer to Backing up user data - V10.1.0.<br>• For V10.1.1, refer to Backing up user data - V10.1.1.<br>• For V10.1.2, refer to Backing up user data - V10.1.2. |
| 2 | Uninstall Rational® Test Automation Server V10.1.0 or later. | Uninstalling the server software from Ubuntu on page 66 |
| 3 | Install the latest version of the server software, and then restore the user data. | Installing the server software on Ubuntu by using k3s on page 50 |

If your existing version of server software is 10.0.2, Fix Pack 1, or earlier, then you cannot directly upgrade to the latest version of the server software due to the steps involved in the migration process. Therefore, you must perform the tasks in sequence as listed in the following table:

| Tasks | Task description | More information |
|---|---|---|
| 1 | Upgrade Rational® Test Automation Server V10.1.2. | Installing Rational® Test Automation Server V10.1.2 |
| 2 | Back up the user data from Rational® Test Automation Server V10.1.2. | Backing up user data - V10.1.2 |
| 3 | Uninstall Rational® Test Automation Server V10.1.2. | Uninstalling the server software |
| 4 | Install the server software, and then restore the user data on the server software. | Installing the server software on Ubuntu by using k3s on page 50 |

! **Important:**

After you migrate the server data from a previous version to a newer version, the **Project Owner** must log in to HCL OneTest™ Server so that other members of the project can access the test assets in that project.

# Backing up and restoring the user data

You might want to restore an earlier environment state if you install a new version of HCL OneTest™ Server, move environments to different systems (both real and virtual), or set up a test environment ready for testing. To address any of these needs, you can back up and restore HCL OneTest™ Server user data installed either on Ubuntu or on Red Hat OpenShift platform.

## Backing up and restoring the user data on Red Hat OpenShift

To secure the user data in HCL OneTest™ Server that is installed on the OpenShift server, you can back up the user data. At any point in time, after you install HCL OneTest™ Server, you can restore the backed-up user data.

**Before you begin**

You must have been granted access to the cluster.

**About this task**

This topic provides you with an example of backing up and restoring the user data by using Velero. Velero is one of the methods for backing up and restoring the user data. You can also use other methods for backing up and restoring the user data. If you use a different method to back up, then you must include the Persistent Volumes in the cluster.

## Preparing your cluster to backup and restore the user data

You must prepare your cluster before you back up or restore the user data in HCL OneTest™ Server.

1. Log in to the cluster by using oc login.
2. Install and configure Velero with the Restic Integration.
3. Set the name of the namespace in which HCL OneTest™ Server is installed by running the following command:

```
NS=<namespace>
```

4. Update stateful sets to apply the annotations required by Velero to back up the PVs used by the pods by running the following commands:

```
for sts in $(oc get sts -n "$NS" -o name); do \
  if ! oc -n "$NS" patch --type=json "$sts" -p \
    '[{"op":"add","path":"/spec/template/metadata/annotations/backup.velero.io~1backup-volumes",
 "value":"data"}]' 2>/dev/null; then
      oc -n "$NS" patch --type=json "$sts" -p \
       '[{"op":"add","path":"/spec/template/metadata/annotations",
 "value":{"backup.velero.io/backup-volumes":"data"}}]'
  fi \
done
```

5. Change the security restriction of the pods by running the following commands:

```
for sts in $(oc get sts -n "$NS" -oname); do \
  if oc get -n "$NS" "$sts" -ojsonpath='{.spec.template.spec.securityContext}' | grep -q
 "runAsNonRoot:true"; then \
    echo $sts
    oc patch -n "$NS" "$sts" --type json \
      -p '[{"op":"replace", "path":"/spec/template/spec/securityContext", "value":
 {"runAsNonRoot": false}}]'
  fi \
done
```

## Backing up the user data

You can back up the user data by using Velero.

### Before you begin

You must have completed the steps in the Preparing your cluster to backup and restore the user data on page 58.

### About this task

This procedure is an example of backing up the user data by using Velero.

1. Log in to the cluster by using oc login.
2. Start the backup process by running the following commands:

```
velero backup create <backup-name> --include-namespaces=<hcl-onetest-namespace>
```

## Restoring the user data

If you want to restore the Velero backup on to a cluster in which HCL OneTest™ Server is not installed, you can restore the user databy using Velero.

### Before you begin

You must have backed up the user data. See Backing up the user data on page 59.

1. Log in to the cluster by using oc login.
2. Start the restore process by entering the following command:.

```
velero restore create --from-backup=<backup-name> --restore-volumes
```

**Results**

You have backed up or restored the user data for HCL OneTest™ Server.

**What to do next**

After you migrate the server data from a previous version to a newer version, the **Project Owner** must log in to HCL OneTest™ Server so that other members of the project can access the test assets in that project.

Related information

#unique_83_Connect_42_section_ukg_bxt_slb

Velero Documentation

## Backing up and restoring the user data on Ubuntu

To secure the user data in HCL OneTest™ Server that is installed on the Ubuntu server, you can back up the user data. At any point in time, after you install HCL OneTest™ Server, you can restore the backed-up user data.

**Before you begin**

You must have completed the following tasks:

- Been granted with the same permissions that are required for installing and uninstalling the software.
- Been granted with the *sudo* access.
- Communicated to the users that HCL OneTest™ Server might be unavailable for some time until the process is complete.

**About this task**

You may need to back up and restore the user data in the following scenarios:

- Move the existing environment to a new system.
- Change the release name or the namespace.
- Back up the user data at regular intervals and restore the backed-up data when required.

## Backing up the user data

You can back up the user data in HCL OneTest™ Server and use the backed-up data to restore at any point in time.

1. Log in to the Ubuntu server and open a terminal.
2. Add the software registry to Helm by running the following command:

   ```
   helm repo add hclsoftware https://hclcr.io/chartrepo/ot --username {okta-email-address}
    --password {harbor-cli-secret}
   ```

   **Note:** You must replace `{okta-email-address}` with the user name of the Harbor repository and replace `{harbor-cli-secret}` with the secret key that you copied from the Harbor repository.

   If the user name contains any special characters, such as $, you must enclose it within single quotes.

3. Run the following command to extract the base:

   ```
   helm pull --untar hclsoftware/hcl-onetest-base --version 4.1013.0
   ```

4. Change to the `hcl-onetest-base` directory.

   This directory contains the `backup.sh` script which is required to complete the backup operation.

5. Create a directory that contains metadata related to the Persistent Volume Claims and their Persistent Volumes by running the following command:

   ```
   sudo ./backup.sh create-pvc-links
   ```

   You can use the following optional parameter along with the **create-pvc-links** command:

   --volumes or -v: Use this parameter to specify the directory path of the Volumes.

   For example, `backup.sh create-pvc-links -v hcl-onetest-base/my-pvc-links` creates a directory called `my-pvc-links` that stores the metadata required for backup.

   **Remember:**

   - If you did not provide the volume parameter, by default, the command creates a sub-directory in the same directory where you have the backup script.
   - You must ensure that the directory you create must be empty or not exist to avoid script failure.
   - If you specify a directory by using the -v parameter, then you must use the same parameter value when you restore the data.

6. Run the following command to stop the cluster and HCL OneTest™ Server:

   ```
   k3s-killall.sh
   ```

7. Run the following command to create a backup of the existing user data:

   ```
   sudo ./backup.sh create [options] <backup-file-name>
   ```

After you run this command, a backup of the local Persistent Volumes is created. The backup is created as tar archives that is compressed by using gzip (.tar.gz). The **create** command archives the Persistent Volumes into the *<backup-file-name>*.

You can use the following parameters along with the **create** command:

- --namespace or -n: Use this parameter to back up the Persistent Volumes in the specified namespace. If you do not mention the namespace, then all the Volumes from all the namespaces are included in the backup. The syntax is:

  ```
  -namespace <name of the namespace>
  ```

- --volumes or -v: Use this parameter to specify the directory path of the Volumes. The syntax is:

  ```
  -volumes <path-of-the-directory>
  ```

  **Remember:**

  - If you did not provide the volume parameter, by default, the command creates a sub-directory in the same directory where you have the backup script.
  - If you specify a directory by using the -v parameter, then you must use the same parameter value when you restore the data.

For example,

```
sudo ./backup.sh create --namespace test-system my-backup.tar.gz
```

This command creates a backup file named *my-backup.tar.gz* that contains all of the Persistent Volumes associated with pods available in the *test-system* namespace.

8. Run the following command to restart the cluster and HCL OneTest™ Server:

   ```
   sudo systemctl start k3s
   ```

## Restoring the user data

You can restore the user data that is backed up in HCL OneTest™ Server at any point in time after you install latest version of server software.

### About this task

The following procedure is for the restoration of the server data from the current release. If you are upgrading the server software, then you can restore the server data from the earlier version to the latest version during the installation of the server software. For more information, see the **Related information** section.

1. Log in to the Ubuntu server and open a terminal.
2. Add the software registry to Helm by running the following command:

   ```
   helm repo add hclsoftware https://hclcr.io/chartrepo/ot --username {okta-email-address}
    --password {harbor-cli-secret}
   ```

**Note:** You must replace `{okta-email-address}` with the user name of the Harbor repository and replace `{harbor-cli-secret}` with the secret key that you copied from the Harbor repository.

If the user name contains any special characters, such as $, you must enclose it within single quotes.

3. Run the following command to extract the base:

```
helm pull --untar hclsoftware/hcl-onetest-base --version 4.1013.0
```

4. Change to the `hcl-onetest-base` directory.
   This directory contains the `backup.sh` script which is required to complete the restore operation.
5. If you have not already done so, create a directory that contains metadata related to the Persistent Volume Claims and their Persistent Volumes by running the following command:

```
sudo ./backup.sh create-pvc-links
```

You can use the following optional parameter along with the **create-pvc-links** command:

--volumes or -v: Use this parameter to specify the directory path of the Volumes.

For example, `backup.sh create-pvc-links -v hcl-onetest-base/my-pvc-links` creates a directory called `my-pvc-links` that stores the metadata required to restore your user data.

**Remember:**

- If you did not provide the volume parameter, by default, the command creates a sub-directory in the same directory where you have the backup script.
- You must ensure that the directory you create must be empty or not exist to avoid script failure.

6. Run the following command to stop the cluster and HCL OneTest™ Server:

```
k3s-killall.sh
```

7. Run the following command to restore the backed-up user data:

```
sudo ./backup.sh restore [options] <backup-file-name>
```

The **restore** command overwrites the existing Persistent Volumes with data from the *<backup-file-name>*.

You can use the following parameters along with the **restore** command:

- --namespace or -n: Use this parameter to restore a specific namespace from the backup file. If you do not mention the namespace, then volumes from all the namespaces in the backup file are restored. You can map one namespace to another namespace by using colon (:). The syntax is:

```
--namespace <name of the namespace> [:<target-namespace>]
```

- --release or -r: Use this parameter if the helm release name of the server to which the backup is being restored is different than the helm release name of the server where the backup was taken. The syntax is:

  ```
  --release <backup-release>:<target-release>
  ```

- --volumes or -v: Use this parameter to specify the directory path of the Volumes. The syntax is:

  ```
  --volumes <path-of-the-directory>
  ```

- -k or --confirm: Use this parameter to skip the confirmation step.

For example,

```
sudo ./backup.sh restore --namespace test-system:new-test-system -r rel-1:rel-2 -k
 my-backup.tar.gz
```

This command restores the volumes that are backed up from the *rel-1* release in the *test-system* namespace to the *rel-2* release in the *new-test-system* namespace and skips the confirmation step.

8. Run the following command to restart the cluster and HCL OneTest™ Server:

   ```
   sudo systemctl start k3s
   ```

   **Note:**

   - The cluster and HCL OneTest™ Server might take some time to restart and until that time, you cannot access the HCL OneTest™ Server URL.
   - You can run the `kubectl get pods -A` command to verify that the Kubernetes environment is working. After a while, the status of the pods must be `Running` state.

**Results**

You have backed up or restored the user data for HCL OneTest™ Server.

**What to do next**

After you migrate the server data from a previous version to a newer version, the **Project Owner** must log in to HCL OneTest™ Server so that other members of the project can access the test assets in that project.

Related information

Server software upgrade on Ubuntu on page 57

## Uninstallation of the server software

When you no longer require HCL OneTest™ Server, you can uninstall the server software. You can uninstall the server software depending on the platform on which you installed the server software.

## Uninstalling the server software from Red Hat OpenShift

When you want to install a new version of server software or to reinstall when an installation fails, you can uninstall the server software and its components from the Red Hat OpenShift platform.

### Before you begin

You must have completed the following tasks:

- Installed HCL OneTest™ Server software.
- Closed HCL OneTest™ Server, any open web browsers, and all other applications that are enabled by HCL OneTest™ Server.
- **Optional**: Backed up data from the previous version of the product.

1. Open and log in to the terminal.
2. Run the following command to stop the workload that is running:

   ```
   oc delete all,cm,secret -lexecution-marker -n test-system
   ```

   **Remember:** `test-system` is the name of the namespace. If you created a namespace by using a different value, then you must use that value in place of the `test-system` in all the instances in this procedure.

3. Run the following command to uninstall the server software:

   ```
   helm uninstall {my-ots} -n test-system
   ```

   The PersistentVolumeClaims and PersistentVolumes that were created during the installation are not deleted automatically. If you reinstall the server software, the user data is reused unless you specifically delete those volumes.

   **Note:** You must substitute `{my-ots}` with the same release name that you used during the installation of the server software.

4. Run the following command to delete everything including user data contained in PersistentVolumeClaims and PersistentVolumes:

   ```
   oc delete project test-system
   ```

   **Important:** If you want to migrate data from a previous version to the latest version of the product, then you must have taken the backup of data before you run this command.

### Results

You have uninstalled the server software from the Red Hat OpenShift platform.

## Uninstalling the server software from Ubuntu

When you want to install a new version of server software or to reinstall when an installation fails, you can uninstall the server software and its components from the Ubuntu platform.

**Before you begin**

You must have completed the following tasks:

- Installed HCL OneTest™ Server software.
- Closed HCL OneTest™ Server, any open web browsers, and all other applications that are enabled by HCL OneTest™ Server.
- **Optional**: Backed up data from the previous version of the product.

1. Use an SSH session to log in to the Ubuntu server.
2. Run the following command to stop the workload that is running:

   ```
   kubectl delete all,cm,secret -lexecution-marker -n test-system
   ```

   **Remember:** `test-system` is the name of the namespace. If you created a namespace by using a different value, then you must use that value in place of the `test-system` in all the instances in this procedure.

3. Run the following command to uninstall the server software:

   ```
   helm uninstall {my-ots} -n test-system
   ```

   The PersistentVolumeClaims and PersistentVolumes that were created during the installation are not deleted automatically. If you reinstall the server software, the user data is reused unless you specifically delete those volumes.

   **Note:** You must substitute `{my-ots}` with the same release name that you used during the installation of the server software.

4. Run the following commands to delete the entire Kubernetes environment, including all user data:

   ```
   #Run the following command if you are on Ubuntu 18.04
   cd hcl-onetest-base
   sudo ./ubuntu-wipe.sh --confirm

   #Run the following command if you are on Ubuntu 20.04
   cd hcl-onetest-base
   sudo HOME=$HOME ./ubuntu-wipe.sh --confirm
   ```

After you run this command, the system returns to the same state as if `ubuntu-init.sh` was never run.

**Important:** If you want to migrate data from a previous version to the latest version of the product, then you must have taken the backup of data before you run this command.

**Results**

You have uninstalled the server software from the Ubuntu platform.

## Configuration of the server software

You can find information about the tasks that you must perform after you installed the server software to configure HCL OneTest™ Server.

The following table describes the tasks that you can perform as an administrator after you installed the server software:

| Requirement | Tasks | More information |
|---|---|---|
| Required | Review the default user management to know how the server software manages the users and their authentication. | User administration on page 68 |
| Required | Import Certificate Authority (CA) into a browser to prevent certificate errors when you access the HCL OneTest™ Server UI. | Importing Certificate Authority into a browser |
| Optional | Copy the third-party application Jars to Kubernetes to run API Suites that uses a transport. | Copying third-party application Jars to Kubernetes on page 83 |
| Optional | Change the password seed to provide enhanced security to HCL OneTest™ Server. | Changing the password seed on page 85 |
| Optional | Disable the server extensions that are enabled during the installation of the server software if you do not want the server extensions to run on the cluster to save server resources. | Disabling server extensions on page 88 |

## User administration

After you install HCL OneTest™ Server, you must consider how the software manages users and authentication.

You can review the default user management provided by the server and decide what additional controls you might want to add. If you manage users and authentication through an existing LDAP/AD server, you can review how to use that server to manage HCL OneTest™ Server users.

## Default user administration

You installed HCL OneTest™ Server and you want to know about how the software manages user access.

HCL OneTest™ Server uses Keycloak V9.0.2 (https://www.keycloak.org/) to manage and authenticate users.

If you manage and authenticate users by using an LDAP and Active Directory server, you can configure Keycloak to connect to that server. For more information, see LDAP user administration on page 70.

Keycloak uses the concept of a realm to manage and authenticate users. When you install the server software, a realm called testserver is created for you in Keycloak. All server users belong to this realm and when they log in to the server, they log into that realm.

As an administrator, it is important to consider the following points about the server administration:

- By default, there is no administrator for HCL OneTest™ Server.

  Such an administrator is required for accessing additional functions, which includes claiming ownership of server projects and unarchiving them. But you can assign administrative privileges to any user. You must do this by adding the admin role to the user in Keycloak.
- You must sign up a user that you want to be the administrator. You must go to the Login page at `https://<fully-qualified-dns-name>:443` and sign up.

  **Note:** Do not use that admin user to perform non-administration tasks. Instead, sign up another user.
- After you sign up the user that you want to be the administrator for HCL OneTest™ Server, you must log in to the Keycloak Admin Console at `https://<fully-qualified-dns-name>:443/auth/admin/` to make that user the server administrator.

  The default user name for the Keycloak administrator is `keycloak`. The password is randomly generated when the software is installed. You can see the password by using the following kubectl command:

  ```
  kubectl get secret -n test-system {my-ots}-keycloak-postgresql -o jsonpath="{.data.password}" |
   base64 --decode; echo
  ```

**Note:** You must substitute `{my-ots}` with the release name of your choice.

After you log in to the Keycloak Admin Console, from the Users page, you can search and select the user that you want to make an administrator. From the **Groups** tab, you can join the user to the **Admins** group.

For more information about assigning user roles, see Groups in the Keycloak documentation.

Now that you are the server administrator, it is important to consider the following points about the default user management and authentication:

- Minimum password length defaults to 8 characters
- Email verification of new users is turned off
- The Forgot Password feature is turned on by default but no instructions are sent to the user to reset their password
- Forgotten user passwords are changed by you if you do not enable Keycloak to send instructions to reset a password

You can review the following sections about changing the default authentication controls.

### Email settings

By default, the testserver realm sets the Forgot Password switch on. However, as an administrator, you must enable Keycloak to send an email to the user with instructions to reset their password. If you want to verify an email, you must also enable Keycloak to send an email to the user to verify their email address.

You must provide SMTP server settings for Keycloak to send an email. After you log in to the Keycloak Admin Console, see Email Settings in the Keycloak documentation.

Then, to set up the email verification, see Forgot Password in the Keycloak documentation.

### Password policy

By default, the testserver realm has a password policy where the minimum length of a password is 8. As an administrator, you can update password policies in Keycloak.

After you log in to the Keycloak Admin Console, see Password Policies in the Keycloak documentation.

### User password

If you did not enable Keycloak to send instructions to a user about how to reset a password, you must use the Keycloak Admin Console to change their password for them.

After you log in to the Keycloak Admin Console, see User Credentials in the Keycloak documentation.

### User deletion

When a user is inactive or no longer needs to access the server, you can delete that user.

After you log in to the Keycloak Admin Console, see Deleting Users in the Keycloak documentation.

## LDAP user administration

You installed HCL OneTest™ Server and you are using a Lightweight Directory Access Protocol and Active Directory HTTP server to manage users and authentication. You want to know the best practices to use that LDAP/AD server to manage user access to HCL OneTest™ Server.

HCL OneTest™ Server uses Keycloak V9.0.2 (https://www.keycloak.org/) to manage and authenticate users.

Existing user databases hold user credentials. Keycloak federates these existing external user databases through the concept of storage providers. By default, Keycloak supports an LDAP and Active Directory storage provider. By adding a storage provider, you can map LDAP user attributes into Keycloak. You can also configure more mappings.

Before you configure Keycloak to use an existing LDAP/AD provider, you must consider the following best practices:

- Set up your LDAP/AD provider as a read-only repository so that HCL OneTest™ Server cannot change it.
- Add and remove users in LDAP/AD and not the Keycloak local user database.
- Import and synchronize your LDAP/AD users to your Keycloak local database.
    - An import for an LDAP/AD user can fail if the LDAP/AD field chosen for the username mapping in Keycloak is not filled in for that user in LDAP/AD.
    - Filter LDAP/AD users by using the Custom User LDAP Filter, so you can import a subset of all your LDAP users. For example, you can set up a Server user group in LDAP and only import those users to Keycloak.
- Map a login style name, for example, `user1@server.com`, by using the `UserPrincipalName` attribute in LDAP/AD to a `username` in Keycloak. If you want the full name of the user as your login style, use the `cn` attribute in LDAP/AD.

    **Note:** The LDAP/AD user name attribute must match the LDAP/AD provider user name attribute (**Username LDAP attribute**) in Keycloak for the LDAP/AD provider to connect with Keycloak.

The following sections use these best practices to guide you to set up Keycloak to connect to your LDAP/AD HTTP server.

### LDAP provider selection in Keycloak

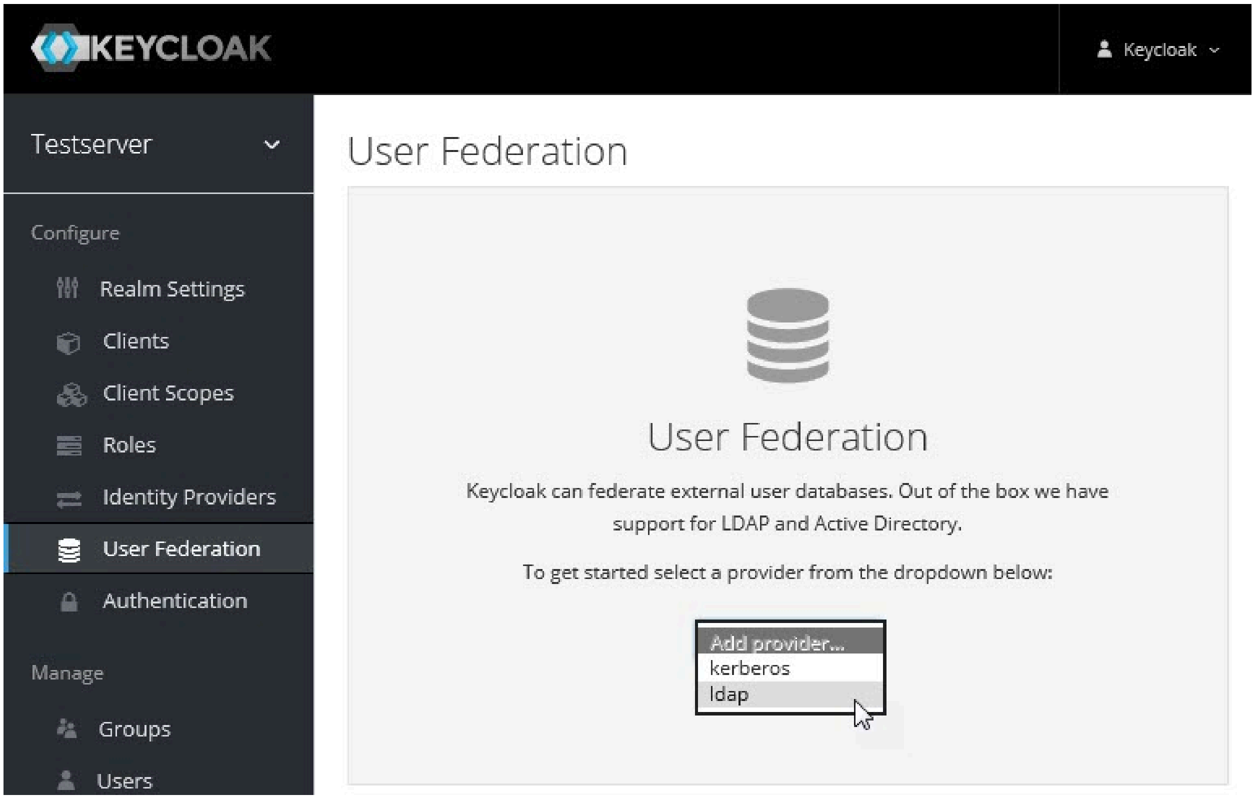
You can use the Keycloak Admin Console to add an LDAP/AD provider.

You can log in to the Keycloak Admin Console at `https://<fully-qualified-dns-nam>/auth/admin`. The default user name for the Keycloak administrator is `keycloak`. The password is randomly generated when you installed the HCL OneTest™ Server software. You can see the password by using the following kubectl command:

```
kubectl get secret -n test-system {my-ots}-keycloak-postgresql -o jsonpath="{.data.password}" | base64
 --decode; echo
```

**Note:** You must substitute `{my-ots}` with the release name of your choice.

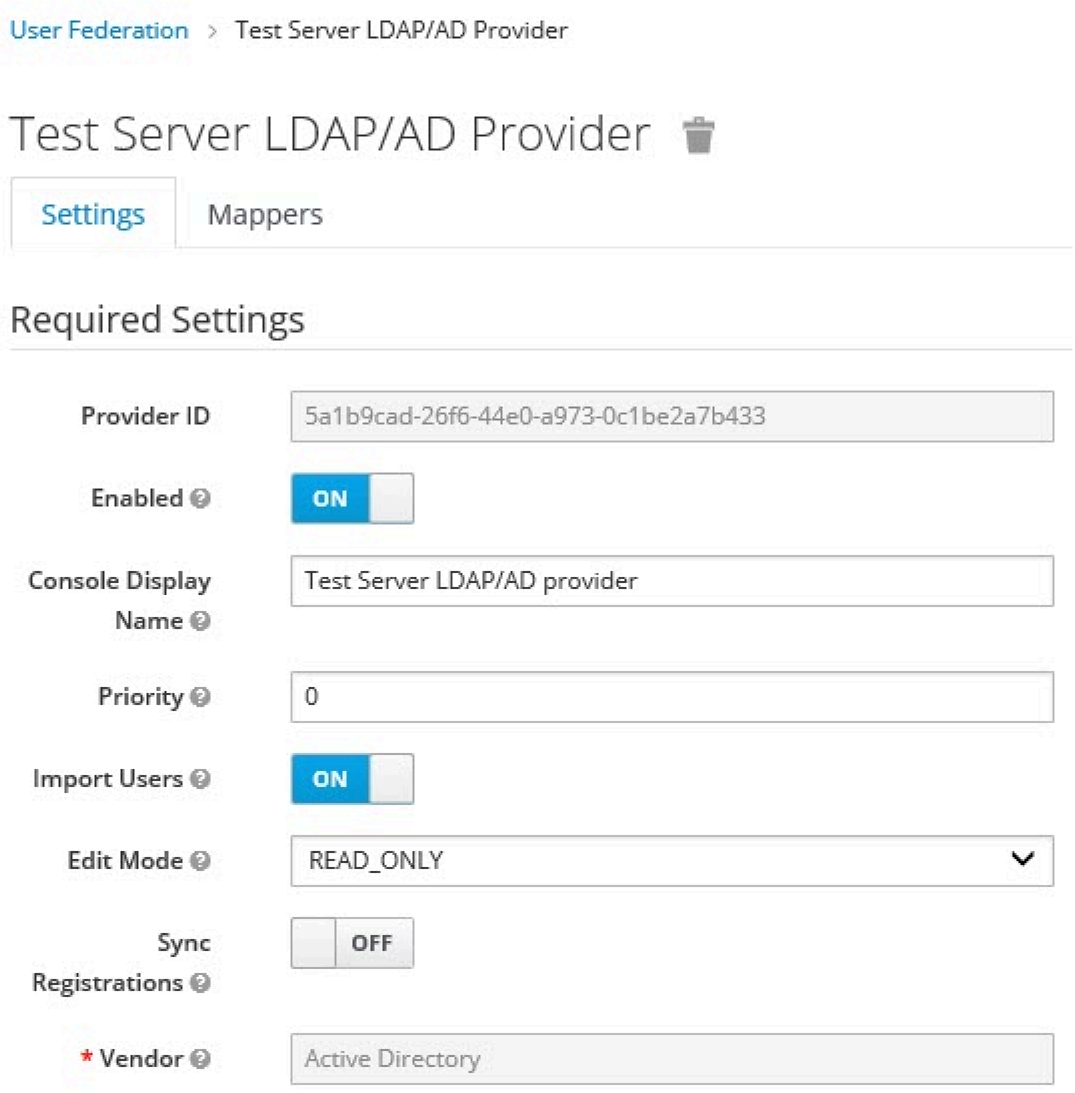
After you log in, you can go to the User Federation page to add your provider.



You want to select the provider named **ldap** from the list. From the Add User Federation Provider page, you can use a form to complete your LDAP/AD connection parameters.

### Required settings for a successful connection to your LDAP/AD provider

The form includes many fields and several fields include default values. Tables describe the important fields that you must complete to ensure a successful HTTP connection to your LDAP/AD provider.



User Federation > Test Server LDAP/AD Provider

Test Server LDAP/AD Provider

Required Settings

Provider ID: 5a1b9cad-26f6-44e0-a973-0c1be2a7b433

Enabled: ON

Console Display Name: Test Server LDAP/AD provider

Priority: 0

Import Users: ON

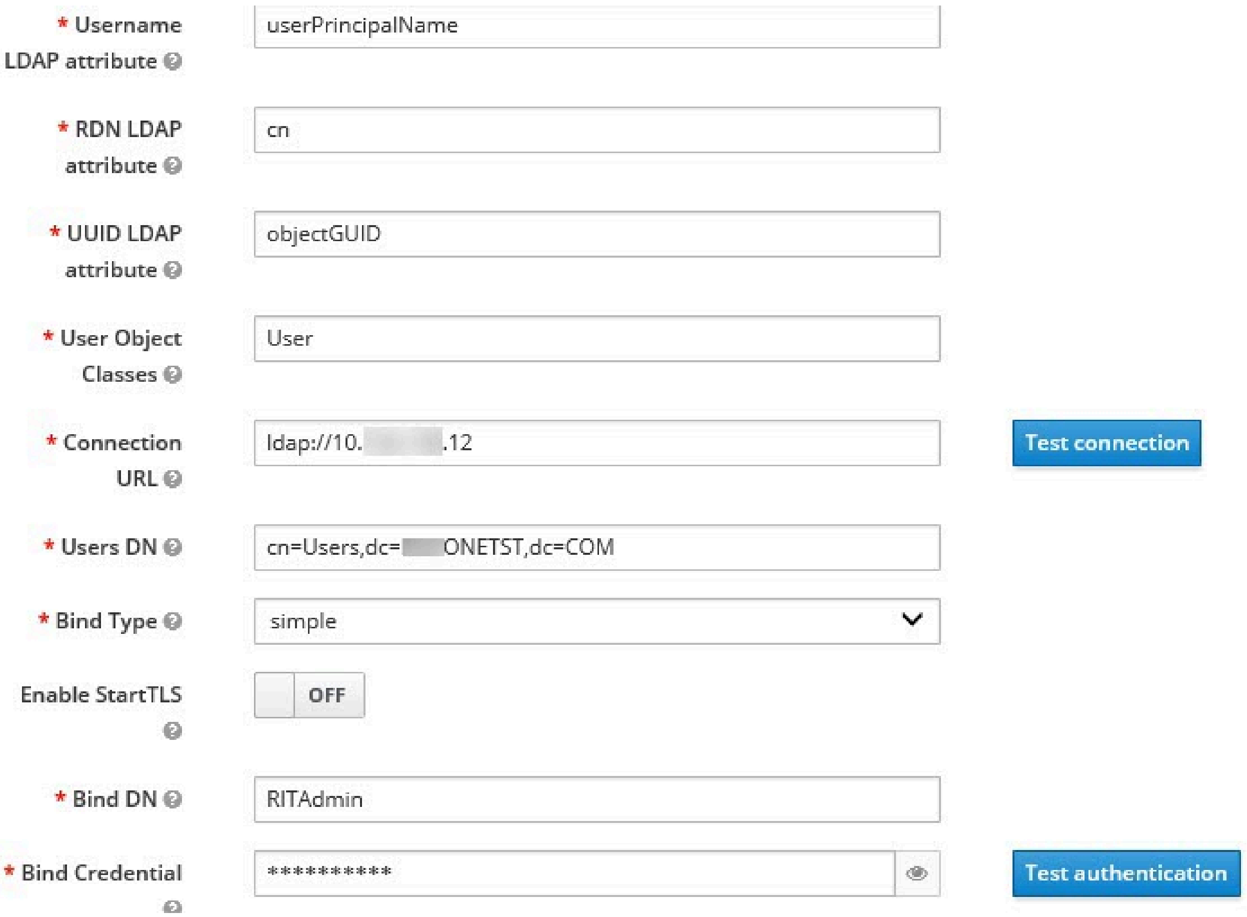
Edit Mode: READ_ONLY

Sync Registrations: OFF

* Vendor: Active Directory

| Field | Description |
|---|---|
| Console Display Name | You want to make your console name recognizable, for example, Test Server LDAP/AD Provider. This name is shown in the Keycloak Admin Console. |
| Priority | The priority must be set to 0 so that when a user logs in to HCL OneTest™ Server, Keycloak looks up the user first in the list of users managed in the LDAP/AD user database. If the user is not found in your LDAP/AD user database, Keycloak then looks up the user in the local Keycloak user database. |
| Import Users | You want to leave import users **ON** so that users in your LDAP/AD user database are synchronized (imported) automatically into the Keycloak local user database. The import uses the settings that you are defining now.<br>After the initial import, when you add a user to your LDAP/AD user database and that user logs in to HCL OneTest™ Server, the LDAP/AD provider imports the LDAP/AD user |

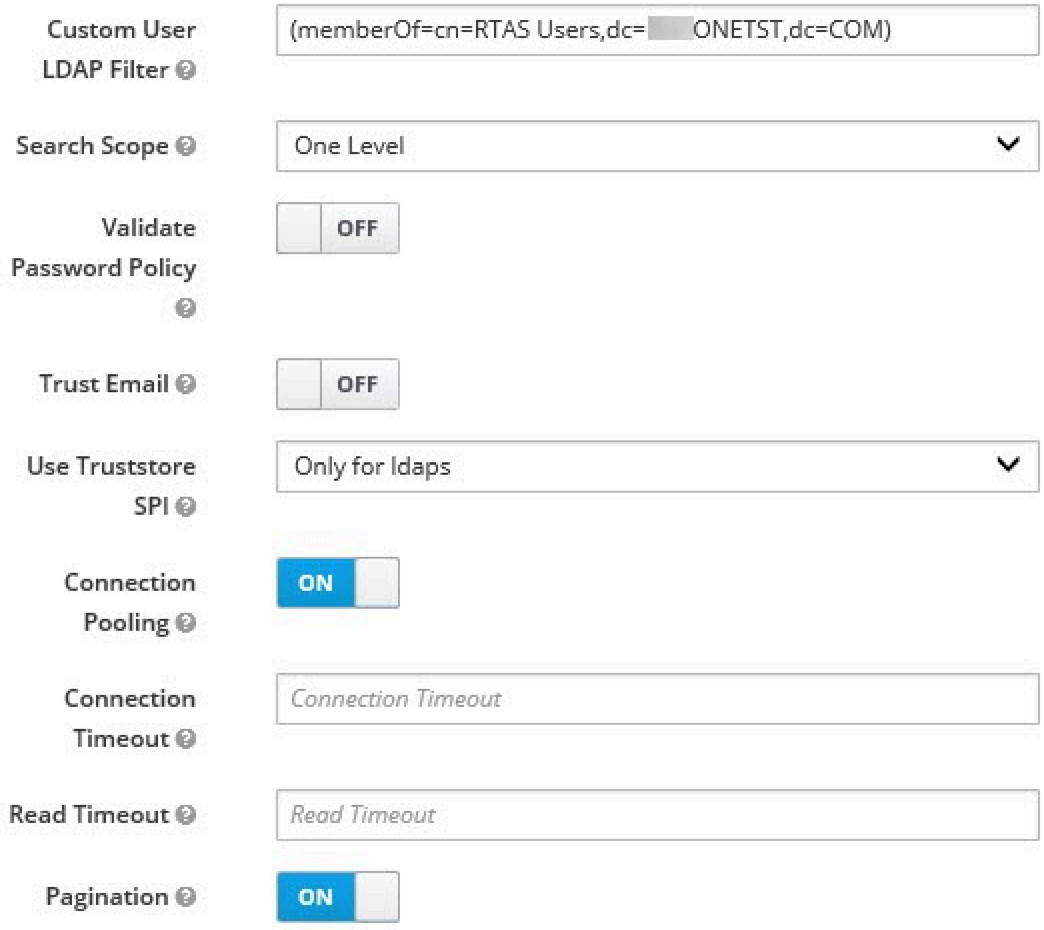
| Field | Description |
|---|---|
| | into the Keycloak user database and authenticates the user against the LDAP/AD password.<br>Similarly, after the initial import, when you remove a user from your LDAP/AD user database that user cannot log in to HCL OneTest™ Server.<br>**Note:** For consistent user management, continue to manage users in your LDAP/AD user database. |
| Edit Mode | Make sure that edit mode is set to **READ_ONLY**.<br>This setting means that your LDAP/AD user database is read only. No user data defined through the mapping of attributes in Keycloak such as the `username` is written back from the Keycloak user database to your LDAP/AD user database.<br>This read-only setting also means that a new user cannot sign up from HCL OneTest™ Server as no user data can be written to your LDAP/AD user database. |
| Vendor | You want to select **Active Directory** from the list of vendors. Many fields complete with default values based on this selection. |



| | | |
|---|---|---|
| * Username LDAP attribute | userPrincipalName | |
| * RDN LDAP attribute | cn | |
| * UUID LDAP attribute | objectGUID | |
| * User Object Classes | User | |
| * Connection URL | ldap://10.[redacted].12 | Test connection |
| * Users DN | cn=Users,dc=[redacted]ONETST,dc=COM | |
| * Bind Type | simple | |
| Enable StartTLS | OFF | |
| * Bind DN | RITAdmin | |
| * Bind Credential | ********** | Test authentication |

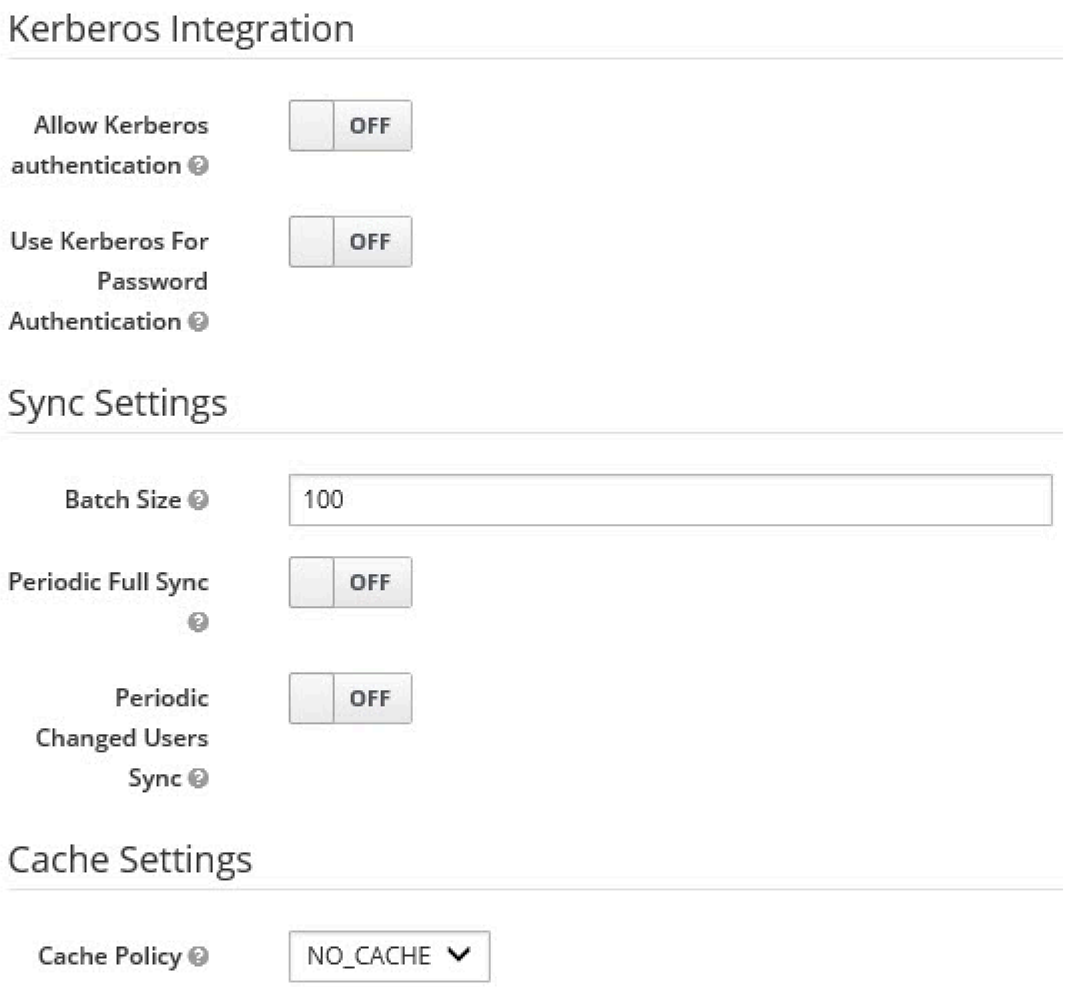
| Field | Description |
|---|---|
| Username LDAP attribute | You can use the default value `cn` for the `username`, which is first name, last name, or you can use `userPrincipalName`, which is username@domain.<br>You might want a login that matches your company style. For example, you might prefer joetester@mycompany.com instead of Joe Tester.<br>If you do make this change the `username` that you specify now must match the `username` in the Mapper. Keycloak makes this change for you. |
| User Object Classes | Because LDAP user records are found based on a user object class, you must set the User Object Classes to `User`. |
| Connection URL | So that Keycloak can connect to your LDAP/AD user database, you must enter your LDAP/AD URL, for example, `ldap://<hostname>.<domain>`<br>Make sure that you test the connection and confirm that the connection is successful. |
| Users DN | You must provide the directory where the LDAP users are listed, for example, `cn=Users,dc=MYCOMPANY,dc=COM`. |

| Field | Description |
|---|---|
| Bind DN<br>Bind Credential | You must also provide the LDAP/AD user database administrator user ID for **BIND DN** and password for the **BIND Credential**. These credentials are used by Keycloak to access the LDAP/AD user database.<br>Make sure that you test the authentication and confirm that the authentication is successful. |



| | |
|---|---|
| Custom User LDAP Filter | (memberOf=cn=RTAS Users,dc=[redacted]ONETST,dc=COM) |
| Search Scope | One Level |
| Validate Password Policy | OFF |
| Trust Email | OFF |
| Use Truststore SPI | Only for ldaps |
| Connection Pooling | ON |
| Connection Timeout | Connection Timeout |
| Read Timeout | Read Timeout |
| Pagination | ON |

| Field | Description |
|---|---|
| Custom User LDAP Filter | You can filter your LDAP/AD users to import a subset of all your LDAP users.<br>For example, you can set up a testers user group for your LDAP/AD user database such that only those users are imported to Keycloak.<br>`(memberOf=cn=TESTERS,dc=MYCOMPANY,dc=COM)` |



Kerberos Integration

Allow Kerberos authentication: OFF

Use Kerberos For Password Authentication: OFF

Sync Settings

Batch Size: 100

Periodic Full Sync: OFF

Periodic Changed Users Sync: OFF

Cache Settings

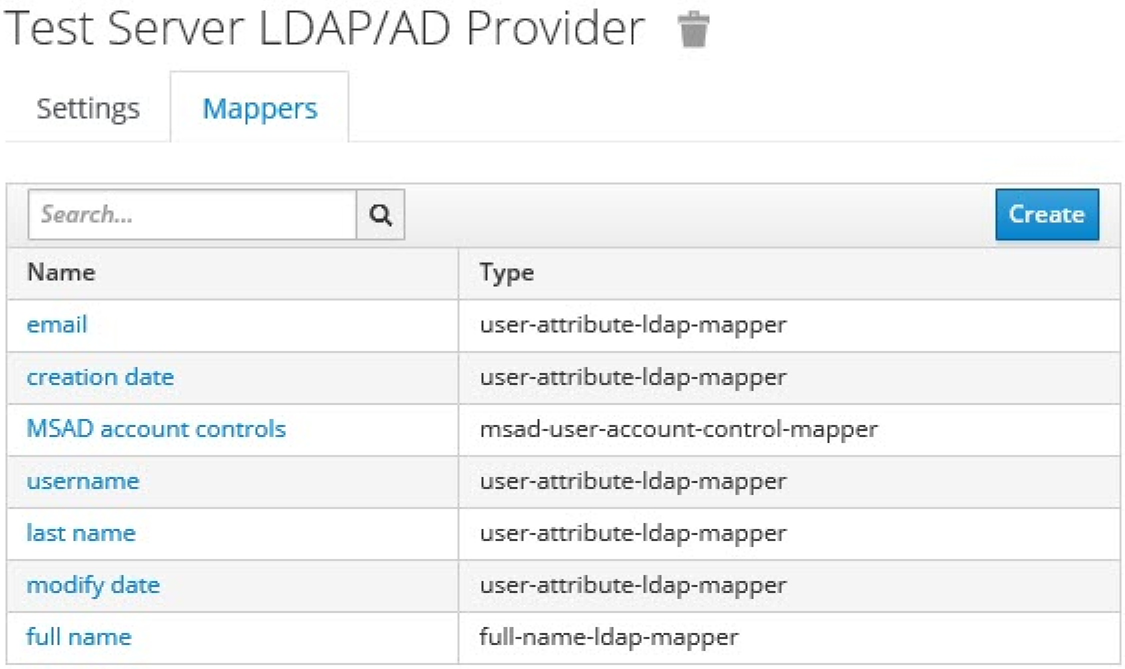
Cache Policy: NO_CACHE

| Field | Description |
|---|---|
| Batch size | You can change the number of users to import in a single transaction by using the batch size. |
| Cache Policy | If you have many users or performance concerns, you can change the cache policy to no cache. |

You can save the settings, which creates your LDAP/AD provider.

### Mappers

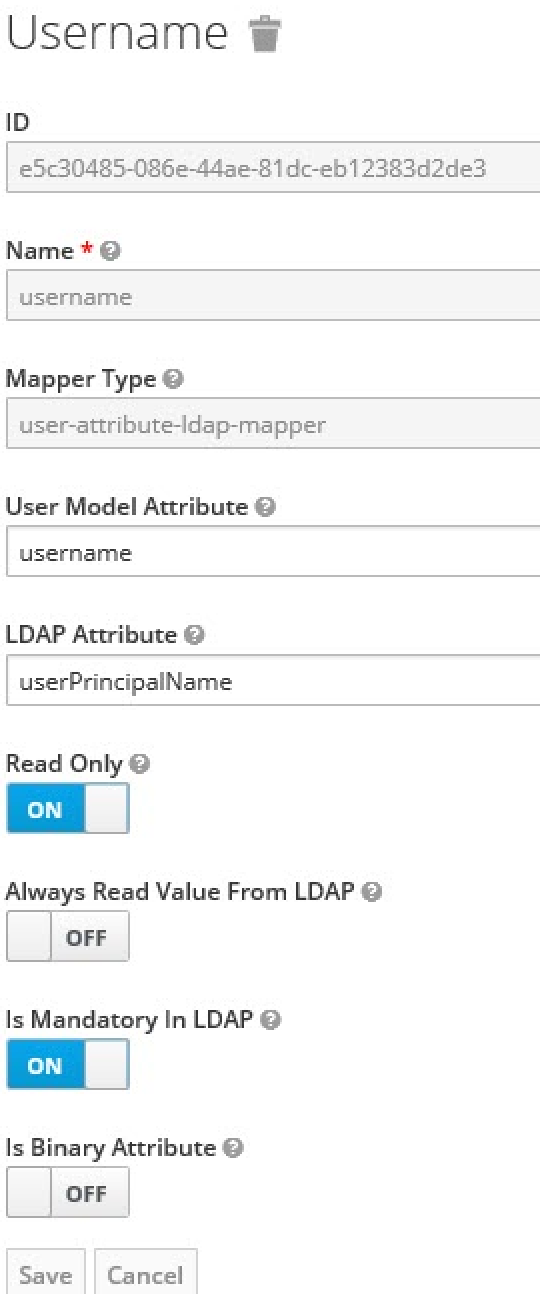
Keycloak uses mappers to map the user attributes defined in the Keycloak user model such as **username** and **email** to the corresponding user attributes in the LDAP/AD user database. By default, when you saved your settings and created your LDAP/AD provider, the following mappers were created.



| Name | Type |
|---|---|
| email | user-attribute-ldap-mapper |
| creation date | user-attribute-ldap-mapper |
| MSAD account controls | msad-user-account-control-mapper |
| username | user-attribute-ldap-mapper |
| last name | user-attribute-ldap-mapper |
| modify date | user-attribute-ldap-mapper |
| full name | full-name-ldap-mapper |

The username attribute that you specified in the **Username LDAP attribute** must match the **username** t attribute defined in the Keycloak mapper for the LDAP/AD user database to connect with Keycloak.

Because you changed the **Username LDAP attribute** from the default value `cn` to `userPrincipalName`, Keycloak made the same change in the mapper called **username** to match.



Username

ID
e5c30485-086e-44ae-81dc-eb12383d2de3

Name *
username

Mapper Type
user-attribute-ldap-mapper

User Model Attribute
username

LDAP Attribute
userPrincipalName

Read Only
ON

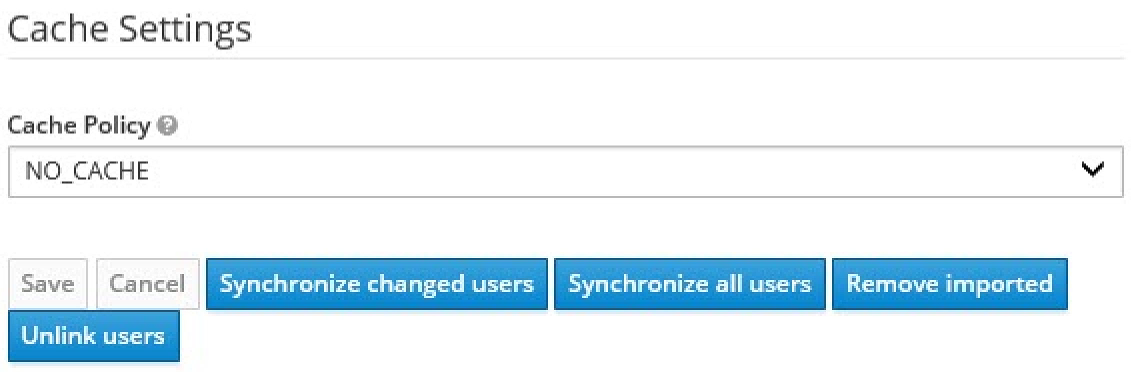
Always Read Value From LDAP
OFF

Is Mandatory In LDAP
ON

Is Binary Attribute
OFF

### User synchronization

You must import all users from your LDAP/AD user database by using the option to **Synchronize all users**. Users are imported based on your saved settings when you set up your LDAP/AD provider.

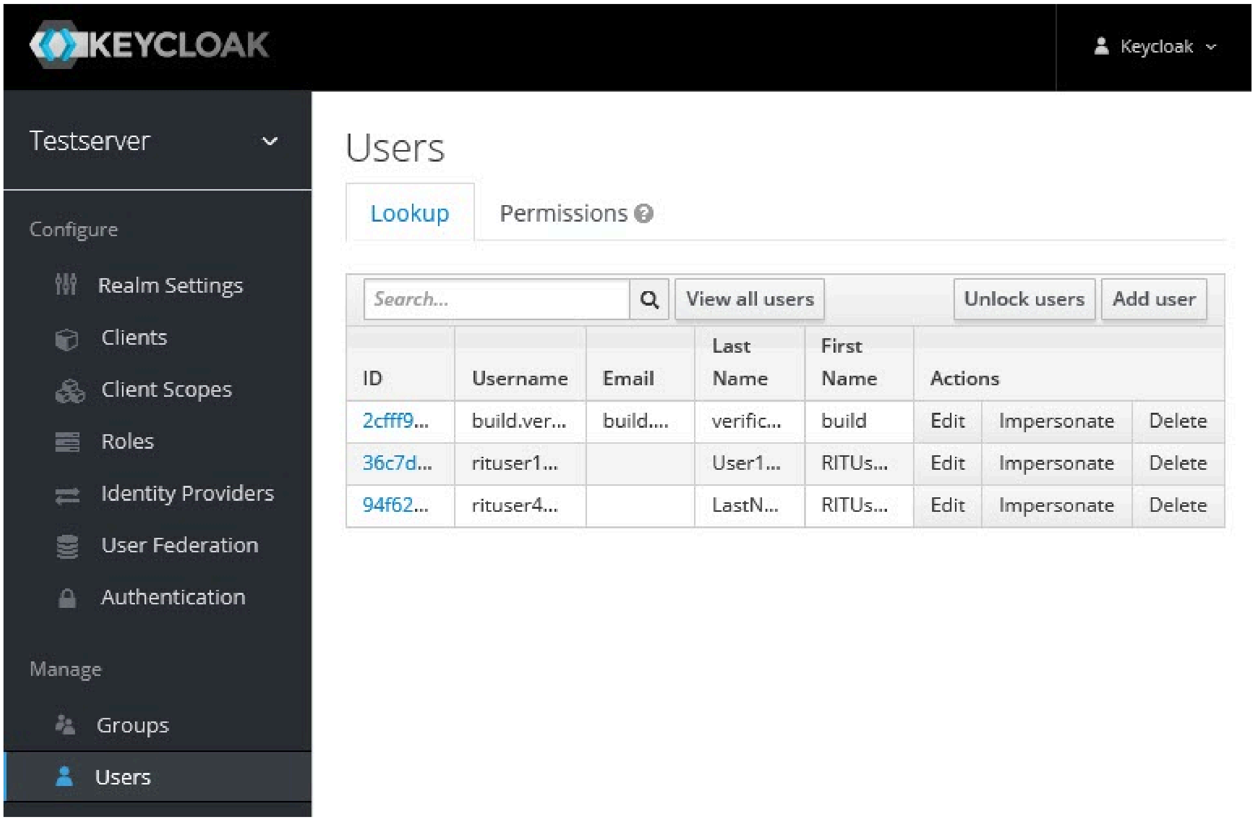


Cache Settings

Cache Policy: NO_CACHE

A successful import is followed by a success message with the number of users imported. A failed import typically results when there is a mismatch between user attributes in the Keycloak user database and the LDAP/AD user database.

You can view all the LDAP/AD database users that were imported and authenticated from the Users page in the Keycloak Admin Console.



Users

| ID | Username | Email | Last Name | First Name | Actions | | |
|---|---|---|---|---|---|---|---|
| 2cfff9... | build.ver... | build.... | verific... | build | Edit | Impersonate | Delete |
| 36c7d... | rituser1... | | User1... | RITUs... | Edit | Impersonate | Delete |
| 94f62... | rituser4... | | LastN... | RITUs... | Edit | Impersonate | Delete |

Users are listed with ID, Username, Email, Last Name, and First Name. The ID is generated by Keycloak. The value of the other attributes is fetched from the LDAP/AD user database by using mappers.

## Managing account settings

You might want to change your password or other settings for your account.

### About this task

As an account owner, you can view your account settings, which include your user name, email, and first and last name. You can also change all of these settings except the user name. Changing your password is also possible from your account settings.

**Note:** Account settings cannot be changed if HCL OneTest™ Server was configured to use an LDAP/AD provider. For more information about server security, see the related links.

- Edit your account details by following these steps:

  1. Click the **User** icon from the menu bar, and select **Account Settings**.

     The Edit account page is displayed.

  2. Edit your user details and save the changes. You can also reset your changes. If you cancel your changes, you return to the Home page.

- Change your password by following these steps:

  1. Click the **User** icon from the menu bar, and select **Account Settings**.

     The Edit account page is displayed.

  2. Click **Change password**.

     The Change password page is displayed.

  3. Type your current password followed by your new password

  4. Confirm the new password by typing it again and save the changes. You can also reset your changes. If you cancel your changes, you return to the Home page.

Related information

Default user administration on page 68

## Certificate authority: Importing and extending lists

When you want browsers and other applications to access HCL OneTest™ Server, you must enable browsers to trust the certificate authority (CA) of HCL OneTest™ Server. For some applications that integrate with HCL OneTest™ Server, you must also extend the CA lists that are trusted by the applications.

You can find the following tasks to import the certificate authority in the browser that you want to use:

- Importing Certificate Authority into the Google Chrome browser on page 81
- Importing Certificate Authority into the Microsoft Edge browser on page 81
- Importing Certificate Authority into the Mozilla Firefox browser on page 82

## Importing Certificate Authority into the Google Chrome browser

You must import certificate authority into the Google Chrome browser to prevent certificate errors when accessing the HCL OneTest™ Server UI.

**Before you begin**

You must have completed the following tasks:

- Saved a certificate authority (CA) during the installation of HCL OneTest™ Server.

  You can run the following command to get the certificate from the system:

  ```
  kubectl get secret ingress -n test-system -o jsonpath={.data.ca\\.crt} | base64 -d
  ```

  **Remember:** *test-system* is the name of the namespace. If you created a namespace by using a different value, then you must use that value in place of the *test-system*.

- Installed the browser that you want to use to access HCL OneTest™ Server.

1. Open a Google Chrome browser.
2. Navigate to **Customize and control Google Chrome > Settings**.
3. Click **Security** from the **Privacy and security** pane.
4. Click **Manage certificates**, and then **Import**.
5. Click **Next** in the **Certificate Import Wizard** window.
6. Click **Browse** and select the CA that you want to import, and then click **Next**.
7. Select the **Place all certificates in the following store** option to store the CA securely.
8. Click **Browse**, and then select **Trust Root Certification Authorities** as certificate store.
9. Click **OK**, and then **Finish**.

**Results**

You have imported the CA into the Google Chrome browser.

## Importing Certificate Authority into the Microsoft Edge browser

You must import Certificate Authority (CA) into the Microsoft Edge browser to prevent certificate errors when accessing the HCL OneTest™ Server UI.

**Before you begin**

You must have completed the following tasks:

- Saved a certificate authority (CA) during the installation of HCL OneTest™ Server.

  You can run the following command to get the certificate from the system:

  ```
  kubectl get secret ingress -n test-system -o jsonpath={.data.ca\\.crt} | base64 -d
  ```

  **Remember:** *test-system* is the name of the namespace. If you created a namespace by using a different value, then you must use that value in place of the *test-system*.

- Installed the browser that you want to use to access HCL OneTest™ Server.

1. Type `certmgr.msc` in the **Start** menu **Search** filed, and then press **Enter**.
2. Expand **Trusted Root Certification Authorities**, and then select **Certificate**.
3. Right-click on the empty space, and then select **All tasks > Import**.
4. Click **Next** in the **Certificate Import Wizard** window.
5. Click **Browse** and select the CA that you want to import, and then click **Next**.
6. Select the **Place all certificates in the following store** option to store the CA securely.
7. Click **Browse**, and then select **Trust Root Certification Authorities** as certificate store.
8. Click **Next**, and then **Finish** to import the certificate.

### Results

You have imported the CA into the Microsoft Edge browser.

## Importing Certificate Authority into the Mozilla Firefox browser

You must import Certificate Authority (CA) into the Mozilla Firefox browser to prevent certificate errors when accessing the HCL OneTest™ Server UI.

### Before you begin

You must have completed the following tasks:

- Saved a certificate authority (CA) during the installation of HCL OneTest™ Server.

  You can run the following command to get the certificate from the system:

  ```
  kubectl get secret ingress -n test-system -o jsonpath={.data.ca\\.crt} | base64 -d
  ```

  **Remember:** *test-system* is the name of the namespace. If you created a namespace by using a different value, then you must use that value in place of the *test-system*.

- Installed the browser that you want to use to access HCL OneTest™ Server.

1. Open a Mozilla Firefox browser.
2. Enter "`about:preferences#privacy`" in the address bar, and then press **Enter**.
3. Locate and click the **View Certificates** option.

4. Select **Authorities** tab, and then click **Import**.
5. Select the CA that you want to import, and then click **Open**.
6. Select **Trust this CA to identify websites** in the **Downloading Certificate** window.
7. Click **OK** to import the CA into the browser, and then close the window.
8. Restart Firefox.

### Results

You have imported the CA into the Mozilla Firefox browser.

## Copying third-party application Jars to Kubernetes

You can run API Suites in a project on HCL OneTest™ Server. If the API Suite uses a transport and the transport requires third-party application `Jar` files for a successful run, you must ensure that the third-party application `Jar` files are available at the test run time. To achieve this, you must copy the third-party application `Jar` files to the computer where HCL OneTest™ Server is installed on Kubernetes.

### Before you begin

If you want to copy the third-party application `Jar` files to the remote Docker host, see Copying third-party application Jars to a remote Docker host on page 229.

You must have server administrator or cluster administrator privileges.

You must have completed the following tasks:

- Identified the third-party application `Jar` files that are required and copied the files. See Test run considerations for API Suites on page 208.
- Copied the third-party application `Jar` files from HCL OneTest™ API to the directory or folder in Kubernetes from where you can run the kubectl commands.

### About this task

You can copy the third-party application `Jar` files to the folder that is specific for the application under the `/data/<application_name>` folder. You need not extract the files.

You can perform this task any time after you have installed HCL OneTest™ Server and you plan to run an API Suite that meets any of the following conditions:

- The API Suite uses transports and the transports require third-party application `Jar` files to be available at the test run time.
- The API Suite in the project has a results database configured.

You can copy the JDK version `Jar` files that you have used to create the JUnit tests.

You can get help on the kubectl commands by running the command: kubectl cp --help from the `/kube` directory. For more information, refer to the kubectl documentation.

1. Use the following table to find the name of the folder that corresponds to the specific third-party application for the transport used in the API Suite.

    **Table 2. Name of the folder for the application**

    | Application | Name of the folder to use |
    |---|---|
    | Camel | Camel |
    | CentraSite | CentraSite |
    | CICS | CICS |
    | Coherence | Coherence |
    | Database | JDBC |
    | IMS | IMS |
    | Integra | Integra |
    | JMS | JMS |
    | SAP RFC | SAP |
    | Software AG Universal Messaging | SoftwareAGUM |
    | TIBCO EMS | TIBCO |
    | TIBCO Rendezvous | |
    | TIBCO SmartSockets | |
    | WebSphere Application Server Service Integration Bus (SiBus) | WAS |
    | WebLogic | WebLogicJMX |
    | Software AG webMethods | webMethods |
    | WebSphere MQ | WMQ |

2. Open the command prompt from the `/kube` directory and copy the `Jar` files to the folder that corresponds to the application by using the following command:

    ```
    kubectl cp <compressed_files> test-system/{my-ots}-userlibs-0:/data/<application_name>/
    ```

    For example, if you are copying the Database `Jar` files, then the name of the folder is *JDBC* and the command to use is:

    ```
    kubectl cp mysql-connector-java.jar test-system/{my-ots}-userlibs-0:/data/JDBC/
    ```

**Note:** You must substitute `{my-ots}` with the release name of your choice.

3. Verify whether the `Jar` files are copied by running the following command:

   ```
   kubectl exec {my-ots}-userlibs-0 -n test-system -- ls /data/<folder_created>
   ```

   For example, if you copied the Database `Jar` files, `mysql-connector-java.jar` to the folder *JDBC*, then the command to verify is:

   ```
   kubectl exec {my-ots}-userlibs-0 -n test-system -- ls /data/JDBC
   ```

   The following information is displayed for the `JDBC` folder:

   ```
   mysql-connector-java.jar
   ```

   **Note:** You must substitute `{my-ots}` with the release name of your choice.

### Results

You have successfully copied the third-party application `Jar` files to the folder in Kubernetes.

### What to do next

You can configure the API Suite run. See Configuring an API Suite run on page 249.

## Changing the password seed

As an administrator, you can change the password seed that is used when you install the server software to provide enhanced security to HCL OneTest™ Server.

### Before you begin

You must have completed the following tasks:

- Installed HCL OneTest™ Server. See Installation of the server software on page 35.
- Installed the JSON command-line tool, jq, and ensured that the jq is in your environment PATH. For information about jq, refer to jq documentation.
- Installed the Curl command line tool. For more information refer to curl documentation.

### About this task

When you install HCL OneTest™ Server, you supply a password seed when you run the `helm install` command. This password seed is used to generate several Kubernetes secrets. Kubernetes Secrets can contain the following information:

- The authentication credentials for micro-services.
- An encryption key for the user-created secrets collection or other secrets.

When you change the password seed for HCL OneTest™ Server, you must consider the following scenarios:

- HCL OneTest™ Server cannot communicate until you reconcile the passwords which are in Kubernetes Secrets by using the old and a new password seed.
- Users cannot read secret collections or other secrets that they have created in HCL OneTest™ Server until you re-encrypt them using a new password seed.

**Important:** You must provide an offline token and old password seed that you used during the installation of server software to re-encrypt user secrets.

1. Run the following command to change the password seed for HCL OneTest™ Server:

   ```
   helm upgrade {my-ots} ./hcl-onetest-server -n test-system \
   --reuse-values \
   --set global.hclOneTestPasswordAutoGenSeed= {my-new-super-secret}
   ```

   **Notes:** You must substitute the value of the following variables with the actual value in the command:

   - `{my-ots}` with the release name that you used during the installation of the server software.
   - `{my-new-super-secret}` with a new value of your choice as the password seed.
   - You must run the following `helm upgrade` command from the same directory where the `helm install` command was run during the installation of the server software. Because the upgrade is dependent on the helm charts and .yaml file values used during the run time of the `helm install` command.

2. Run the following script to generate new server secrets from the updated password seed and to save them to the persistent storage:

   ```
   ./hcl-onetest-server/files/reconcile-secrets.sh -n test-system {my-ots}
   ```

3. Run the following command to restart all the pods:

   ```
   kubectl delete pods -n test-system \
     -lapp.kubernetes.io/instance={my-ots} \
     -lapp.kubernetes.io/managed-by=Helm
   ```

4. Run the following commands to re-encrypt the user-created secrets collection or other secrets by providing the old password seed:

   ```
   export ACCESS_TOKEN=$(curl -k -X POST ${SERVER_URL}/rest/tokens/ \
   -H "Content-Type: application/x-www-form-urlencoded" \
   -H "accept: application/json" \
   -d "refresh_token=${OFFLINE_TOKEN}" | jq -r '.access_token')
   ```

```
curl -k -X POST {$SERVER_URL}/rest/secrets/re-encrypt/ \
     -H "Authorization: Bearer $ACCESS_TOKEN" \
     -H "Content-Type: application/json" \
     -d "{\"type\":\"helm\",\"password_auto_gen_seed\":\"${OLD_SEED}\"}"
```

**Note:** You must substitute the value of the following variables with the actual value in the following commands:

- `{SERVER_URL}` with the URL of your HCL OneTest™ Server UI.
- `{OFFLINE_TOKEN}` with the offline token that belongs to a user with the administrator role.
- `{OLD_SEED}` with the previous password seed that you used during the installation of the server software.

5. Run the following command to display the log file of the gateway pod:

```
kubectl logs {my-ots}-gateway-abcdefghij-abcde -n test-system
```

**Note:** You must substitute the value of the following variables with the actual value in the command:

- `{my-ots}` with the release name that you used during the installation of the server software.
- `abcdefghij-abcde` with an identifier that is assigned to the name of the gateway pod.

  You can run the `kubectl get pods -n test-system` command to obtain the identifier that is assigned to the gateway pod.

**Result**

The following message is displayed when re-encryption is completed:

```
reEncrypt complete. StringyReEncryptor [total=100, fixed=100, broken=0, noop=0]
```

**Results**

You have successfully changed the password seed for HCL OneTest™ Server.

## Server extensions

You can run Postman, JMeter, or JUnit tests in HCL OneTest™ Server by enabling the server extensions, if they were disabled. The server extensions are enabled at the time of installation of the server software.

When the server extensions are enabled, you can perform the following tasks on HCL OneTest™ Server:

- Add repositories that contain the test assets for the Postman collections, JMeter tests, or JUnit tests.
- You can configure a test run for the following test types:

    - **Postman**
    - **JMeter**
    - **JUnit**

Related information

Configuring a JMeter test run on page 289

Configuring a Postman test run on page 297

Configuring a JUnit test run on page 293

## Disabling server extensions

When you do not want the server extensions to run on the cluster to save server resources or when you no longer want to run any of the tests supported by the server extensions, you can disable them.

### Before you begin

You must have ensured that the server extension that you want to disable is enabled and running.

**Note:** You can verify the status by running the kubectl get pods -n test-system command. The status of an enabled server extension is displayed as `Running`.

### About this task

You can disable server extensions depending on the operating system on which you have installed the server software:

- Disabling server extensions on Ubuntu on page 88
- Disabling server extensions on OpenShift on page 89

## Disabling server extensions on Ubuntu

1. Use an SSH session to log in to the Ubuntu server.
2. Run the following command to disable all the three extensions:

```
helm upgrade {my-ots} -n test-system ./hcl-onetest-server \
--reuse-values \
--set global.hclOneTestPostmanEnabled=false \
--set global.hclOneTestJMeterEnabled=false \
--set global.hclOneTestJUnitEnabled=false
```

**Note:** You must substitute {my-ots} with the release name that you provided during the installation of the server software.

If you want to disable any specific extension, then you can set the respective helm parameter value to *false*. For example, to disable the JMeter extension alone, then you can run the following command:

```
helm upgrade {my-ots} -n test-system ./hcl-onetest-server \
--reuse-values \
--set global.hclOneTestJMeterEnabled=false
```

3. Run the following command to verify the status of the extensions:

```
kubectl get pods -n test-system
```

**Note:** The pod for each server extension that you disabled no longer displays.

## Disabling server extensions on OpenShift

1. Open and log in to the terminal.
2. Run the following command to disable all the three extensions:

```
helm upgrade {my-ots} -n test-system ./hcl-onetest-server \
--reuse-values \
--set global.hclOneTestPostmanEnabled=false \
--set global.hclOneTestJMeterEnabled=false \
--set global.hclOneTestJUnitEnabled=false
```

**Note:** You must substitute {my-ots} with the release name that you provided during the installation of the server software.

If you want to disable any specific extension, then you can set the respective helm parameter value to *false*. For example, to disable the JMeter extension alone, then you can run the following command:

```
helm upgrade {my-ots} -n test-system ./hcl-onetest-server \
--reuse-values \
--set global.hclOneTestJMeterEnabled=false
```

3. Run the following command to verify the status of the extensions:

```
oc get pods -n test-system
```

**Note:** The pod for each server extension that you disabled no longer displays.

### Results

You have disabled the required server extensions on HCL OneTest™ Server.

## Enabling server extensions

If you find that the server extension is disabled and you want to either add the test assets to the repository or run the tests supported by the server extension, you must enable the server extension.

**About this task**

You can enable server extensions depending on the operating system on which you have installed the server software:

- Enabling server extensions on Ubuntu on page 90
- Enabling server extensions on OpenShift on page 90

## Enabling server extensions on Ubuntu

1. Use an SSH session to log in to the Ubuntu server.
2. Run the following command to enable all the three extensions:

```
helm upgrade {my-ots} -n test-system ./hcl-onetest-server \
--reuse-values \
--set global.hclOneTestPostmanEnabled=true \
--set global.hclOneTestJMeterEnabled=true \
--set global.hclOneTestJUnitEnabled=true
```

**Note:** You must substitute {my-ots} with the release name that you provided during the installation of the server software.

If you want to enable any specific extension, then you can set the respective helm parameter value to *true*. For example, to enable the Postman extension alone, then you can run the following command:

```
helm upgrade {my-ots} -n test-system ./hcl-onetest-server \
--reuse-values \
--set global.hclOneTestPostmanEnabled=true
```

3. Run the following command to verify that all the three or specific extensions are enabled:

```
kubectl get pods -n test-system
```

**Result**

The status of the specific extension or all the three extensions in the pod is displayed as `Running`.

## Enabling server extensions on OpenShift

1. Open and log in to the terminal.
2. Run the following command to enable all the three extensions:

```
helm upgrade {my-ots} -n test-system ./hcl-onetest-server \
--reuse-values \
--set global.hclOneTestPostmanEnabled=true \
--set global.hclOneTestJMeterEnabled=true \
--set global.hclOneTestJUnitEnabled=true
```

**Note:** You must substitute {my-ots} with the release name that you provided during the installation of the server software.

If you want to enable any specific extension, then you can set the respective helm parameter value to *true*. For example, to enable the Postman extension alone, then you can run the following command:

```
helm upgrade {my-ots} -n test-system ./hcl-onetest-server \
--reuse-values \
--set global.hclOneTestPostmanEnabled=true
```

3. Run the following command to verify that all the three or specific extensions are enabled:

   ```
   oc get pods -n test-system
   ```

   **Result**

   The status of the specific extension or all the three extensions in the pod is displayed as `Running`.

**Results**

You have enabled the required server extensions on HCL OneTest™ Server.

Related information

Configuring a JMeter test run on page 289

Configuring a Postman test run on page 297

Configuring a JUnit test run on page 293

# Team space administration

You can find information about team spaces including the initial team space on HCL OneTest™ Server along with the tasks that you can perform to administer a team space.

You can find the following information about team spaces:

- Team Space overview on page 91
- Creating a team space
- Tasks in a team space on page 93
- Tasks in projects in a team space on page 97
- License management on page 99
- Configuring email notifications

## Team Space overview

You can find information about the initial Team Space that is displayed when you log in to HCL OneTest™ Server. After you install HCL OneTest™ Server, you enter a Team Space. Team Space is a logical partition of HCL OneTest™ Server to share a single installation among multiple teams.

The initial Team Space is the only logical partition created on an instance of HCL OneTest™ Server. The Team Space facilitates users of the same team space to work together.

A Team Space is an exclusive, separate, and secure space on HCL OneTest™ Server. Apart from being a container for projects, a Team Space facilitates the registration of remote Docker hosts, and viewing the registered agents, Dockers, or intercepts.

## Setting up a Team Space

The initial Team Space is the only Team Space that is available, and you cannot set up any other Team Space.

When licensed users log in to HCL OneTest™ Server, they can view the **Projects** page of the initial Team Space. If the user does not have a license, then they can view the **License Configuration** page. Until a user is configured as a licensed user, the user cannot perform any operations on HCL OneTest™ Server. If you are a Server Administrator, you must first configure the licensed users so that the licensed users are enabled to perform operations on HCL OneTest™ Server.

## Tasks or operations in a Team Space

If you are a server administrator, you can perform the following operations in the initial Team Space:

| Tasks | More information |
|---|---|
| Configuring the licensed users | License management on page 99 |
| Configuring an SMTP server to send email notifications | Configuring email notifications |

If you are a licensed user, you can perform the following tasks:

| Tasks | More information |
|---|---|
| Adding a project in the initial Team Space | Adding a project on page 499 |
| Viewing existing projects | Viewing projects on page 502 |
| Becoming a member of other projects | Becoming a project member on page 503 |
| Viewing details about the licensed users or license server | Viewing license details on page 104 |
| Viewing the SMTP configurations | Viewing email configurations |
| Installing remote agents and configuring them to register with HCL OneTest™ Server | Agents overview on page 217 |
| Registering a remote Docker host with HCL OneTest™ Server | Registering a remote Docker host with HCL OneTest Server on page 231 |
| Editing configurations of a remote Docker host | Editing configurations of a remote Docker host on page 235 |

| Tasks | More information |
|---|---|
| Unregistering a remote Docker host | Unregistering a remote Docker host from HCL OneTest Server on page 239 |
| Viewing remote agents, remote Dockers, or intercepts that are registered with HCL OneTest™ Server in the initial Team Space | • Viewing remote Docker hosts that are registered with HCL OneTest Server on page 233<br>• Viewing agents that are registered with HCL OneTest Server on page 219<br>• Viewing intercepts that are registered with HCL OneTest Server on page 325 |
| Creating a model of the system under test | System modeling on page 180 |
| Adding a Team Space repository | Adding a repository to a team space on page 184 |
| Viewing an existing system model | Viewing the system model on page 205 |

### System Model

The system model feature enables you to model the system under test by creating components that are a logical representation of test assets or resources. You also create inter-relationships among the components. All licensed users of the initial Team Space can design a system model. You must configure a team space repository before you create the system model. See System modeling on page 180.

**Note:** The repository that you add to the Team Space is only used to store the system model. You cannot view this repository from the **Projects** page.

### Working with projects in a Team Space

All projects are created within the initial Team Space. When you open a project, the options that are displayed in the navigation pane are specific to working with projects within a Team Space. See Working with projects on page 497.

When you are working on a project in a Team Space and you want to return to the options available for working in the Team Space, you can click the **Home** icon ⌂ in the navigation pane.

## Tasks in a team space

You can find information about the tasks that you can perform as a *Member*, *Project Owner*, *Architect*, or *Team Space Owner* in a team space on HCL OneTest™ Server.

The roles assigned in a team space for users are as follows:

| Persona | Initial team space | | Any team space | |
|---|---|---|---|---|
| | **Role** | **Notes** | **Role** | **Notes** |
| Server administrator | *Team Space Owner* | Default role assigned | *Team Space Owner* | Default role assigned |
| Licensed user | *Team Space Owner* | Role assigned by Server administrator | *Team Space Owner* | Role assigned by Server administrator |
| | *Architect* | Role assigned by *Team Space Owner* | *Architect* | Role assigned by *Team Space Owner* |
| | *Project Creator* | Default role assigned | *Project Creator* | • Default role assigned in a *Permissive* team space.<br>• Role assigned by *Team Space Owner*<br>in a *Restrictive* team space. |
| | *Member* | Default role assigned | *Member* | Default role assigned in either a *Permissive* or *Restrictive* team space. |

The following table shows the tasks that can be performed by licensed users with their assigned roles in a team space:

| Tasks | Team Space Owner | Architect | Project Creator | Member |
|---|---|---|---|---|
| Configuring licenses | ✓ | | | |
| Configuring a Simple Mail Transfer Protocol (SMTP) server to send email notifications. | ✓ | | | |
| Changing the license server configuration | ✓ | | | |
| Adding members in a team space | ✓ | | | |
| Assigning different roles to other team space members | ✓ | | | |
| Modifying configuration details of a team space | ✓ | | | |
| Deleting a team space | ✓ | | | |

| **Tasks** | **Team Space Owner** | **Architect** | **Project Creator** | **Member** |
|---|---|---|---|---|
| Renaming a team space | ✓ | | | |
| Adding or removing the team space repository from a team space | ✓ | ✓ | | |
| Designing a system model | ✓ | ✓ | | |
| Creating projects in a team space | ✓ | | ✓ | |
| Adding, removing, or modifying remote Docker hosts | ✓ | ✓ | ✓ | ✓ |
| Adding agents | ✓ | ✓ | ✓ | ✓ |
| Adding, modifying, or removing resource monitoring sources | ✓ | | | |
| Connecting resource monitoring agents | ✓ | | | |
| Adding resource monitoring labels | ✓ | | | |
| Working with projects in a team space | ✓ | ✓ | ✓ | ✓ |

The following table provides the links to the information about the tasks that can be performed by users with different roles in a team space:

| **Tasks** | **Roles** | **More information** |
|---|---|---|
| Configuring a team space license server | *Team Space Owner* | Configuring named user licenses on page 101 |
| Configuring the Named Users | *Team Space Owner* | Adding users to the Named Users list on page 102 |
| Modifying the configuration of a team space | *Team Space Owner* | Modifying the configuration of a team space on page 488 |
| Viewing the configuration of a team space | All members of a team space | Viewing the configuration of a team space on page 490 |
| Reassigning the role of team space members | *Team Space Owner* | Managing members and their roles in a team space on page 493 |
| Becoming a team space member | • *Team Space Owner*<br>• All users can send a request | Becoming a team space member on page 492 |
| Viewing team spaces | All members of a team space | Viewing team spaces on page 484 |

| Tasks | Roles | More information |
|---|---|---|
| | **Notes:**<br>- The **Permissive** and public **Restrictive** team spaces are visible to all the users.<br>- The private **Restrictive** team spaces are only visible to its members. | |
| Deleting a team space | *Team Space Owner* | Deleting a team space on page 495 |
| Adding a project in a team space | *Project Creator* | Adding a project on page 499 |
| Viewing existing projects | All members of a team space | Viewing projects on page 502 |
| Viewing details about the users or license server | All members of a team space | Viewing license details on page 104 |
| Installing remote agents and configuring them to register with HCL OneTest™ Server | All members of a team space | Agents overview on page 217 |
| Registering a remote Docker host with HCL OneTest™ Server | All members of a team space | Registering a remote Docker host with HCL OneTest Server on page 231 |
| Editing configurations of a remote Docker host | All members of a team space | Editing configurations of a remote Docker host on page 235 |
| Unregistering a remote Docker host | All members of a team space | Unregistering a remote Docker host from HCL OneTest Server on page 239 |
| Viewing remote agents, remote Dockers, or intercepts that are registered with HCL OneTest™ Server in the initial team space | All members of a team space | - Viewing remote Docker hosts that are registered with HCL OneTest Server on page 233 |

| Tasks | Roles | More information |
|---|---|---|
| | | • Viewing agents that are registered with HCL OneTest Server on page 219<br>• Viewing intercepts that are registered with HCL OneTest Server on page 325 |
| Adding resource monitoring sources and agents | *Team Space Owner* | Resource Monitoring service on page 370 |
| Adding resource monitoring labels to the sources | *Team Space Owner* | Controlling Resource Monitoring sources in a schedule |
| Viewing the performance metrics | All members of a team space | Viewing the performance metrics |
| Creating a model of the system under test | *Architect* | System modeling on page 180 |
| Adding a repository to a team space | • *Team Space Owner*<br>• *Architect* | Adding a repository to a team space on page 184 |
| Viewing an existing system model | All members of a team space | Viewing the system model on page 205 |

Related information

Team space administration on page 91

## Tasks in projects in a team space

You can find information about the roles that you can assign in a project for members of a team space and the tasks that you can perform in projects in a team space.

Members of a team space with the role of a *Project Creator* can create projects in the team space and become the *Owner* of a project. The owners of projects can add *Members* of the team space to a project and assign them the following roles:

- *Owner*
- *Tester*
- *Viewer*

For the list of actions that members assigned any of the roles can perform, see Managing access to server projects on page 504.

All projects are created in a team space. When you open a project, the options that are displayed in the navigation pane are specific to the projects within a team space.

When you are working on a project in a team space and you want to return to the options available to work in the team space, you can click the **Home** icon ⌂ in the navigation pane.

**Note:** You must be assigned the role of an *Owner*, a *Tester*, or a *Viewer* of a project.

The following table lists the tasks that members of a team space can perform while working with projects in a team space:

| **Tasks** | **More information...** |
|---|---|
| Viewing projects in a team space. | Viewing projects on page 502 |
| Adding a project. | Adding a project on page 499 |
| Adding a repository that contains test assets and resources that you want to run on the server. | Adding repositories to a server project on page 499 |
| Adding secrets to a secrets collection. | Secrets configuration on page 501 |
| Encrypting the data set resources. | Data security |
| Configuring a change management system. | Change management system |
| Adding members to a project and configuring their roles. | Adding users to a server project on page 501 |
| Selecting the global branch in a project to view the test assets and resources. | Selecting the global branch in a project on page 511 |
| Adding agents to your project, if you are using remote agents as a location to run certain tests. | Adding an agent to a project on page 220 |
| Adding remote Dockers, if you are using a remote Docker host as a location to run tests. | Adding a remote Docker host to the project for running tests on page 234 |
| Reading the considerations that you must take into account before you configure a test run. | Prerequisites to running tests on page 207 |
| Reading the considerations that you must take into account before you configure a run or start a virtual service. | Prerequisites for running virtual services on page 321 |
| Configuring a run of the test assets. | Test run configurations on page 240 |
| Configuring a run of the virtual service resource. | Configuring a run of a virtual service on page 328 |
| Viewing the progress of a test run. | Viewing the progress of running test assets on page 312 |

| Tasks | More information... |
|---|---|
| Managing a running test. | Management of running tests on page 312 |
| Viewing the results of a test run after the run is completed. | Test results on page 359 |
| Reviewing the test results and creating defects for test runs from the **Results** page. | Creating Jira defects on page 401 |

## HCL license portal

Licensing for HCL OneTest™ Server is administered through HCL Software License & Download portal. This portal is a FlexNet-based web application that helps to manage software entitlements and licenses.

When a software order is placed and acknowledged, a software entitlement is created. You must then follow the instructions in the Software Order Acknowledgment document that you receive to activate your entitlement, create devices, and download the software from the portal.

The license portal provides both software distribution and management of your software entitlements that are purchased from HCL Software. The portal provides control and flexibility on how to consume your licenses. An organization identifies one of its resources as a License Manager (also called Tech or Portal Admin) who is familiar with the language of licenses.

For more information about the HCL Software License & Download portal, you can refer to the following knowledge articles:

- What is the HCL Software License & Download portal (FlexNet portal)?
- How to find HCL Product Releases in HCL Software License & Download portal
- Managing Users on the HCL Software License & Download portal

## License management

You as a server administrator can configure and manage licenses. As a licensed user you can only view the details of the licenses configured. If you are not a licensed user you cannot view or perform any operations on HCL OneTest™ Server.

When you log in to HCL OneTest™ Server after you install the server software, you can encounter the following conditions:

- The **No licenses configured** dialog box is displayed.
- The **No licenses configured** dialog box is displayed if you click **Manage > Licenses**.
- The **New Project** button is disabled.

Depending on your privileges you can perform the following tasks in HCL OneTest™ Server:

| Privileges | Tasks | More information |
|---|---|---|
| Server administrator | Read about licensing on HCL OneTest™ Server. | Licensing overview on page 100 |
| | Configure licenses. | Configuring named user licenses on page 101 |
| | Configure named users. | |
| | Remove or replace named users. | Removing or replacing named users on page 103 |
| Licensed user | Perform tasks or operations in a Team Space. | Tasks or operations in a Team Space on page 92 |
| | Viewing details about licenses. | Viewing license details on page 104 |

## Licensing overview

You can find information about the licensing on HCL OneTest™ Server.

To use the features offered by HCL OneTest™ Server, you must purchase the named user licenses for the number of licensed users that you want to configure. You must then install the licenses on the FlexNet Operations server and configure the license server details on HCL OneTest™ Server.

The named user licenses are automatically self-acquired on a first-come-first-serve basis. As an administrator, you can add users to the **Named Users** list. The licenses are always available for the added users even though the self-acquisition of named user licenses option is disabled in the **License Configuration** page.

A single named user license is used from the total number of named user licenses under any of the following conditions:

- When you create a project after you log in if you are not a member of any other project.
- When you are approved as a member of a project if you are not the owner of any project.

If the automatic self-acquisition option is disabled, only the users who are added to the **Named Users** list can perform the these project tasks in HCL OneTest™ Server. However, the other users receive an error notification about the unavailability of the license when they try to perform the project tasks.

The named user license is returned to the license pool only under any of the following conditions:

- When the named user is no longer a member of any projects.
- When the named user is removed as a user of HCL OneTest™ Server.

**Important:** The named user license takes a few minutes to be returned to the license pool.

Related information

Configuring named user licenses on page 101

## Configuring named user licenses

As an administrator, you can configure certain options in the **License Configuration** page, so that licensed users can perform various operations such as creating a project, running tests, and so on.

**Before you begin**

You must have completed the following tasks:

- Installed the product and signed up as a user on HCL OneTest™ Server.
- Been assigned the role as an administrator of HCL OneTest™ Server.

**About this task**

As an administrator, you can assign the role of an administrator to any user. For information about how to change the role, see Default user administration at the Related information section.

**Note:** When your role is changed from a user to an administrator, you must log out and log in again so that HCL OneTest™ Server can apply the change in your role.

1. Log in to HCL OneTest™ Server.

   The **License Configuration** page is displayed for the first time you log in if the licenses are not configured.

   Alternatively, you can click **Manage > Licenses**.
2. Select the **Disable automatic self-acquisition of named user licenses** option if you do not want to self-acquire named users license on a first-come-first-serve basis.

   **Note:**

- When you clear this option, any user can get a license when sufficient licenses are available.
- When you select this option, only such users who are added to the **Named Users** list get a license to use HCL OneTest™ Server. If other users try to perform any project operations in HCL OneTest™ Server, they cannot get a license even if sufficient licenses are available.

3. Enter the values in the **OneTest License Server URL** and **OneTest License Server ID** fields to install the licenses on the FlexNet Operations server.
4. Click **Configure** to apply the license configuration.
5. Enter the name or the email ID of the user in the **Add named users** field, and then select the name from the list that displays to add members to the **Named Users** list.

   **Note:** You can press the **Ctrl** key to select multiple users simultaneously.

6. Click **Add** to make selected users as the named users.

**Results**

You have configured the license server. You can view the number of named users licenses that are currently in use in the **License usage** section.

Related information

Default user administration on page 68

Removing or replacing named users on page 103

## Adding users to the Named Users list

As an administrator, you can add users to the **Named Users** list so that the licenses are always available for those users.

**Before you begin**

You must have completed the following tasks:

- Been assigned the role as an administrator of HCL OneTest™ Server.

1. Log in to HCL OneTest™ Server and open the team space for which you want to add users.
2. Enter the name or the email ID of the user in the **Add named users** field, and then select the name from the list that displays to add members to the **Named Users** list.

**Note:** You can press the **Ctrl** key to select multiple users simultaneously.

3. Click **Add** to make selected users as **Named users**.

**Results**

You have added users to the **Named Users** list.

## Removing or replacing named users

When you want to remove a user from the list of named users or you want to replace an existing user with another named user, you as an administrator can perform these changes from the **License Configuration** page.

**Before you begin**

You must have completed the following tasks:

- Been assigned the role as an administrator of HCL OneTest™ Server.
- Configured licenses and added users to the list of named users on HCL OneTest™ Server.

1. Log in to HCL OneTest™ Server.
   **Result**
   The **Projects** page of the initial Team Space is displayed.
2. Click **Manage > Licenses**.
   **Result**
   The **License Configuration** page is displayed with licensing details.
3. Perform the actions described in the following table to remove or replace users from the **Named Users** list:

| **Tasks** | **Actions** |
|---|---|
| Remove a user from the list of named users. | Perform the following steps:<br>a. Locate the user from the **Named Users** list that you want to remove.<br>b. Click the **Menu** icon ⋮ next to the name of the user, and then click **Remove**.<br>c. Click **Remove** on the **Remove named user** dialog box to complete the action.<br>**Remember:**<br>◦ When the **Disable automatic self-acquisition of named user licenses** option is cleared, and if the user who is removed from the **Named Users** list tries to perform any project operation, then the name of the |

| Tasks | Actions |
|---|---|
| | 🔔 user is added back to the list. The name of the user is added only if sufficient licenses are available.<br>◦ When the **Disable automatic self-acquisition of named user licenses** option is selected and you remove the user from the **Named Users** list, then the following operations are impacted by the user who is removed from the list:<br>▪ The test runs scheduled by the user might fail to run.<br>▪ The test results which are in progress by the user might fail to record complete details.<br>▪ The user cannot perform any of the future project operations that includes generation of offline token that are used for long-run tests.<br>If the user is currently on HCL OneTest™ Server, then the license can be used while the current session remains active. |
| Replace a user in the list of named users. | Perform the following steps:<br>a. Locate the user from the **Named Users** list that you want to replace with another user.<br>b. Click the **Menu** icon ⋮ next to the name of the user, and then click **Replace**.<br>c. Enter the email ID or name of another user that you want to add to the **Named Users** list.<br>d. Click **Replace** to complete the action. |

**Results**

You have removed or replaced a user from the named users list.

## Viewing license details

You can find information about the number of licenses configured, licensed users, or details of the license server on the **License Configuration** page.

**Before you begin**

The server administrator must have configured the licenses on HCL OneTest™ Server.

**About this task**

You as a licensed user can create projects or use the other features in HCL OneTest™ Server only after the server administrator has configured the licenses. If you are not a licensed user, you cannot view or perform any operations on HCL OneTest™ Server.

1. Log in to HCL OneTest™ Server.
   **Result**
   The **Projects** page of the initial Team Space is displayed.
2. Click **Manage > Licenses**.
   **Result**

   The **License Configuration** page is displayed with the following license information:

   - The number of named user licenses that are currently in use
   - The URL and ID of OneTest License Server
   - Whether the self-acquisition of named user licenses option is enabled or disabled
   - The name and email ID of the users that are configured as named users by the administrator

# Chapter 5. Test Author Guide

This guide describes how to create test assets in HCL OneTest™ Server and publish test assets to the Git repository that you have configured. This guide is intended for testers and test managers.

## Datasets overview

A dataset provides tests with variable data during a run. The test that uses a dataset at run time replaces a value in the recorded test with variable test data that is stored in the dataset.

In HCL OneTest™ Server, you can create a dataset and use it to replace the dataset values with original values during run time, when you want to run test assets that contain the dataset.

From the **Datasets** page, you can perform the following actions:

**Table 3. Options available in Datasets page**

| **Actions** | **Descriptions** |
|---|---|
| View | You can view the contents of a dataset. See Viewing a dataset on page 117. |
| Create | You can create a dataset to use it during a test or schedule run. See Creating a dataset on page 107. |
| Edit | You can edit a dataset when you want to run a test asset with different dataset values. See Editing a dataset on page 108. |
| Configure | You can configure the string values in a dataset that contains variable data for tests to use when they run. See Editing a dataset on page 108. |
| Delete | You can delete the dataset when it is not required in your test environment. See Deleting a dataset on page 119. |
| Save As | You can save a copy of a dataset in a different folder or make a copy with a different name by clicking the **Menu** icon ⋮ and then by selecting the **Save As** option. |
| Branch selection | You can view the dataset listed in the other branches of the repository by selecting the name of the branch from the **Branch** list. When you access the **Datasets** page for the first time after adding the repository, the default branch that it displayed is **master**.<br>When multiple repositories are added to the same project, the following events occur:<br>• The datasets stored in the **master** branch of all the repositories are displayed.<br>• All the branches in all the repositories are listed in the drop-down list. |

**Table 3. Options available in Datasets page (continued)**

| Actions | Descriptions |
|---|---|
| | To differentiate a common branch across multiple repositories added to the project, the tooltip is not displayed in front of the common branch name in the list. |

**Notes:**

- Only a project **Owner** or a project member whose role is a **Tester** can view and edit the dataset.
- To view the content of an encrypted dataset, you must provide an encryption key that you set while encrypting the dataset column.
- As a **Viewer**, when you try to access the **Datasets** page, an error message is displayed because you do not have the permission to view this page.

## Creating a dataset

You can create datasets in HCL OneTest™ Server to replace the existing dataset values with new dataset values during a test or schedule run.

### Before you begin

You must be a member of the project with the **Owner** or **Tester** role and must have completed the following tasks:

- Created a project in HCL OneTest™ Server. See Adding a project on page 499.
- Configured the Git repository in your project. See Adding repositories to a server project on page 499.

### About this task

When you create a dataset in HCL OneTest™ Server, it always creates 1 Row, 2 Column (1R X 2C) dataset. Later, you can edit the dataset by adding some rows and columns that you want, to add the data in it.

1. Go to the HCL OneTest™ Server URL.
2. Enter your user name and password, and then click **Login**.
3. Open your project from the HCL OneTest™ Server UI.
4. Go to the **Datasets** page, and then click **Create dataset**.
5. Enter a name for the dataset in the **Asset name** field, and then select the place where you want to save the dataset in the **Location** drop-down list.
6. Click **Create**.

   **Result**

   The new dataset opens in a CSV Editor in a web browser. The dataset created is listed on the **Datasets** page.

### Results

You have created a dataset in your project.

### What to do next

- You can add, modify, or remove data in the dataset. See Editing a dataset on page 108.
- You can publish the dataset to the Git repository so that other members of the project can use the dataset. See Publishing a dataset on page 116.

## Editing a dataset

You can add, modify, remove, import, or export 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 or configured a repository that contains the dataset.

### About this task

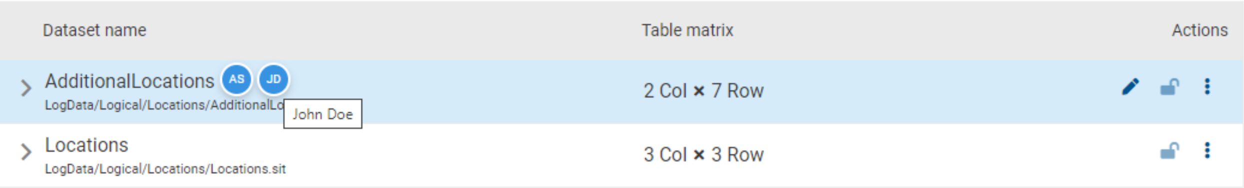
If you are a project **Owner** or **Tester** in HCL OneTest™ Server V10.1.0 or later, you can perform basic tasks in the CSV Editor by right-clicking any row, column, or cell in the dataset to organize your data in a better way. For example, you can perform tasks such as updating data in a cell, inserting or deleting rows and columns, or renaming column names.

When you edit the dataset in the CSV Editor, you can use the following keyboard shortcuts to control the cursor selection in the CSV Editor:

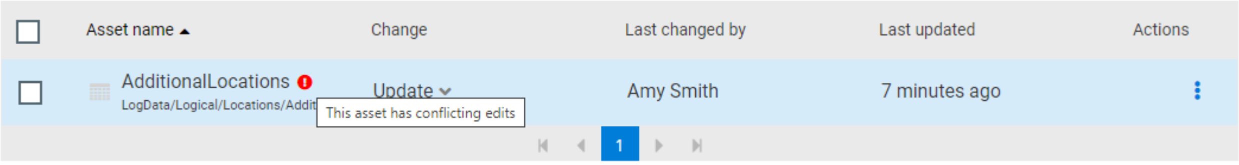
- **Tab** - To move the cursor control to the next available option.
- **Shift-Tab** – To move the cursor control to the previous option.
- **Shift+F10** – To open the context menu from the dataset cell.

After you edit the dataset, you can save the changes made to the dataset, and then you can publish the dataset to the Git repository. If you save and close the edited dataset, the **Changes** page lists the edited dataset and later you can publish to the Git repository for other members to use.

When you or other members of the project edit the same dataset, you can see an icon with the initials of the member next to the name of the dataset on the **Datasets** and **Changes** pages. If you do not see the icon, you must refresh the **Datasets** page to view the icons. The **Changes** page denotes conflicting edits when another member publishes their edited dataset first.



| Dataset name | Table matrix | Actions |
|---|---|---|
| AdditionalLocations AS JD<br>LogData/Logical/Locations/AdditionalLo John Doe | 2 Col × 7 Row | ✎ 🔓 ⋮ |
| Locations<br>LogData/Logical/Locations/Locations.sit | 3 Col × 3 Row | 🔓 ⋮ |



| □ | Asset name ▲ | Change | Last changed by | Last updated | Actions |
|---|---|---|---|---|---|
| □ | AdditionalLocations ❗<br>LogData/Logical/Locations/Addit | Update<br>This asset has conflicting edits | Amy Smith | 7 minutes ago | ⋮ |

When a member with conflicting edits tries to publish the edited dataset, an error about conflicting changes is displayed. However, the member can use the **Save As** option to save and publish a copy of the dataset edits under a new asset name. The member must discard the edits that were made to the original dataset.

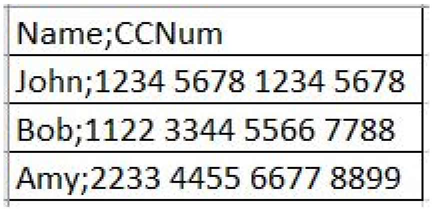
For example, consider a scenario when Amy Smith and John Doe edit the same dataset, and Amy Smith edits and publishes the dataset. On the **Changes** page, a message that indicates a conflict is displayed when John Doe tries to publish the same dataset.

**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.

When you have a CSV file that has data separated from a character, then you can import that CSV file into the dataset. You can select any of the following separator characters from the **Configure Dataset** window, and the selection can be the separator character that you used in the CSV file:

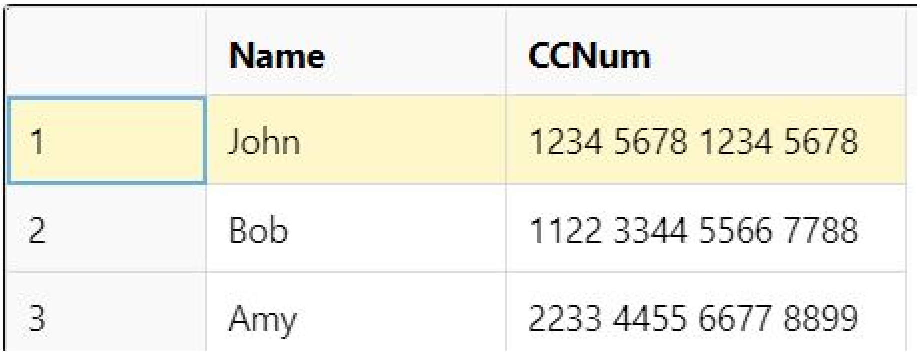
- Comma
- Semicolon
- Space
- Tab
- Other

Consider that you have the data in the CSV file in the following format:



```
Name;CCNum
John;1234 5678 1234 5678
Bob;1122 3344 5566 7788
Amy;2233 4455 6677 8899
```

When you import the CSV file in the dataset, and then select the separator value as **Semicolon**, the data in the dataset is displayed as follows:



| | Name | CCNum |
|---|---|---|
| 1 | John | 1234 5678 1234 5678 |
| 2 | Bob | 1122 3344 5566 7788 |
| 3 | Amy | 2233 4455 6677 8899 |

If you want the data in its original format, that is, a semicolon (;) character to separate the data, then you can choose any other separator value from the **Configure Dataset** window.

**Note:** The default separator value is **Comma**.

1. Go to the **Datasets** page and find the dataset which you want to edit.
2. Click the **Edit** icon ✎ from the **Actions** column of the dataset.

   **Result**

   The dataset opens in the CSV Editor in a web browser.
3. Perform the following actions to use the options available in the CSV Editor:

| Options | Actions |
|---|---|
| Find and Replace 🔍 | To find:<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>▪ Select the **Case sensitive** checkbox 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** checkbox to search for cells that contain only the characters that you have entered in the **Find** field.<br>▪ Select the **Search using regular expression** checkbox to search the pattern that matches strings.<br>For example, to search a cell that contains any number between 0 to 9, do the following:<br>i. Enter *\d* in the **Find** filed.<br>ii. Select the **Search using regular expression** checkbox.<br>iii. Click **Find**.<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.<br>To find and replace: |

| Options | Actions |
|---|---|
| | a. Click the **Find and Replace** icon 🔍 .<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** checkbox 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** checkbox 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** checkbox 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>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.<br>The CSV Editor saves the unlimited redo action. |
| Import | When you have a large amount of data stored in a CSV file, you can import that into a dataset instead of creating a new dataset. You must have a .csv file that contains variable data to import into a dataset.<br>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>c. Select one of the following options to append or overwrite data in the dataset:<br>▪ Click **Overwrite** to add the rows and columns from the selected CSV file from the beginning of the dataset.<br>▪ Click **Append** to add rows and columns from the selected CSV file to the end of the dataset.<br>d. Select the **First row contains headers** checkbox if your CSV file contains the header. |
| Export | You can export variable data from the dataset into a CSV file to reuse in future tests when required. You must have a dataset that you want to export. |

| Options | Actions |
|---|---|
| | Click the **Export** icon to download the dataset as a CSV file. |
| Set as current row | During the test run, if you want variable data to be selected from a current row instead of the first row in a dataset, right-click any cell in a row and select **Set as current row**.<br>Also, you can set the current row from the **Datasets** page by clicking **Menu**, and then the **Configure** option.<br>**When rows are deleted:**<br>If you delete any row between row 1 to current row, the current row data is taken from the next row. 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>**When rows are inserted:**<br>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. |
| 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>a. Click the **Menu** icon ⋮ , and then select the **Configure** option.<br>b. Select any of the separator values that you used in the CSV file.<br>The available options are **Comma**, **Semicolon**, **Space**, **Tab**, and **Other**. In the CSV file, if you have any other separator characters other than the available options, then you can select the **Other** option, and then can specify a value.<br>For example, if the data in the CSV file is separated by a character #, then select the **Other** option and enter # in the field.<br>c. Configure the following options to change the row and column settings:<br>▪ **Column header** - Use an up-down control button to increment or decrement the value of the column header.<br>▪ **Data start point** - Use an up-down control button to increment or decrement the value of the data starting pointer.<br>▪ **Current row** - Use an up-down control button to increment or decrement the value of the current row. |

| Options | Actions |
|---|---|
| | d. Configure the following options to change the string values in the dataset:<br>▪ **Treat as null** - Enter a string value that is to be treated as null when running the test.<br>▪ **Treat as empty** - Enter a string value that is to be treated as empty when running the test.<br>For example, when you run the test and the data *123* in the dataset to be treated as empty, then you can specify *123* in the **Treat as empty** field.<br>▪ **Treat empty text as null** - Select this field when you want the dataset that contains any blank cells, and the value of those blank cells to be interpreted as null.<br>e. Click **Update** to apply the changes. |
| Discard | Click the **Menu** icon ⋮ and select **Discard** to discard the changes made to the dataset. |

4. Click the **Save** icon to save the changes made to the dataset.

5. Click the **Publish** icon to publish the dataset to the Git repository, and then close the CSV editor.

**Results**

You have edited the dataset.

## Dataset encryption

Encrypted datasets are useful when you want to run tests that contain confidential information such as a set of passwords or account numbers.

When you run a test that uses an encrypted dataset, then you must provide an encryption key for decrypting the encrypted data in columns so that the data can be used in the test. If the test uses data from the multiple encrypted dataset columns, you must enter the same encryption key for every encrypted dataset column that the test uses.

When you run the test that uses the dataset with the encrypted column, the value of the column is decrypted at a run time. The data in the column is sent as a clear-text string in 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.

You can use only one encryption key to encrypt data in the columns in any dataset.

**Important:**

The encryption keys that 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 perform the following operations:

- View the encrypted values
- Decrypt data
- Enable the use of the encrypted dataset during test runs

## 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 on page 107.

1. Go to the HCL OneTest™ Server URL.
2. Enter your user name and password, and then click **Login**.
3. Open your project from the HCL OneTest™ Server UI.
4. Go to the **Datasets** page and find the dataset that you want to encrypt.
5. Click the **Edit** icon ✎ from the **Actions** column of the dataset.

   **Result**

   The dataset opens in the CSV Editor in a web browser.
6. Right-click any cell in a column that you want to encrypt and select **Encrypt column data**.

   **Result**

   The **Encrypt Column** window is displayed.
7. 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 that 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 perform the following operations:

- View the encrypted values
- Decrypt data
- Enable the use of the encrypted dataset during test runs

8. Click **Encrypt Column**.

   **Result**

   Asterisks are displayed instead of actual data for the encrypted column.

**Results**

You have encrypted the dataset column in your project.

**What to do next**

You can publish the dataset to the Git repository so that other members of the project can use the dataset. See Publishing a dataset on page 116.

## Decrypting a dataset column

To view the content of an encrypted dataset, you can decrypt the dataset column. Removing encryption from a dataset revokes the protection offered to the test data.

**Before you begin**

You must have created at least one dataset and encrypted the dataset with an encryption key. See Creating a dataset on page 107 and Encrypting a dataset column on page 114.

1. Go to the HCL OneTest™ Server URL.
2. Enter your user name and password, and then click **Login**.
3. Open your project from the HCL OneTest™ Server UI.
4. Go to the **Datasets** page and find the dataset that you want to decrypt.
5. Click the **Edit** icon ✎ from the **Actions** column of the dataset.

   **Result**

   The dataset opens in the CSV Editor in a web browser.
6. Right-click encrypted cells that display the contents with asterisks, and then select **Decrypt column data**.

   **Result**

   The **Decrypt Column** window is displayed.
7. Enter the encryption key that you used to encrypt the data in the column in the **Encryption Key** field.
8. Click **Decrypt Column**.

   **Result**

   Asterisks are replaced with the actual data in the decrypted column.

**Results**

The encryption is 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 original data in the recorded test without prompting for the encryption key.

**What to do next**

You can publish the dataset to the Git repository so that other members of the project can use the dataset. See Publishing a dataset on page 116.

## Publishing a dataset

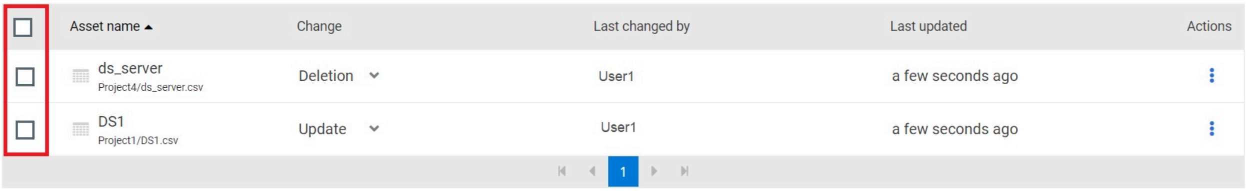
When you create or edit any datasets in HCL OneTest™ Server, you can publish your changes to the Git repository. Therefore, when you publish a dataset, other members in the project can use your dataset in their test assets run, if required.

**Before you begin**

You must have created, edited, or deleted dataset assets in HCL OneTest™ Server.

**About this task**

The **Changes** page lists dataset assets that are modified. You can publish all datasets or a single dataset listed in the **Changes** page to the Git repository by selecting the appropriate checkboxes.



| □ | Asset name ▲ | Change | Last changed by | Last updated | Actions |
|---|---|---|---|---|---|
| □ | ds_server<br>Project4/ds_server.csv | Deletion | User1 | a few seconds ago | ⋮ |
| □ | DS1<br>Project1/DS1.csv | Update | User1 | a few seconds ago | ⋮ |

HCL OneTest™ Server processes one publish request at a time. Therefore, when multiple users attempt to publish the same dataset, the request that reaches first from the users is processed. The other users receive an error message, and they are unable to publish.

When you modify the dataset and publish it to the Git repository, the other members who have access to that dataset can view the updated dataset.

You can also perform the following actions from the Changes page by clicking the **Menu** icon ⋮.

- **Edit** - If you want to make any last-minute updates to the dataset before publishing.
- **Discard** - To remove the changes that you made to the dataset asset.
- **Save As** - To save a copy of dataset.

1. Go to the HCL OneTest™ Server URL.
2. Enter your user name and password, and then click **Login**.
3. Open your project from the HCL OneTest™ Server UI.

4. Go to the **Changes** page, find the dataset that you want to publish.
5. Select the checkbox that precedes the **Dataset name** to publish to the Git repository, and then click **Publish**.
6. Enter a description for the changes made to the dataset, and then click **Publish**.

   **Result**

   A message is displayed for successful pushing of changes to the Git repository.

### Results

You have published the modified dataset into the Git repository.

## Viewing a dataset

When test assets includes datasets, you can view the contents of a dataset from HCL OneTest™ Server. The datasets residing in the Git repository are listed in the **Datasets** page.

### Before you begin

You must be a member of the project with the **Owner** or **Tester** role and must have completed the following tasks:

- Created a project in HCL OneTest™ Server. See Adding a project on page 499.
- Configured the Git repository in your project. See Adding repositories to a server project on page 499.
- Created at least one dataset and encrypted the dataset with an encryption key. See Creating a dataset on page 107 and Encrypting a dataset column on page 114.
- Created a classification for an encrypted dataset. See Creating a classification on page 519.

1. Go to the HCL OneTest™ Server URL.
2. Enter your user name and password, and then click **Login**.
3. Open your project from the HCL OneTest™ Server UI.
4. Go to the **Datasets** page, find the dataset that you are interested in, and then expand the dataset by clicking the **Expand** icon >.

   **Note:** You can click the **Dataset Name** field to sort the datasets by name in alphabetical order. Alternatively, you can use the **Search** field to search the dataset by name.

### Results

You have viewed the contents of the dataset.

## Viewing an encrypted dataset

You can use the **Dataset** page to view the contents of an encrypted dataset from HCL OneTest™ Server.

1. Go to the **Datasets** page, find the encrypted dataset that you are interested in, and then expand the dataset by clicking the **Expand** icon >.
2. Perform the following steps, if you have created a classification but yet to add the encrypted dataset to it:
    a. Find the encrypted dataset that you are interested in, and then click the **Lock** icon 🔒 from the **Actions** column of the dataset.
    b. In the **Change the classification for the dataset** window, select a classification from the list and enter the encryption key set for the dataset.
    c. Click **Save** to save the classification details.
3. Right-click the encrypted column and click **Show encrypted data**.
   You can right-click the decrypted column and select **Hide encrypted data** to encrypt the data again.

> **Important:** The **Show encrypted data** option is available only when you have added the encrypted dataset to the respective classification.

**Results**

You have viewed the contents of an encrypted dataset.

## Changing classification for an encrypted dataset

To move the encrypted dataset from one classification to another, you can change the classification for an encrypted dataset from the **Dataset** page.

**Before you begin**

You must have created at least two or more classifications. See Creating a classification on page 519.

1. Go to the **Datasets** page and find an encrypted dataset for which you want to change the classification.
2. Click the **Lock** icon 🔒 from the **Actions** column of a dataset.
3. In the **Change the classification for the Dataset** window:
    a. Choose the classification from the list.
    b. Enter the encryption key for the dataset that was set while encrypting the dataset column.
    c. **Save** the classification details.

> **Remember:** You can perform this task only if you know the encryption key set for the dataset while encrypting the dataset column.

**Results**

You have changed the classification for an encrypted dataset.

Related information

Managing an encrypted dataset on page 518

Creating a classification on page 519

## Deleting a dataset

You can delete the dataset when it is not required in your test environment.

**Before you begin**

You must have at least one dataset asset in your Git repository that you have configured.

**About this task**

You can delete datasets in HCL OneTest™ Server in the following scenarios:

- Datasets that you created in the desktop clients and that are cloned to the repository in your project on HCL OneTest™ Server.
- Datasets that you created in HCL OneTest™ Server and that are published to the repository in your project.
- Datasets that are in the `.csv` file format.

You cannot delete the following datasets:

- Datasets that you created or edited in HCL OneTest™ Server and that are not published to the repository in your project.
- Datasets that are in the `.sit` file format.

1. Go to the HCL OneTest™ Server URL.
2. Enter your user name and password, and then click **Login**.
3. Open your project from the HCL OneTest™ Server UI.
4. Go to the **Datasets** page, find the dataset that you want to delete.
5. Click the **Menu** icon ⋮ from the **Actions** column of a dataset, and then click **Delete**.
6. Clear the **Publish delete to *<branch name of the configured Git repository>*** checkbox to delete the dataset from your project.

   When you clear this option, the deletion of the dataset does not reflect in the configured Git repository. Therefore, the other members of the project can still use the dataset that you have deleted. If you want, later you can publish the dataset to the Git repository from the **Changes** page.

   **Note:** By default, **Publish delete to *<branch name of the configured Git repository>*** field is selected.

7. Enter a description for deleting the dataset in the **Description of change** field.

8. Perform the following action:

| Option | When | Action |
|---|---|---|
| ***Publish delete to <branch name of the configured Git repository>*** | Selected ☑ | Click **Delete and Publish** |
| | Cleared □ | Click **Delete** |

**Result**

A message is displayed for the successful deletion of the dataset.

# Generation of test data

To test an application, you use data that is passed to get real-time results. Since creating test data manually is time-consuming, you can use HCL® OneTest™ Data, an automated tool to generate random test data. You can generate the test data in various file formats.

Test data is a core element in the process of testing any application. A sufficient amount of data is necessary to test all the possible scenarios of any application. HCL® OneTest™ Data requires a schema to generate the test data.

When you want to generate the test data, then you must fabricate a schema or generate a schema by using any external resource. After you fabricated a schema, you must define types of the schema and set properties for each type. You can then generate the test data for one or multiple schemas at a time. You must generate multiple schemas in HCL® OneTest™ Data by using the schemas created in the JDBC supported databases.

## Schema fabrication

A schema is a graphical structure to represent different types of data. You can fabricate a schema for the generation of test data.

HCL® OneTest™ Data supports the following ways to fabricate the schema:

- Import a schema from local file system into HCL® OneTest™ Data.
- Generate a schema in external resource by using HCL® OneTest™ Data.
- Create a schema by using HCL® OneTest™ Data.

You can fabricate a schema by using various methods. You can also manage the schema for your project to generate the test data.

## Schema import

HCL® OneTest™ Data requires a schema to generate the test data. If you have a schema in your local file system, then you can import the same schema into your project.

HCL OneTest Data supports the import of the schema, which is either in JavaScript Object Notation (JSON) or XML Schema Definition (XSD) file format.

After you import the schema, you can perform all the following tasks to the imported schema:

- Defining types
- Setting type properties
- Setting restrictions
- Applying functions

🚫 **Restriction:** You cannot modify the properties of the types in an XSD or a JSON schema.

## Importing XSD or JSON files from the local file system

When you have an XSD or a JSON format file in your local file system, you can import the same files from the local file system into your HCL® OneTest™ Data project.

### Before you begin

- You must have created a project.
- You must have an XSD or a JSON format file in your local file system.

### About this task

XSD or JSON files are in schema format. To create a schema, you can import an existing schema that is in the XSD or JSON format from your local file system.

🚫 **Restriction:** You cannot modify all the properties of the types in an XSD or a JSON schema.

The following table shows the list of properties supported by JSON or XSD schema:

| Properties | JSON schema | XSD schema |
|---|---|---|
| JavaScript rules | Yes | No |
| Regular expression | Yes | No |
| Restrictions | No | No |
| Default values | No | No |

1. Select your project in Rational® Test Automation Server and go to the **Data Fabrication** page.
2. Click the **Schemas** tab.
3. Click **XSD/JSON** from the **Schemas menu**.

   **Result**

The **Import XSD/JSON** dialog box is displayed.

4. Select the JSON or XSD file that you want to import.
5. Provide a schema name with a brief description and keywords as **Tags**.
6. Click **OK** and save the changes.

### Results

You have successfully imported XSD or JSON files from the local file system.

### What to do next

You must define the types in the schema. See Defining Types on page 128.

You can also perform the following actions:

- View the imported file listed under the **Schemas** tab.
- Use the **Search** field to search for any schema from the existing schemas.
- Sort the list of schemas by name, folder, and recently used schemas.

## Schema generation

When you want to generate the test data in any external resource, you must generate the schema by using that specific resource.

To generate the schema, you must establish a connection between HCL® OneTest™ Data and the external resource.

HCL® OneTest™ Data supports the establishment of connections with the following external resources:

- Java Database Connectivity (JDBC)
- Systems, Applications, and Products in Data Processing (SAP)
- MongoDB
- Excel

**Note:** After you generate schema by using any external resource, you can perform all the following tasks to the generated schema in HCL® OneTest™ Data:

- Defining types
- Setting type properties
- Setting restrictions
- Applying functions

The following topics show how HCL® OneTest™ Data connects with an external resource and generates the test data.

| Tasks | More information |
|---|---|
| Generate test data in JDBC-supported databases | HCL OneTest Data support to write the generated test data in JDBC-supported databases on page 146 |
| Generate test data for SAP IDOC or DXOB | HCL OneTest Data support to generate test data for an SAP IDoc or DXOB file on page 153 |
| Generate test data for SAP BAPI | HCL OneTest Data support to generate test data for SAP BAPI on page 156 |
| Generate test data in MongoDB | HCL OneTest Data support to write the generated test data in MongoDB on page 159 |
| Generate test data for an Excel file | HCL OneTest Data support to generate test data for an Excel file on page 163 |

## Schema creation

When you do not have an existing schema or you do not want to generate a schema in any database, then you can create a schema by using HCL® OneTest™ Data.

To create a schema, you need to design a schema by defining the types, setting the properties for each type of schema, and designing a structural view of schema. You can also set restrictions or apply functions to the defined types.

## Composing a schema

To generate the test data, you require a schema. You can compose a new schema if you do not have an existing schema or you do not want to generate the schema by using any external resource.

**Before you begin**

You must have a project.

1. Log in to HCL OneTest™ Server and open the team space that contains your project.
2. Open your project.

   **Result**

   The **Overview** page is displayed.
3. Click **Author > Data Fabrication**.

   **Result**

   The **Data Fabrication** page is displayed.
4. Click **New** from the **Schemas**. Otherwise, select **Create a schema** from **Quick Links**.

   **Result**

   The **New Schema** dialog box is displayed.
5. Provide a schema name.
6. Select the default project folder or enter any project folder name.

7. Enter keywords as tags which can help for quick search.
8. Provide a brief description of the schema and click **OK**.

   **Result**

   The schema opens in the schema designer as Root as the data dictionary. You can view the schema name as a tab in the workspace.

**Results**

The schema is displayed in the schema designer.

**What to do next**

You must define the types in your schema. For information about defining types, see Defining types on page 128.

## Importing data from local file system

If you have data or a sample file, you can import it into your project and can use that data to design a schema.

**Before you begin**

- You must have created a project.
- You must have a data file in your local file system in CSV or Excel files.

**About this task**

The **Files** tab on the **Data Fabrication** page is one of the design components of a project. By clicking this tab, you can create a file. This tab helps you to import an existing file from the local file system, for which you want to generate test data.

1. Select your project in Rational® Test Automation Server and go to the **Data Fabrication** page.
2. Click the **Files** tab.
3. Click **New** from the **Files** to create a file.

   **Result**

   The **New File** dialog box is displayed.
4. Select a file that you want to import from your local file system.
5. Provide a name to a file, folder details, keyword as **Tag** for quick search, and a brief description of the file.
6. Click **OK** to import and create a file.

**Results**

You achieved the following results:

- View the list of all the files imported for this project in the **Files** tab.
- Use the **Search** field to search any file from the existing files.
- Sort the list of files by name, folder, and recently used files.

### What to do next

After you have created a file into your project, you can import a file into schema designer to create a schema.

Related information

Integrating sample files into a schema on page 136

## Importing files into schema designer

To create a schema by using the data you imported, you must import the data file into schema designer.

### Before you begin

You must have imported a data file into your project from the local file system.

### About this task

When you want to create a schema, you require data. You can use the data which you have imported from your local file system.

1. Click **Import** from the **Schemas**.

   **Result**

   The **Import** dialog box is displayed.
2. Select the file type as CSV.
3. Select the file that you want to import from the **Sample File Path** and define the import properties of the data.
4. Provide a schema name for this data and click **Import**.

   **Result**

   The schema appears in the schema designer.
5. Save the changes.

### Results

You can view the imported file listed under the **Schemas** tab.

### What to do next

You must define the types in the schema. For information about defining types, see Defining types on page 128.

## Copying artifacts of a schema to another schema

When a schema in your project includes artifacts, and you want to reuse those artifacts in another schema in the same project, then you can copy the artifacts of a schema to another schema by using HCL® OneTest™ Data.

### Before you begin

You must have a schema in your HCL® OneTest™ Data project.

### About this task

While you copy the artifacts of the schema, you can either create a schema or select an existing schema.

**Restriction:** You cannot copy an XSD or a JSON schema to another schema.

1. Log in to HCL OneTest™ Server and open the team space that contains your project.
2. Open your project.

   **Result**

   The **Overview** page is displayed.
3. Click **Author > Data Fabrication**.

   **Result**

   The **Data Fabrication** page is displayed.
4. Open your schema from the **Schemas** tab.
5. Select a category, group, or item type from **Dictionary**, and then click **Copy to Schema** from the menu.

   **Result**

   The **Copy Type** dialog box is displayed.
6. Toggle the **Create New Schema** switch to select the destination schema. You can either create a schema or select an existing schema.

   **Choose from:**

   - To copy the artifacts of a schema into a new schema, go to step 5.
   - To copy the artifacts of a schema into an existing schema, go to step 6.
7. Copy the artifacts to a new schema by performing the following steps:

   a. Toggle the **Create New Schema** switch to the on position to create a schema.

   b. Enter a name for the new schema.

   c. Toggle the **Copy Child Types** switch to the on position to copy all the child types of the selected type.

   d. Toggle the **Copy Referenced Types** switch to the on position if you want to copy the types referenced as the components of the group.

   **Note:** The referenced types are always copied to the same path as the source schema.
8. Copy the artifacts to an existing schema by performing the following steps:

   a. Toggle the **Create New Schema** switch to the off position to select an existing schema.

   b. Select a schema from the **Schema** drop-down list.

   c. Toggle the **Retain Same Type Paths** switch to the on position if you want to copy the types to the same path in the destination schema.

> **Note:** If you toggle the switch to the off position, then you must provide the type path in the **Type Path** field.

d. Toggle the **Copy Child Types** switch to the on position to copy all the types and the child types of the selected type.

e. Toggle the **Copy Referenced Types** switch to the on position if you want to copy the types referenced as components of the group.

f. Toggle the **Overwrite Existing Types** switch to the on position to overwrite the type in the destination schema that already exists.

9. Click **Copy**.
10. Save the changes.

**Results**

You have successfully copied the artifacts of a schema into another schema.

## Schema management

In a schema you can perform certain tasks such as editing and deleting of a schema. You can also split a schema view.

## Editing a schema

At any point in time, you can edit the details of a schema of your project.

1. Select a schema that you want to edit from the list of schemas of your project.

   **Result**

   The schema appears in the schema designer.
2. Click the edit icon ✎.

   **Result**

   The **Edit Schema** dialog box appears.
3. Edit the schema details and save the changes.

## Deleting a schema

If you do not need a schema anymore, you can remove that specific schema from your project.

1. Select the schema that you want to delete from the list of schemas of your project.

   **Result**

   The schema appears in the schema editor.
2. Click the delete icon 🗑.
3. Click **Yes** to confirm and save the changes.

## Splitting a schema view

The schema designer displays a single schema. However, you can split the schema designer to view two schemas simultaneously. This is useful for copying, editing, or viewing different components of very large schemas.

## Schema design overview

When you compose a schema by using HCL® OneTest™ Data you must define the types of the schema. You can also modify the types of the schema when you import any schema from local file system or use the schema generated by using any external resource.

You can find information about the tasks that you must perform to design and create the structural view of the schema in your project.

## Defining types

For optimum test data generation, you must define entire data of schema into types. Types are the components of schemas. After you design schema in the schema designer, you must define entire data as types by using type designer.

## Adding types to a schema

The types in the schema are arranged in a hierarchy. A new schema by default has Root type. You can add subtypes to the Root type to define a schema.

**Before you begin**

You have a schema for your project.

**About this task**

Types are the set of data objects. A new schema has a single Root type. All types are added as subtypes of the Root type and then as subtypes of other types.

1. Select the type under which you want to add a subtype. For example, select "Root".
2. From the menu of the selected type, click **Add**.

   **Result**

   The subtype is created of the similar type as of the selected type and it appears under the selected type.

   **Note:** You cannot add any type to a JSON or an XSD schema.

3. Save the changes.

**Results**

You can view the nested structure of types.

**What to do next**

You must define the properties of the types of the schema. For information about how to setup type properties, see related links at the end of this page.

## Copying a type

To create more types with similar properties, you can copy an existing type.

### About this task

In addition to using the drag-and-drop method, you can use the menu to copy a type.

1. Select the type that you want to copy and click **Copy** from the menu.
2. Select the target type—the type under which you want to copy the given type.
3. Click **Paste** from the menu of the target type.

### Results

You can view a new type with the properties similar to an existing type.

### What to do next

You can modify the name, description, and the properties of the copied type.

## Deleting a type

When you want to remove any type from the schema, you can delete it.

1. Select the type that you want to delete and click **Delete** from the menu.
2. Click **Yes** to confirm and save the changes.

   You can select more than one type simultaneously by using **Multiselect** option.

   **Note:** You cannot delete any type of a JSON or an XSD schema.

### Results

The selected type is deleted.

Related information

Objects, types, and classes

## Setting type properties

After you add types to your schema, you must set type properties.

### Before you begin

You must have a schema with types.

1. Select a type and click the menu of the type.
2. Click **Properties**.

   **Result**

   The **Properties** dialog box is displayed.
3. Enter the name, class, and description details of the type.
4. Choose the class of the type as **Category**, **Group**, or **Item**.

   **Result**

   The properties of the selected type is displayed.
5. Define the required properties for the selected type.
6. Click **Advanced properties** to provide additional property details of the type.

   You can set multiple properties for an item type.

   **Note:** If you want to import any JSON or XSD format schema, you cannot modify the properties of any type of the schema.
7. Save your changes.

**Results**

You have set the type properties of schema and saved them successfully.

**What to do next**

When you finish setting the properties for all the types of your schema, you must prepare a structural view of the schema by using the drag-and-drop method. See Designing a structural view of the schema on page 141.

**Note:**

If you set multiple properties for an item type, then HCL® OneTest™ Data considers the following order of priority while executing the properties:

- Data generation for multiple schemas based on schema integrity
- Pairwise data property
- Property to set Javascript rules
- Setting output rule in **Rule Editor**
- Property to set **Implied Default** value
- Regular expression property
- Property to set weightage values by using a file
- Property to set restrictions

Read the following topics to understand how to set certain properties for any item type:

Related information

Type properties

## Value assignation for item types

When you want to use specific item type values to generate the test data, then you can assign those values to the item types. To assign the values to any item type, you can either import the item type values or add the values manually.

### Prerequisites

You must have completed the following tasks:

- Created a project in HCL OneTest™ Server.
- Created a schema in HCL® OneTest™ Data.

You can assign values to any item type by using any one of the following methods:

- Assigning values for item types from a file on page 131
- Inserting values for item types manually on page 132

## Assigning values for item types from a file

When a file contains values for an item type on your computer, then you can assign those values to the item type to generate the test data.

#### Before you begin

You must have created a file with item type values on your computer.

1. Log in to HCL OneTest™ Server and open the team space that contains your project.
2. Open your project.

   **Result**

   The **Overview** page is displayed.
3. Click **Author > Data Fabrication**.

   **Result**

   The **Data Fabrication** page is displayed.
4. Click **Files** and import the file of item type values from your computer into your project.
5. Open your schema from the **Schemas** tab.
6. Select the item type for which you want to assign the values.
7. Select **Restrictions** from the **Item Properties** drop-down list of the **Properties** dialog box.
8. Enable the **Values from file** toggle button.
9. Provide the details of the file that you want to use to get the values of the item type in the **Values file** field.

   You can use any one of the following methods to provide the details of the file:

   **Choose from:**
   - Upload the file from your computer.
   - Provide the name of the imported file with an extension.

10. Enter the column number of the file where the item type values are specified in the **Values column number** field.
11. Save the changes.

**Results**

You have successfully assigned the values for item types from a file.

**What to do next**

You must prepare the structural view of the schema. See Designing a structural view of the schema on page 141.

## Inserting values for item types manually

When you want to assign specific values to an item type that does not exist in any file on your computer, then you can assign those values by inserting them manually.

1. Select your project in Rational® Test Automation Server and go to the **Data Fabrication** page.
2. Open your schema from the **Schemas** tab.
3. Select the item type for which you want to assign the values.
4. Assign a default value or multiple values.
5. Perform the following steps to assign a default value to the item type:
   a. Select **Item Properties** from the **Properties** dialog box.
   b. Click the **Advanced Properties** icon ☰ .
   c. Enter a default value in the **Implied Default** field.
6. Perform the following steps if you want to assign multiple values to the item type:
   a. Select **Restrictions** from the **Item Properties** drop-down list of the **Properties** dialog box.
   b. Click **Include**.
   c. Enter the value for the selected item type in the first row.
   d. **Optional:** Click **Description** to provide the details of the values of the item type.
   e. Click Menu ⋮, and then click the **Value include restrictions** icon ⬇ to add another row.
7. Save the changes.

**Results**

You have successfully inserted the values for item types manually.

**What to do next**

You must prepare the structural view of the schema. See Designing a structural view of the schema on page 141.

## Setting weightage values

You can set weightage values for an item type to determine how often the value of that item type can be displayed in the generated test data.

### Before you begin

- You must have a project and a schema.
- You must have created a weightage file that can be of CSV or Microsoft Excel Open XML format.

**Note:** The weightage file consists of one column for values of item types and another column for weightage values for each item type value. The weightage values must be integers.

### About this task

The property of weightage values is set as a restriction on an item type. The probability to get higher accuracy increases in the results of the weightage values in the following scenarios:

- When HCL® OneTest™ Data processes a large amount of data.
- When there is a large difference between the weightage values.

### Computation of weightage value

The probability of occurrence of any item type value in the generated test data can be computed by using the following formula:

```
Probability of occurrence of any item type value = Weight of each item type value / Total weight of all
 the item type values
```

The following table shows an example when you have a weightage file with five item type values and its corresponding weightage values:

| Item type values | Weightage values |
|---|---|
| `Red` | `50000` |
| `Blue` | `5000` |
| `Green` | `500` |
| `Orange` | `50` |
| `Purple` | `5` |

The total weightage value of all the item type values = `50000 + 5000 + 500 + 50 + 5 = 55555`.

The probability of occurrence of each item type value in the generated test data can be computed as shown in the following table:

| Item type values | Probability of occurrence | Computed values |
|---|---|---|
| Red | 50000/55555 | 0.9 |
| Blue | 5000/55555 | 0.09 |
| Green | 500/55555 | 0.009 |
| Orange | 50/55555 | 0.0009 |
| Purple | 5/55555 | 0.00009 |

From the computed values, `Red` has a higher probability value than other colors.

When you use this weightage file to set the weightage value for any specific item type and generate the test data, then you can find the generated test data with more occurrences of `Red` as compared to the other colors.

1. Select your project in Rational® Test Automation Server and go to the **Data Fabrication** page.
2. Click **Files** and import the weightage file from the local file system into your project.
3. Open your schema from the **Schemas** tab.
4. Select the item type on which you want to apply the weightage value.
5. Select **Restrictions** from the **Item Properties** drop-down list of the **Properties** dialog box.
6. Enable the **Values from file** toggle button.
7. Provide the name of the weightage file in the **Values file** field.
8. Enter the column number in **Values column number** and **Weights column number**.
9. Click **Save**.
10. Select the group and generate the test data.

**Results**

You can view more occurrences of the values that are set with higher weightage values as a result of the test data generation.

**What to do next**

You must prepare the structural view of the schema. See Designing a structural view of the schema on page 141.

Related information

Value assignation for item types on page 131

## Setting regular expressions

When you want values of an item type to appear in a certain pattern in the generated test data, you must set the **Regular Expression** property for that item type in a defined schema.

**Before you begin**

- You must have logged in to HCL OneTest™ Server.
- You must have created a project and a schema.

### About this task

The regular expression is a special text to describe a pattern. For example, you can specify regular expressions for email address, phone number, or website. HCL® OneTest™ Data provides you some sample regular expressions.

To set regular expressions to any item type, you can either insert a value or select from the sample list of regular expressions that best suits your testing requirements. HCL® OneTest™ Data then validates the value you inserted for the **Regular Expression** field and generates the values of the selected item type in a similar pattern.

1. Log in to HCL OneTest™ Server and open the team space that contains your project.
2. Open your project.

   **Result**

   The **Overview** page is displayed.
3. Click **Author > Data Fabrication**.

   **Result**

   The **Data Fabrication** page is displayed.
4. Open your schema from the **Schemas** tab.
5. Select the item type for which you want to set regular expressions.
6. Click **Regular Expression** from the **Properties** dialog box.

   The sample list of regular expressions is displayed.
7. Set the regular expression in the **Regular Expressions** field by using any one of the following methods:
   - Type the regular expression. HCL® OneTest™ Data then validates the regular expression.

     For example, `[0-9]{18}`

     **Tip:** If you want to reuse the validated regular expression then click **Add to sample list**, to add the expression in the list.
   - Select the regular expression from the sample list. The selected regular expression is displayed in the **Regular Expressions** field.
8. Click **Save**.

### Results

You have successfully set the regular expression for the selected item type.

### What to do next

You must prepare the structural view of the schema. See Designing a structural view of the schema on page 141.

Related reference

Regular expressions

## Integrating sample files into a schema

When you want to generate the test data, you must define a schema. HCL® OneTest™ Data provides sample files with item type values so that you can integrate the sample file into a schema by mapping it to any item type.

### Before you begin

- You must have logged in to HCL OneTest™ Server.
- You must have created a project and a schema.

### About this task

The **Files** tab on the **Data Fabrication** page is one of the design components of a project. From the **Files** tab, you can integrate a file into a schema by using any one of the following options:

- Create a file by retrieving the data.
- Upload a file from the local file system.
- Use any sample file.

Sample files contain the values for a specific item type. The sample files are in CSV and Excel file formats. You can map these sample files to an item type in a schema.

**Note:** You can map only one sample file to one item type in a schema.

You can use the **Search** field to search for any specific sample file from the **Sample Files** list. After you select a sample file, you can use the menu to manage the file for the following actions:

- Edit: You can modify the file name and the description of the selected sample file.
- Delete: You can delete the selected sample file from the list of files you can view in the **File** tab.
- Download: You can download the selected sample file into your local file system.

  **Note:** If you want to modify the selected sample file, you must first download the file into your local file system, and then edit the file for changes.

1. Select the project in Rational® Test Automation Server and go to the **Data Fabrication** page.
2. Click **Use sample** from the **Files** tab.

   The **Sample Files** dialog box is displayed.
3. Select one or more sample files that you want to import into your project, and then click **Select**.

4. Click the **Files** tab, and then download the selected sample file that you imported into your project.

   **Note:** You can open the sample file and check the column number of the file that you want to map with the item type.

5. Open your schema from the **Schemas** tab.
6. Select **Restrictions** from the **Item Properties** drop-down list of the **Properties** dialog box.
7. Enable the **Values from file** toggle button.
8. Provide the file name of the selected sample file with an extension in the **Values file** field.
9. Enter the column number of the sample file where the item type values are specified in the **Values column number** field.
10. Click **Save**.

### Results

- You can view the list of all the sample files you have selected in the **Files** tab.
- You have successfully integrated the sample file into a defined schema.

### What to do next

You must prepare the structural view of the schema. See Designing a structural view of the schema on page 141.

## Adding JavaScript rules

When you have a JavaScript file with defined rules to create values for any item type, then you can add the JavaScript rules to create the values for the mapped item type.

### Before you begin

- You must have created a project and a schema.
- You must have created a JavaScript file with defined rules in your local file system.

### About this task

If you want to add the JavaScript rules to create the values for the mapped item type, then you must use the following command to define the rules in the JavaScript file:

function getData**()**

You can additionally set any one of the following properties for the mapped item type by using the JavaScript rules:

- Regular expression: The JavaScript rule uses the regular expression that you enter in the **Properties** dialog box as an input for the regular expression rule.
- Seed value: The JavaScript rule uses the seed value that you enter in the **Generate Data** dialog box as an input for the seed value rule.

To set either regular expression or seed value property, you must define the rules in the JavaScript file by using the following commands:

| Property | Command | Parameter description |
|---|---|---|
| Regular expressions | function getData**(param)** | **(param)**: Represents the regular expression pattern in the **Regular Expression** field. |
| Seed value | function getData**(param)** | **(param)**: Represents the seed value in the **Numeric Seed Value** field of the **Generate Data** dialog box. |

**Note:** If you set the values for both regular expression and seed value properties in HCL® OneTest™ Data and define the rule in the JavaScript file by using the command function getData**(param)**, then only the value set for regular expression property is accepted while creating the item type values.

After you define rules in the JavaScript files, you can import the JavaScript file from your local file system into your project, and then map the file to any item type in a schema.

1. Select the project in Rational® Test Automation Server and go to the **Data Fabrication** page.
2. Click **Files** and import the JavaScript file from the local file system into your project.
3. Open your schema from the **Schemas** tab.
4. Select the item type for which you want to create the values by using the JavaScript file.
5. Select **Item Properties** from the drop-down list of the **Properties** dialog box.
6. Select **Text** in the **Item Subclass** field.
7. Click the **Advanced Properties** icon ▤ .
8. Optional. Provide the pattern in the **Regular Expression** field, if you want to set the regular expression for the mapped item type.
9. Provide the name of the JavaScript file with an extension in the **JavaScript Name** field.

   **Important:** You must enter the correct file name of the JavaScript file that you imported into your project. For example, `address.js`.
10. Click **Save**.

**Results**

You have successfully added the JavaScript rules for the item type.

**What to do next**

You must prepare the structural view of the schema. See Designing a structural view of the schema on page 141.

## The pairwise data generation method

When you test any application, you might want to ensure that the test data that you use covers all possible combinations of item type values. To achieve this goal of maximum coverage of all possible scenarios, you must apply the pairwise data generation method on the required item types of a schema.

For testing any application, you must attain the maximum test coverage. You can achieve the maximum test coverage and discover more defects with less effort and time by applying the pairwise data generation method. The pairwise data generation method in HCL® OneTest™ Data helps to generate all the possible combinations of the item type values for each pair of item types. The use of this method decreases the number of the generated test data but covers all the possible combinations of the item type values.

For example, consider that there are three item types in a schema such as a list box with five values (zero, one, two, three, four), one radio button with two values (selected or unselected), and one check box with two values (selected or cleared). Therefore, the combination becomes 5x2x2 as the generated test data, which is equal to 20 values. These 20 values are required to cover all combinations of the item types of the schema without applying the pairwise data generation method. After you apply the pairwise data generation method, you can observe that the generated test data comprises all combinations of item type values for the three pairs of item types among the three item types. This pairing of item types results in 12 combinations of values of the generated test data.

For more information about how to set the pairwise data generation method in HCL® OneTest™ Data, see Setting the pairwise data property on page 139.

## Setting the pairwise data property

When you want to generate an optimum test data that covers all possible combinations of item type values for each pair of item types in a schema, you can set the pairwise data property for those item types.

### Before you begin

You must have completed the following tasks:

- Created a schema in HCL® OneTest™ Data.
- Read the details about how to assign the values to an item type. See Value assignation for item types on page 131.

### About this task

The pairwise data generation method is a combinational method to find all the possible values of the selected pairs of item types. The pairwise data generation method is efficient when you have a large number of item types and the item type values in a schema. When you apply the pairwise data generation method in such cases, the number of generated test data decreases drastically although covering all the combinations. To implement the pairwise data generation method, you must have multiple item types in a group type of a schema. To assign item type values, you must either import the values of item type from a file or insert values for each item type.

1. Select your project in Rational® Test Automation Server and go to the **Data Fabrication** page.
2. Open your schema from the **Schemas** tab.
3. Select the item type from **Dictionary** for which you want to set the pairwise data property.
4. Select **Restrictions** from the **Item Properties** drop-down list of the **Properties** dialog box.
5. Set values for the selected item type by using any one of the following methods:

   **Choose from:**

   - Import item type values from the file by using **Values from file**.
   - Include values for the selected item type in the **Properties** dialog box.
6. Save the changes.
7. Select **Item Properties** from the drop-down list of the **Properties** dialog box.
8. Click the **Advanced Properties** icon ▤.
9. Select **Yes** from the **Pairwise Data** drop-down list.

   **Note:** The default value of the **Pairwise Data** drop-down list is No.
10. Save the changes.

    **Important:** You must select multiple item types to apply the pairwise data generation method.

    **Note:** When you want to apply the pairwise data generation method for each item type, you must repeat step 4 through step 10.

**Results**

You have successfully set the pairwise data property for the selected item type.

**What to do next**

You must select a group type to generate the test data of a schema. See Generating test data on page 167.

**Note:** The group type that you select to generate the test data can comprise both the item types set for the pairwise data property and the item types that are not set for this property.

**Important:** When you generate the test data for a group type with a few or all of the item types set for the pairwise data property, then during the test data generation, the value you enter for the **Number of records** field is ignored. Based on the pairwise data property, HCL® OneTest™ Data auto calculates the number of records.

Related information

Value assignation for item types on page 131

## Designing a structural view of a schema

To define how the item and group types are organized and are related to each other, you must design the structural view of a schema before generating the test data.

### Before you begin

You must have a group type in your schema.

### About this task

When you design the structural view of a massive schema, you can filter the item or group types of your schema by using **Filter**.

1. Double-click any group type from **Dictionary**.

   Alternatively, you can click the menu of the selected group, and then select **Structure**.

   **Result**

   The **Structure** dialog box appears.
2. Drag and drop the types from **Dictionary** to the **Structure** dialog box to map **Item** and **Group** types.

   **Note:** You can use the **Category** type to organize the data dictionary entities. You cannot map the **Category** type in the structure view.

3. Save the changes.

### What to do next

You can set restrictions and apply functions to define the component rules to any item type. You can also generate the test data without applying restrictions and functions. See Generating test data on page 167.

## Setting restrictions

When you define any item type for a schema, you can specify valid values to that type. You can define validations to the types by setting the restrictions.

### Before you begin

- You must have a group type in your schema.
- You must have a structured view of your schema.

### About this task

You can set restrictions only for item types of your schema.

1. Open the structure view of your schema.
2. Click the context menu for each item type and set the required restrictions.

   You can set the following restriction rules:

    - Optional
    - Required
    - Set Range
    - Unlimited
3. Save the changes.

   You can modify the properties of the selected item type by using **Properties** in the context menu.

   For information about item restrictions, see related links at the end of this page.

**What to do next**

You can apply functions to define component rules to any item type. You can also generate test data without applying functions. See Generating test data on page 167.

Related information

Restrictions settings

## Applying functions

You can apply functions on the item types of a schema to define the component rules.

**Before you begin**

You must have a defined schema in a schema designer. See Schema design overview on page 128.

**About this task**

You can also apply functions on the types of the imported JSON or XSD schemas.

1. Log in to HCL OneTest™ Server and open the team space that contains your project.
2. Open your project.

   **Result**

   The **Overview** page is displayed.
3. Click **Author > Data Fabrication**.

   **Result**

   The **Data Fabrication** page is displayed.
4. Click **Schemas**, and then select the schema for which you want to generate the test data.
5. Select the group type, and then open the structural view of the schema.
6. Select the item type on which you want to apply the function, in the **Structure** dialog box.
7. Click the menu on the right side of the **Structure** dialog box.

   **Result**

   A blank field is displayed next to the item type.
8. Click the blank field.

   **Result**

The **Rule Editor** pane is displayed on the page.

9. Enter the equal sign (=) in the **Rule Editor** pane.
10. Provide the function in the **Rule Editor** pane.

    You can enter the function by performing any one of the following methods:

    **Choose from:**

    - Enter the function manually.
    - Select from the list of functions by using the following steps:
        a. Click the **Insert mapping functions** icon *fx*.
        b. Select the function that you want to apply as a component rule, and then click **Insert** from the menu of the selected function.

    For example, if you want the generated value of an item type in uppercase, then you must enter or select the `UPPERCASE()` function.

    ```
    =UPPERCASE(RANDDATA())
    ```
11. **Optional:** Enter a parameter for the selected function.

    You can provide the parameter value either as a constant value or as a reference of the item type.

    You must use the following syntax to provide the parameter in any function:

    ```
    <item type name>:Out1
    ```

    where:

    - *item type name* - Represents the name of the item type.
    - `:Out1` - Represents a standard output argument.

    For example, consider that you have a group type as `Flowers` with `Colors` and `Pattern` as two item types. If you want to generate the value of `Colors` item type in uppercase, then you can select the item type `Pattern` in the **Structure** dialog box. You must open the **Rule Editor** pane, and then select the `UPPERCASE()` function with reference of `Colors` item type as the parameter.

    ```
    = Uppercase (Colors:Out1)
    ```
12. Press the **Enter** key.

    **Result**

    The selected function is displayed in the item type field of the **Structure** dialog box.
13. Save the changes.

### Results

You have successfully applied the function on an item type of the schema.

### What to do next

You can generate the test data for the designed schema. See Generating test data on page 167.

Related information

Functions and expressions

Designing a structural view of a schema on page 141

Setting a function to concatenate the values of item types on page 144

## Setting a function to concatenate the values of item types

You can set a function to concatenate the values of multiple item types so that the values are displayed as the value of another item type.

**Before you begin**

You must have completed the following tasks:

- Defined a schema. See Schema design overview on page 128.
- Created the structural view of the schema. Designing a structural view of a schema on page 141.

1. Log in to HCL OneTest™ Server and open the team space that contains your project.
2. Open your project.

   **Result**

   The **Overview** page is displayed.
3. Click **Author > Data Fabrication**.

   **Result**

   The **Data Fabrication** page is displayed.
4. Click **Schemas**, and then select the schema for which you want to generate the test data.
5. Open the structural view of the schema.

   **Important:** You must have all the item types in the same group type for the values that you want to concatenate.

   **Result**

   The **Structure** dialog box is displayed.
6. Select the item type for which you want the values of the generated test data in the concatenated format.

   **Note:** You must not place the item type, for which you want the values in the concatenated format, in the same group type.

   For example, consider that a schema includes a group type, `group_name1` and three item types such as `item type1`, `item type2`, and `item type3`. If you want to concatenate the values of `item type1` and `item type2` as the generated value of `item type3`, then you must add `item type1` and `item type2` item types in the `group_name1` group type. You must ensure that `item type3` does not exist in the `group_name1` group type.

7. Click the menu on the right side of the **Structure** dialog box.

   **Result**

   A blank field is displayed next to all the item types.
8. Click the blank field of the item type for which you want to set the function for concatenation.

   **Result**

   The **Rule Editor** pane is displayed on the page.
9. Enter the equal sign (=) in the **Rule Editor** menu.
10. Select the `SERIESTOTEXT()` function from the **Insert mapping functions** *fx* icon.

    **Note:** You can also provide the following rule manually in the **Rule Editor** pane: `"item type1" + "item type2"`.
11. Provide the name of the group type as the parameter for this function.

    For example, if you have a `group_name1` group type, then you must provide the following function:

    ```
    SERIESTOTEXT(group_name1:OUT1)
    ```
12. Press Enter.

    **Result**

    The selected function is displayed in the item type field of the **Structure** dialog box.
13. Save the changes.

**Results**

You have successfully set the function to concatenate the values of multiple item types.

**What to do next**

You can generate the test data for the designed schema. See Generating test data on page 167.

Related information

Functions and expressions

## Test data generation from structured schemas

When you want to test your application by using randomly generated test data, then you must generate the test data from the structured schemas.

You can generate the test data for schemas that you import, create, or generate in HCL® OneTest™ Data by using any external resource. When you generate schemas from any JDBC-supported databases, then you can generate the test data simultaneously for multiple schemas that comprise parent and child schemas. The parent and child schemas are related to each other by using common item types. The item types of the child schema refer to the item types of the parent schema. After you verify the relationship among the selected schemas, you can generate the test data. When you generate the test data by using the related schemas, the generated values of referring item types of the child schema are available in the generated values of the referred item types of the parent schema, based on referential integrity.

The referential integrity among the parent and child schemas in HCL® OneTest™ Data supports the following relations:

- One-to-one
- One-to-many
- Many-to-one
- Self-reference

**Note:** When you generate the test data by using multiple schemas, you must select schemas that are generated by using the same JDBC connection.

When you generate the test data by using the schemas generated in HCL® OneTest™ Data, then you can either download the generated test data or can insert it into the connected database. You can insert the test data into the following databases if there is a successful connection between HCL® OneTest™ Data and the database:

- JDBC-supported databases
- MongoDB

## HCL OneTest Data support to write the generated test data in JDBC-supported databases

When you want the generated test data in any Java Database Connectivity (JDBC) supported database, you can generate the test data by using HCL® OneTest™ Data. Subsequently, HCL OneTest Data inserts the generated test data directly in the configured JDBC-supported database.

To enable HCL® OneTest™ Data to write the generated test data in the database, you must create a sample schema with tables in the installed JDBC-supported database, establish a connection with the installed database, select the generated schema in HCL® OneTest™ Data, and then generate the test data of the selected schema.

HCL® OneTest™ Data supports the following JDBC-supported databases:

- MySQL Server
- Microsoft SQL Server
- IBM DB2

HCL® OneTest™ Data supports the test data generation for the following data types:

- INTEGER
- VARCHAR
- CHAR
- DATETIME

- DATE
- TIME
- BOOLEAN

**Note:** You must add the restriction values for BOOLEAN data type as `0` and `1`.

### Prerequisites

The following prerequisites must be met to enable HCL OneTest Data to write the generated test data:

- You must have access to HCL OneTest™ Server.
- You must have installed any JDBC-supported database, such as MySQL, Microsoft SQL Server, and so on.

### Task flow

You must perform the following tasks in sequence to enable HCL OneTest Data to write the generated test data directly in the JDBC-supported database. The table also provides you the links to the information about the tasks.

| Task | More Information |
|---|---|
| Create a sample schema in a database | Creating a sample schema in a database on page 147 |
| Establish a JDBC connection with HCL OneTest Data | Establishing a JDBC connection with HCL OneTest Data on page 149 |
| Generate a schema in HCL OneTest Data | Generating a schema in HCL OneTest Data on page 150 |
| Insert the generated test data in the database | Inserting the generated test data in a database on page 152 |

## Creating a sample schema in a database

To generate a schema in HCL® OneTest™ Data, you must create a schema with a table in the database.

### About this task

After you download the connector jar files of the JDBC-supported database, you must copy the jar files from your computer to the pods such as *HIP REST* and *HIP Server*. These pods are responsible for the following actions:

- *HIP Server* is responsible to establish a connection with the JDBC-supported database. This pod also generates a schema by using the tables of the database. The structure of a schema is a flat hierarchy, where all the item types are in a single group called a *Row*. The item types in a schema refer to the columns of the table.
- *HIP REST* is responsible to run the map and generate the data. This pod also writes the generated data to the JDBC-supported database.

Both the pods use the connector jar file to communicate with the JDBC-supported database.

You can download the connector jar files from the following locations:

| Connector jar | Location |
|---|---|
| Mysql Server connector jar | https://dev.mysql.com/downloads/connector/j/ |
| Microsoft SQL Server connector jar | https://docs.microsoft.com/en-us/sql/connect/jdbc/download-microsoft-jdbc-driver-for-sql-server?view=sql-server-ver15 |
| IBM DB2 | https://www.ibm.com/support/pages/db2-jdbc-driver-versions-and-downloads |

1. Download the JDBC-supported database connector jar file to your local file system.

   **Note:** You must download the connector jar file that is compatible with the database server.

2. Copy the connector jar file from your local file system to the pods by using the following commands from the command prompt.

   For example, run the following commands if you are using the MySQL database:

   For *HIP REST*

   ```
   kubectl cp /<source path>/mysql-connector-java-8.0.19.jar {my-ots}-<hip rest podname>:/opt/runtime/
   extjar/mysql-connector-java-8.0.19.jar -n test-system
   ```

   For *HIP Server*

   ```
   kubectl cp /<source path>/mysql-connector-java-8.0.19.jar {my-ots}-<hip server podname>:/opt/hcl/hip/
   extjar/mysql-connector-java-8.0.19.jar -n test-system
   ```

   **Notes:**

   - Depending on the database you use, you must copy different connector jar files to the pods.
   - While copying the jar files to the pods, you must ensure that the names of pods are same as those installed on HCL OneTest™ Server.

     For example:

        - *HIP Rest* is displayed as `{my-ots}-hip-rest-6757c99d9-4qjtl`
        - *HIP Server* is displayed as `{my-ots}-hip-server-0`
    - When you install or restore HCL® OneTest™ Data, you must copy the connector jars files again.

3. Create a schema with a table in the database by using the following commands from the command prompt.

   For example, you can use the following command to create a table in a schema:

```
create schema <schema_name>;
use <schema_name>;
    CREATE TABLE IF NOT EXISTS <table_name> (
    Column1 <DataType> Constraint,
    Column2 <DataType> Constraint,
    Column2 <DataType> Constraint,,,,,,
);
```

   **Remember:** In HCL® OneTest™ Data, you must refer to the schema name and the table name that you created by using this command.

#### Results

You have created a sample schema with a table in the database.

#### What to do next

After you create a schema in the database, you must establish the connection between HCL® OneTest™ Data and the JDBC-supported database. See Establishing a JDBC connection with HCL OneTest Data on page 149.

You can view the schema name that you created in the database as the value of the **Catalog** or **Schema** property in HCL® OneTest™ Data.

## Establishing a JDBC connection with HCL OneTest Data

You must establish a connection between HCL® OneTest™ Data and the JDBC supported database to generate the schema and to write the generated test data in the database.

#### Before you begin

You must have created a sample schema in the database. See Creating a sample schema in a database on page 147.

1. Select your project in Rational® Test Automation Server and go to the **Data Fabrication** page.
2. Click **New** from the **Connections** tab to create a new connection.
3. Select *JDBC* as a connection type and click **Next**.
4. Enter the following properties of the connection in the **Properties** dialog box.

a. Enter the URL of the database.

   You must provide the URL in the following format:`jdbc:<DatabaseType>://<HostName/IPAddress>:<PortNumber>/<CustomSettings>`

   > **Note:** You must provide the database name in `<CustomSettings>` for establishing connection with IBM DB2 database.

   For example,

   - MySQL Server - `jdbc:mysql://10.0.2.15:8081`
   - Microsoft SQL Server - `jdbc:sqlserver://10.134.59.16:1433`
   - IBM DB2 - `jdbc:db2://10.134.59.16:50000/testdb`

b. Enter the name and password to access the database.

5. Click **Test** to verify whether the connection is established successfully.

   > **Note:** If the connection fails to establish, it can be because of one of the following reasons:
   > - Mismatch in the version of the drivers.
   > - Unreachable database.
   > - Missing database.
   > - Unsuccessful copying of the connector jar file.

**Results**

You have successfully established the connection between HCL® OneTest™ Data and the database.

**What to do next**

You must provide a name for the connection, and then generate a schema. See Generating a schema in HCL OneTest Data on page 150.

## Generating a schema in HCL OneTest Data

After you establish the connection between HCL® OneTest™ Data and the JDBC-supported database, you can generate a schema in HCL® OneTest™ Data from the configured database.

1. Click the **Connections** tab and select the connection that you created.
2. Click the context menu and select **New Action**.
3. Select **Target** as the **Action Type** in the **New Action** dialog box and click **Next**.
4. Enter the following properties of the schema that you want to generate, and then click **Next**.

| Properties | Actions | Required/Optional |
|---|---|---|
| Target | Select the target as *Table* that you created in the database from the list. | Required |
| Write Mode | Select *Insert* to insert the data in the database. | Required |
| Catalog | Select the schema name that you created in the database from the list. | Required |
| Schema | Select the schema name that you created in the database from the list.<br>The schema name is displayed either in the **Catalog** or in the **Schema** field for the selected database. | Optional |
| Table | Select the name of the table that you created in the database from the list. | Required |
| Logging | Select logging as *ON*, if you want the actions performed in the database to be logged in the log files. | Optional |
| Failure Action | Select the action to be performed when any failure occurs while generating the schema.<br>The default failure action is *Rollback*. | Optional |
| REST Output | Select to override the action after deploying in *HIP REST*. | Optional |

5. Click **Generate**.

   The schema is generated in HCL® OneTest™ Data by using the schema you created in the database. HCL® OneTest™ Data generates the schema name as *schema_<number>*.

   **Note:** You must select this schema name as the schema for HCL® OneTest™ Data to generate the test data.

6. Select the schema type as **Row** and click **Next**.

7. Provide the identification details of the new action, and then click **OK**.

   The new action is created and is listed under the JDBC connection.

### Results

You have successfully generated the schema in HCL® OneTest™ Data.

**Important:** If you want to generate schemas for other tables in the same database, then instead of creating a new connection, you must click **New Action** on the existing JDBC connection.

### What to do next

You can provide the action name under the **Identification** dialog box, and then generate the test data in the database. See Inserting the generated test data in the database on page 152.

## Inserting the generated test data in a database

After you generate the schema from the configured database in HCL® OneTest™ Data, you can directly insert the generated test data into the database.

### Before you begin

You must have generated the schema by using the database. See Generating a schema in the database on page 150.

1. Select the schema that you generated with the following name from the **Schema** tab:

   *schema_<number>*
2. Click the menu of the group type of the schema and select **Structure**.

   You can view the list of all item types associated with the selected group type in the structure view.
3. Click the menu for all the item types to set the property as **Set Required**, and then click **Save**.

   **Note:** If you have a numeric item type, you must set **Min size (digits)** and **Max size (digits)** properties.

   For example, when you select `Integer` as **Presentation** then you can set the **Min size (digits)** as `1` and the **Max size (digits)** as `9`.
4. Select the group type in the **Dictionary** dialog box and click **Generate Data** from the menu.

   The **Generate Data** dialog box is displayed.
5. Select the **Connection** option and provide values for the **Number of records** and **Numeric Seed Value**.
6. Select the connection type from the list, and then click **OK**.

### Results

You have successfully inserted the generated test data from HCL® OneTest™ Data in the JDBC supported database.

**What to do next**

You can click the **Jobs** tab to view the status of the job. The tooltip of the job displays the following details:

- URL of the connection.
- Name of the table.
- Schema name from which the test data is generated.

**Important:**

You cannot download the generated test data from the **Jobs** page because the test data is written to the configured database and is not available in HCL® OneTest™ Data.

You can log in to the configured database to view the generated test data.

## HCL OneTest Data support to generate test data for an SAP IDoc or DXOB file

When you want to generate the test data for an SAP Intermediate Document (IDoc) or Data Transfer Object (DXOB) file, you can generate a schema in HCL® OneTest™ Data by using the structured schema of the SAP IDoc or DXOB file. You can then generate the test data by using the generated schema in HCL® OneTest™ Data.

The SAP applications provide file-based interfaces, such as the SAP IDoc and DXOB interfaces. The SAP IDoc interface supports Application Link Enabling (ALE) and Electronic Data Interchange (EDI) file formats, and the SAP DXOB interface supports the DXOB files.

You must perform the following tasks to generate the test data for any SAP IDoc or DXOB file:

- Schema generation for an SAP IDoc or DXOB file on page 153
- Generating test data on page 167

Related information

SAP applications

## Schema generation for an SAP IDoc or DXOB file

When you want to generate the test data for an SAP IDoc or DXOB file, then you must generate a schema for the selected SAP IDoc or DXOB file.

The SAP IDoc or DXOB file is a standard data structure in SAP applications to transfer data between SAP system applications and external resources. Every SAP IDoc or DXOB file has a unique number.

If you have an SAP IDoc or DXOB file that you want to use for test data generation in your local file system, then you must import the file into your HCL® OneTest™ Data project. After importing the file, you must establish a connection between the SAP IDoc or DXOB interface and HCL® OneTest™ Data. The established connection helps to transfer and interpret the data of the imported SAP IDoc or DXOB file. The interpreted information then helps HCL® OneTest™ Data to generate a schema.

You must perform the following tasks to generate a schema in HCL® OneTest™ Data for the SAP IDoc or DXOB file:

## Importing an SAP IDoc or DXOB file

When you want to establish a connection with an SAP IDoc or DXOB interface, you must import the SAP IDoc or DXOB file into your HCL® OneTest™ Data project from your local file system.

### Before you begin

You must have completed the following tasks:

- Created a project.
- Created the SAP IDoc or DXOB file in your local file system.

1. Select your project in Rational® Test Automation Server and navigate to the **Data Fabrication** page.
2. Click **Files**, and then click **New** to import the SAP IDoc or DXOB file from the local file system into your project.

### Results

You have successfully imported the SAP IDoc or DXOB file from the local file system into your project.

### What to do next

You must establish a connection between the SAP IDoc or DXOB interface and HCL® OneTest™ Data. See Establishing an SAP IDoc or DXOB interface connection with HCL OneTest Data on page 154.

## Establishing an SAP IDoc or DXOB interface connection with HCL OneTest Data

When you want to generate a schema for an SAP IDoc or DXOB file, you must establish a connection with the SAP IDoc or DXOB interface.

### Before you begin

You must have imported the SAP IDoc or DXOB file into your project.

1. Select your project in Rational® Test Automation Server and navigate to the **Data Fabrication** page.
2. Click **Connections**.
3. Click **New** to create a new connection type.

   The **New Connection** dialog box is displayed.
4. Select **SAP IDoc** from the **Type** drop-down list, and then click **Next**.
5. Set the following properties under **Connection Properties**:

    - Enable **Bypass Client Connection**.
    - Select `2 (Unicode)` as **Bytes Per Character**.
6. Click **Next**, and then provide a name to the established connection.
7. Click **OK**.

   You can see the name of the SAP IDoc connection under the **Connections** tab.

#### Results

You have successfully established the connection between HCL® OneTest™ Data and the SAP IDoc or DXOB interface.

#### What to do next

You must generate a schema in HCL® OneTest™ Data by using the SAP IDoc or DXOB file. See Generating schema by using an SAP IDoc or DXOB file. on page 155

## Generating schema by using an SAP IDoc or DXOB file

When you want to generate the test data for an SAP IDoc or DXOB file, then you must generate a schema for the selected SAP IDoc or DXOB file.

#### Before you begin

You must have established the connection with the SAP IDoc or DXOB interface.

1. Select your project in Rational® Test Automation Server and navigate to the **Data Fabrication** page.
2. Click **Connections**.
3. Select the SAP IDoc connection name that you established, and then click **New Action** from the menu.

   The **New Action** dialog box is displayed.
4. Select **Target** as the action type, and then click **Next**.
5. Enter the following properties for the new action:
    a. Select one of the following file types from the **Data record type** drop-down list.
        - ALE - 1000 byte segments (ALE/RFC port)
        - EDI - Segments terminated by line feed (file port)
        - DXOB - Data Transfer Object
    b. Select the imported SAP IDoc or SAP DXOB file name from the **IDoc or DXOB File** drop-down list.
    c. Select **Native** from the **Character set** drop-down list.
6. Click **Next**.
7. Click **Generate** to generate a new schema by using the imported SAP IDoc or SAP DXOB file.

   A schema is generated as `anonymous_schema_<number>`.
8. Select **Schema type**.
9. Click **Next**.
10. Provide the name to the new action, and then click **OK**.

The new action is created and listed under the SAP IDoc or DXOB connection name that you established.

**Note:** You can create multiple actions for each established connection.

11. Click **Save**.

**Results**

You have successfully generated a schema for the SAP IDoc or DXOB file in HCL® OneTest™ Data.

**Note:** You can modify the schema name `anonymous_schema_<number>` from the **Schemas** tab.

**What to do next**

You can generate the test data by using the generated schema. See Generating test data on page 167.

## HCL OneTest Data support to generate test data for SAP BAPI

When you want to generate test data for SAP Business Application Programming Interface (BAPI), you can generate a schema in HCL® OneTest™ Data based on structured schema of SAP BAPI function modules. You can then generate the test data by using the generated schema in HCL® OneTest™ Data.

SAP BAPIs are API methods of SAP Business Object Types. HCL® OneTest™ Data uses SAP BAPI to access the application layer of the SAP System.

**Prerequisites**

You must have access to HCL OneTest™ Server.

**Task flow**

You must perform the following tasks in sequence to generate the test data for SAP BAPI in HCL® OneTest™ Data.

The table also provides you the links to the information about the tasks:

| Task | More Information |
|---|---|
| Install SAP connector files in HCL® OneTest™ Data | Installing the SAP connector files in HCL OneTest Data on page 157 |
| Establish an SAP BAPI connection with HCL® OneTest™ Data | Establishing an SAP BAPI connection with HCL OneTest Data on page 157 |
| Generate a schema in HCL® OneTest™ Data | Generating a schema in HCL OneTest Data on page 158 |

| Task | More Information |
|---|---|
| Generate test data in HCL® OneTest™ Data | Generating test data on page 167 |

Related information

SAP applications

## Installing the SAP connector files in HCL OneTest Data

To generate a schema in HCL® OneTest™ Data, you must install the SAP connector files in HCL® OneTest™ Data.

1. Download the SAP connector files libsapjco3.so and sapjco3.jar to your local file system.
2. Copy the connector files from your local file system to the *HIP Server* pod by using the following commands from the command prompt:

   ```
   kubectl cp /source path/libsapjco3.so {my-ots}-hip-server-0:/opt/hcl/hip/libs/libsapjco3
   ```

   ```
   kubectl cp /source path/sapjco3.jar {my-ots}-hip-server-0:/opt/hcl/hip/libs/sapjco3.jar
   ```

   **Note:** When you install or restore HCL® OneTest™ Data, you must copy the connector files again.

3. Enter the following command to change the current user as the root user:

   ```
   chown <user>:root libsapjco3.so sapjco3.jar
   ```

4. Run the following command to modify the access permissions of the connector file:

   ```
   chmod 777 libsapjco3.so sapjco3.jar
   ```

### Results

You have successfully installed the SAP connector files in HCL® OneTest™ Data.

### What to do next

You must establish the connection between HCL® OneTest™ Data and the SAP BAPI connector. See Establishing an SAP BAPI connection with HCL OneTest Data on page 157.

## Establishing an SAP BAPI connection with HCL OneTest Data

When you want to generate a schema for SAP BAPI, you must establish a connection between HCL® OneTest™ Data and SAP BAPI.

### Before you begin

You must have installed the SAP connector files in HCL® OneTest™ Data. Ensure that you have installed the connector files a few minutes back. See Installing SAP connector files in HCL OneTest Data on page 157

1. Select your project in Rational® Test Automation Server and navigate to the **Data Fabrication** page.
2. Click **Connections**.
3. Click **New** to create a new connection type.

   The **New Connection** dialog box is displayed.
4. Select **SAP BAPI** from the **Type** drop-down list, and then click **Next**.
5. Set the following properties under **Connection Properties**, and then save the changes:
   - Enter the SAP System host name in the **Host ID** field.
   - Enter **Client Number** and **System ID** of the SAP System.
   - Provide the SAP System credentials in the **User** and **Password** fields for authentication.
6. Go to the **Connection Properties** pane of the **Data Fabrication** page, and then click **Test** to verify whether the connection is established successfully.

   **Note:** Your connection might fail to establish because of any one of the following reasons:
   - Trouble at the SAP server port.
   - Firewall blockage between Rational® Test Automation Server and the SAP server.
   - Unsuccessful copying of the connector files.
   - Unsuccessful refresh of the *HIP Server* pod.
7. Click **Next**, and then provide a name for the SAP BAPI connection that you established.
8. Click **OK**.

   You can view the name of the SAP BAPI connection under the **Connections** tab.

**Results**

You have successfully established the connection between HCL® OneTest™ Data and SAP BAPI.

**What to do next**

You must generate a schema in HCL® OneTest™ Data by using the SAP BAPI. See Generating a schema in HCL OneTest Data on page 158.

## Generating a schema in HCL OneTest Data

After you establish the connection between HCL® OneTest™ Data and an SAP BAPI, you can generate a schema in HCL® OneTest™ Data.

1. Select your project in Rational® Test Automation Server and navigate to the **Data Fabrication** page.
2. Click **Connections**.
3. Select the SAP BAPI connection name that you established, and then click **New Action** from the menu.

   The **New Action** dialog box is displayed.
4. Select **Target** as the action type, and then click **Next**.
5. Enter the following properties for the new action:

   a. Click **Fetch** to select the function module category from the **BAPI Function Module category** drop-down list.
   b. Click **Fetch** to select the function module of the selected function module category from the **BAPI Function Module** drop-down list.
6. Click **Next**.
7. Select **New**, and then click **Generate** to generate a new schema in HCL® OneTest™ Data.

   The schema with a name as *schema_<number>* is generated in HCL® OneTest™ Data.

   The **Schema Type** is populated automatically based on the selected BAPI function module.

   **Note:** You must select this schema name as the schema for HCL® OneTest™ Data to generate the test data.

8. Click **Next**, and then provide the identification details of the new action.
9. Click **OK**.

   The new action is created and is listed under the SAP BAPI connection.

**Results**

You have successfully generated a schema for SAP BAPI in HCL® OneTest™ Data.

**Note:** You can modify the schema name `schema_<number>` from the **Schemas** tab.

**What to do next**

You can generate the test data by using the generated schema. See Generating test data on page 167.

## HCL OneTest Data support to write the generated test data in MongoDB

When you use MongoDB as a database in your application testing environment and you want the generated test data in MongoDB, then you can generate the test data by using HCL® OneTest™ Data. Subsequently, HCL® OneTest™ Data inserts the generated test data directly in MongoDB.

To enable HCL® OneTest™ Data to write the generated test data in the database, you must either import or create a schema in HCL® OneTest™ Data, and then establish a connection with MongoDB. After you establish the connection, you must associate the selected schema with MongoDB, and then generate the test data by using HCL® OneTest™ Data. Subsequently, HCL® OneTest™ Data writes the generated test data directly in MongoDB.

**Important:** HCL® OneTest™ Data always writes the generated test data in MongoDB in the JSON format.

**Prerequisites**

You must have access to HCL OneTest™ Server.

**Task flow**

You must perform the following tasks in sequence to enable HCL® OneTest™ Data to write the generated test data directly in MongoDB. The table also provides you the links to the information about the tasks.

| Task | More information |
|---|---|
| Create a schema in HCL® OneTest™ Data<br>Import a JSON or XSD into HCL® OneTest™ Data project | Create a schema in HCL OneTest Data on page 123<br>Import a JSON or XSD into HCL OneTest Data project on page 121 |
| Establish a MongoDB connection with HCL® OneTest™ Data | Establishing a MongoDB connection with HCL OneTest Data on page 160 |
| Associate a schema with MongoDB | Associating a schema with MongoDB on page 161 |
| Insert the generated test data in a database | Inserting the generated test data in a database on page 162 |

## Establishing a MongoDB connection with HCL OneTest Data

You must establish a connection between HCL® OneTest™ Data and MongoDB to associate a schema with the database.

**Before you begin**

You must have completed the following tasks:

- Created or imported a schema in the HCL® OneTest™ Data project.
- Installed MongoDB on your computer and the MongoDB instance must be ready to connect. For more information, refer to Install MongoDB.
- Created a database in MongoDB. For more information, refer to Database Methods.

1. Select your project in Rational® Test Automation Server and go to the **Data Fabrication** page.
2. Click **Connections**.
3. Click **New** to create a new connection.

   The **New Connection** dialog box is displayed.
4. Select *MongoDB* as a connection type and click **Next**.
5. Enter the following properties of the connection in the **Properties** dialog box:

   a. Enter the host name of MongoDB in the **Host** field.
   b. Enter the server port number of the computer where MongoDB is running in the **Port** field.
   c. Enter the user name and password to access MongoDB.

6. Click **Test** to verify whether the connection is established successfully.

   **Note:** If the connection fails to establish, it can be because of any one of the following reasons:

   - Unreachable database.
   - Missing database.
   - Incorrect host name and port number.

7. Click **Fetch** to upload all the databases available in the configured MongoDB in the **Database** drop-down list, and then select the database that you want to use to write the generated test data.
8. Click **Next**, and then provide a name to the established MongoDB connection.
9. Click **OK**.

   You can see the name of the MongoDB connection under the **Connections** tab.

**Results**

You have successfully established the connection between HCL® OneTest™ Data and MongoDB.

**What to do next**

You must associate a schema with a database of MongoDB. See Associating a schema with MongoDB on page 161.

## Associating a schema with MongoDB

After you establish the connection between HCL® OneTest™ Data and MongoDB, you must associate a schema with a database of MongoDB. You must select the schema that you want to use to generate the test data.

**Before you begin**

You must have created a collection in the selected MongoDB database.

For more information, refer to db.createCollection() section of Database Methods.

**About this task**

You can associate any schema with the MongoDB database that can generate the test data in the JSON format.

The following schema formats support the output format in JSON:

- Excel
- JDBC
- XSD
- CSV

1. Log in to HCL OneTest™ Server and open the team space that contains your project.
2. Open your project.

   **Result**

   The **Overview** page is displayed.
3. Click **Author > Data Fabrication**.

   **Result**

   The **Data Fabrication** page is displayed.
4. Click the **Connections** tab and select the MongoDB connection name that you created.
5. Click the menu and select **New Action**.
6. Select **Target** as the **Action Type** in the **New Action** dialog box, and then click **Next**.
7. Click **Fetch** to extract a list of all the collections of the selected database, and then select a collection from the **Collection** drop-down list.

   **Note:**

   You can retain the default property settings for all other boxes.
8. Click **Next** to provide the schema details.
9. Click the **Schema** drop-down list to select the schema that you want to use to generate the test data.
10. Click the **Schema Type** drop-down list to select the group type for which you want to generate the test data, and then click **Next**.
11. Provide the identification details of the new action, and then click **OK**.

    The new action is created and listed under the MongoDB connection.

#### Results

You have successfully associated the schema with the database.

**Tip:** If you want to associate any other schema in the same database, then you must click **New Action** on the existing MongoDB connection. You need not create a new connection.

#### What to do next

You can generate the test data from the schema that you associated with the database of MongoDB. The generated test data is directly inserted into the database. See Inserting the generated test data in a database on page 162.

## Inserting the generated test data in a database

After you associate a schema with a database of MongoDB, you can generate the test data and insert the generated test data directly in the database by using HCL® OneTest™ Data.

#### Before you begin

You must have associated the schema with MongoDB. See Associating a schema with MongoDB on page 161.

1. Select your project in Rational® Test Automation Server and go to the **Data Fabrication** page.
2. Select the schema that you associated with the database from the **Schemas** tab.
3. Select the group type of the schema for which you want to generate the test data.
4. Click the menu of the selected group type, and then click **Generate Data**.

   You can view the list of all item types associated with the selected group type in the structure view.

   **Result**

   The **Generate Data** dialog box is displayed.
5. Select the **Connection** option.
6. Select the **MongoDB** connection name that you established from the **Connections** drop-down list.

   **Note:**

   When you select MongoDB to insert the generated test data, you can generate only one record of test data at a time.
7. Provide a value for **Numeric Seed Value**, and then click **OK**.

**Results**

You have successfully generated and inserted the test data in MongoDB.

**What to do next**

You can click the **Jobs** tab to view the status of the job. When you hover over the status of the job, you can view the following details in the tooltip:

- Connection URL that includes details of host name and port number.
- Name of the collection.
- Database name with which the schema is associated.

**Important:**

You cannot download the generated test data from the **Jobs** page because the test data is written directly in the MongoDB database and the generated test data is not available in HCL® OneTest™ Data.

You must log in to the database to view the generated test data.

## HCL OneTest Data support to generate test data for an Excel file

When you use Microsoft Excel as a database in your application testing environment and you want to use the structure of an Excel file to generate test data, then you can generate the test data for the selected Excel file by using HCL® OneTest™ Data.

When you want to generate the test data for the Excel file, then you must import the file into your HCL® OneTest™ Data project. After importing the file, you must establish a connection between Excel and HCL® OneTest™ Data. The established connection helps to transfer the data and interpret the data of the imported Excel file into your HCL® OneTest™ Data project. The interpreted information then helps HCL® OneTest™ Data to generate a schema. You can then use the generated schema to generate the test data in HCL® OneTest™ Data.

**Prerequisites**

The following prerequisites must be met to enable HCL® OneTest™ Data to write the generated test data:

- You must have access to HCL OneTest™ Server.
- You must have an Excel file on your computer.

**Task flow**

You must perform the following tasks in sequence to enable HCL® OneTest™ Data to generate the test data for the Excel file. The table also provides you the links to the information about the tasks.

| Task | More Information |
|---|---|
| Import an Excel file. | Importing an Excel file. on page 164 |
| Establish a connection between Excel and HCL® OneTest™ Data. | Establishing a connection between Excel and HCL OneTest Data on page 165. |
| Generate a schema based on an Excel file. | Generating a schema based on an Excel file on page 165. |
| Generate test data by using the generated schema. | Generating test data by using the generated schema on page 166. |

## Importing an Excel file

When you want to establish a connection between HCL® OneTest™ Data and an Excel file, you must import the Excel file into your HCL® OneTest™ Data project from your computer.

**Before you begin**

You must have completed the following tasks:

- Created a project in HCL OneTest™ Server.
- Opened an Excel file or created a file if it did not exist.

1. Select your project in HCL OneTest™ Server and navigate to the **Data Fabrication** page.
2. Click **Files**, and then click **New** to import the Excel file from the computer into your project.

**Results**

You have successfully imported the Excel file from the computer into your project.

**What to do next**

You must establish a connection between Excel and HCL® OneTest™ Data. See Establishing a connection between HCL OneTest Data and Excel on page 165.

## Establishing a connection between HCL OneTest Data and Excel

When you want to generate a schema based on an Excel file, you must establish a connection between HCL® OneTest™ Data and Excel.

**Before you begin**

You must have imported an Excel file from your computer into your project. See Importing an Excel file on page 164.

1. Select your project in Rational® Test Automation Server and navigate to the **Data Fabrication** page.
2. Click **Connections**.
3. Click **New** to create a new connection.

   The **New Connection** dialog box is displayed.
4. Select **Excel** as a connection type, and then click **Next**.
5. Provide a name for this connection.
6. Click **OK**.

   You can see the name of the connection under the **Connections** tab.

**Results**

You have successfully established the connection between HCL® OneTest™ Data and Excel.

**What to do next**

You must generate a schema in HCL® OneTest™ Data based on the Excel file. See Generating a schema based on Excel file on page 165.

## Generating a schema based on an Excel file

After you establish the connection between HCL® OneTest™ Data and Excel, you can generate a schema that is based on an Excel file in HCL® OneTest™ Data.

1. Select your project in Rational® Test Automation Server and navigate to the **Data Fabrication** page.
2. Click the **Connections** tab and select the connection that you created.
3. Click the menu, and then select **New Action**.
4. Select **Target** as the **Action Type** in the **New Action** dialog box, and then click **Next**.

5. Enter the following properties of the schema that you want to generate, and then click **Next**:
   - Select the Excel file that you imported into your project from the **Excel File** drop-down list.
   - Click **Fetch** to upload all the worksheets of the Excel file in the **Worksheet** drop-down list.
   - Select the worksheet from the **Worksheet** drop-down list.

   **Note:**

   You can retain the default property settings for all other fields.

6. Select **New**, and then click **Generate**.

   The schema is generated in HCL® OneTest™ Data. HCL® OneTest™ Data generates the schema name as *schema_<number>*.

   **Note:** You must select this schema name as the schema for HCL® OneTest™ Data to generate the test data.

7. Select the schema type as **Sheet**.
8. Click **Next** to provide the identification details of the new action, and then click **OK**.

   The new action is created and listed under the connection that is established between HCL® OneTest™ Data and Excel.

### Results

You have successfully generated the schema in HCL® OneTest™ Data.

**Note:**

You can modify the schema name *schema_<number>* from the **Schemas** tab.

**Tip:** If you want to generate schemas for other Excel files, then you must click **New Action** on the existing Excel connection. You need not create a new connection.

### What to do next

You can view the generated schema listed under the **Schemas** tab. You can generate the test data for the generated schema in HCL® OneTest™ Data. See Generating test data by using the generated schema on page 166.

## Generating test data by using the generated schema

After you generate the schema in HCL® OneTest™ Data, you can use this schema to generate the test data in HCL® OneTest™ Data.

#### Before you begin

You must have generated the schema by using the connection that is established between HCL® OneTest™ Data and Excel. See Generating a schema based on Excel file on page 165.

1. Select your project in Rational® Test Automation Server and navigate to the **Data Fabrication** page.
2. Select the schema name *schema_<number>* from the **Schemas** tab.

   **Result**

   The schema is displayed.
3. Select the **Sheet** group type, and then select **Structure** from the menu.

   You can view a group as **Row** in the structure of the schema. All item types of a schema are listed under the **Row** group type.
4. Click the menu for the **Row** group type, and then set the property as **Set Required**.
5. Click the menu for all the item types that you want to use to generate the test data, and then set the property as **Set Required**.

   **Note:** If you have a numeric item type, you must set **Min size (digits)** and **Max size (digits)** properties.

   For example, when you select `Integer` as **Presentation**, then you can set the **Min size (digits)** as `1` and the **Max size (digits)** as `9`.
6. Click **Save**.
7. Select the **Sheet** group type in the **Dictionary** dialog box, and then click **Generate Data** from the menu.

   The **Generate Data** dialog box is displayed.
8. Provide the number of records that you want to generate in **Number of records**.
9. **Optional:** Provide a seed value that acts as a reference number to generate the same set of data in **Numeric Seed Value**.
10. Select the output file format for the generated test data from the list, and then click **OK**.

#### Results

You have successfully generated the test data by using the generated schema.

#### What to do next

You can click the **Jobs** tab to view the status of the job. You can then download the generated test data of the jobs that are completed successfully.

## Generating test data

When you have a defined and structured schema for the entire data, you can generate test data to perform application testing.

#### Before you begin

The following prerequisites must be met to generate the test data:

- You must have created a project.
- The schema must have a group type.
- You must have a structured view of your defined schema.

**About this task**

You can generate the sample test data from the group type of schema. When you import the XSD format schema, then HCL® OneTest™ Data generates only XML test data. If you import JSON format as input, the application generates only JSON test data.

1. Select your project in Rational® Test Automation Server and navigate to the **Data Fabrication** page.
2. Click the **Schemas** tab and select the schema for which you want to generate the test data.
3. Select the group type that you want to use to generate the test data from the **Dictionary** dialog box.
4. Click **Generate Data** from the menu.

   **Result**

   The **Generate Data** dialog box is displayed.
5. Enter the number of records that you want to generate in the **Number of records** field.

   **Important:** When you select the pairwise data generation property for any item type, then the number of records you enter is ignored. Based on the pairwise data generation approach, HCL® OneTest™ Data autocalculates the number of records.

6. Enter seed value in the **Numeric Seed Value** field.

   The seed value acts as an instance of random data that can produce the same set of data multiple times. By default, the seed value is blank.

   **Note:** The following table lists the maximum values for numeric seed value and number of records:

   | Description | Maximum values |
   |---|---|
   | Numeric Seed Value | 8 digits (99,999,999) |
   | Number of records | 9 digits (999,999,999) |

7. Choose the output file format of the test data that you want to generate from the **Output File Format** list. HCL® OneTest™ Data supports the following formats:
   - CSV
   - Native
   - JSON

    - XML
    - Excel

    **Notes:**

    - If you create a schema by using the schema designer, by default, the output file format is CSV.
    - If you import a schema into your project or have a schema with nested groups, you can select only the Plain text file as the output file format.
    - You can generate line numbers in the output by creating an auto-increment rule by using functions and expressions.

8. Click **OK**.

**Results**

The test data is generated successfully.

**Note:** The column headings for the generated test data are available only for the CSV and Excel file output formats.

**What to do next**

You can view and download the generated test data on the **Jobs** page. See Status of generated test data on page 174.

## Generating test data by using multiple schemas

When you have multiple schemas generated by external resources in HCL® OneTest™ Data, then you can generate the test data by using multiple schemas.

**Before you begin**

You must have completed the following tasks:

- Logged into Rational® Test Automation Server.
- Installed any JDBC supported database, such as Oracle, MySQL, and so on.

**About this task**

To generate the test data for multiple schemas, you must create tables in a database that are supported by the JDBC connection. After establishing the JDBC connection between the database and HCL® OneTest™ Data, you must generate multiple schemas in HCL® OneTest™ Data. You can then select multiple schemas for which you want to generate the test data and view the structure of all the selected schemas with the representation of their relationships with each other.

The output formats that are supported in HCL® OneTest™ Data for the generation of test data for multiple schemas by using referential integrity are CSV, Excel, and Native. You can also insert the generated test data into the database by using the JDBC connection.

You must perform the following tasks in sequence to generate the test data in HCL® OneTest™ Data for multiple schemas:

1. Create a sample schema in a database on page 147.
2. Establish a JDBC connection with HCL OneTest Data on page 149.
3. Generate a schema in HCL OneTest Data on page 150.
4. Define properties of schemas on page 171.
5. Select multiple schemas on page 170.
6. Generate test data for selected schemas on page 169.

**Results**

The test data is generated successfully by using multiple schemas simultaneously in HCL® OneTest™ Data.

## Selecting multiple schemas

After you define the properties of schemas, you must select schemas and view the relationships among them to generate the test data for multiple schemas simultaneously.

**Before you begin**

You must have completed the following tasks:

1. Created a project.
2. Generated schemas in HCL® OneTest™ Data by using a JDBC connection. See Generating a schema in HCL OneTest Data on page 150.
3. Defined properties for the generated schemas. See Defining properties of schemas on page 171.

1. Select your project in Rational® Test Automation Server and navigate to the **Data Fabrication** page.
2. Click **Schema Integrity** to select schemas.

   **Result**

   The **Select Schemas** dialog box is displayed.
3. Click the field to view the list of schemas that are generated by using the JDBC connection.
4. Select multiple schemas.

   **Note:** You can select a maximum of 10 schemas.

5. Click **Import** to import the selected schemas into the **Schema Integrity** page.

   You can find the structural view of all the selected schemas with their relationships with each other.

   **Note:** You can use the scroll wheel or pinch-to-zoom option on the **Schema Integrity** page to view the structural view of all schemas and relationships.

6. Verify the relationships among the selected schemas.

7. Select schemas based on the following criteria:
   - If a schema does not have a relationship with any other schema on the **Schema Integrity** page, then you must select that schema manually. Otherwise, click **Select All** to select all schemas to generate the test data.
   - If a schema has a relationship with other schemas and is selected, then all the related schemas get selected automatically.

   **Notes:**
   - You cannot select schemas that belong to different JDBC connections.
   - You can use **Modify canvas** to add or remove the selected schemas.
   - If any of the selected schema is deleted from the workspace, then the view of the **Schema Integrity** page changes as the deleted schema is removed from the **Schema Integrity** page.

### Results

You have successfully selected multiple schemas for the generation of test data.

### What to do next

You can generate the test data for the selected schemas. See Generating test data for selected schemas on page 172.

## Defining properties of schemas

After you generate the schema from the configured database in HCL® OneTest™ Data, you must define the required properties for all schemas.

### Before you begin

You must have completed the following tasks:

1. Created a project.
2. Generated schemas in HCL® OneTest™ Data by using a JDBC connection. See Generating a schema in HCL OneTest Data on page 150.

1. Log in to HCL OneTest™ Server and open the team space that contains your project.
2. Open your project.

   **Result**

   The **Overview** page is displayed.
3. Click **Author > Data Fabrication**.

   **Result**

   The **Data Fabrication** page is displayed.
4. Select the generated schema from the **Schemas** tab.

   For example, *schema_<number>*

5. Click the **Menu** icon ⋮ of the group type of the schema and select **Structure**.

   You can view the list of all item types associated with the selected group type in the structural view of the schema.

6. Click the **Menu** icon ⋮ of all item types in the **Structure** dialog box to set the following properties:

   a. Click **Set Required** to get at least one occurrence of the item type in the generated test data, and then click **Save**.

   b. Select item types with the values of **Item Subclass** as `Number` and **Presentation** as `Integer`.

   c. Click **Properties** for the selected item types and perform the following steps for **Item Properties**:

      i. Set the value of **Min size (digits)** and **Max size (digits)** between 1-9, where the value for **Min size (digits)** should be less than or equal to **Max size (digits)**.

         For example, you can set the **Min size (digits)** as `1` and the **Max size (digits)** as `9`.

   **Note:** The values of both the **Min size (digits)** and **Max size (digits)** properties for item types that act as a primary key and foreign key must be the same.

7. Save the changes.
   You must repeat steps 2 through step 5 to set the properties for all schemas.

### Results

You have defined the properties for all schemas.

### What to do next

You can click **Schema Integrity** to select the defined and structured schemas that you want to use to generate the test data. See Selecting multiple schemas on page 170.

## Generating test data for selected schemas

After you select multiple schemas on the **Schema Integrity** page, you can generate the test data of all the selected schemas simultaneously.

### Before you begin

You must have completed the following tasks:

1. Created a project.
2. Generated schemas in HCL® OneTest™ Data by using a JDBC connection. See Generating a schema in HCL® OneTest™ Data on page 150.

3. Defined properties for the generated schemas. See Defining properties of schemas on page 171.
4. Selected all schemas that are generated by using the same JDBC connection. See Selecting multiple schemas on page 170.

1. Select your project in Rational® Test Automation Server and navigate to the **Data Fabrication** page.
2. Click **Generate Data** to generate the test data for the selected schemas on the **Schema Integrity** page.

   **Result**

   The **Generate Data** dialog box is displayed.
3. Select the output location from the following options:

   **Choose from:**

   - **File**: You must select the **File** option to generate the output in CSV, Excel, or Native format.
   - **Connections**: You must select the **Connections** option to insert the generated test data into the database by using the JDBC connection.
4. Enter the number of records that you want to generate in the **Number of records** field.
5. Enter a seed value in the **Numeric Seed Value** field.

   The seed value acts as an instance of random data that can generate the same set of data multiple times. The default seed value is blank.

   **Note:** The following table lists the maximum values for numeric seed value and number of records:

   | Description | Maximum values |
   |---|---|
   | Numeric seed value | 8 digits (99,999,999) |
   | Number of records | 9 digits (999,999,999) |

6. Choose the output file format of the test data that you want to generate from the **Output File Format** list.
7. Click **Analyze** to validate the selected schemas.

   If the selected schemas are validated successfully, then you can view the order of schemas execution for generating the test data. All the parent schemas are executed first followed by the child schemas.

   **Note:** If the validation fails, it might be because of one of the following reasons:

   - The parent schemas of all the selected schemas are missing.
   - The connection and actions of selected schemas do not exist in the project.
   - Any one of the following properties of all item types of the selected schemas are not set:
     - **Min size (digits)** and **Max size (digits)** properties for each integer item type.
     - **Set Required** property.

You must resolve the cause of failure for the successful validation of the schemas.

8. Click **OK**.

**Results**

You have generated the test data by using multiple schemas simultaneously.

HCL® OneTest™ Data first generates the test data of the parent schema followed by the child schemas.

**What to do next**

You can view and download the generated test data of all the selected jobs on the **Jobs** page. See Status of generated test data on page 174.

## Status of generated test data

You can view the status of all the test data generated jobs performed in your project, on the **Jobs** page.

You can view the status of test data generated jobs in the **Status** column on the **Jobs** page. When the test data generation of any schema initiates, the status of that job is shown as *Started*. As the test data generation progresses, the status of that job changes to *In Progress*. After the test data generation completes successfully, the status changes to *Complete*. If the generation of test data fails, you can view the status of that particular job as *Failed*.

## Usage of test data generation job

The generation of test data for a single or multiple schemas in HCL® OneTest™ Data is considered as a job. You can view the list of all jobs on the **Jobs** page.

When the job to generate the test data is complete, you can download the test data generated job from the **Jobs** page. You can use the job details to troubleshoot the failed jobs and to regenerate the successful jobs.

You can access API history for each of the test data generation jobs. If you want to reuse any schema, you can regenerate the test data.

**Note:** You cannot regenerate the test data for the jobs that are generated for multiple schemas.

You can find different methods of using the generated test data jobs.

## Downloading test data

When you want to test any application by using the generated test data or to reuse the data of any specific schema definition for some other project, you can download the generated test data into your local file system.

**Before you begin**

You must have completed the following tasks:

- Created a project.
- Generated test data.

1. Select your project in Rational® Test Automation Server and go to the **Data Fabrication** page.
2. Click **Jobs**.

   **Result**

   You can view the list of all test data generated jobs in the **Jobs** page.

   **Note:** If you want to view only the jobs related to referential integrity, then select **Schema integrity jobs** from the menu.
3. Select a job for which you want to download the generated test data.

   **Notes:**

   - You can select multiple jobs at a time to download the test data.
   - You can download the test data only for the jobs that are completed successfully.
   - The download option is disabled for the jobs that are set to insert the generated test data into the database.
4. Click the **Download job(s)** button or ⤓ icon.

**Results**

The generated test data is downloaded in the zip format in your local file system. When you download multiple jobs, then the zip format of the jobs are organized by time.

**Note:** You can also download the generated test data file from the following location of the HCL® OneTest™ Data pod:

`/opt/hcl/hip-rest/output/<accountId>/<userID>/<projectId>/<schemaId>/<genMapPath>`.

## Regenerating test data

When you want to generate another set of test data by using an existing schema, then instead of defining a schema again, you can regenerate test data from the **Jobs** page.

### Before you begin

- You must have a project.
- You must have generated test data.

### About this task

On the **Jobs** page, you can view the list of all test data generated jobs. If you want to use the existing data model or schema to generate different values of test data, you must regenerate with an empty seed value. However, if the seed value is not empty, then the same test data is generated as it was generated earlier for that schema.

**Note:** You cannot regenerate the test data for the jobs that are generated for multiple schemas by using referential integrity.

1. Select your project in Rational® Test Automation Server and go to the **Data Fabrication** page.
2. Click **Jobs**.

   **Result**

   You can view the list of data generated jobs.
3. Select a job for which you want to regenerate the test data.

   You can select multiple jobs at a time to regenerate the test data.
4. Click **Regenerate data** from the menu or click the **Regenerate data** icon .

**Results**

The test data for the required schema is regenerated and is listed on the **Jobs** page.

**What to do next**

You can download the regenerated test data. See, Downloading test data on page 174.

## Configuring job settings

HCL® OneTest™ Data periodically deletes all the generated test data and the jobs of test data generation with API history to clear the memory. However, if you want to change the default number of days to retain the generated test data jobs in memory, you must configure the job settings.

**Before you begin**

- You must have cluster-admin permissions.
- You must have the IP address of the computer where HCL OneTest™ Server is installed.

**About this task**

As a default configuration, HCL® OneTest™ Data retains the jobs with their generated test data and API history for five days after the jobs are completed. All the jobs that are created before five days are deleted. If you want to retain the jobs with their generated data for more number of days in HCL® OneTest™ Data, then you must modify the value of the **data_rotation_time** property to the number of days you want to retain the jobs in the `configmap` file.

**Note:** If you modify the value of the **data_rotation_time** property to zero, then you can disable the deletion of generated test data jobs and API history.

1. Log in to the SSH console of HCL OneTest Server.
2. Run the following command to edit the configuration file:

   ```
   kubectl edit configmap -n test-system {my-ots}-data-config -o yaml
   ```

3. Search for the **data_rotation_time** property in the configuration file, and then set the value to the number of days you want the jobs to retain in the server.

   **Note:** The default value of **data_rotation_time** is *5*.

   For example, if you want to retain the jobs for 10 days, set the value of **data_rotation_time** property as *10*.

4. Save your changes, and then exit from the configuration file.
5. Run the following command to delete and restart the *<onetest data>* pod:

   ```
   kubectl delete pod {my-ots}-data-app-0
   ```

**Results**

You have successfully configured the setting to retain the jobs with generated test data and API history in HCL® OneTest™ Data.

## Jobs management

When you run a job to generate the test data, you can view the job details on the **Jobs** page. The **Jobs** page helps you to view and to manage the list of all the jobs that you execute.

You can perform a quick search for the test data generated jobs based on the name of the schema or the date range by selecting the *From* and *To* dates. If the *To* field is blank, you can view the results till the most recent date. After you perform the search, if you want to continue to search for any schema or specific date, you must clear the **Search** field by using the **Clear Filter** button.

You can delete and cancel the generated test data jobs.

## Deleting a test data generated job

When any of the generated test data is of no further usage, you can delete that specific test data from your project.

**Before you begin**

- You must have a project.
- You must have generated test data.

**About this task**

On the **Jobs** page, you can view the list of all the projects for which you have generated the test data. You can remove the reference of any specific test data.

1. Select your project in Rational® Test Automation Server and go to the **Data Fabrication** page.
2. Click **Jobs**.

   **Result**

   You can view the list of data generated jobs.
3. Select a job for which you want to delete the test data.

   You can select multiple jobs at a time to delete.
4. Click **Delete Job** from the menu or click the **Delete** icon 🗑.

**Results**

The selected test data is deleted from your project.

## Canceling a test data generation job

After initiating a test data generation job, if you want to modify the schema or terminate the job before the job completes, then you can cancel the running test data generation job.

**Before you begin**

You must have a test data generation job in the `In Progress` status.

**About this task**

You can cancel only one test data generation job at a time. You can view the details of API endpoints of the canceled job on the **API History** page. Later, if you want to reuse the same schema to generate the test data, you must select **Regenerate Job** to regenerate the test data. However, you cannot download the test data of the canceled job.

**Note:** You can hover the cursor over the job to view the details of the member who canceled the job along with the timestamp.

1. Log in to HCL OneTest™ Server and open the team space that contains your project.
2. Open your project.

   **Result**

   The **Overview** page is displayed.
3. Click **Author > Data Fabrication**.

   **Result**

   The **Data Fabrication** page is displayed.
4. Go to the **Jobs** page and select the test data generation job that you want to cancel.
5. Click **Cancel Job** from the menu of the selected job.

   A notification is displayed that the selected job is canceled.

**Results**

You successfully canceled the running job. The canceled job appears on the **Jobs** page with the status as `Canceled`.

**What to do next**

You can modify the schema by using the schema designer and generate the test data.

# Chapter 6. Test Execution Specialist Guide

This guide describes tasks that you must complete before you can configure and run tests in HCL OneTest™ Server. You can find information about configuring runs for the different test types that are supported. You can also find information about other tasks that you can perform on the Resource Monitoring Service, Virtualization, and Integrations with third-party applications. This guide is intended for testers or test execution specialists.

## System modeling

In this topic, you can find information about using the system model feature. You can also find the tasks that you can perform to create a system model. You must configure a repository to save the system model that you created for the application under test.

### Introduction to system model

A system model is a logical presentation of the components contained in a system under test and the relationships between components. You can create a system model when you want to visualize the different components and their relationships as a diagrammatic representation.

You can create only one system model within a team space on HCL OneTest™ Server. A team space repository is required to contain the system model that you create. Before you can create a system model, you must add a team space repository. The system model that is created or edited is stored in the team space repository. All projects within a team space share the same system model. All members of a team space can create, edit, and view the system model.

After you log in to HCL OneTest™ Server, you can go to the **System Model** page to create a system model for your team space. Alternatively, you can go to any of the projects in your team space and create a system model from the **System Model** page within a project. The system model is accessible from all the projects of your team space repository.

### Components in a system model

Components are the basic building blocks which are used to represent a piece of software in an application.

You can model a system by using components. You can create components that represent the type of asset or resource used by the system or application under test and can be of the following types:

| Component type | Description |
|---|---|
| Database | The component type represents the database resource. You can select this component type when you want to represent a database asset or resource in the system model. |
| Service | The component type represents the virtual service resource. You can select this component type in the system model to represent a virtual service in the team space repository or associate a virtual service resource that is in the project repository. |

| Component type | Description |
|---|---|
| UI | The component type represents the User Interface (UI) resource. You can select this component type when you want to represent the User Interface resource in the system model. |

You can create multiple components to depict the different resources in the system or application under test. You can also associate virtual service resources with the components. The components in the system model can then be associated with the virtual services stored in the repositories that are configured for each of the projects.

You can create the following types of relationships between the components:

- Components can be related as *child* components to *parent* components.
- Components can be related as *dependent* components. The dependency is displayed by a line-arrow with the arrow-head pointing to the dependent component. For example, if *Component A* depends on *Component B*, then the relationship is displayed with a line-arrow from *Component A* to *Component B* and the arrow points to *Component B*. If Component B is also dependent on *Component A*, then the relationship is displayed with a line-arrow from *Component B* to *Component A* and the arrow points to *Component A*.

## Using the system model UI

The following table lists the icons used in a system model and the icons that are displayed on the **System Model** page:

| Icon | Description |
|---|---|
| ⛬ | Represents a system model. |
| ☰ | Represents Suites and Tests in a repository. |
| ⛁ | Represents a database resource. The component name is prefixed with this icon when you select the component type as database. |
| ⚙ | Represents a service resource. The component name is prefixed with this icon when you select the component type as service. |
| ▭ | Represents a User Interface (UI) resource. The component name is prefixed with this icon when you select the component type as User Interface. |
| ⊕ | Used to add components, child components, and dependent components. |
| 🔍 | Used to search for components in the system model. |
| 🖧 | Represents the virtual services. |
| ☁ | Used to publish the system model to the team space repository. |
| - - - -▶ | Represents the dependency between the components. |

| Icon | Description |
|---|---|
| —— | Represents the inherited dependencies. |

You can perform the following operations when you view the model displayed on the **System Model** page:

| Operation | Action |
|---|---|
| Pan the view of the model | 1. Click the **Pan** icon ✋.<br>2. Click on the area of the model that you want to pan and move the area keeping the mouse pressed.<br>**Note:** You must click the **Pan** icon to exit from the pan mode. |
| Zoom-in the view of the model | 1. Click the **Zoom-in** icon 🔍.<br>The view is enlarged or zoomed-in.<br>**Note:** You can repeatedly click the Zoom-in icon to further enlarge the view. |
| Zoom-out the view of the model | 1. Click the **Zoom-out** icon 🔍.<br>The view is diminished or zoomed-out.<br>**Note:** You can repeatedly click the **Zoom-out** icon to further diminish the view. |
| Reset the view | 1. Click the **Menu** icon ⋮ on the **System Model** page.<br>2. Select **Reset Zoom**.<br>The view is reset to the default view without any zoom applied. If the view was zoomed-in or zoomed-out, the view is reset to 100% of the size. |

| Operation | Action |
|---|---|
| Position the model view | 1. Click the **Menu** icon ⋮ on the **System Model** page.<br>2. Select from the following options:<br>◦ **Center**: Select this option to align the view of the model to the center of the page.<br>◦ **Fit to View**: Select this option to fit the view of the model in the window on the **System Model** page. |

### Tasks in system modeling

When you want to model the system under test for the first time after you install the server software and you are a licensed user in the team space, you must first add the team space repository. See Adding a repository to a team space on page 184.

After a repository is configured as the team space repository, you can work with the system model. See Tasks for working with a system model on page 183.

When you want to change, update or delete the repository, you can work with the team space repository. See Tasks for working with the team space repository on page 184.

## Task flows for working with a system model

You can find information about the tasks that you can perform when you want to create a system model, create components, associate the components, create linkages among the components, and then publish the system model to the team space repository. Also, you can find information about the tasks that you must perform when you want to modify an existing system model.

### Tasks for working with a system model

You can find the tasks when you want to create a system model for the application under test in your team space.

| Tasks | More information |
|---|---|
| Add a repository to the team space. | Adding a repository to a team space on page 184 |
| Create a system model. | Creating a system model on page 186 |
| Create components in a system model. | Creating components on page 188 |
| Create child components to existing components in a system model. | Creating child components on page 189 |

| Tasks | More information |
|---|---|
| Define the relationship between the components in a system model. | Creating linkages between components on page 191 |
| Delete the relationship between the components in a system model. | Deleting linkages between components on page 193 |
| Modify components in a system model. | Modifying components on page 194 |
| Delete components in a system model. | Deleting components on page 196 |
| Associate resources with components in a system model. | Associating resources with components on page 198 |
| View resources associated with components in a system model. | Viewing resources associated with components on page 199 |
| Remove resources that are associated with components in a system model. | Removing resources associated with components on page 203 |
| Publish changes made to the system model. | Publishing changes to the system model on page 197 |
| Publish changes of associating resources in a project with components in the system model. | Publishing changes of associating resources with components on page 204 |
| View system model. | Viewing the system model on page 205 |
| Delete system model. | Deleting a system model on page 205 |

### Tasks for working with the team space repository

You can find the tasks that you can perform when you want to work with the team space repository.

| Tasks | More information |
|---|---|
| Add a repository to the team space. | Adding a repository to a team space on page 184 |
| Delete a team space repository. | Deleting a repository that is added to a team space on page 206 |

## Adding a repository to a team space

You must add a repository to the team space when you want to create, modify, or store a system model.

**Before you begin**

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Team Space Owner* or an *Architect* in the team space. See Managing members and their roles in a team space on page 493.
- Been provided valid credentials to access the Git repository that you want to add.

**About this task**

As a *Team Space Owner* or an *Architect*, you can add a repository to the team space. You can add a repository to the team space in any of the following methods:

- Add an existing repository that can contain test assets, which are created in the desktop products.
- Add a bare repository.

**Attention:** The repository that you want to add must not contain any projects or `.project` files under the root directory.

When you add a repository, the Git repository is cloned to the team space. While adding a repository, you must provide the necessary authentication credentials that are set for the Git repository. For example, if the authentication type is SSH, then you must provide the Git URL, a deploy key, and a passphrase.

**Important:** You can add only one repository to the team space. After you add a repository to the team space, the option to add another repository is not available. You must delete an existing repository before you can add a different repository.

1. Log in to HCL OneTest™ Server.
2. Open the team space to which you want a team space repository.

   The **Active projects** page is displayed.
3. Click the **Settings** icon ⚙ from the left navigation pane, and then select **Manage team space**.

   The **Details** page is displayed.
4. Click **Repository** on the left navigation pane.

   The **Repository** page is displayed.
5. Click **Add repository**.

   The **Add repository** dialog is displayed.
6. Enter the URL of the Git repository, which you want to add to the team space, in the **Git Repository** field.

   The required fields are displayed based on the type of Git URL that you entered.

7. Enter the required credentials based on any of the following authentication methods configured in the repository.

   To gain access to the repository, you must use any one of the authentication methods:

   | Authentication method | Credentials required |
   |---|---|
   | SSH | ◦ Deploy key<br>◦ Passphrase |
   | HTTPS | ◦ User name<br>◦ Password |
   | HTTP | ◦ User name<br>◦ Password |

   **Notes:**

   ◦ You must have defined the authentication type and set the authentication credentials in the Git repository.

   ◦ If you use SSH to connect to your remote repository and HCL OneTest™ Server displays an `Auth Fail` exception while using the deliver changes option, you can resolve this exception error by regenerating your SSH keys by using the -m PEM option.

8. Click **Add**.

   The Git repository is added to the team space on HCL OneTest™ Server.

   **Note:** Depending on the size of the repository you are cloning, it can take a few to several minutes to clone the repository.

**What to do next**

You can perform the following actions on the repository that you added:

- Add a system model to the repository. See Creating a system model on page 186.
- Delete a repository that is added to the team space, if it is no longer required.

Related information

Managing repositories on page 508

## Creating a system model

Before you can create components in a system model, you must create an empty system model and publish the system model to the team space repository.

**Before you begin**

You must have completed the following tasks:

- Ensured that you are assigned a role as an *Architect* in the team space, a *Project Owner* or a *Tester* in the project. See Managing members and their roles in a team space on page 493 or Managing access to server projects on page 504.
- Added a repository to the team space. See Adding a repository to a team space on page 184.

1. Log in to HCL OneTest™ Server.
2. Open the team space to which a team space repository is added.

   The **Active projects** page is displayed.
3. Select an option to create a system model based on your role in the team space or in a project in the team space:
   - If you are a *Team space owner* or an *Architect*, go to Step 4 on page 187.
   - If you are not a *Team space owner* or an *Architect* but a project *Owner* or *Tester*, go to Step 5 on page 187.
4. Perform the following steps, if you are a *Team space owner* or an *Architect*:

   a. Click **System Model** on the left navigation pane.

      The **System Model** page is displayed.

   b. Click the **Add system model** icon ⊕.

      An empty system model is created.

   c. Go to Step 6 on page 187.
5. Perform the following steps, if you are not a *Team space owner* or an *Architect* but a *Project owner*:

   a. Open your project that is displayed under **My projects** on the **Active projects** page.

      The **Overview** page is displayed.

   b. Click **System Model** on the left navigation pane.

      The **System Model** page is displayed.

   c. Click the **Add system model** icon ⊕.

      An empty system model is created.

   d. Go to Step 6 on page 187.
6. Click the **See unpublished commits** icon ⤓, and then select the **Publish changes** option.

**Note:** If you do not want to commit the changes that you made to the system model, you can select the **Discard changes** option.

7. Enter a commit message in the **Description of change** field, and then click **Publish**.

   The empty system model is committed and published to the team space repository. When the system model is published to the team space repository, other members of the team space can work with the same system model from any of their projects in the team space.

### Results

You have created an empty system model and published it to the team space repository.

### What to do next

You can perform any of the following tasks:

- Create components in the system model. See Creating components on page 188.
- Create child components for the components that you added to the system model. See Creating child components on page 189.

## Creating components

The first step in building a system model is to create components. The components serve as the building blocks for the system model.

### Before you begin

You must have completed the following tasks:

- Created and published the system model. See Creating a system model on page 186.

1. Log in to HCL OneTest™ Server.
2. Open the team space to which a team space repository is added.

   The **Active projects** page is displayed.
3. Click **System Model** on the left navigation pane.

   The **System Model** page is displayed.
4. Perform the following steps to add a component to the system model:

   a. Click the **Add component** icon ⊕ in the **Components** row on the right pane.

      The **Add a component** dialog is displayed.

   b. Enter a name for the component in the **Name** field.

c. Select the type of component that you want to add from the following options displayed in the **Type** field.

    - **Service**
    - **Database**
    - **UI**

d. Click **Add**.

    The component is displayed as a circular block on the system model pane with the icon and name of the component below the block. The added component is also displayed under the **Components** list in the right pane.

    You create components of other types or the same type in the system model following the preceding steps.

**Results**
You have created components in the system model.

### What to do next

You can perform any of the following tasks:

- Publish the changes that you made to the system model. See Publishing changes to the system model on page 197.
- Create child components for the components that you added to the system model. See Creating child components on page 189.
- Create linkages between components in the system model. See Creating linkages between components on page 191.

## Creating child components

You can create components as children of existing components in the system model.

### Before you begin

You must have completed the following tasks:

- Created and published the system model. See Creating a system model on page 186.
- Created components in the system model. See Creating components on page 188.

### About this task

You can create multiple components in your system model and multiple child components under any component in the system model. After you create a child for a component, the child is represented as a circular block within the circular component block in the system model.

1. Log in to HCL OneTest™ Server.
2. Open the team space to which a team space repository is added.

   The **Active projects** page is displayed.
3. Click **System Model** on the left navigation pane.

   The **System Model** page is displayed with the existing components.
4. Create child components in any of the following methods:

   - To create child components from the left pane on the **System Model** page, go to Step 5 on page 190.
   - To create child components from the right pane on the **System Model** page, go to Step 6 on page 190.
5. Perform the following steps to create child components from the left pane:

   a. Click the component under which you want to create child components in the left pane.

      The details of the component are displayed in the right pane.

   b. Click the **Add child** icon ⊕ in the **Children** row on the right pane.

      The **Add a child to...** dialog is displayed.

   c. Enter a name for the component in the **Name** field.

   d. Select the type of component that you want to add from the following options displayed in the **Type** field.

      - ⚙ **Service**
      - 🛢 **Database**
      - **UI**

   e. Click **Add**.

      The child component is displayed as a circular block within the circular block of the parent component on the system model pane. The added child component is also displayed under the **Children** list under the component in the right pane.

      You create child components of other types or the same type for the same or other components in the system model following the preceding steps.
6. Perform the following steps to create child components from the right pane of the stem model page:

   a. Click the component name listed under the **Components** in the right pane.

   b. Click the **Add child** icon ⊕ in the **Children** row on the right pane.

      The **Add a child to...** dialog is displayed.

c. Enter a name for the component in the **Name** field.

d. Select the type of component that you want to add from the following options displayed in the **Type** field.

   - **Service**
   - **Database**
   - **UI**

e. Click **Add**.

   The child component is displayed as a circular block within the circular block of the parent component on the system model pane. The added child component is also displayed under the **Children** list under the component in the right pane.

   You create child components of other types or the same type for the same or other components in the system model following the preceding steps.

**Results**

A component is created and it is associated as a child with the selected component and the name of the child is displayed under the **Children** section in the right pane.

**What to do next**

You can perform any of the following tasks:

- Publish the changes that you made to the system model. See Publishing changes to the system model on page 197.
- Create linkages between components in the system model. See Creating linkages between components on page 191.

## Creating linkages between components

You can create linkages between components that display the relationship between them. The linkages can be between components or between child components of the same or different parent components.

**Before you begin**

You must have completed the following tasks:

- Created and published the system model. See Creating a system model on page 186.
- Created components in the system model. See Creating components on page 188.
- Created child components. See Creating child components on page 189.

**About this task**

You can create multiple components in your system model and multiple child components under any component in the system model. You can specify the dependency among the components.

1. Log in to HCL OneTest™ Server.
2. Open the team space to which a team space repository is added.

   The **Active projects** page is displayed.
3. Click **System Model** on the left navigation pane.

   The **System Model** page is displayed.
4. Perform the following steps to create a relationship between the components in the system model:

   a. Click the component that you want.

      The details of the component are displayed in the right pane.

   b. Select from the following types of relationships that you want to create:

| Option | Action |
|---|---|
| Depends on | Perform the following steps, if the component you selected is dependent on another component:<br>i. Click the **Add dependency** icon ⊕ in the row of the **Depends on** in the right pane.<br>The **Add dependency** dialog is displayed.<br>ii. Select the component from the list of components displayed under **Depends on**.<br>iii. Click **Add**.<br>The components linked are listed under **Depends on** in the right pane. The linkage between the first to the second component you select is shown with an arrow that starts from the first and ends in the second component. |
| Is a dependency of | Perform the following steps, if the component you selected has a dependency of another component:<br>i. Click the **Add dependent** icon ⊕ in the row of the **Is a dependency of** in the right pane.<br>The **Add dependent** dialog is displayed.<br>ii. Select the component from the list of components displayed under **Is a dependency of**.<br>iii. Click **Add**. |

| Option | Action |
|---|---|
| | The components linked are listed under **Is a dependency of** in the right pane. The linkage between the first to the second component you select is shown with an arrow that starts from the second and ends in the first component. |

### Results

You have created linkages between components in the system model.

### What to do next

You can publish the changes that you made to the system model. See Publishing changes to the system model on page 197.

## Deleting linkages between components

When you do not want to retain the linkages between components that you created in the system model, you can delete the linkages.

### Before you begin

You must have completed the following tasks:

- Created and published the system model. See Creating a system model on page 186.
- Created components in the system model. See Creating components on page 188.
- Created child components. See Creating child components on page 189.
- Created linkages between components in the system model. See Creating linkages between components on page 191.

1. Log in to HCL OneTest™ Server.
2. Open the team space to which a team space repository is added.

   The **Active projects** page is displayed.
3. Click **System Model** on the left navigation pane.

   The **System Model** page is displayed.
4. Perform the following steps to delete a relationship between the components in the system model:

   a. Click the component that you want.

      The details of the component are displayed in the right pane.

   b. Select from the following types of relationships that you want to delete:

| When... | Action |
|---|---|
| The component depends on another component. | Perform the following steps, if the component you selected is dependent on another component:<br>i. Select the component from the list of components displayed under the **Depends on** section in the right pane.<br>ii. Click the **Delete selected** icon 🗑 inline with **Depends on** section.<br>The **Delete relationship** dialog is displayed with the details of the component and the dependent component.<br>iii. Click **Delete**.<br>The component linkage is removed from the system model pane. |
| The component has a dependency by another component. | Perform the following steps, if the component you selected has a dependency of another component:<br>i. Select the component from the list of components displayed under the **Is a dependency of** section in the right pane.<br>ii. Click the **Delete selected** icon 🗑 inline with **Is a dependency of** section.<br>The **Delete relationship** dialog is displayed with the details of the component and the component dependency.<br>iii. Click **Delete**.<br>The component linkage is removed from the system model pane. |

### Results

You have created linkages between components in the system model.

### What to do next

You can publish the changes that you made to the system model. See Publishing changes to the system model on page 197.

## Modifying components

At any point in time, after you create a component, you can modify and update the details of the components such as the component type, and the relationships with other components.

### Before you begin

You must have completed the following tasks:

- Created and published the system model. See Creating a system model on page 186.
- Created components in the system model. See Creating components on page 188.
- Created child components. See Creating child components on page 189.

1. Log in to HCL OneTest™ Server.
2. Open the team space to which a team space repository is added.

   The **Active projects** page is displayed.
3. Click **System Model** on the left navigation pane.

   The **System Model** page is displayed.
4. Click the component that you want to edit.

   The component details are displayed in the right pane.
5. Perform the actions for the task to modify components as listed in the following table:

| Task | Action |
|---|---|
| Change the component type. | Perform the following steps:<br>a. Identify the component that you want to modify from the components listed under **Components** in the right pane.<br>b. Click the component checkbox.<br>The **Edit selected** icon ✎ is enabled.<br>c. Click the **Edit** icon ✎.<br>The **Edit component** dialog box is displayed.<br>d. Select the component type.<br>e. Click **Modify**<br>The type of the component is modified. |
| Change the name of a component. | Perform the following steps:<br>a. Identify the component that you want to modify from the components listed under **Components** in the right pane.<br>b. Click the component checkbox.<br>The **Edit selected** icon ✎ is enabled.<br>c. Click the **Edit** icon ✎.<br>The **Edit component** dialog box is displayed. |

| Task | Action |
|---|---|
|  | d. Edit the component name in the **Name** field.<br>e. Click **Modify**.<br>The name of the component is modified. |
| Change the parent of a component. | Perform the following steps:<br>a. Identify the component that you want to modify from the components listed under **Components** in the right pane.<br>b. Click the component checkbox.<br>The **Change parent** icon is enabled.<br>c. Click the **Change parent** icon .<br>The **Change parent** dialog box is displayed.<br>d. Select another parent for the selected components that are listed under the **Parent** list.<br>e. Click **Change**.<br>The parent of the component is changed. |

### Results

You have edited components and updated the details successfully.

### What to do next

You can publish the changes that you made to the system model. See Publishing changes to the system model on page 197.

## Deleting components

When you do not require a component, you can delete the component.

### Before you begin

You must have completed the following tasks:

- Created and published the system model. See Creating a system model on page 186.
- Created components in the system model. See Creating components on page 188.

1. Log in to HCL OneTest™ Server.
2. Open the team space to which a team space repository is added.

   The **Active projects** page is displayed.

3. Click **System Model** on the left navigation pane.

   The **System Model** page is displayed.

4. Click the component that you want to delete.

   The component details are displayed in the right pane.

5. Click the **Delete component** icon 🗑 inline with the component name.

   The **Delete component** dialog is displayed.

6. Click **Delete**.

#### Results

You have deleted a component from the system model.

#### What to do next

You can perform any of the following tasks:

- Publish the changes that you made to the system model. See Publishing changes to the system model on page 197.
- Create components in the system model. See Creating components on page 188.

## Publishing changes to the system model

When you create, delete, or modify components in a system model, the changes are saved locally and are not visible to the other project members or team space members. You must publish the system model changes to the team space repository to make your changes visible for all users of the team space.

#### Before you begin

You must have completed the following tasks:

- Created and published the system model. See Creating a system model on page 186.
- Performed actions to add or modify the system model that you want to publish.

1. Log in to HCL OneTest™ Server.
2. Open the team space to which a team space repository is added.

   The **Active projects** page is displayed.

3. Click **System Model** on the left navigation pane.

   The **System Model** page is displayed.

4. Perform any of the following actions:

    - Create a system model.
    - Delete a system model.
    - Modify a system model.
5. Click the **See unpublished commits** icon, and then select the **Publish changes** option.

    **Note:** If you do not want to commit the changes that you made to the system model, you can select the **Discard changes** option.

6. Enter a commit message in the **Description of change** field, and then click **Publish**.

**Results**

You have published the changes that you made in the system model to the team space repository. Other members of the team space can now view and use the system model.

## Associating resources with components

After creating components in the system model, you can navigate to a project in the team space to associate the Suites and Tests or virtual services resources in the project repositories with the components in the system model.

**Before you begin**

- You must have created components.
- You must be a member of the project that contains the Suites and Tests or virtual service resources.

1. Click **Projects**.

    The **Projects** page is displayed.

2. Click the project that contains the Suites and Tests or virtual service resources that you want to associate with components.
3. Click **System Model**.
4. Click the component that you want to associate with the Suites and Tests or virtual services resources.

    The component details are displayed in the right pane.

5. Click the **Associate** tab.

    The associated resources are displayed. To associate more resources, click the **Add association** icon ⊕ next to the resource type.

    **Note:** Click the **Add** button to associate resources with the component for the first time.

    To associate a resource with a component for the first time, you must perform the following steps:

a. Click **Add**.

   A drop-down list is displayed.

b. Select one of the following resource types:

   - **Suites and Tests**: Click this option if you want to associate your component with the different types of test suites.
   - **Virtual Services**: Click this option if you want to associate your component with the virtual service resources.

   The **Associate Assets** page is displayed.

   **Notes:**

   - You can view all the available resources of Suites and Tests or virtual services by selecting the **Suites and Tests** and **Virtual Resources** check boxes.
   - You can use the **Search** field to search for virtual service resources by entering the name of the Suites and tests or virtual service resources.

c. Select one or more resources that you want to associate, and then click **Add**.

   The added resources are displayed in the **Associate** and **Overview** tabs of the component.

6. **Optional**: Associate more resources by clicking the **Add association** icon ⊕ next to the resource type.

**Results**

You have associated components with the Suites and Tests or virtual service resources.

## Viewing resources associated with components

After you associate Suites and Tests or virtual services resources with components in the system model, you can view the associated resources from the **System Model** page. You can opt to start an instance of the associated resource or stop a running instance of the associated resource.

**Before you begin**

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to server projects on page 504.
- Created and published the system model. See Creating a system model on page 186.
- Created components in the system model. See Creating components on page 188.
- Associated resources in a project with components in the system model. See Associating resources with components on page 198.

1. Log in to HCL OneTest™ Server.
2. Open the team space to which a team space repository is added.

The **Active projects** page is displayed.

3. Click **Active projects > My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.

4. Click **System Model** on the left navigation pane.

   The **System Model** page is displayed.

5. Click the component that you associated with the resources in the project repository.

   The resources associated with the component are displayed under the resource type in the **Associate** tab.

6. Click the **Overview** tab, if it is not already displayed.

   The resources that are associated with the component are displayed under the **Suites and Tests** or **Virtual Services** sections.

7. View and perform the actions for the resource type that you want from the following options:

   - For **Suites and Tests** resources, go to 8 on page 200.
   - For **Virtual Services** resources, go to 9 on page 201.

8. Identify the test resource that are listed under the **Suites and Tests** section, and perform the action that you want described in the following table:

| Task | Description | Action |
|---|---|---|
| **Collapse** icon ⌄ | You can collapse the resource sections by clicking the **Collapse** icon ⌄.<br>**Note:** When the section is collapsed, you can view the number that is displayed of the resources that are associated. | Click the icon to toggle between the collapsed and expanded display of the **Suites and Tests** section. |
| **Type** | You can view the type of the Suite or test. | Hover over the Suites and Tests icon under the **Type** column to view the type of Suite or test. |
| **Name** | You can view the name of the associated Suite or test and the path in the repository. | There is no action to perform. |
| **Status** | You can view the details of the test runs. | Click each of the squares to view the details of the specific test run. For example, if the status of a test run is represent- |

| Task | Description | Action |
|---|---|---|
| | | ed as shown in the image ▨▨▨🟥●, then it indicates that the Suite or test has run only two times. After you click any colored square, the **Results** page is displayed. |
| **Action** | You can start the execution of the Suite or test from the **Execution** page when you click the **Show in execution page** icon ↗. | Click the **Show in execution page** icon ↗ to view the Suite or test on the **Execution** page.<br>The Suite or test is displayed on the **Execution** page. |

9. Identify the virtual service resource that are listed under the **Virtual Services** section, and perform the action that you want described in the following table:

| Field | Description | Action |
|---|---|---|
| **Collapse** icon ⌄ | You can collapse the resource sections by clicking the **Collapse** icon ⌄.<br>**Note:** When the section is collapsed, you can view the number that is displayed of the resources that are associated. | Click the icon to toggle between the collapsed and expanded display of the **Virtual Services** section. |
| **Show all** icon ↗ | You can opt to view the virtual services that are associated with the component either on the **Resources** or **Instances** page.<br>You can start an instance of the virtual service from the **Resources** page while you can stop a running instance from the **Instances** page. | Perform the following steps:<br>a. Click the **Show all** icon ↗.<br>b. Select any of the following actions:<br>▪ To start an instance of the virtual service:<br>i. Select **Show all in resources page** to go to the **Resources** page.<br>ii. Click the **Execute** icon ▶ in the row of the virtual service.<br>iii. Configure the settings for the run of the virtual service, if you want to change any of the settings, else click **Execute.**<br>The virtual service starts to run. |

| Field | Description | Action |
|---|---|---|
| | | ▪ To stop a running instance of the virtual service:<br>i. Select **Show all in instances page** to go to the **Instances** page.<br>ii. Identify the instance that you want to stop and click the **Stop** icon ⏹ in the **Actions** column of the selected virtual service instance.<br>iii. Click **Ok** in the **Stop virtual service instance** dialog that is displayed.<br>The running virtual service instance is stopped. |
| **Name** | You can view the name of the associated virtual service and the path in the repository. | There is no action to perform. |
| **Instances** | You can view the state of the running instance of the virtual service resource. The state is displayed as `Running` only if the instance is running. | There is no action to perform. |
| **Action** | You can go to the **Resources** page to view the virtual service resource when you click the **Show resource** icon ↗.<br>You can start an instance of the virtual service from the **Resources** page. | Click the **Show resource** icon ↗ to view the virtual service on the **Resources** page.<br>The virtual service resource is displayed on the **Resources** page. |
| **Expand** icon › | You can expand the virtual service card only if the virtual service is a running instance.<br>In the expanded card, you can view the details such as the Environment, the total number of requests received by the virtu- | Perform the following steps:<br>a. Expand the virtual service card for a running instance.<br>You can view the details of the instance. |

| Field | Description | Action |
|---|---|---|
| | al service, and the relative time from when the virtual service was started.<br>You can stop a running instance of the virtual service from the **Instances** page. | b. Click the **Show instance** icon ↗ to go to the **Instances** page to view the running instance of the virtual service.<br>The running instance of the virtual service is displayed on the **Instances** page. |

**Results**

You have viewed the resources that are associated with a component from the **System Model** page.

## Removing resources associated with components

When you do not want to retain the resource that you associated with a component, you can delete the resource in the system model.

**Before you begin**

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to server projects on page 504.
- Created and published the system model. See Creating a system model on page 186.
- Created components in the system model. See Creating components on page 188.
- Associated resources in a project with components in the system model. See Associating resources with components on page 198.

1. Log in to HCL OneTest™ Server.
2. Open the team space to which a team space repository is added.

   The **Active projects** page is displayed.
3. Click **Active projects > My projects > *project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.
4. Click **System Model** on the left navigation pane.

   The **System Model** page is displayed.
5. Click the component that you associated with the resources in the project repository.

   The resources associated with the component are displayed under the resource type in the **Associate** tab.
6. Select the resource that you want to remove from the component by clicking the checkbox of the resource.

   You can select any or all resources for removal.

7. Click the **Delete selected** icon 🗑 inline with the resource type.

   The **Disassociate...** dialog is displayed for the type of resource that you selected. For example, the **Disassociate Suites and Tests** dialog is displayed when you selected the test assets.

8. Click **Delete**.

   For the changes to be reflected in the system model for all members of the project, you must publish the changes to the project repository.

9. Open the **Changes** page by clicking **Author > Changes**.
10. Verify the changes listed and then click **Publish changes**.
11. Enter a commit message in the **Description of change** field, and then click **Publish**.

**Results**

You have removed resources that were associated with components in the system model.

## Publishing changes of associating resources with components

After you associate resources in a project with a component in the system model, you must publish the changes made to the branch in the project repository, if you want the associations to be visible to the other members in the project.

**Before you begin**

You must have completed the following tasks:

- Created and published the system model. See Creating a system model on page 186.
- Created components in the system model. See Creating components on page 188.
- Associated resources in a project with components in the system model. See Associating resources with components on page 198.

1. Log in to HCL OneTest™ Server.
2. Open the team space to which a team space repository is added.

   The **Active projects** page is displayed.

3. Click **Active projects > My projects >** ***project_name*** to open the project that contains the test assets.

   The **Overview** page of the project is displayed.

4. Open the **Changes** page by clicking **Author > Changes**.

   The **Changes** page is displayed. You can find the following information about the changes that you made to the system model when you associated resources with components:

   - The resources that you associated with components are listed under the **Asset name** column.
   - The type of change is listed under the **Changes** column.
   - The name of the member who made the change is displayed under the **Last changed by** column.

    - The relative time when the change was made is displayed under the **Last updated** column.
    - The number of changes that the *Edit* branch is ahead of from the project repository branch is indicated by the **commits ahead** option. You can view the details by clicking the **commits ahead** option.
    - The number of changes that the *Edit* branch is behind from the project repository branch is indicated by the **commits behind** option. The *Edit* branch can be behind the project repository branch if another member has published or committed other changes in the system model to the same project repository branch. You can view the details by clicking the **commits behind** option.
5. Click the **Publish changes** option.

   **Note:** If you do now want to commit the changes you made to the system model, you can select the **Discard changes** option.

6. Enter a commit message in the **Description of change** field, and then click **Publish**.

### Results

You have published the changes that you made in the system model to the repository in the project. Other members of the team space can now view and use the system model.

## Viewing the system model

You can view an existing system model or you can create a system model if a system model does not exist on the **System Model** page.

### Before you begin

You must be a licensed user.

1. Log in to HCL OneTest™ Server.
2. Click **System Model**.

   The **System Model** page is displayed.
3. Perform any of the following actions:
    - Add a team space repository, if no repository is configured as the team space repository.
    - Create a system model, if no model is created or exists.
    - View the system model that exists.
    - Modify the existing system model.

### Results

You have viewed an existing system model from the **System Model** page.

## Deleting a system model

When you want to remove an existing system model to create a new system model, you can delete the existing system model.

**Before you begin**

You must have ensured that a system model exists in the **Initial Team Space**.

1. Log in to HCL OneTest™ Server.
2. Click **System Model**.

   The **System Model** page is displayed.

3. Click the **Open action menu** icon ⋮ in the right-pane.
4. Click the **Delete** icon 🗑.

   The **Delete System Model** dialog is displayed.

5. Click **Delete**.

**Results**

You have deleted the system model from the team space repository.

**What to do next**

You must publish the changes to the team space repository. See Publishing changes to the system model on page 197.

## Deleting a repository that is added to a team space

You can delete a repository that you added to the team space when you do not want to use a team space repository.

**Before you begin**

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Team Space Owner* or an *Architect* in the team space. See Managing members and their roles in a team space on page 493.
- Added a repository to the team space. See Adding a repository to a team space on page 184.

1. Log in to HCL OneTest™ Server.
2. Open the team space to which a team space repository is added.

   The **Active projects** page is displayed.

3. Click the **Settings** icon ⚙ from the left navigation pane, and then select **Manage team space**.

   The **Details** page is displayed.

4. Click **Repository** on the left navigation pane.

   The **Repository** page is displayed.

5. Click the **Open action menu** icon ⋮.
6. Click **Delete** in the action menu list.

The **Delete repository** dialog is displayed.

7. Select the **I understand...** checkbox, and then click **Delete**.

**Results**

You have deleted the ream space repository.

🚫 **Restriction:** When you delete the team space repository, the system model if created is also deleted, and you or other members of the team space cannot add a system model.

# Prerequisites to running tests

Before you can configure and run a test in a project on HCL OneTest™ Server, you must read the information about the different tests. You might want to add a remote static agent or a remote Docker host to the project as an alternate location to run the tests.

You can find the following information:

Related information

Tests configurations and test runs

## Test run considerations for AFT Suites

Before you configure an AFT Suite run, you must read the considerations you must take into account.

When you want to run an AFT Suite from a project on HCL OneTest™ Server, you can check if the AFT Suite meets any of the following conditions:

| If... | Then... |
|---|---|
| The AFT Suite uses the settings configured in an AFT XML file for a remote agent | You must ensure that you have configured the location element in the AFT XML to point to a remote agent.<br>For more information about setting the location element, refer to Using an XML file to run multiple Web UI tests and compound tests simultaneously. |
| You want to run an AFT Suite on a remote agent | You must run the test on the remote agent before you commit the test asset to the remote repository.<br>When you run the test on the remote agent, the following events occur: |

| If... | Then... |
|---|---|
| | - The agent is added as the location in the AFT XML file, on which the test is to be run.<br>- The agent is displayed as the agent on which the test is to be run, under the **Host** column within the **Location** tab in the **Execute test asset** dialog box. |
| You commit the test asset to the remote repository without running the test on the remote agent | The agent is not displayed under the **Host** column within the **Location** tab in the **Execute test asset** dialog box. |

Before you configure the run for an AFT Suite, you must complete the following tasks, if they are applicable:

- Add the remote agents, on which the test is to be run, to the project on HCL OneTest™ Server. See Adding an agent to a project on page 220.
- After you add the agent, you can select the agent configured in the `AFT XML` file as the location for the test run or select the remote agents that you add to the project, when you configure a run for an AFT Suite.

  The remote agents are shown as the agents available for selection under the **Override** column within the **Location** tab in the **Execute test asset** dialog box.

  You can then select the remote agent as the location to run the AFT Suite when you are configuring the test run. Alternatively, you can enter the argument `-swaplocation <configured_agent_location>:<overriding_agent_location>` in the **Program Arguments** field in the **Advanced Setting** dialog box when you are configuring the test run.

For running an AFT Suite, see Configuring an AFT Suite run on page 241.

Related information

Tests configurations and test runs

## Test run considerations for API Suites

Before you configure an API Suite run, you must read the considerations you must take into account.

You can find the following information about API Suites:

- Important information about API Suites on page 209
- Important information about report configuration in API Suites on page 211

## Important information about API Suites

When you want to run an API Suite from a project on HCL OneTest™ Server and the test suite in HCL OneTest™ API meets any of the following conditions:

- The test suite refers to local stubs that use a transport other than HTTP or MQ.
- The test suite refers to local stubs that use the HTTP transport.
- The transport used in the tests in the test suite is configured with the host name set to *localhost* for the HTTP/TCP proxy.
- The test suite has tests that use a transport and the transport requires third-party application `Jar` files for a successful run.

You must then refer to the following table for the next steps:

| If... | Then... |
|---|---|
| The tests in an API Suite refer to local stubs that use a transport other than HTTP or MQ. | You must perform any of the following actions before you commit the test asset to the remote repository:<br>• Remove the reference to the local stub from the test suite.<br>• Publish the stubs to HCL® Quality Server.<br>To publish and edit stubs, see Publishing stubs. |
| The tests in an API Suite refer to local stubs that use the HTTP transport. | You must perform the following actions before you commit the test asset to the remote repository:<br>1. Create a test suite that contains the stub and tests that run on the stub.<br>2. Configure the physical HTTP transport with the following parameters in the **Client** tab in the **Physical View**:<br>◦ Set the `Host name or IP address of the HTTP Proxy server` in the **Proxy Host** field.<br>◦ Set *3128* as the proxy server port in the **Proxy Port** field.<br>3. Set the reference to the stub for the tests in the test suite scenario from the **Scenario Editor** dialog box from the **Test Factory** view by selecting the *Live system* option in the **Satisfied by** column |

| If... | Then... |
|---|---|
| | to run the stubs on the HTTP proxy registered with HCL OneTest™ Server.<br>4. Save the project. |
| The transport used in the tests in an API Suite is configured with the host name set to *localhost* for the HTTP/TCP proxy. | You must replace the host name with *fully-qualified-domain-name* or *IPAddress* of the proxy host in the tests before you commit the test asset to the remote repository. |
| You plan to run the API Suite on the computer where HCL OneTest™ Server is installed and the API Suite contains tests that use a transport from any of the following third-party applications:<br>- Camel<br>- CentraSite<br>- CICS<br>- Coherence<br>- Database<br>- IMS<br>- Integra<br>- JMS<br>- SAP RFC<br>- Software AG Universal Messaging<br>- TIBCO EMS<br>- TIBCO Rendezvous<br>- TIBCO SmartSockets<br>- WebSphere Application Server Service Integration Bus (SiBus)<br>- WebLogic<br>- Software AG webMethods<br>- WebSphere MQ | You must complete the following tasks:<br>1. Identify the location of the third-party application JAR files. You can use Library Manager to know the location where the third-party application JAR files are saved.<br>2. Copy the third-party application JAR files to the computer where HCL OneTest™ Server is installed.<br>See Copying third-party application Jars to Kubernetes on page 83. |
| You plan to run the API Suite on a remote Docker host and the API Suite contains tests that use a transport from any of the following third-party applications: | You must complete the following tasks: |

| If... | Then... |
|---|---|
| • Camel<br>• CentraSite<br>• CICS<br>• Coherence<br>• Database<br>• IMS<br>• Integra<br>• JMS<br>• SAP RFC<br>• Software AG Universal Messaging<br>• TIBCO EMS<br>• TIBCO Rendezvous<br>• TIBCO SmartSockets<br>• WebSphere Application Server Service Integration Bus (SiBus)<br>• WebLogic<br>• Software AG webMethods<br>• WebSphere MQ | 1. Identify the location of the third-party application JAR files. You can use Library Manager to know the location where the third-party application JAR files are saved.<br>2. Copy the third-party application JAR files to the computer where the remote Docker host is installed.<br>See Copying third-party application Jars to a remote Docker host on page 229. |

## Important information about report configuration in API Suites

You must configure test suites in HCL OneTest™ API to use a results database so that the details of the test results can be captured and displayed in the HCL OneTest™ Server API results reports. See Configuring the project results database.

If your server projects use either a Microsoft SQL Server, MYSQL, or both as a project results database, you must copy the Microsoft SQL Server and MySQL database JAR files to the JDBC folder in the path `/data/JDBC` on HCL OneTest™ Server. See Copying third-party application Jars to Kubernetes on page 83.

The database JAR files that you need are based on the version of Microsoft SQL Server or MQSQL that you use with HCL OneTest™ API. See Adding Microsoft SQL Server and MySQL drivers.

To run an API Suite, see Configuring an API Suite run on page 249.

Related information

Scenario reference setting

Tests configurations and test runs

# Test run considerations for schedules

Before you configure a Rate Schedule or VU Schedule run, you must read the considerations that you must take into account.

## Important information about JMeter tests in schedules

When you want to run JMeter tests as part of the VU Schedule or Rate Schedule on HCL OneTest™ Server, you must have completed the following tasks:

- Installed the HCL OneTest™ Performance Agent on a computer.
- Installed the JMeter application on the computer where you have installed the HCL OneTest™ Performance Agent.
- Set the environment variable **JMETER_HOME** that points to the JMeter installation directory.

## Important information about Resource Monitoring sources in schedules

When you want to run schedules on HCL OneTest™ Server, you must have completed the following tasks in HCL OneTest™ Performance:

- Created a performance schedule of **Rate** or **VU** type.
- Added Resource Monitoring sources, or the labels of Resource Monitoring sources to a schedule. Refer to Adding resource monitoring sources to a performance schedule by using labels.

Before you configure a run of a Schedule, you must complete the following tasks in HCL OneTest™ Server:

- Add Resource Monitoring sources to your project in HCL OneTest™ Server. See Monitoring host resources on page 375.
- Enable HCL OneTest™ Server as Resource Monitoring service only if you want to replace Resource Monitoring labels set in HCL OneTest™ Performance with the labels that you created in HCL OneTest™ Server. See Controlling Resource Monitoring sources in a schedule.
- Perform the following steps when you want to override the Resource Monitoring labels in the test asset with the Resource Monitoring labels that you create in HCL OneTest™ Server:
    1. Click the **Resource Monitoring** tab.

        **Note:**

        The **Resource Monitoring** tab displays only if you enabled the **Resource Monitoring from Service** option in HCL OneTest™ Performance when you created the test asset.

    2. Click +, and press the `Ctrl` + `Space bar` keys, or enter the initial letter of a label to select a label in the list.

**Note:**

You can select or add labels only if you added the labels in the **Resource Monitoring Sources** page in HCL OneTest™ Server.

For running a Rate Schedule or VU Schedule, see Configuring a run of a Rate Schedule or VU Schedule on page 302.

Related information

Tests configurations and test runs

## Considerations for using Jaeger traces in reports

Before you can use Jaeger to report the test results in the supported tests, you must read the considerations that you must take into account.

When you want to enable Suites, Tests, or Schedules that are run from HCL OneTest™ Server to use Jaeger traces for reporting the test results, you must have completed the following tasks:

- Installed Jaeger at the time of installation of the server software. See Prerequisites for installing the server software on Red Hat OpenShift on page 36 or Prerequisites for installing the server software on Ubuntu on page 44.
- Verified that Jeager is enabled and working.

After you install and verify that Jaeger is enabled, and when you want Jaeger to report the test results for the tests that you configure for a run, you must perform the following steps:

1. Click **Advanced**.
2. Enter `-history jaeger` in the **Program Arguments** field.

Related information

Tests configurations and test runs

## Test run considerations for JMeter tests

Before you can run JMeter tests on HCL OneTest™ Server, you must read the considerations that you must take into account and complete the tasks indicated.

When you want to run JMeter tests that you create in JMeter, you must have completed the following tasks:

- You must ensure that the JMeter extension on HCL OneTest™ Server is enabled at the time of installation of the server software or is running before you add the repository that contains JMeter tests to the server project. See Prerequisites for installing the server software on Red Hat OpenShift on page 36 or Prerequisites for installing the server software on Ubuntu on page 44.
- You must have committed the following JMeter assets or resources as a JMeter project to the remote repository:

  **Note:** You must have created a project directory and added the associated tests or assets in sub directories, that might be required for the JMeter tests to run successfully.

  - The `jmeterRoot.jprj` project file in the project directory.
  - The JMeter tests in a sub directory with a name you provide within the project directory.
  - Optionally, the Java Key Store that is required at test run time in the `certs` sub directory.
  - Optionally, the JMeter properties files that are required at test run time in the `properties` sub directory.
  - Optionally, the library jar files that are required at test run time in the `lib` sub directory.
  - Optionally, the test dependencies that are required at test run time in the `deps` sub directory.

  For example, the project directory can be *myproject* that contains the `jmeterRoot.jprj` project file. In the project directory, you can create the following sub directories:

  - The `mytests` sub directory that contains the JMeter tests.
  - Optionally, the `certs` sub directory that contains the Java Key Store.

    For example, the path to the key store can be specified in the `system.properties` file as follows:

    ```
    javax.net.ssl.trustStore=$(deploy.home)/certs/simpleStore.jks
    ```

    Where <*simpleStore.jks*> is the Java Key Store and the <*deploy.home*> is the keyword to access the key store at the test run time.
  - Optionally, the `properties` sub directory that contains properties files such as `system.properties`, `jmeter.properties`, `reportGenerator.properties`, or `user.properties`.
  - Optionally, the `lib` sub directory that contains the **lib** and **lib/ext** content to be used at test run time. For example, the **amq client** and the **plugins** required to parse the **jmx** that contains **amqp** calls.
  - Optionally, the `deps` sub directory that contains the test dependencies.

  The resulting project directory can be as follows:

  ```
  myproject (that contains the jmeterRoot.jprj)
  myproject/mytests
  myproject/certs
  myproject/properties
  myproject/lib
  myproject/deps
  ```

- You must ensure that you use a relative path when JMeter tests refer to other test assets. For example, if the JMeter test that is in the `myproject/mytests` folder refers to a `mycsv.csv` file that is in the `myproject/deps` folder, then the `mycsv.csv` is referred as follows: `../deps/mycsv.csv` in the JMeter test.
- You can create the JMeter project file named as `jmeterRoot.jprj`.

  The project file is a java properties file. The JMeter project file is required so that HCL OneTest™ Server can identify the asset as a JMeter project. You can specify additional information in the project file. You can specify the names of the sub directories created and through keywords you can specify the path in the properties files whether they are relative to the root directory or to the tests sub directory.

  For example, if you have created the following sub directories in the project directory called *myproject* that contains the `jmeterRoot.jprj` project file:

  - `myproject/mytests`
  - `myproject/properties`

  If the JMeter tests are contained in the following path: `myproject/mytests/testplan/test1.jmx`, and some dependencies that are also required to be deployed at test run time exists in the following path: `myproject/mytests/testplan/localdir`. You must specify these file paths in the project file as follows:

  ```
  testDir.localtest.sub.dir=localdir
  aSecondRootDir.sub.dir=dir2
  athirdRootDir.sub.dir=dir3
  ```

Related reference

Troubleshooting issues on page 537

Related information

Configuring a JMeter test run on page 289

## Test run considerations for JUnit tests

Before you can run JUnit tests on HCL OneTest™ Server, you must read the considerations that you must take into account and complete the tasks indicated.

When you want to run JUnit tests, you must have completed the following tasks:

- You must ensure that the JUnit extension on HCL OneTest™ Server is enabled at the time of installation of the server software or is running before you add the repository that contains the JUnit tests to the server project. See Prerequisites for installing the server software on Red Hat OpenShift on page 36 or Prerequisites for installing the server software on Ubuntu on page 44.

- You must have created the JUnit tests that use JDK 8, as part of a Maven V3.6.3 project. The Maven project must contain the Maven Surefire plugin, which is required to run the JUnit tests contained in the Maven project. The Maven project as a `pom.xml` must be committed to the remote repository.

  **Note:** If you create the JUnit tests on other versions of JDK, you must copy the JDK version to the `userlibs` pod under the `data/junit-ext/jdks` directory. See Copying third-party application Jars to Kubernetes on page 83.

Related reference

Troubleshooting issues on page 537

Related information

Configuring a JUnit test run on page 293

## Test run considerations for Postman tests

Before you can run Postman tests on HCL OneTest™ Server, you must read the considerations that you must take into account and complete the tasks indicated.

When you want to run Postman tests that you create in Postman, you must have completed the following tasks:

- You must ensure that the Postman extension on HCL OneTest™ Server is enabled at the time of installation of the server software or is running before you add the repository that contains Postman collections to the server project. See Prerequisites for installing the server software on Red Hat OpenShift on page 36 or Prerequisites for installing the server software on Ubuntu on page 44.

  **Restriction:** If you add the repository with Postman collections to a server project before you enable the Postman microservice, the Postman collections might not be displayed in the **Execution** page for you.

- You must have exported the following resources from Postman and saved the resources on your computer:
  - Collections and their associated variables and environments from Postman in which all collections are exported as a single file.

    **Note:** You can export as many collections as you want.

    For more information, refer to Exporting a collection from Postman and Exporting data dumps.
  - Environments configured in Postman, if you are exporting only the environments as separate JSON files.

    For more information, refer to Exporting environments from Postman.
- You must have committed the following Postman assets or resources to the remote repository:

- Collections and their associated variables and environments that you exported. You can export all resources that are in your workspace in Postman. You can also export the collections along with the variables as a single file and the environments separately as another file.

  For example, you can save the exported file as `MyCollectionsVarEnv.json` that contains the collections, variables, and environments or you can export only the collections and variables as `MyCollectionsVar.json`.
- Environments that you exported, if you are exporting only the environments as separate JSON files. For example, you can save the exported file as `MyEnv.json`.
- An empty text file with the `.postman` extension that is contained in the same directory as the other Postman resources, so that the contents in the directory are attributed as Postman resources. For example, you can create `MyProject.postman`.

  **Note:** The `.postman` file enables HCL OneTest™ Server to identify the test resources as Postman resources.

For example, if you have saved the Postman resources that you downloaded from Postman as follows:

- `MyCollectionsVarEnv.json` or `MyCollectionsVar.json` that contains the Postman collections.
- `MyEnv.json` that contains the environments associated with the Postman collection.

The project directory can be as follows:

```
/Project
/Project/MyProject.postman
/Project/MyCollectionsVarEnv.json
```

Alternatively, the directory can be as follows:

```
/Project
/Project/MyProject.postman
/Project/MyCollectionsVar.json
/Project/MyEnv.json
```

Related reference

Troubleshooting issues on page 537

Related information

Configuring a Postman test run on page 297

## Agents overview

You can find information about the agents that you must install on remote computers and configure the agents with the offline user token so that the agents register with HCL OneTest™ Server. After the agents are registered, you can add them to your projects and then select them as locations to run tests.

When you plan to run any of the tests on a remote agent from a project on HCL OneTest™ Server, you must have completed the following tasks:

- Installed the agent of the same version as the version of HCL OneTest™ Server.
- Configured the agent with your offline user token so that the agent can connect and also register with HCL OneTest™ Server.

  **Note:** When you configure the agent to connect to the server by using the offline user token that you generated on HCL OneTest™ Server, you become the owner of the agent.

- Verified that you have write permissions to the `majordomo.config` file so that certain transactions can be written by HCL OneTest™ Server to the `majordomo.config` file.

  **Note:** You can verify if you have the *Read* and *Write* permissions by checking the properties of the `majordomo.config` file.

- Viewed the agents that are registered with HCL OneTest™ Server. See Viewing agents that are registered with HCL OneTest Server on page 219.
- Added the remote agents that are registered with HCL OneTest™ Server to your project on HCL OneTest™ Server. See Adding an agent to a project on page 220.

  **Note:** After you add the remote agents to your project, HCL OneTest™ Server determines the capabilities that are provided by the remote agent and displays the details and capabilities of the agent on the **Infrastructure** page. You can also define the capabilities that you want to add, and then view, edit, or delete the capabilities that you defined for the agents in your project. See Working with agent capabilities on page 222.

Related information

Agent installation

## Test run considerations for running tests on remote agents

Before you configure a test to run on a remote agent from HCL OneTest™ Server, you must read the considerations that you must take into account.

### Supported tests

You can run the following tests that are supported to be run on a remote agent:

- AFT Suite
- Compound Tests that contains performance tests

- Compound Tests that contains Web UI tests
- Compound Tests that contains traditional HTML tests
- Rate Schedule
- VU Schedule

### Prerequisite tasks

Before you can run any of the supported tests on a remote agent from a project on HCL OneTest™ Server, you must have completed the following tasks:

- Installed the agent on the remote computer. See Agents overview on page 217.
- Configured the agent with your offline user token so that the agent can connect to HCL OneTest™ Server.
- Added the remote agents to your project on HCL OneTest™ Server. See Adding an agent to a project on page 220.

**Note:** If you want to run an AFT Suite (that contains Web UI tests) on an agent, you must start the agent as a non-admin user so that certain browsers that are configured start correctly during the test run time.

Before you add the tests to the remote repository, you must have completed the following tasks:

- Associated the agents with the performance schedule in HCL OneTest™ Performance.
- Specified the host name of the remote workstation in the AFT XML file.

You can continue to configure a run for any of the following supported test types:

- Configuring an AFT Suite run on page 241
- Configuring a run of a Compound Test that contains performance tests on page 281
- Configuring a run of a Compound Test that contains Web UI tests on page 264
- Configuring a run of a Compound Test that contains HTML tests on page 257
- Configuring a run of a Rate Schedule or VU Schedule on page 302

## Viewing agents that are registered with HCL OneTest™ Server

After you install the remote agents and configure them with your offline user token, the agents register with HCL OneTest™ Server and you can view the registered agents on the **Agents and Intercepts** page.

#### Before you begin

You must have completed the following tasks:

- Installed the agent of the same version as the version of HCL OneTest™ Server.
- Configured the agent with your offline user token so that the agent can register with HCL OneTest™ Server.

  **Note:** When you configure the agent to connect to the server, you must specify the offline user token that you generated on HCL OneTest™ Server. This token defines your ownership over the agent.

1. Log in to HCL OneTest™ Server.
2. Click **Infrastructure > Agents and Intercepts** in the navigation pane.

   The **Agents and Intercepts** page is displayed.

   You can view the agents, intercepts or Dockers that are registered with HCL OneTest™ Server.
3. You can view the agents that you own in any of the following ways:
   - Search for the agent by entering the name of the agent in the **Search** field.

     **Note:** You can enter either the full name or any text that is in the name. The search is enabled for case sensitive text that you can enter.
   - Sort the **Type** column to sort the items and display the agents in the rows at the top of the table.
   - Sort the **Agents** column and identify the agents by their names or by their owner.

   You can view the following details about the agents that are registered with HCL OneTest™ Server:
   - The projects to which the agent is added, are displayed in the **Projects** column.
   - The status of the agent is displayed in the **Status** column.

     **Note:** You can add agents that are in the `Ready` state to your project, and then use them as locations to run tests.

### Results

You have viewed the registered agents from the **Agents and Intercepts** page.

### What to do next

You can add the agents that you own to your project. See Adding an agent to a project on page 220.

## Adding an agent to a project

If you have performance tests that distribute workload across different workstations or Accelerated Functional Testing (AFT) Suites that need to run on different platforms and browsers, you can add HCL OneTest™ Performance Agents to your projects to run tests on remote workstations.

### Before you begin

You must have completed the following tasks:

- Read about the considerations that you must take into account before you configure a test run to run on a remote agent. See Test run considerations for running tests on remote agents on page 218.
- Been assigned the *Owner* or *Tester* role in the project to which you want to add the agent.
- Owned at least one agent that is not added to the project.

### About this task

Only you can add or remove this agent from the project. However, any member of your project with the permission to run the tests can initiate the run of the performance schedule and AFT Suites that are associated with the agent.

1. Log in to HCL OneTest™ Server, open your project, and click **Manage > Infrastructure** to go to the **Infrastructure** page.
2. Click **Add > Add agent**.

   The agents that you configured with your user token are displayed.
3. Select the agents that you want to use and click **Add**.

   The agents that you added are displayed.
4. Perform the following steps to view details and capabilities of the agents:

   a. Expand the agent to view the details panel.

   You can find the following details of the agent displayed:

| Parameter | Description |
|---|---|
| Version | The version of the installed agent. |
| Address | The IP address of the computer on which the agent is installed. |
| License type | The license type that is enabled for you to use the agent. |
| Load generation service | The status of the agent whether it can be used for load generation service. |
| Last contact | The time of the last contact with the agent by HCL OneTest™ Server. |

   b. Expand the capabilities panel to view the system and application capabilities.

   You can find the following capabilities of the agent displayed:

| | Capability | Description |
|---|---|---|
| System capabilities | Architecture | The architecture of the CPU processor of the computer on which the agent is installed. |
| | Operating system | The operating system of the computer on which the agent is installed. |

| | Capability | Description |
|---|---|---|
| | RAM | The memory that is allotted to RAM in the computer on which the agent is installed. |
| Application capabilities | Browser | The browsers along with their version that are installed on the computer on which the agent is installed. |
| User Defined capabilities | User defined | The capabilities that you add for an agent. You can also tag an agent so that you can identify the agent to select as a location. |

c. Check for the status of the agent from the **Status** column.

**Note:** You can add agents that are in any state to the project.

**Results**

You have added remote agents to your project and viewed the agent capabilities displayed for each agent that you own.

**What to do next**

You can perform any of the following tasks:

- Run the tests on the agent that is added to the project from the **Execution** page. See Test run configurations on page 240.
- Define the capabilities that you want to add, and then view, edit, or delete the capabilities that you defined for the agents in your project. See Working with agent capabilities on page 222.

## Working with agent capabilities

When you have installed static agents V10.1.2 or later, and added them to your project, you can view capabilities of the agents from the **Infrastructure** page. When you have installed static agents V10.1.3 or later, you can add, view, edit, or delete capabilities of agents from the **User Defined** panel of that agent.

**Before you begin**

You must have completed the following tasks:

- Been assigned the *Owner* or *Tester* role in the project to which you want to add the agent.
- Owned at least one agent that is not added to the project.
- Installed the agent V10.1.3 or later. See Test run considerations for running tests on remote agents on page 218.
- Added agents to your project. See Adding an agent to a project on page 220.

1. Log in to HCL OneTest™ Server, open your project, and click **Manage > Infrastructure** to go to the **Infrastructure** page.

   The agents that you added to your project are displayed.

2. Expand the agent.

   The **Details** and **Capabilities** tabs are displayed.

3. Click **Capabilities** to view the capabilities of the agent.
4. Click **User Defined** to view the capabilities that you can add, view, edit, or delete.
5. Complete the steps for the task that you want to perform as listed in the following table:

| Task | Action |
|---|---|
| Adding a capability | a. Click **Add new**.<br>b. Enter the name of the capability in the **Name** field.<br>c. Enter the value of the capability in the **Value** field.<br>d. Click **Save**.<br>The capability that you add is displayed. |
| Viewing capabilities that you added | The **User Defined Capabilities** panel displays the capabilities that you have added to the selected agent. |
| Editing capabilities that you added | a. Click ✏ in the row of the capability that you want to edit.<br>b. Edit the name of the capability in the **Name** field or the value of the capability in the **Value** field or edit both fields.<br>c. Click **OK**.<br>The updated capability of the agent is displayed. |
| Deleting capabilities that you added | a. Click 🗑 in the row of the capability that you want to delete.<br>b. Click **Delete**.<br>The capability that you selected is deleted. |

### Results

You have added, viewed, edited, or deleted the capability of an agent.

### What to do next

You can run the tests on the agent from the **Execution** page. See Test run configurations on page 240.

# Managing Docker hosts

You can find information about the tasks that you can perform on remote Docker hosts. After you set up a remote Docker host, you can register the Docker host with HCL OneTest™ Server. You must add the registered remote Docker host to your project before you can run tests on the remote Docker host location.

## Test run considerations for running tests on remote Docker hosts

Before you can configure a test to run on a remote Docker host, you must read the considerations that you must take into account.

You can run all the tests that are supported on HCL OneTest™ Server on remote Docker hosts.

**Important:** You can run virtual services only in the **Default Cluster** location of HCL OneTest™ Server. You cannot run virtual services on a remote Docker host.

When you plan to run any of the tests on a remote Docker host from a project on HCL OneTest™ Server, you must complete the following tasks:

| | Tasks | For information, go to... |
|---|---|---|
| 1.1 | Set up non-secure remote Docker hosts. | Setting up a remote Docker host computer on page 225 |
| 1.2 | Set up secure remote Docker hosts. | Setting up a secure remote Docker host computer on page 227 |
| 2 | Register the remote Docker host on HCL OneTest™ Server. | Registering a remote Docker host with HCL OneTest Server on page 231 |
| 3 | View the registered Docker hosts. | Viewing remote Docker hosts that are registered with HCL OneTest Server on page 233 |
| 4 | Add a remote Docker host to the HCL OneTest™ Server project. | Adding a remote Docker host to the project for running tests on page 234 |

| | Tasks | For information, go to... |
|---|---|---|
| 5 | Configure a run for your test that you want to run on the remote Docker host by performing the following actions:<br>1. Go to Test run configurations on page 240.<br>2. Check for the prerequisite tasks that you must perform before you configure a test run for the test.<br>3. Select the task for the test that you want to configure for a run, and then follow the steps to run the test. | 1. Test run configurations on page 240<br>2. Prerequisites to running tests on page 207<br>3. Test run configurations on page 240 |

If you plan to run API Suites that use a transport and the transport requires third-party application `Jar` files for a successful run on a remote Docker host, then you must perform the following task:

- You must copy the third-party application `Jar` files to the third-party application folder on the computer where the remote Docker host is installed. See Copying third-party application Jars to a remote Docker host on page 229.

After you set up and register the remote Docker host with HCL OneTest™ Server, you can add the remote Docker hosts to your project on HCL OneTest™ Server. See Adding a remote Docker host to the project for running tests on page 234.

Related information

Tests configurations and test runs

Managing Docker hosts on page 224

## Setting up a remote Docker host computer

You must set up a computer to host the remote Docker host. You can set up the remote host with a non-secure mode of connection with HCL OneTest™ Server.

### Before you begin

You must have identified a remote computer or VM that has a minimum of 16 GB RAM. You must have installed Ubuntu V18.04 or later, on the remote computer.

You must have installed Docker Enterprise Edition for a non-secure mode of connection with HCL OneTest™ Server on the remote Docker host computer. Refer to Get Docker Engine.

You must have the entitlement key that is required for you to connect to the Harbor repository to be able to pull in the software image to the remote Docker host.

**About this task**

You must set up a remote computer or a Virtual Machine (VM) on which you want to set up the remote Docker host. You must be a system administrator or get the remote Docker host set up by a system administrator. You must ensure that the system administrator completes the following actions:

- Enables SSH.
- Creates user credentials for you with root permissions.
- Provides the name of the Docker container.
- Provides the IP address of the remote Docker host computer.

1. As a system administrator of the remote system, specify a port to access the Docker daemon in the `docker.service` file located in the directory `/lib/systemd/system`. You must add the configured port at the line in the file:

   ```
   ExecStart=/usr/bin/dockerd -H fd:// -H tcp://0.0.0.0:PORT
   ```

   For example, if the port configured is *4342*, then the line must read as:

   ```
   ExecStart=/usr/bin/dockerd -H fd:// -H tcp://0.0.0.0:4342
   ```
2. Restart the Docker service by using the following commands:

   $ sudo systemctl daemon-reload

   $ sudo systemctl restart docker
3. Perform the following steps as a user with root permissions or ask the system administrator to complete the following steps:

   a. Open a terminal and connect to the remote Docker host by using an SSH tool by running the following command:

      ssh root@<IP_address_of_remote_host>

   b. Log in to HCL Software's Harbor container registry  by running the following command:

      sudo docker login hclcr.io -u <username> -p <password>

      **Note:** You must provide the following username and password in the command to log in to the registry:

- Username as email address registered with HCL
- Password as Harbor CLI secret

c. Run the following commands to pull the product images from the registry on to the remote Docker host computer:

**Note:** You must pull in the images with the version which is the same as the version of HCL OneTest™ Server that you use to run the tests.

- sudo docker pull cp.hclcr.io/ot/hcl-onetest/studio<image_identifier>
- sudo docker pull hclcr.io/ot/hcl-onetest/virtualization<image_identifier>

### Results

You have set up the remote Docker host computer that can be accessed in a non-secure mode.

### What to do next

You can set up a secure remote Docker host computer optionally, that enables a secure connection with HCL OneTest™ Server. See Setting up a secure remote Docker host computer on page 227.

You can register the remote Docker host on HCL OneTest™ Server. See Registering a remote Docker host with HCL OneTest Server on page 231.

## Setting up a secure remote Docker host computer

You must set up a computer to host the remote Docker host. You can set up the remote host with a secure mode of connection with HCL OneTest™ Server by using certificates issued by the remote host computer that authenticate the connection.

### Before you begin

You must have identified a remote computer or VM that has a minimum of 16 GB RAM. You must have installed Ubuntu V18.04 or later, on the remote computer.

You must have installed Docker Community Edition V18.09 or later. For instructions, refer to Get Docker Engine.

**Note:** For the remote Docker host to be reached in a safe manner through the network, you can enable *TLS* by specifying the **tlsverify** flag and by pointing the **tlscacert** flag of the remote Docker host to a trusted CA certificate. Refer to Protect the Docker daemon socket.

You must have the entitlement key that is required for you to connect to the Harbor repository to be able to pull in the software image to the remote Docker host.

### About this task

You must set up a remote computer or a Virtual Machine (VM) on which you want to set up the remote Docker host. You must be a system administrator or get the remote Docker host set up by a system administrator. You must ensure that the system administrator completes the following actions:

- Enables SSH.
- Creates user credentials for you with root permissions.
- Provides the name of the Docker container.
- Provides the IP address of the remote Docker host computer.

Perform the following steps as a user with root permissions or ask the system administrator to complete the following steps:

a. Open a terminal and connect to the remote Docker host by using an SSH tool by running the following command:

   ```
   ssh root@<IP_address_of_remote_host>
   ```

b. Log in to HCL Software's Harbor container registry  by running the following command:

   ```
   sudo docker login hclcr.io -u <username> -p <password>
   ```

   **Note:** You must provide the following username and password in the command to log in to the registry:

   - Username as email address registered with HCL
   - Password as Harbor CLI secret

c. Run the following commands to pull the product images from the registry on to the remote Docker host computer:

   **Note:** You must pull in the images with the version which is the same as the version of HCL OneTest™ Server that you use to run the tests.

   - sudo docker pull cp.hclcr.io/ot/hcl-onetest/studio<image_identifier>
   - sudo docker pull hclcr.io/ot/hcl-onetest/virtualization<image_identifier>

### Results

You have set up the remote Docker host computer that can be accessed in a secure mode.

### What to do next

You can optionally, set up a non-secure remote Docker host. See Setting up a remote Docker host computer on page 225.

You can register the remote Docker host on HCL OneTest™ Server. See Registering a remote Docker host with HCL OneTest Server on page 231.

Related reference

Task flow for working with remote Docker hosts

## Copying third-party application Jars to a remote Docker host

You can run API Suites in a project on HCL OneTest™ Server on a remote Docker host. If the API Suites use a transport and the transport requires third-party `Jar` files for a successful run, you must ensure that the third-party application `Jar` files are available at the test run time. To achieve this, you must copy the third-party application `Jar` files to the computer where you have set up the remote Docker host.

### Before you begin

If you want to copy the third-party application `Jar` files to the to the computer where HCL OneTest™ Server is installed on Kubernetes, see Copying third-party application Jars to Kubernetes on page 83.

You must have server administrator privileges.

You must have completed the following tasks:

- Set up and configured the remote Docker host. See Managing Docker hosts on page 224.
- Identified the third-party application `Jar` files that are required and copied the `Jar` files. See Test run considerations for API Suites on page 208.
- Copied the `Jar` files of the third-party application jars to the directory or folder on the remote Docker host from where you can run the docker commands.

### About this task

You can copy the required third-party application `Jar` files to the folder that is specific for the application under the `/myFiles/Userlibs/<application_name>` folder. You need not extract the files.

You can perform this task any time after you have configured the remote Docker host and you plan to run an API Suite on the remote Docker host. The API Suite uses transports and the transports require third-party application JAR files to be available at the test run time.

You can get help on the docker commands by running the command: $ docker help from the docker command line. For more information about the docker commands, refer to the Docker command line documentation.

1. Use the following table to find the name of the folder that corresponds to the specific third-party application for the transport used in the API Suite.

   **Note:** You must provide the name of the folder listed for the third-party application as the `<application_name>` in the docker command.

   **Table 4. Name of the folder for the application**

   | Application | Name of the folder to use |
   |---|---|
   | Camel | Camel |
   | CentraSite | CentraSite |
   | CICS | CICS |
   | Coherence | Coherence |
   | Database | JDBC |
   | IMS | IMS |
   | Integra | Integra |
   | JMS | JMS |
   | SAP RFC | SAP |
   | Software AG Universal Messaging | SoftwareAGUM |
   | TIBCO EMS | TIBCO |
   | TIBCO Rendezvous | |
   | TIBCO SmartSockets | |
   | WebSphere Application Server Service Integration Bus (SiBus) | WAS |
   | WebLogic | WebLogicJMX |
   | Software AG webMethods | webMethods |
   | WebSphere MQ | WMQ |

2. Run the following docker command to create or update the volume that contains the application `Jar` files:

   ```
   docker run --rm -v /<anyFolder>/UserLibs/<application_name>:/ulsrc -v userlibs/<application_name>:/uldest
   alpine:latest cp -r /ulsrc/. /uldest/UserLibs/<application_name>
   ```

   **Attention:** You must run the docker command when no other tests are running in the Docker host to prevent concurrent access problems.

## Results

You have successfully copied the third-party application `Jar` files to the folder in the remote Docker host.

**What to do next**

- You can register the remote Docker host, if you have not already done so. See Registering a remote Docker host with HCL OneTest Server on page 231.
- You can configure the API Suite run. See Configuring an API Suite run on page 249.

## Registering a remote Docker host with HCL OneTest™ Server

You must register the remote Docker host with HCL OneTest™ Server before you add the remote Docker host to your project, which contains the test assets that you want to run on the remote Docker host.

**Before you begin**

Depending on the mode of connection with HCL OneTest™ Server, you must have set up the remote Docker host computer either for a secured mode or for an unsecured mode of connection. See Setting up a remote Docker host computer on page 225 or Setting up a secure remote Docker host computer on page 227.

1. Log in to HCL OneTest™ Server.
2. Click **Infrastructure > Agents and Intercepts** in the navigation pane.

   The **Agents and Intercepts** page is displayed.

   You can view the agents, intercepts or Dockers that are registered with HCL OneTest™ Server.
3. Click **New docker host**.

   The **New docker host** dialog box is displayed.
4. Enter the *host name* and the *port* of the remote Docker host in the format *<host name or IP_address>:<port>* in the **Remote Docker Host** field.

   If you do not provide the port that you configured for the remote Docker host, then the default port that is used is as follows:
   - Port *2375* for the unsecured mode
   - Port *2376* for the secured mode
5. Choose from any of the following authentication modes to establish the connection between HCL OneTest™ Server and the remote Docker host computer:

   - For the secured mode, which is the default mode and the checkbox is displayed as selected.

     Perform the following steps:
     a. Select **Secure mode**, if it is not selected.
     b. Click the action labels to browse and select the `.pem` files that correspond to `ca_certificate.pem`, `client_certificate.pem`, and `client_key.pem`.

**Note:** You must have generated the `.pem` files on the remote Docker host computer and copied the files to your local drive.

- For the unsecured mode, the **Secure mode** checkbox must not be selected.

  Perform the following step:

  a. Clear the **Secure mode** checkbox, if selected.

6. Click **Test connection** to test if a connection is established between HCL OneTest™ Server and the remote host computer.

   Any of the following events occur when **Test connection** is clicked:

   - On a successful connection, a message is displayed.
   - On a failure to connect to the remote host computer, an error message is displayed. You must resolve the error and reattempt to establish a successful connection.

   **Important:** The remote Docker host computer must be connected successfully before you register the remote Docker host with HCL OneTest™ Server.

7. Click **Register**.

   The remote Docker host that you registered is displayed.

   **Important:** The remote Docker host must be registered successfully with HCL OneTest™ Server before you add the remote Docker host to your project.

8. Click **Close** to exit.

### Results

You have registered the remote Docker host with HCL OneTest™ Server.

### What to do next

You can view the Docker hosts that are registered with HCL OneTest™ Server. See Viewing remote Docker hosts that are registered with HCL OneTest Server on page 233.

You must add the registered remote Docker host to your project before you use the remote Docker host as a location to run tests in your project. See Adding a remote Docker host to the project for running tests on page 234.

Related reference

Task flow for working with remote Docker hosts

## Viewing remote Docker hosts that are registered with HCL OneTest™ Server

You must register the remote Docker host with HCL OneTest™ Server before you add the remote Docker host to your project, which contains the test assets that you want to run on the remote Docker host.

### Before you begin

Depending on the mode of connection with HCL OneTest™ Server, you must have set up the remote Docker host computer either for a secured mode or for an unsecured mode of connection. See Setting up a remote Docker host computer on page 225 or Setting up a secure remote Docker host computer on page 227.

1. Log in to HCL OneTest™ Server.
2. Click **Infrastructure > Agents and Intercepts** in the navigation pane.

   The **Agents and Intercepts** page is displayed.

   You can view the agents, intercepts or Dockers that are registered with HCL OneTest™ Server.
3. You can view the Dockers that you own in any of the following ways:
   - Search for the Docker by entering the name of the Docker host in the **Search** field.

     **Note:** You can enter either the full name or any text that is in the name. The search is enabled for case sensitive text that you can enter.

   - Sort the **Type** column to sort the items and then identify the Docker by the name displayed.
   - Sort the **Agents** column to sort the items and identify the Docker by the owner.

   You can view the following details about the Dockers that are registered with HCL OneTest™ Server:
   - The projects to which the Docker host is added, are displayed in the **Projects** column.
   - The status of the Docker host is displayed in the **Status** column.

     **Note:** You can add Docker hosts that are in the `Ready` state to your project, and then use them as locations to run tests.

### Results

You have viewed the remote Docker hosts that are registered with HCL OneTest™ Server.

### What to do next

You must add the registered remote Docker host to your project before you use the remote Docker host as a location to run tests in your project. See Adding a remote Docker host to the project for running tests on page 234.

## Adding a remote Docker host to the project for running tests

You can choose to run tests on Docker hosts that you have set up on remote host computers. You must register the Docker host with HCL OneTest™ Server and then add them to your project before you run tests on the remote Docker hosts.

**Before you begin**

You must have completed the following tasks:

- Read about the considerations you must take into account before you configure a test run to run on a remote Docker host. See Test run considerations for running tests on remote Docker hosts on page 224.
- Set up the remote Docker host system. See Setting up a remote Docker host computer on page 225 or Setting up a secure remote Docker host computer on page 227.
- Registered the remote Docker host with HCL OneTest™ Server. See Registering a remote Docker host with HCL OneTest Server on page 231.

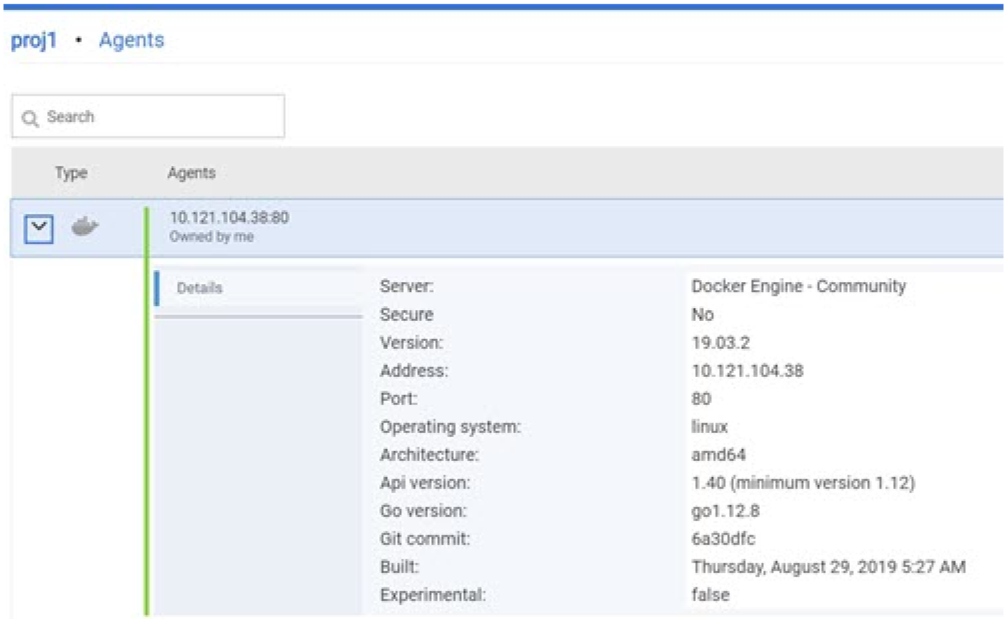
1. Log in to HCL OneTest™ Server.
2. Open your project by clicking **Projects > My Projects > *project_name***.
   The **Overview** page is displayed.
3. Click **Manage > Infrastructure**.

   The **Infrastructure** page is displayed.
4. Click **Add > Add docker**.
5. Select the Docker host that you want to add to the project from the list of Docker hosts, and then click **Add**.

   **Note:** You can add the Docker hosts that you have registered. You can add any number of Docker hosts to your project.

The Docker hosts that you added to the project are displayed.

You can view the details of the Docker host by clicking the **Expand** icon >.



10.121.104.38:80
Owned by me

Details

| | |
|---|---|
| Server: | Docker Engine - Community |
| Secure | No |
| Version: | 19.03.2 |
| Address: | 10.121.104.38 |
| Port: | 80 |
| Operating system: | linux |
| Architecture: | amd64 |
| Api version: | 1.40 (minimum version 1.12) |
| Go version: | go1.12.8 |
| Git commit: | 6a30dfc |
| Built: | Thursday, August 29, 2019 5:27 AM |
| Experimental: | false |

### What to do next

You can select any of the remote Docker hosts as an alternate location to run the test asset in your project while configuring a test run from the **Execution** page.

Related reference

Task flow for working with remote Docker hosts

Related information

Tests configurations and test runs

## Editing configurations of a remote Docker host

You can edit the host name or the port of the registered Docker host computer instead of registering it as a new remote host if any of the parameters are changed on the remote host computer. You can also change the mode of authentication of a registered remote Docker host.

### Before you begin

You must have completed the following tasks:

- Set up the remote Docker host computer. See Setting up a remote Docker host computer on page 225 or Setting up a secure remote Docker host computer on page 227.
- Registered the remote Docker host with HCL OneTest™ Server. See Registering a remote Docker host with HCL OneTest Server on page 231.

1. Log in to HCL OneTest™ Server.

   The **My Projects** page is displayed.
2. Click **Infrastructure > Agents and Intercepts** in the navigation pane.

   The **Agents and Intercepts** page is displayed.

   You can view the agents, intercepts or Dockers that are registered with HCL OneTest™ Server.
3. You can view the Dockers that you own in any of the following ways:
   - Search for the Docker by entering the name of the Docker host in the **Search** field.

     **Note:** You can enter either the full name or any text that is in the name. The search is enabled for case sensitive text that you can enter.
   - Sort the **Type** column to sort the items and then identify the Docker by the name displayed.
   - Sort the **Agents** column to sort the items and identify the Docker by the owner.

   You can view the following details about the Dockers that are registered with HCL OneTest™ Server:
   - The projects to which the Docker host is added, are displayed in the **Projects** column.
   - The status of the Docker host is displayed in the **Status** column.
4. Identify the Docker host that you want to modify the configuration, and then click the **Edit** icon ✎.

   **Note:** The ✎ icon is displayed in the **Actions** column in the row of the Docker only if you own the remote Docker host.

   The **Update docker host** dialog box is displayed.
5. Change any of the configurations.

| To change... | Do this... |
|---|---|
| Change the host name or the port of the remote Docker host. | Enter the changed value of the host name or the port in the **Remote hostname** field. |
| Change the mode of authentication from a non-secure mode to a secure mode. | a. Select the **Secure mode** option.<br>b. Click the action labels to browse and select the `.pem` files that correspond to `ca_certificate.pem`, `client_certificate.pem`, and `client_key.pem`. |

| To change... | Do this... |
|---|---|
| | **Note:** You must have generated the `.pem` files on the remote host computer and copied the files to your local drive. |
| Change the mode of authentication from a secure mode to a non-secure mode. | Clear the **Secure mode** option that is selected. |

6. Click **Test connection** to test whether a connection is established between HCL OneTest™ Server and the remote host computer:
    - On a successful connection, a message is displayed.
    - On a failure to connect to the remote host computer, an error message is displayed. You must resolve the error and reattempt to establish a successful connection.
7. Click **Update**.

    The remote Docker host with the updated details is displayed.

#### Results

You have updated the registered remote Docker host with the changed parameters.

#### What to do next

You can now add the updated Docker host to your project.

Related reference

Task flow for working with remote Docker hosts

Related information

Setting up a remote Docker host computer on page 225

Registering a remote Docker host with HCL OneTest Server on page 231

Adding a remote Docker host to the project for running tests on page 234

Removing a remote Docker host from a project on page 237

Unregistering a remote Docker host from HCL OneTest Server on page 239

## Removing a remote Docker host from a project

You can delete a remote Docker host that you added to your project in HCL OneTest™ Server when you no longer need it to run tests.

### Before you begin

You must have completed the following tasks:

1. Set up the remote Docker host computer. See Setting up a remote Docker host computer on page 225 or Setting up a secure remote Docker host computer on page 227.
2. Registered the remote Docker host with HCL OneTest™ Server. See Registering a remote Docker host with HCL OneTest Server on page 231.
3. Added the registered Docker host to a project. See Adding a remote Docker host to the project for running tests on page 234.

1. Log in to HCL OneTest™ Server.

   The **My Projects** page is displayed.
2. Click **Manage > Infrastructure** in the navigation pane.

   The **Infrastructure** page is displayed.

   You can view the agents, intercepts or Dockers that are added to your project.
3. You can view the Dockers that you own in any of the following ways:

   - Search for the Docker by entering the name of the Docker host in the **Search** field.

     **Note:** You can enter either the full name or any text that is in the name. The search is enabled for case sensitive text that you can enter.

   - Sort the **Type** column to sort the items and then identify the Docker by the name displayed.
   - Sort the **Agents** column to sort the items and identify the Docker by the owner.

   You can view the following details about the Dockers that are registered with HCL OneTest™ Server:
   - The projects to which the Docker host is added, are displayed in the **Projects** column.
   - The status of the Docker host is displayed in the **Status** column.
4. Identify the Docker host that you want to delete, and then click the **Remove docker host** icon 🗑.

   **Note:** The 🗑 icon is displayed in the **Actions** column in the row of the Docker only if you own the remote Docker host.

   A message is displayed that the Docker host is removed from the project successfully.

### Results

You have deleted the remote Docker host from your project. The remote Docker host is no longer available as a location while configuring a test run in your project.

### What to do next

You might have to add a remote Docker host to your project to use the remote Docker host as a location to run tests.

Related reference

Task flow for working with remote Docker hosts

Related information

Unregistering a remote Docker host from HCL OneTest Server on page 239

Editing configurations of a remote Docker host on page 235

## Unregistering a remote Docker host from HCL OneTest™ Server

You can unregister a remote Docker host that is registered with HCL OneTest™ Server when you no longer require it.

### Before you begin

You must have completed the following tasks:

1. Set up the remote Docker host computer. See Setting up a remote Docker host computer on page 225 or Setting up a secure remote Docker host computer on page 227.
2. Registered the remote Docker host with HCL OneTest™ Server. See Registering a remote Docker host with HCL OneTest Server on page 231.

1. Log in to HCL OneTest™ Server.
2. Click **Infrastructure > Agents and Intercepts** in the navigation pane.

   The **Agents and Intercepts** page is displayed.

   You can view the agents, intercepts or Dockers that are registered with HCL OneTest™ Server.
3. Identify the Docker host that you want to unregister.

   **Note:** The ✏ and 🗑 icons are displayed in the **Actions** column in the row of the Docker only if you own the remote Docker host.

4. Unregister the Docker host by completing the following steps:
   a. Click the **Unregister docker host** icon 🗑.
   b. Click **Unregister** in the **Unregister docker host** dialog box.

   A message is displayed that the Docker host is unregistered successfully.

### Results

You have unregistered the remote Docker host from HCL OneTest™ Server. You cannot run tests on this remote Docker host if it is added to your project. You might have to register it again with HCL OneTest™ Server, if you want to use the remote Docker host in your projects.

Related reference

Task flow for working with remote Docker hosts

Related information

Editing configurations of a remote Docker host on page 235

Adding a remote Docker host to the project for running tests on page 234

Removing a remote Docker host from a project on page 237

# Test run configurations

You can configure and run tests in HCL OneTest™ Server after you add the test resources to your project.

Before you configure a test run, you must have completed the following tasks:

- Created a project on HCL OneTest™ Server or you must have been granted access to a project with the *Tester* role assigned. See Test assets and a server project on page 496.
- Created tests in the desktop clients and committed the test assets and test resources to a remote repository. You must have added the remote repository to the project.
- Read the considerations you must take into account for certain test types. See Prerequisites to running tests on page 207.

You must be a project *Owner* or a member with the *Tester* role assigned to configure and run a test.

You can find information about how to configure and run the following types of tests:

## Configuring a test for a quick run

You can configure any type of test to be run on HCL OneTest™ Server when you want to quickly ensure that the test runs correctly. You might not want to set the different options for the test nor want to schedule the run.

**Before you begin**

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to server projects on page 504.
- Read and completed the tasks mentioned in Prerequisites to running tests on page 207, if they apply to the test that you want to configure for a run.

1. Open the project that contains the test assets or resources that you added from the Git repository, and then click **Execution**.
2. Select the branch of the repository that contains the test assets or resources that you want to run.

All test assets in the selected branch are displayed on the **Execution** page.

3. Identify and select the test asset or resource that you want to run from the test assets listed.
4. Click the **Execute** icon ▶ in the row of the identified test asset.

   The **Execute test asset** dialog box is displayed.
5. Select the version of the test resources that you want to run, if you want to run a different version other than the latest version.
6. Select **Now** to initiate the test run immediately after you click **Execute**.
7. Enter a label for the test run that helps you to identify the test on the **Results** page.

   After the test run completes, the label that you entered is displayed for the test under the **Labels** column on the **Results** page.
8. Click **Execute**.
   The test run is initiated.

**Results**
You have started a test run.

**What to do next**

You can choose to perform any of the following tasks:

- View the progress of the test from the **Progress** page. See Viewing the progress of running test assets on page 312.
- View the results, reports, and logs of the test from the **Results** page after the test completes the run. See Test results on page 359.

Related information

Resetting the configuration settings for a test run on page 315

Viewing the progress of running test assets on page 312

Monitoring a test run

Stopping a test run on page 317

Canceling a scheduled test run on page 318

## Configuring an AFT Suite run

After you added the test resources that you created in the desktop client to the project, you can configure an AFT Suite to be run on HCL OneTest™ Server.

**Before you begin**

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to server projects on page 504.
- You must have read Test run considerations for AFT Suites on page 207, if you want to configure a run for an AFT Suite that has the agent location configured in the `AFT XML` file.

1. Open the project that contains the test assets or resources that you added from the Git repository, and then click **Execution**.
2. Select the branch of the repository that contains the test assets or resources that you want to run.

   All test assets in the selected branch are displayed on the **Execution** page.
3. Identify the test asset or resource that you want to run by performing any of the following steps:

   a. Identify the test asset or resource by scrolling through the list.

   **Note:** You can also identify the type of the asset from the icon that represents the test type as shown in the following table:

   | Icon | Represents the test asset or resource |
   |---|---|
   | 🌐 | AFT Suite |
   | ⇹ | API Suite |
   | ▦ | Compound Test |
   | JMeter | JMeter Test |
   | Ju | JUnit Test |
   | ⊘ | Postman resources |
   | 📶 | Rate Schedule |
   | 📶 | VU Schedule |

   b. Search for the test asset or resource by entering any text contained in the test asset or resource name in the **Search** text box.

   c. Create a filter query by using the **New filter** option by performing the following steps:

      i. Click **New filter**.

      ii. Select an operator, and add a rule, or a group of rules.

      iii. Add or enter the relevant parameters and either select or enter the condition and the criteria for the condition.

      You can select a parameter from the following list:

            - Type
            - Test Asset Name
            - Test Asset Path
            - Last Result
            - Next Run
            - Components

        iv. Apply the filter query to filter the assets based on the query.

            The test assets that match the filter criteria are displayed.

        v. Save the filter query by performing the following steps, if you want to reuse the filter query later:

            1. Click **Save**.
            2. Enter a name for the filter query.
            3. Click **Save**.

    d. Retrieve a saved filter, if you have saved filter queries earlier by performing the following steps:

        **Note:** To open the filter query, you must have created and saved a filter query.

        i. Click the **Open filters** icon ☰.
        ii. Select the saved filter in the **Filters** dialog box.
        iii. Click **Apply** to apply the filter.

            The test assets that match the filter criteria are displayed.

4. Click the **Execute** icon ▶ in the row of the identified test asset.

    The **Execute test asset** dialog box is displayed.

5. Select the version of the test resources that you want to run by completing any of the following actions:

    **Note:** The test resources in the version can contain the test assets, datasets, `AFT XML` files, API environment tags, and other resources specific to projects created from any of the desktop clients.

    - Expand the list in the **Version** field, find the version of the test resources, and then select the version.

        Use the following details about the version of the test resources that are displayed to identify the version that you want:

        - Commit message.
        - Tags labeled by the user for the version committed.
        - The user who committed the version to the repository.
        - Relative time of the commit. For example, *2 hours ago* or *3 days ago*.

The list displays the versions of the test resources committed by all users to the branch in the repository. The versions are arranged with the latest version that is committed, and then followed by the versions committed previously.

- Expand the list in the **Version** field, and then search for the version that you want to select by entering a partial or the complete commit message of that version.

  The version that matches the search criteria is displayed and it is selected for the test run.

The default value for the version selected for the run is the latest version in the selected branch in the repository. If you do not select any version, then the latest version is selected for the test run.

**Notes:**

- If you selected a version but you do not want to use that version in the test run, you can remove the selected version by clicking the ✖ icon, then the default version is selected for the test run.
- If you repeated a test or ran the test again from the **Results** page, then the version of the test resources that you had selected for the earlier run is shown as selected. You can either retain this version or select any other version from the list. You can also remove the previous version by clicking the ✖ icon.

6. Select the time for scheduling the test run from the following options:

   - Select **Now** to initiate the test run immediately after you click **Execute**.

     **Important:** Click **Execute** only after you have configured the other settings in this dialog box.

   - Select **Later** and configure the date and time for scheduling a test to run at the scheduled date and time.

   The default time for scheduling a run is **Now**.

   **Notes:**

   - If you have configured some or all of the settings for the current test run, and you do not want to continue with those settings, you can reset the settings by clicking **Reset**.

   - If you want to repeat a test run and do not want to use the saved settings from a previous run, you can reset all the saved settings to their default values by clicking **Reset**.

7. Enter a label for the test run that helps you to identify the test on the **Results** page.

   After the test run completes, the label that you entered is displayed for the test under the **Labels** column on the **Results** page. After you have created a label, any member of the project can use that label.

The default value for the **Label** field is null or an empty field.

**Important:** The configuration that you set for the test run in the **Execute test asset** dialog box is preserved when you run the same test again. Those changes are not visible when another user logs in to HCL OneTest™ Server. For example, if you created new variables on the server, those variables are available only for you when the same test is run again.

If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

If you are running an AFT Suite and the following conditions are true, then you must perform the next step:

- You do not want to run the test on the agent configured in the `AFT XML` file.
- You have not selected the agent from the **Override** column in the **Location** tab.

8. Click **Advanced** to make the following advanced configurations:

    a. Add the following setting in the **Program Arguments** field:

    ```
    -swaplocation <configured_agent_location>:<overriding_agent_location>
    ```

    For example, if the configured agent location is *91.2.352.24* and the remote agent where you want to run the test is *100.35.117.164*, then the entry in the **Program Arguments** field is as follows:

    ```
    -swaplocation 91.2.352.24:100.35.117.164
    ```

    If the test that you want to configure supports Jaeger tracing, then do the required tasks in the following scenarios:

| If... | Then... |
|---|---|
| You want to obtain the test results as a Jaeger trace. | Enter `-history jaeger` in the **Program Arguments** field. |
| Jaeger is already set as a program argument and you want to remove the Jaeger trace for your test. | Delete the `-history jaeger` entry in the **Program Arguments** field. |

| If... | Then... |
|---|---|
| You want the report as a test log and as a Jaeger trace. | Enter `-history jaeger,testlog` in the **Program Arguments** field. |

**Note:** The default report format is the *test log* format for the test reports.

**Note:** You must separate the arguments or variables with a white space when you enter them in the same line or start each argument or variable on a new line.

The default value for each of the fields for the advanced settings is null or an empty field.

If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

9. Follow the instructions if you are running a test asset that contains datasets:

    a. Click the **DATA SOURCES** tab, if it is not already open.

    b. Consider the following information about datasets before you select a dataset:

    The default value for the datasets in the **DATA SOURCES** tab is null if the test asset did not have an associated dataset. If the asset had an associated dataset, the default value is the associated dataset.

    You can utilize the dataset stored as an Excel or CSV file to override the original dataset associated with the Suite, test, or schedule. For example, when you have associated a dataset in `.xlsx`, `.xls`, or `.csv` format with the test or schedule in desktop clients and if you have another set of data stored in an Excel or CSV file, then you can select that dataset from the **Override** list. If you want to run a test or schedule by using the schema created from the **Data Fabrication** page, see related links.

    **Remember:** You must have uploaded the dataset as an Excel or CSV file into the Git repository, and ensured that both the original dataset (from the test asset) and new datasets (added to the project) have the same column names.

    c. Select the dataset that you want to use in the test run from any of the following options:

    - Select the dataset that is displayed as the default dataset when the test asset contains a single dataset.

      **Note:** If there is only one dataset in the test asset, then that dataset is displayed as the default dataset.

    - Select the dataset from the list.

> **Note:** If there are multiple datasets in the test asset, the datasets are listed in their increasing alphabetical order.

- Select the dataset from the **Override** list to override the dataset that was associated with the test in the desktop client.

  > **Important:** If the test contains an encrypted dataset, the Project Owner must classify it in the **DATA SECURITY** tab on the Project page before you can select it. You must have added datasets to your project from the **Dataset** page for the datasets to be displayed in the **Override** list.

If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

10. Follow the instructions if the test requires a variable that must be passed to the test at the test run time.

    a. Click the **VARIABLES** tab, if it is not already open.

    b. Choose one of the following methods to add the variables:

    - To add new variables manually, click the **Add Variable** icon ⊕, enter the name, and value of the variable.
    - To add new variables from your local computer or from the Git repository that is associated with your server project, click the **Upload** icon ⬆ and select the **Upload from local system** or **Browse from server** to select the variable file.

      > **Note:** You must have created a file with the variables before you can select the file.

    The default value for the variables is null or an empty field.

    If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

11. Follow the instructions if you are running a test that has static agents configured:

    a. Click the **LOCATION** tab, if it is not already open.

    The static agents that are configured in the test asset are listed under the **Host** column. The information about the availability of the agent is displayed.

**Note:** You must have added agents to your project from the **Infrastructure** page for the agents to be displayed under the **Override** column.

b. Select the agent where you want to run the test asset.

You can select the same agent that is configured in the test asset. Alternatively, you can override the agent with any other agent added to the project by selecting it from the list in the **Override** column.

The default value for the agents is null or an empty field if no agents were configured in the test asset. If the test asset contains agents that are configured, then the default agent is the first item to be displayed on the list of agents listed in the increasing alphabetical order.

If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

12. Follow the instructions if you want to change the location for running the test:

a. Click the **LOCATION** tab, if it is not already open.

The **Default Cluster** is the default location where the test runs, and it is listed under the **Host** column. The information about the availability of the default location is displayed.

**Important:** You must have added remote Docker hosts that are registered with HCL OneTest™ Server to your project from the **Infrastructure** page. The remote Docker hosts are then displayed under the **Override** column.

**Notes:**

- If remote Docker hosts are not added to your project, the option **No override options** is displayed as the default value and the test runs in the Kubernetes cluster of HCL OneTest™ Server.
- If remote Docker hosts are added to your project, the added Docker hosts are displayed along with their availability status and ownership information.

b. Select the location where you want to run the test asset from the following options:

- Select the **Default Cluster** when no remote Docker hosts are available in your project.
- Select the remote Docker host from the list when a remote Docker host is available in your project.
- Select **No override options**, if you selected any remote Docker host and want to revert to the **Default Cluster** to run the test asset.

If you want to run the test immediately or at the scheduled time, click **Execute**.

13. Click **Execute**.
The test run is initiated.

**Results**

You have configured and either started or scheduled a test run of an AFT Suite.

**What to do next**

You can choose to perform any of the following tasks after you have initiated or scheduled a run:

- Stop the test run at any point after the test run is initiated, from the **Execution** page. See Stopping a test run on page 317.
- Cancel a scheduled test run, from the **Execution** page. See Canceling a scheduled test run on page 318.
- View the progress of the test from the **Progress** page. See Viewing the progress of running test assets on page 312.
- Monitor the test from the **Progress** page. See Monitoring a test run.
- View the results, reports, and logs of the test from the **Results** page after the test completes the run. See Test results on page 359.

Related information

Resetting the configuration settings for a test run on page 315

Tests configurations and test runs

Test run configurations on page 240

## Configuring an API Suite run

After you added the test resources that you created in the desktop client to the project, you can configure an API Suite to be run on HCL OneTest™ Server.

**Before you begin**

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to server projects on page 504.
- You must have completed the following tasks if you are running API Suites that use a transport and the transport requires third-party application `Jar` files for a successful run:
  - Read Test run considerations for API Suites on page 208.
  - Copied the third-party application `Jar` files to the third-party application folder on the computer where HCL OneTest™ Server is installed, if you are running the API Suite in Kubernetes. See Copying third-party application Jars to Kubernetes on page 83.
  - Copied the third-party application `Jar` files to the third-party application folder on the computer where the remote Docker host is installed, if you are running the API Suite in a remote Docker host. See Copying third-party application Jars to a remote Docker host on page 229.

1. Open the project that contains the test assets or resources that you added from the Git repository, and then click **Execution**.
2. Select the branch of the repository that contains the test assets or resources that you want to run.

   All test assets in the selected branch are displayed on the **Execution** page.
3. Identify the test asset or resource that you want to run by performing any of the following steps:

   a. Identify the test asset or resource by scrolling through the list.

      **Note:** You can also identify the type of the asset from the icon that represents the test type as shown in the following table:

      | Icon | Represents the test asset or resource |
      |---|---|
      | 🌐 | AFT Suite |
      | ⇔ | API Suite |
      | ▦ | Compound Test |
      | JMeter | JMeter Test |
      | Ju | JUnit Test |
      | ⊘ | Postman resources |
      | 📶 | Rate Schedule |
      | 📶 | VU Schedule |

   b. Search for the test asset or resource by entering any text contained in the test asset or resource name in the **Search** text box.

   c. Create a filter query by using the **New filter** option by performing the following steps:

      i. Click **New filter**.

      ii. Select an operator, and add a rule, or a group of rules.

      iii. Add or enter the relevant parameters and either select or enter the condition and the criteria for the condition.

         You can select a parameter from the following list:

         - Type
         - Test Asset Name
         - Test Asset Path
         - Last Result

        - Next Run
        - Components
    iv. Apply the filter query to filter the assets based on the query.

        The test assets that match the filter criteria are displayed.

    v. Save the filter query by performing the following steps, if you want to reuse the filter query later:

        1. Click **Save**.
        2. Enter a name for the filter query.
        3. Click **Save**.

   d. Retrieve a saved filter, if you have saved filter queries earlier by performing the following steps:

    **Note:** To open the filter query, you must have created and saved a filter query.

    i. Click the **Open filters** icon ☰.
    ii. Select the saved filter in the **Filters** dialog box.
    iii. Click **Apply** to apply the filter.

        The test assets that match the filter criteria are displayed.

4. Click the **Execute** icon ▶ in the row of the identified test asset.

   The **Execute test asset** dialog box is displayed.

5. Select the version of the test resources that you want to run by completing any of the following actions:

   **Note:** The test resources in the version can contain the test assets, datasets, `AFT XML` files, API environment tags, and other resources specific to projects created from any of the desktop clients.

   - Expand the list in the **Version** field, find the version of the test resources, and then select the version.

     Use the following details about the version of the test resources that are displayed to identify the version that you want:
       - Commit message.
       - Tags labeled by the user for the version committed.
       - The user who committed the version to the repository.
       - Relative time of the commit. For example, *2 hours ago* or *3 days ago*.

     The list displays the versions of the test resources committed by all users to the branch in the repository. The versions are arranged with the latest version that is committed, and then followed by the versions committed previously.

  - Expand the list in the **Version** field, and then search for the version that you want to select by entering a partial or the complete commit message of that version.

    The version that matches the search criteria is displayed and it is selected for the test run.

  The default value for the version selected for the run is the latest version in the selected branch in the repository. If you do not select any version, then the latest version is selected for the test run.

  **Notes:**

  - If you selected a version but you do not want to use that version in the test run, you can remove the selected version by clicking the ✖ icon, then the default version is selected for the test run.
  - If you repeated a test or ran the test again from the **Results** page, then the version of the test resources that you had selected for the earlier run is shown as selected. You can either retain this version or select any other version from the list. You can also remove the previous version by clicking the ✖ icon.

6. Select the time for scheduling the test run from the following options:

  - Select **Now** to initiate the test run immediately after you click **Execute**.

    **Important:** Click **Execute** only after you have configured the other settings in this dialog box.

  - Select **Later** and configure the date and time for scheduling a test to run at the scheduled date and time.

  The default time for scheduling a run is **Now**.

  **Notes:**

  - If you have configured some or all of the settings for the current test run, and you do not want to continue with those settings, you can reset the settings by clicking **Reset**.

  - If you want to repeat a test run and do not want to use the saved settings from a previous run, you can reset all the saved settings to their default values by clicking **Reset**.

7. Enter a label for the test run that helps you to identify the test on the **Results** page.

  After the test run completes, the label that you entered is displayed for the test under the **Labels** column on the **Results** page. After you have created a label, any member of the project can use that label.

  The default value for the **Label** field is null or an empty field.

**Important:** The configuration that you set for the test run in the **Execute test asset** dialog box is preserved when you run the same test again. Those changes are not visible when another user logs in to HCL OneTest™ Server. For example, if you created new variables on the server, those variables are available only for you when the same test is run again.

If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

8. Click **Advanced** to make the following advanced configurations:

    a. Enter any JVM arguments that must be passed to the test run at run time in the **JVM Arguments** field, if applicable for the test.

    For example, you can set a maximum Java heap size.

    b. Enter program arguments that must be passed to the test run at run time in the **Program Arguments**, if applicable for the test.

    If the test that you want to configure supports Jaeger tracing, then do the required tasks in the following scenarios:

| If... | Then... |
|---|---|
| You want to obtain the test results as a Jaeger trace. | Enter `-history jaeger` in the **Program Arguments** field. |
| Jaeger is already set as a program argument and you want to remove the Jaeger trace for your test. | Delete the `-history jaeger` entry in the **Program Arguments** field. |
| You want the report as a test log and as a Jaeger trace. | Enter `-history jaeger,testlog` in the **Program Arguments** field. |

    **Note:** The default report format is the *test log* format for the test reports.

    c. Enter the environment variables that must be passed to the test run at run time in the **Environment Variables** field, if applicable for the test.

    For example, enter the environment variables when the third-party libraries that are used in the test run refer to the environment variables for configuration.

**Note:** You must separate the arguments or variables with a white space when you enter them in the same line or start each argument or variable on a new line.

The default value for each of the fields for the advanced settings is null or an empty field.

If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

9. Follow the instructions if the API Suite has an environment or secrets configured:
    a. Click the **ENVIRONMENT** tab, if it is not already open.
    b. Select the API test environment from the list if there are multiple environments configured in the test asset.
    c. Select the secrets collection that contains the secrets to be used for the test run.

    **Notes:**

    - The test asset that was created in the desktop client and added to the Git repository must have the environments defined as part of the API test project.
    - If the test asset contained secrets, then you must create those secrets in secrets collections in the project.

    The default value for the environment is the environment configured in the test asset. The default value for secrets is null or an empty field.

    If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

10. Follow the instructions if you are running a test asset that contains datasets:

    a. Click the **DATA SOURCES** tab, if it is not already open.

    b. Consider the following information about datasets before you select a dataset:

    The default value for the datasets in the **DATA SOURCES** tab is null if the test asset did not have an associated dataset. If the asset had an associated dataset, the default value is the associated dataset.

    You can utilize the dataset stored as an Excel or CSV file to override the original dataset associated with the Suite, test, or schedule. For example, when you have associated a dataset in `.xlsx`, `.xls`, or `.csv` format with the test or schedule in desktop clients and if you have another set of data stored in an Excel or CSV file, then you can select that dataset from the **Override** list. If you want to run a test or schedule by using the schema created from the **Data Fabrication** page, see related links.

    **Remember:** You must have uploaded the dataset as an Excel or CSV file into the Git repository, and ensured that both the original dataset (from the test asset) and new datasets (added to the project) have the same column names.

    c. Select the dataset that you want to use in the test run from any of the following options:

- Select the dataset that is displayed as the default dataset when the test asset contains a single dataset.

  **Note:** If there is only one dataset in the test asset, then that dataset is displayed as the default dataset.

- Select the dataset from the list.

  **Note:** If there are multiple datasets in the test asset, the datasets are listed in their increasing alphabetical order.

- Select the dataset from the **Override** list to override the dataset that was associated with the test in the desktop client.

  **Important:** If the test contains an encrypted dataset, the Project Owner must classify it in the **DATA SECURITY** tab on the Project page before you can select it. You must have added datasets to your project from the **Dataset** page for the datasets to be displayed in the **Override** list.

If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

11. Follow the instructions if the test requires a variable that must be passed to the test at the test run time.

    a. Click the **VARIABLES** tab, if it is not already open.

    b. Choose one of the following methods to add the variables:

    - To add new variables manually, click the **Add Variable** icon ⊕, enter the name, and value of the variable.
    - To add new variables from your local computer or from the Git repository that is associated with your server project, click the **Upload** icon ⬆ and select the **Upload from local system** or **Browse from server** to select the variable file.

      **Note:** You must have created a file with the variables before you can select the file.

    The default value for the variables is null or an empty field.

    If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

12. Follow the instructions if you want to change the location for running the test:

a. Click the **LOCATION** tab, if it is not already open.

   The **Default Cluster** is the default location where the test runs, and it is listed under the **Host** column. The information about the availability of the default location is displayed.

   **Important:** You must have added remote Docker hosts that are registered with HCL OneTest™ Server to your project from the **Infrastructure** page. The remote Docker hosts are then displayed under the **Override** column.

   **Notes:**

   - If remote Docker hosts are not added to your project, the option **No override options** is displayed as the default value and the test runs in the Kubernetes cluster of HCL OneTest™ Server.
   - If remote Docker hosts are added to your project, the added Docker hosts are displayed along with their availability status and ownership information.

b. Select the location where you want to run the test asset from the following options:

   - Select the **Default Cluster** when no remote Docker hosts are available in your project.
   - Select the remote Docker host from the list when a remote Docker host is available in your project.
   - Select **No override options**, if you selected any remote Docker host and want to revert to the **Default Cluster** to run the test asset.

If you want to run the test immediately or at the scheduled time, click **Execute**.

13. Click **Execute**.
    The test run is initiated.

**Results**

You have configured and either started or scheduled a test run of an API Suite.

**What to do next**

You can choose to perform any of the following tasks after you have initiated or scheduled a run:

- Stop the test run at any point after the test run is initiated, from the **Execution** page. See Stopping a test run on page 317.
- Cancel a scheduled test run, from the **Execution** page. See Canceling a scheduled test run on page 318.
- View the progress of the test from the **Progress** page. See Viewing the progress of running test assets on page 312.
- View the results, reports, and logs of the test from the **Results** page after the test completes the run. See Test results on page 359.

Related information

Resetting the configuration settings for a test run on page 315

Tests configurations and test runs

Test run configurations on page 240

## Configuring a run of a Compound Test that contains HTML tests

After you added the test resources that you created in the desktop client to the project, you can configure a Compound Test that contains HTML tests to be run on HCL OneTest™ Server.

**Before you begin**

You must have completed the following tasks:

- Been assigned the *Owner* or *Tester* role in the project to configure and run tests.
- Authored a Compound Test in HCL OneTest™ UI and must have completed the following tasks when you created the test assets:
  - Created a Compound Test under a Web UI project in your workspace.
  - Created a Functional HTML Test under a Compound Test in a Functional test project in the same workspace that contained the Web UI project.
  - Used the **startBrowser(Firefox,<URL>)** attribute with the browser value set to *Firefox* in the Functional test script.

    **Restriction:** Functional test scripts that contain the **startApp()** or **callScript()** attributes are treated as non-HTML scripts and are not supported to be used in test scripts in the HTML tests.
- Ran the Compound Test that contains the HTML tests on HCL OneTest™ UI.
- Committed the projects that include the Web UI project (with the Compound Test) and the Functional test (with the HTML tests in a Compound Test) to the remote repository.

1. Open the project that contains the test assets or resources that you added from the Git repository, and then click **Execution**.
2. Select the branch of the repository that contains the test assets or resources that you want to run.

   All test assets in the selected branch are displayed on the **Execution** page.
3. Identify the test asset or resource that you want to run by performing any of the following steps:

   a. Identify the test asset or resource by scrolling through the list.

      **Note:** You can also identify the type of the asset from the icon that represents the test type as shown in the following table:

| Icon | Represents the test asset or resource |
|---|---|
| 🌐 | AFT Suite |
| ⇄ | API Suite |
| ⊞ | Compound Test |
| APACHE JMeter™ | JMeter Test |
| Ju | JUnit Test |
| ⊘ | Postman resources |
| 📶 | Rate Schedule |
| 📶 | VU Schedule |

b. Search for the test asset or resource by entering any text contained in the test asset or resource name in the **Search** text box.

c. Create a filter query by using the **New filter** option by performing the following steps:

   i. Click **New filter**.

   ii. Select an operator, and add a rule, or a group of rules.

   iii. Add or enter the relevant parameters and either select or enter the condition and the criteria for the condition.

   You can select a parameter from the following list:

   - Type
   - Test Asset Name
   - Test Asset Path
   - Last Result
   - Next Run
   - Components

   iv. Apply the filter query to filter the assets based on the query.

   The test assets that match the filter criteria are displayed.

   v. Save the filter query by performing the following steps, if you want to reuse the filter query later:

   1. Click **Save**.
   2. Enter a name for the filter query.
   3. Click **Save**.

d. Retrieve a saved filter, if you have saved filter queries earlier by performing the following steps:

**Note:** To open the filter query, you must have created and saved a filter query.

i. Click the **Open filters** icon ☰.
ii. Select the saved filter in the **Filters** dialog box.
iii. Click **Apply** to apply the filter.

The test assets that match the filter criteria are displayed.

4. Click the **Execute** icon ⏵ in the row of the identified test asset.

   The **Execute test asset** dialog box is displayed.

5. Select the version of the test resources that you want to run by completing any of the following actions:

   **Note:** The test resources in the version can contain the test assets, datasets, `AFT XML` files, API environment tags, and other resources specific to projects created from any of the desktop clients.

   - Expand the list in the **Version** field, find the version of the test resources, and then select the version.

     Use the following details about the version of the test resources that are displayed to identify the version that you want:
     - Commit message.
     - Tags labeled by the user for the version committed.
     - The user who committed the version to the repository.
     - Relative time of the commit. For example, *2 hours ago* or *3 days ago*.

     The list displays the versions of the test resources committed by all users to the branch in the repository. The versions are arranged with the latest version that is committed, and then followed by the versions committed previously.

   - Expand the list in the **Version** field, and then search for the version that you want to select by entering a partial or the complete commit message of that version.

     The version that matches the search criteria is displayed and it is selected for the test run.

   The default value for the version selected for the run is the latest version in the selected branch in the repository. If you do not select any version, then the latest version is selected for the test run.

   **Notes:**

- If you selected a version but you do not want to use that version in the test run, you can remove the selected version by clicking the ✖ icon, then the default version is selected for the test run.
- If you repeated a test or ran the test again from the **Results** page, then the version of the test resources that you had selected for the earlier run is shown as selected. You can either retain this version or select any other version from the list. You can also remove the previous version by clicking the ✖ icon.

6. Select the time for scheduling the test run from the following options:

   - Select **Now** to initiate the test run immediately after you click **Execute**.

     **Important:** Click **Execute** only after you have configured the other settings in this dialog box.

   - Select **Later** and configure the date and time for scheduling a test to run at the scheduled date and time.

   The default time for scheduling a run is **Now**.

   **Notes:**

   - If you have configured some or all of the settings for the current test run, and you do not want to continue with those settings, you can reset the settings by clicking **Reset**.

   - If you want to repeat a test run and do not want to use the saved settings from a previous run, you can reset all the saved settings to their default values by clicking **Reset**.

7. Enter a label for the test run that helps you to identify the test on the **Results** page.

   After the test run completes, the label that you entered is displayed for the test under the **Labels** column on the **Results** page. After you have created a label, any member of the project can use that label.

   The default value for the **Label** field is null or an empty field.

   **Important:** The configuration that you set for the test run in the **Execute test asset** dialog box is preserved when you run the same test again. Those changes are not visible when another user logs in to HCL OneTest™ Server. For example, if you created new variables on the server, those variables are available only for you when the same test is run again.

   If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

8. Click **Advanced** to make the following advanced configurations:

a. Enter any JVM arguments that must be passed to the test run at run time in the **JVM Arguments** field, if applicable for the test.

   For example, you can set a maximum Java heap size.

b. Enter program arguments that must be passed to the test run at run time in the **Program Arguments**, if applicable for the test.

   If the test that you want to configure supports Jaeger tracing, then do the required tasks in the following scenarios:

| If... | Then... |
|---|---|
| You want to obtain the test results as a Jaeger trace. | Enter `-history jaeger` in the **Program Arguments** field. |
| Jaeger is already set as a program argument and you want to remove the Jaeger trace for your test. | Delete the `-history jaeger` entry in the **Program Arguments** field. |
| You want the report as a test log and as a Jaeger trace. | Enter `-history jaeger,testlog` in the **Program Arguments** field. |

   **Note:** The default report format is the *test log* format for the test reports.

c. Enter the environment variables that must be passed to the test run at run time in the **Environment Variables** field, if applicable for the test.

   For example, enter the environment variables when the third-party libraries that are used in the test run refer to the environment variables for configuration.

**Note:** You must separate the arguments or variables with a white space when you enter them in the same line or start each argument or variable on a new line.

The default value for each of the fields for the advanced settings is null or an empty field.

If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

9. Follow the instructions if you are running a test asset that contains datasets:

   a. Click the **DATA SOURCES** tab, if it is not already open.

   b. Consider the following information about datasets before you select a dataset:

The default value for the datasets in the **DATA SOURCES** tab is null if the test asset did not have an associated dataset. If the asset had an associated dataset, the default value is the associated dataset.

You can utilize the dataset stored as an Excel or CSV file to override the original dataset associated with the Suite, test, or schedule. For example, when you have associated a dataset in `.xlsx`, `.xls`, or `.csv` format with the test or schedule in desktop clients and if you have another set of data stored in an Excel or CSV file, then you can select that dataset from the **Override** list. If you want to run a test or schedule by using the schema created from the **Data Fabrication** page, see related links.

**Remember:** You must have uploaded the dataset as an Excel or CSV file into the Git repository, and ensured that both the original dataset (from the test asset) and new datasets (added to the project) have the same column names.

c. Select the dataset that you want to use in the test run from any of the following options:

- Select the dataset that is displayed as the default dataset when the test asset contains a single dataset.

  **Note:** If there is only one dataset in the test asset, then that dataset is displayed as the default dataset.

- Select the dataset from the list.

  **Note:** If there are multiple datasets in the test asset, the datasets are listed in their increasing alphabetical order.

- Select the dataset from the **Override** list to override the dataset that was associated with the test in the desktop client.

  **Important:** If the test contains an encrypted dataset, the Project Owner must classify it in the **DATA SECURITY** tab on the Project page before you can select it. You must have added datasets to your project from the **Dataset** page for the datasets to be displayed in the **Override** list.

If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

10. Follow the instructions in this step if the test requires a variable that must be passed to the test at the test run time.

You must configure the supported browser by using a variable if the test has a browser configured, which is different from the one that is supported by HCL OneTest™ Server.

a. Click the **VARIABLES** tab, if it is not already open.

b. Choose one of the following methods to add the variables:

  - To add new variables manually, click the **Add Variable** icon ⊕, enter the name, and value of the variable.

    i. Enter `RTW_WebUI_Browser_Selection` as the name.

    ii. Enter `Firefox` as the value, if you want to use Firefox as the browser and override the browser that is specified in the HTML test.

  - To add new variables from your local computer or from the Git repository that is associated with your server project, click the **Upload** icon and select the **Upload from local system** or **Browse from server** to select the variable file.

    **Note:** You must have created a file with the variables before you can select the file.

The default value for the variables is null or an empty field.

If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

11. Follow the instructions if you want to change the location for running the test:

a. Click the **LOCATION** tab, if it is not already open.

The **Default Cluster** is the default location where the test runs, and it is listed under the **Host** column. The information about the availability of the default location is displayed.

**Important:** You must have added remote Docker hosts that are registered with HCL OneTest™ Server to your project from the **Infrastructure** page. The remote Docker hosts are then displayed under the **Override** column.

**Notes:**

- If remote Docker hosts are not added to your project, the option **No override options** is displayed as the default value and the test runs in the Kubernetes cluster of HCL OneTest™ Server.
- If remote Docker hosts are added to your project, the added Docker hosts are displayed along with their availability status and ownership information.

b. Select the location where you want to run the test asset from the following options:

- Select the **Default Cluster** when no remote Docker hosts are available in your project.
- Select the remote Docker host from the list when a remote Docker host is available in your project.
- Select **No override options**, if you selected any remote Docker host and want to revert to the **Default Cluster** to run the test asset.

If you want to run the test immediately or at the scheduled time, click **Execute**.

12. Click **Execute**.

    The test run is initiated.

**Results**

You have configured and either started or scheduled a test run of a Compound Test that contains traditional HTML tests.

**What to do next**

You can choose to perform any of the following tasks after you have initiated or scheduled a run:

- Stop the test run at any point after the test run is initiated, from the **Execution** page. See Stopping a test run on page 317.
- Cancel a scheduled test run, from the **Execution** page. See Canceling a scheduled test run on page 318.
- View the progress of the test from the **Progress** page. See Viewing the progress of running test assets on page 312.
- Monitor the test from the **Progress** page. See Monitoring a test run.
- View the results, reports, and logs of the test from the **Results** page after the test completes the run. See Test results on page 359.

Related information

Resetting the configuration settings for a test run on page 315

Tests configurations and test runs

Test run configurations on page 240

## Configuring a run of a Compound Test that contains Web UI tests

After you added the test resources that you created in the desktop client to the project, you can configure a Compound Test that contains Web UI tests to be run on HCL OneTest™ Server.

**Before you begin**

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to server projects on page 504.
- Read and completed the tasks mentioned in Prerequisites to running tests on page 207, if they apply to the test that you want to configure for a run.
- You must have completed the following tasks:
    - Created Web UI tests in HCL OneTest™ UI and added the test asset to the project repository on HCL OneTest™ Server.
    - Read about the considerations that you must take into account before you configure a test run to run on a remote agent. See Test run considerations for running tests on remote agents on page 218.

1. Open the project that contains the test assets or resources that you added from the Git repository, and then click **Execution**.
2. Select the branch of the repository that contains the test assets or resources that you want to run.

   All test assets in the selected branch are displayed on the **Execution** page.
3. Identify the test asset or resource that you want to run by performing any of the following steps:

   a. Identify the test asset or resource by scrolling through the list.

   **Note:** You can also identify the type of the asset from the icon that represents the test type as shown in the following table:

   | Icon | Represents the test asset or resource |
   |---|---|
   | 🌐 | AFT Suite |
   | 🔌 | API Suite |
   | ▦ | Compound Test |
   | JMeter™ | JMeter Test |
   | Ju | JUnit Test |
   | ⊘ | Postman resources |
   | 📶 | Rate Schedule |
   | 📶 | VU Schedule |

   b. Search for the test asset or resource by entering any text contained in the test asset or resource name in the **Search** text box.

   c. Create a filter query by using the **New filter** option by performing the following steps:

   i. Click **New filter**.
   ii. Select an operator, and add a rule, or a group of rules.
   iii. Add or enter the relevant parameters and either select or enter the condition and the criteria for the condition.

      You can select a parameter from the following list:

      - Type
      - Test Asset Name
      - Test Asset Path
      - Last Result
      - Next Run
      - Components

   iv. Apply the filter query to filter the assets based on the query.

      The test assets that match the filter criteria are displayed.

   v. Save the filter query by performing the following steps, if you want to reuse the filter query later:

      1. Click **Save**.
      2. Enter a name for the filter query.
      3. Click **Save**.

d. Retrieve a saved filter, if you have saved filter queries earlier by performing the following steps:

   **Note:** To open the filter query, you must have created and saved a filter query.

   i. Click the **Open filters** icon.
   ii. Select the saved filter in the **Filters** dialog box.
   iii. Click **Apply** to apply the filter.

      The test assets that match the filter criteria are displayed.

4. Click the **Execute** icon in the row of the identified test asset.

   The **Execute test asset** dialog box is displayed.

5. Select the version of the test resources that you want to run by completing any of the following actions:

   **Note:** The test resources in the version can contain the test assets, datasets, `AFT XML` files, API environment tags, and other resources specific to projects created from any of the desktop clients.

   - Expand the list in the **Version** field, find the version of the test resources, and then select the version.

     Use the following details about the version of the test resources that are displayed to identify the version that you want:

        - Commit message.
        - Tags labeled by the user for the version committed.
        - The user who committed the version to the repository.
        - Relative time of the commit. For example, *2 hours ago* or *3 days ago*.

      The list displays the versions of the test resources committed by all users to the branch in the repository. The versions are arranged with the latest version that is committed, and then followed by the versions committed previously.

    - Expand the list in the **Version** field, and then search for the version that you want to select by entering a partial or the complete commit message of that version.

      The version that matches the search criteria is displayed and it is selected for the test run.

   The default value for the version selected for the run is the latest version in the selected branch in the repository. If you do not select any version, then the latest version is selected for the test run.

   **Notes:**

    - If you selected a version but you do not want to use that version in the test run, you can remove the selected version by clicking the ✖ icon, then the default version is selected for the test run.
    - If you repeated a test or ran the test again from the **Results** page, then the version of the test resources that you had selected for the earlier run is shown as selected. You can either retain this version or select any other version from the list. You can also remove the previous version by clicking the ✖ icon.

6. Select the time for scheduling the test run from the following options:

    - Select **Now** to initiate the test run immediately after you click **Execute**.

      **Important:** Click **Execute** only after you have configured the other settings in this dialog box.

    - Select **Later** and configure the date and time for scheduling a test to run at the scheduled date and time.

   The default time for scheduling a run is **Now**.

   **Notes:**

- If you have configured some or all of the settings for the current test run, and you do not want to continue with those settings, you can reset the settings by clicking **Reset**.
- If you want to repeat a test run and do not want to use the saved settings from a previous run, you can reset all the saved settings to their default values by clicking **Reset**.

7. Enter a label for the test run that helps you to identify the test on the **Results** page.

   After the test run completes, the label that you entered is displayed for the test under the **Labels** column on the **Results** page. After you have created a label, any member of the project can use that label.

   The default value for the **Label** field is null or an empty field.

   **Important:** The configuration that you set for the test run in the **Execute test asset** dialog box is preserved when you run the same test again. Those changes are not visible when another user logs in to HCL OneTest™ Server. For example, if you created new variables on the server, those variables are available only for you when the same test is run again.

   If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

8. Click **Advanced** to make the following advanced configurations:

   a. Enter any JVM arguments that must be passed to the test run at run time in the **JVM Arguments** field, if applicable for the test.

      For example, you can set a maximum Java heap size.

   b. Enter program arguments that must be passed to the test run at run time in the **Program Arguments**, if applicable for the test.

      If the test that you want to configure supports Jaeger tracing, then do the required tasks in the following scenarios:

| If... | Then... |
|---|---|
| You want to obtain the test results as a Jaeger trace. | Enter `-history jaeger` in the **Program Arguments** field. |
| Jaeger is already set as a program argument and you want to remove the Jaeger trace for your test. | Delete the `-history jaeger` entry in the **Program Arguments** field. |

| If... | Then... |
|---|---|
| You want the report as a test log and as a Jaeger trace. | Enter `-history jaeger,testlog` in the **Program Arguments** field. |

**Note:** The default report format is the *test log* format for the test reports.

c. Enter the environment variables that must be passed to the test run at run time in the **Environment Variables** field, if applicable for the test.

For example, enter the environment variables when the third-party libraries that are used in the test run refer to the environment variables for configuration.

**Note:** You must separate the arguments or variables with a white space when you enter them in the same line or start each argument or variable on a new line.

The default value for each of the fields for the advanced settings is null or an empty field.

If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

9. Follow the instructions if you are running a test asset that contains datasets:

a. Click the **DATA SOURCES** tab, if it is not already open.

b. Consider the following information about datasets before you select a dataset:

The default value for the datasets in the **DATA SOURCES** tab is null if the test asset did not have an associated dataset. If the asset had an associated dataset, the default value is the associated dataset.

You can utilize the dataset stored as an Excel or CSV file to override the original dataset associated with the Suite, test, or schedule. For example, when you have associated a dataset in `.xlsx`, `.xls`, or `.csv` format with the test or schedule in desktop clients and if you have another set of data stored in an Excel or CSV file, then you can select that dataset from the **Override** list. If you want to run a test or schedule by using the schema created from the **Data Fabrication** page, see related links.

**Remember:** You must have uploaded the dataset as an Excel or CSV file into the Git repository, and ensured that both the original dataset (from the test asset) and new datasets (added to the project) have the same column names.

c. Select the dataset that you want to use in the test run from any of the following options:

- Select the dataset that is displayed as the default dataset when the test asset contains a single dataset.

  **Note:** If there is only one dataset in the test asset, then that dataset is displayed as the default dataset.

- Select the dataset from the list.

  **Note:** If there are multiple datasets in the test asset, the datasets are listed in their increasing alphabetical order.

- Select the dataset from the **Override** list to override the dataset that was associated with the test in the desktop client.

  **Important:** If the test contains an encrypted dataset, the Project Owner must classify it in the **DATA SECURITY** tab on the Project page before you can select it. You must have added datasets to your project from the **Dataset** page for the datasets to be displayed in the **Override** list.

If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

10. Follow the instructions if the test requires a variable that must be passed to the test at the test run time.

    a. Click the **VARIABLES** tab, if it is not already open.

    b. Choose one of the following methods to add the variables:

    - To add new variables manually, click the **Add Variable** icon ⊕, enter the name, and value of the variable.
    - To add new variables from your local computer or from the Git repository that is associated with your server project, click the **Upload** icon and select the **Upload from local system** or **Browse from server** to select the variable file.

      **Note:** You must have created a file with the variables before you can select the file.

    The default value for the variables is null or an empty field.

    If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

11. Follow the instructions if you are running a compound test that has static agents configured:

a. Click the **LOCATION** tab, if it is not already open.

The static agents that are configured in the test asset are listed under the **Host** column. The information about the availability and capabilities of the agent are displayed.

**Note:** You must have added agents to your project from the **Infrastructure** page for the agents to be displayed under the **Override** column.

b. Select the agent where you want to run the test asset in the following scenarios:

| If... | Then... | Action |
|---|---|---|
| The agent specified in the test assets is available and no other agents are added to the project. | There is no agent displayed for override. | Select the agent in the **Host** column as the location to run the test. |
| The agent specified in the test assets is available and there are other agents that are added to the project. | Agents that have capabilities that match with the agent capabilities specified in the test assets are filtered and displayed in the **Override** column.<br>An agent from among the agents with matching capabilities is automatically selected for an override.<br>**Note:** The agents that have the capabilities that are missing from the agent in the **Host** column are displayed with the ❗ icon. | Perform any of the following actions:<br>- Select the agent in the **Host** column as the location to run the test.<br>- Select an agent in the **Override** column as the location to run the test. |
| The agent specified in the test assets is not available and there are other agents that are added to the project. | Agents that have capabilities that match with the agent capabilities specified in the test assets are automatically selected for an override in the **Override** column. | Select an agent in the **Override** column as the location to run the test. |
| | If there are no agents with any capabilities that match with the agent capabilities specified in | You cannot run the test. |

| If... | Then... | Action |
|---|---|---|
| | the test assets, no agent is selected or displayed for an override in the in the **Override** column. | |

The default value for the agents is null or an empty field if no agents were configured in the test asset. If the test asset contains agents that are configured, then the default agent is the first item to be displayed on the list of agents listed in the increasing alphabetical order.

If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

12. Follow the instructions if you want to change the location for running the test:

    a. Click the **LOCATION** tab, if it is not already open.

    The **Default Cluster** is the default location where the test runs, and it is listed under the **Host** column. The information about the availability of the default location is displayed.

    **Important:** You must have added remote Docker hosts that are registered with HCL OneTest™ Server to your project from the **Infrastructure** page. The remote Docker hosts are then displayed under the **Override** column.

    **Notes:**

    - If remote Docker hosts are not added to your project, the option **No override options** is displayed as the default value and the test runs in the Kubernetes cluster of HCL OneTest™ Server.
    - If remote Docker hosts are added to your project, the added Docker hosts are displayed along with their availability status and ownership information.

    b. Select the location where you want to run the test asset from the following options:

    - Select the **Default Cluster** when no remote Docker hosts are available in your project.
    - Select the remote Docker host from the list when a remote Docker host is available in your project.
    - Select **No override options**, if you selected any remote Docker host and want to revert to the **Default Cluster** to run the test asset.

    If you want to run the test immediately or at the scheduled time, click **Execute**.

13. Click **Execute**.

    The test run is initiated.

### Results

You have configured and either started or scheduled a test run of a Compound Test that contains Web UI tests.

**What to do next**

You can choose to perform any of the following tasks after you have initiated or scheduled a run:

- Stop the test run at any point after the test run is initiated, from the **Execution** page. See Stopping a test run on page 317.
- Cancel a scheduled test run, from the **Execution** page. See Canceling a scheduled test run on page 318.
- View the progress of the test from the **Progress** page. See Viewing the progress of running test assets on page 312.
- Monitor the test from the **Progress** page. See Monitoring a test run.
- View the results, reports, and logs of the test from the **Results** page after the test completes the run. See Test results on page 359.

Related information

Resetting the configuration settings for a test run on page 315

Tests configurations and test runs

Test run configurations on page 240

## Configuring a run of a Compound Test that contains mobile tests

**Disclaimer:** This release contains access to run mobile tests on HCL OneTest™ Server as a Tech Preview. The Tech Preview is intended for you to view the capabilities offered by HCL OneTest™ Server, and to provide your feedback to the product team. You are permitted to use the information only for evaluation purposes and not for use in a production environment. HCL provides the information without obligation of support and "as is" without warranty of any kind.

After you added the test resources that you created in the desktop client to the project, you can configure a Compound Test that contains mobile tests to be run on HCL OneTest™ Server.

**Before you begin**

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to server projects on page 504.
- You must have created the mobile tests in HCL OneTest™ UI and added the test asset to the project repository on HCL OneTest™ Server.

- You must have created a variable file if you want to import the variables file. The file must contain the details of the Appium server that is connected to the remote agent or the device cloud to which the mobile device is connected.
- Read Considerations for using Jaeger traces in reports on page 213, if you want to configure a run and you want Jaeger to report the test results for the test.

### About this task

You can run the mobile tests that are contained in a Compound Test after you create the tests in HCL OneTest™ UI, and then add them to your project on HCL OneTest™ Server. You can then run the mobile tests on the mobile devices that are connected to any of the following agents or clouds:

- Remote agents on which the Appium server is installed.
- The BitBar Cloud.
- The Perfecto Mobile cloud.

You must provide the details of the server or the mobile cloud to which the mobile devices are connected as variables in Step 10 on page 280. You can either enter the variables or use the file in which you entered the variables. You must refer to the following tables for the variables that are required for a successful run:

**Table 5. Variables for the Appium server**

| Name of the Variable | Action for the Value field |
|---|---|
| Mobile_Device_Selection | Specify the name of the mobile device that is connected to the Appium server. |
| appium.server.host | Specify the host name or IP address of the Appium server. |
| appium.server.port | Specify the port on the Appium server that is configured to communicate with HCL OneTest™ Server. |

**Table 6. Variables for the BitBar cloud**

| Name of the Variable | Value |
|---|---|
| Mobile_Device_Selection | Specify the name of the mobile device that is connected to the BitBar Cloud. |
| bitbar.apikey | Specify the user token generated for your BitBar account to authenticate your connection with the BitBar Cloud. |

**Table 6. Variables for the BitBar cloud (continued)**

| Name of the Variable | Value |
|---|---|
| bitbar.host | Specify the host name of the BitBar Cloud instance. |
| bitbar.project | Specify the name of the project that contains the recorded test. |
| bitbar.testrun | Specify a name for the test run that must be displayed in the BitBar dashboard for the test run. |

**Table 7. Variables for the Perfecto mobile cloud**

| Name of the Variable | Value |
|---|---|
| Mobile_Device_Selection | Specify the name of the mobile device that is connected to the Perfecto Mobile cloud. |
| perfecto.securitytoken | Specify the user token generated for your Perfecto account to authenticate your connection with the Perfecto Mobile cloud. |
| perfecto.host | Specify the host name of the Perfecto Mobile cloud instance. |

1. Open the project that contains the test assets or resources that you added from the Git repository, and then click **Execution**.
2. Select the branch of the repository that contains the test assets or resources that you want to run.

   All test assets in the selected branch are displayed on the **Execution** page.
3. Identify the test asset or resource that you want to run by performing any of the following steps:

   a. Identify the test asset or resource by scrolling through the list.

      **Note:** You can also identify the type of the asset from the icon that represents the test type as shown in the following table:

      | Icon | Represents the test asset or resource |
      |---|---|
      | 🌐 | AFT Suite |

| Icon | Represents the test asset or resource |
|---|---|
|  | API Suite |
|  | Compound Test |
| JMeter | JMeter Test |
| Ju | JUnit Test |
| ⊘ | Postman resources |
|  | Rate Schedule |
|  | VU Schedule |

b. Search for the test asset or resource by entering any text contained in the test asset or resource name in the **Search** text box.

c. Create a filter query by using the **New filter** option by performing the following steps:

   i. Click **New filter**.

   ii. Select an operator, and add a rule, or a group of rules.

   iii. Add or enter the relevant parameters and either select or enter the condition and the criteria for the condition.

   You can select a parameter from the following list:

   - Type
   - Test Asset Name
   - Test Asset Path
   - Last Result
   - Next Run
   - Components

   iv. Apply the filter query to filter the assets based on the query.

   The test assets that match the filter criteria are displayed.

   v. Save the filter query by performing the following steps, if you want to reuse the filter query later:

   1. Click **Save**.
   2. Enter a name for the filter query.
   3. Click **Save**.

d. Retrieve a saved filter, if you have saved filter queries earlier by performing the following steps:

> **Note:** To open the filter query, you must have created and saved a filter query.

  i. Click the **Open filters** icon.
  ii. Select the saved filter in the **Filters** dialog box.
  iii. Click **Apply** to apply the filter.

  The test assets that match the filter criteria are displayed.

4. Click the **Execute** icon in the row of the identified test asset.

   The **Execute test asset** dialog box is displayed.

5. Select the version of the test resources that you want to run by completing any of the following actions:

   > **Note:** The test resources in the version can contain the test assets, datasets, `AFT XML` files, API environment tags, and other resources specific to projects created from any of the desktop clients.

   - Expand the list in the **Version** field, find the version of the test resources, and then select the version.

     Use the following details about the version of the test resources that are displayed to identify the version that you want:
     - Commit message.
     - Tags labeled by the user for the version committed.
     - The user who committed the version to the repository.
     - Relative time of the commit. For example, *2 hours ago* or *3 days ago*.

     The list displays the versions of the test resources committed by all users to the branch in the repository. The versions are arranged with the latest version that is committed, and then followed by the versions committed previously.

   - Expand the list in the **Version** field, and then search for the version that you want to select by entering a partial or the complete commit message of that version.

     The version that matches the search criteria is displayed and it is selected for the test run.

   The default value for the version selected for the run is the latest version in the selected branch in the repository. If you do not select any version, then the latest version is selected for the test run.

   > **Notes:**

- If you selected a version but you do not want to use that version in the test run, you can remove the selected version by clicking the ✖ icon, then the default version is selected for the test run.
- If you repeated a test or ran the test again from the **Results** page, then the version of the test resources that you had selected for the earlier run is shown as selected. You can either retain this version or select any other version from the list. You can also remove the previous version by clicking the ✖ icon.

6. Select the time for scheduling the test run from the following options:

   - Select **Now** to initiate the test run immediately after you click **Execute**.

     **Important:** Click **Execute** only after you have configured the other settings in this dialog box.

   - Select **Later** and configure the date and time for scheduling a test to run at the scheduled date and time.

   The default time for scheduling a run is **Now**.

   **Notes:**

   - If you have configured some or all of the settings for the current test run, and you do not want to continue with those settings, you can reset the settings by clicking **Reset**.
   - If you want to repeat a test run and do not want to use the saved settings from a previous run, you can reset all the saved settings to their default values by clicking **Reset**.

7. Enter a label for the test run that helps you to identify the test on the **Results** page.

   After the test run completes, the label that you entered is displayed for the test under the **Labels** column on the **Results** page. After you have created a label, any member of the project can use that label.

   The default value for the **Label** field is null or an empty field.

   **Important:** The configuration that you set for the test run in the **Execute test asset** dialog box is preserved when you run the same test again. Those changes are not visible when another user logs in to HCL OneTest™ Server. For example, if you created new variables on the server, those variables are available only for you when the same test is run again.

   If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

8. Click **Advanced** to make the following advanced configurations:

   a. Enter any JVM arguments that must be passed to the test run at run time in the **JVM Arguments** field, if applicable for the test.

      For example, you can set a maximum Java heap size.

   b. Enter program arguments that must be passed to the test run at run time in the **Program Arguments**, if applicable for the test.

      If the test that you want to configure supports Jaeger tracing, then do the required tasks in the following scenarios:

| **If...** | **Then...** |
|---|---|
| You want to obtain the test results as a Jaeger trace. | Enter `-history jaeger` in the **Program Arguments** field. |
| Jaeger is already set as a program argument and you want to remove the Jaeger trace for your test. | Delete the `-history jaeger` entry in the **Program Arguments** field. |
| You want the report as a test log and as a Jaeger trace. | Enter `-history jaeger,testlog` in the **Program Arguments** field. |

      **Note:** The default report format is the *test log* format for the test reports.

   c. Enter the environment variables that must be passed to the test run at run time in the **Environment Variables** field, if applicable for the test.

      For example, enter the environment variables when the third-party libraries that are used in the test run refer to the environment variables for configuration.

   **Note:** You must separate the arguments or variables with a white space when you enter them in the same line or start each argument or variable on a new line.

The default value for each of the fields for the advanced settings is null or an empty field.

If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

9. Follow the instructions if you are running a test asset that contains datasets:

   a. Click the **DATA SOURCES** tab, if it is not already open.

   b. Consider the following information about datasets before you select a dataset:

The default value for the datasets in the **DATA SOURCES** tab is null if the test asset did not have an associated dataset. If the asset had an associated dataset, the default value is the associated dataset.

You can utilize the dataset stored as an Excel or CSV file to override the original dataset associated with the Suite, test, or schedule. For example, when you have associated a dataset in `.xlsx`, `.xls`, or `.csv` format with the test or schedule in desktop clients and if you have another set of data stored in an Excel or CSV file, then you can select that dataset from the **Override** list. If you want to run a test or schedule by using the schema created from the **Data Fabrication** page, see related links.

> **Remember:** You must have uploaded the dataset as an Excel or CSV file into the Git repository, and ensured that both the original dataset (from the test asset) and new datasets (added to the project) have the same column names.

c. Select the dataset that you want to use in the test run from any of the following options:

- Select the dataset that is displayed as the default dataset when the test asset contains a single dataset.

  > **Note:** If there is only one dataset in the test asset, then that dataset is displayed as the default dataset.

- Select the dataset from the list.

  > **Note:** If there are multiple datasets in the test asset, the datasets are listed in their increasing alphabetical order.

- Select the dataset from the **Override** list to override the dataset that was associated with the test in the desktop client.

  > **Important:** If the test contains an encrypted dataset, the Project Owner must classify it in the **DATA SECURITY** tab on the Project page before you can select it. You must have added datasets to your project from the **Dataset** page for the datasets to be displayed in the **Override** list.

If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

10. Perform the following steps to provide the variables that specify the server or cloud to which the mobile device is attached. You can either enter the variables that must be passed to the test at the test run time or import the file that contains the variables.

a. Click the **VARIABLES** tab, if it is not already open.

b. Choose one of the following methods to add the variables:

- To add new variables manually, click the **Add Variable** icon ⊕, enter the name, and value of the variable.
- To add new variables from your local computer or from the Git repository that is associated with your server project, click the **Upload** icon and select the **Upload from local system** or **Browse from server** to select the variable file.

  **Note:** You must have created a file with the variables before you can select the file.

11. Click **Execute**.

    The test run is initiated.

**Results**

You have configured and either started or scheduled a test run of a Compound Test that contains mobile tests.

**What to do next**

You can choose to perform any of the following tasks after you have initiated or scheduled a run:

- Stop the test run at any point after the test run is initiated, from the **Execution** page. See Stopping a test run on page 317.
- Cancel a scheduled test run, from the **Execution** page. See Canceling a scheduled test run on page 318.
- View the progress of the test from the **Progress** page. See Viewing the progress of running test assets on page 312.
- Monitor the test from the **Progress** page. See Monitoring a test run.
- View the results, reports, and logs of the test from the **Results** page after the test completes the run. See Test results on page 359.

Related information

Resetting the configuration settings for a test run on page 315

Tests configurations and test runs

Test run configurations on page 240

## Configuring a run of a Compound Test that contains performance tests

After you added the test resources that you created in the desktop client to the project, you can configure a Compound Test that contains performance tests to be run on HCL OneTest™ Server.

**Before you begin**

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to server projects on page 504.

1. Open the project that contains the test assets or resources that you added from the Git repository, and then click **Execution**.
2. Select the branch of the repository that contains the test assets or resources that you want to run.

   All test assets in the selected branch are displayed on the **Execution** page.
3. Identify the test asset or resource that you want to run by performing any of the following steps:

   a. Identify the test asset or resource by scrolling through the list.

      **Note:** You can also identify the type of the asset from the icon that represents the test type as shown in the following table:

      | Icon | Represents the test asset or resource |
      |---|---|
      | 🌐 | AFT Suite |
      |  | API Suite |
      |  | Compound Test |
      | JMeter | JMeter Test |
      | Ju | JUnit Test |
      |  | Postman resources |
      |  | Rate Schedule |
      |  | VU Schedule |

   b. Search for the test asset or resource by entering any text contained in the test asset or resource name in the **Search** text box.

   c. Create a filter query by using the **New filter** option by performing the following steps:

      i. Click **New filter**.

      ii. Select an operator, and add a rule, or a group of rules.

      iii. Add or enter the relevant parameters and either select or enter the condition and the criteria for the condition.

         You can select a parameter from the following list:

         - Type
         - Test Asset Name
         - Test Asset Path
         - Last Result

- Next Run
- Components

iv. Apply the filter query to filter the assets based on the query.

The test assets that match the filter criteria are displayed.

v. Save the filter query by performing the following steps, if you want to reuse the filter query later:

1. Click **Save**.
2. Enter a name for the filter query.
3. Click **Save**.

d. Retrieve a saved filter, if you have saved filter queries earlier by performing the following steps:

**Note:** To open the filter query, you must have created and saved a filter query.

i. Click the **Open filters** icon.
ii. Select the saved filter in the **Filters** dialog box.
iii. Click **Apply** to apply the filter.

The test assets that match the filter criteria are displayed.

4. Click the **Execute** icon in the row of the identified test asset.

   The **Execute test asset** dialog box is displayed.

5. Select the version of the test resources that you want to run by completing any of the following actions:

   **Note:** The test resources in the version can contain the test assets, datasets, `AFT XML` files, API environment tags, and other resources specific to projects created from any of the desktop clients.

   - Expand the list in the **Version** field, find the version of the test resources, and then select the version.

     Use the following details about the version of the test resources that are displayed to identify the version that you want:
     - Commit message.
     - Tags labeled by the user for the version committed.
     - The user who committed the version to the repository.
     - Relative time of the commit. For example, *2 hours ago* or *3 days ago*.

     The list displays the versions of the test resources committed by all users to the branch in the repository. The versions are arranged with the latest version that is committed, and then followed by the versions committed previously.

  - Expand the list in the **Version** field, and then search for the version that you want to select by entering a partial or the complete commit message of that version.

    The version that matches the search criteria is displayed and it is selected for the test run.

  The default value for the version selected for the run is the latest version in the selected branch in the repository. If you do not select any version, then the latest version is selected for the test run.

  **Notes:**

  - If you selected a version but you do not want to use that version in the test run, you can remove the selected version by clicking the ✖ icon, then the default version is selected for the test run.
  - If you repeated a test or ran the test again from the **Results** page, then the version of the test resources that you had selected for the earlier run is shown as selected. You can either retain this version or select any other version from the list. You can also remove the previous version by clicking the ✖ icon.

6. Select the time for scheduling the test run from the following options:

  - Select **Now** to initiate the test run immediately after you click **Execute**.

    **Important:** Click **Execute** only after you have configured the other settings in this dialog box.

  - Select **Later** and configure the date and time for scheduling a test to run at the scheduled date and time.

  The default time for scheduling a run is **Now**.

  **Notes:**

  - If you have configured some or all of the settings for the current test run, and you do not want to continue with those settings, you can reset the settings by clicking **Reset**.
  - If you want to repeat a test run and do not want to use the saved settings from a previous run, you can reset all the saved settings to their default values by clicking **Reset**.

7. Enter a label for the test run that helps you to identify the test on the **Results** page.

  After the test run completes, the label that you entered is displayed for the test under the **Labels** column on the **Results** page. After you have created a label, any member of the project can use that label.

  The default value for the **Label** field is null or an empty field.

**Important:** The configuration that you set for the test run in the **Execute test asset** dialog box is preserved when you run the same test again. Those changes are not visible when another user logs in to HCL OneTest™ Server. For example, if you created new variables on the server, those variables are available only for you when the same test is run again.

If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

8. Click **Advanced** to make the following advanced configurations:

    a. Enter any JVM arguments that must be passed to the test run at run time in the **JVM Arguments** field, if applicable for the test.

    For example, you can set a maximum Java heap size.

    b. Enter program arguments that must be passed to the test run at run time in the **Program Arguments**, if applicable for the test.

    If the test that you want to configure supports Jaeger tracing, then do the required tasks in the following scenarios:

| If... | Then... |
|---|---|
| You want to obtain the test results as a Jaeger trace. | Enter `-history jaeger` in the **Program Arguments** field. |
| Jaeger is already set as a program argument and you want to remove the Jaeger trace for your test. | Delete the `-history jaeger` entry in the **Program Arguments** field. |
| You want the report as a test log and as a Jaeger trace. | Enter `-history jaeger,testlog` in the **Program Arguments** field. |

    **Note:** The default report format is the *test log* format for the test reports.

    c. Enter the environment variables that must be passed to the test run at run time in the **Environment Variables** field, if applicable for the test.

    For example, enter the environment variables when the third-party libraries that are used in the test run refer to the environment variables for configuration.

**Note:** You must separate the arguments or variables with a white space when you enter them in the same line or start each argument or variable on a new line.

The default value for each of the fields for the advanced settings is null or an empty field.

If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

9. Follow the instructions if you are running a test asset that contains datasets:

    a. Click the **DATA SOURCES** tab, if it is not already open.

    b. Consider the following information about datasets before you select a dataset:

    The default value for the datasets in the **DATA SOURCES** tab is null if the test asset did not have an associated dataset. If the asset had an associated dataset, the default value is the associated dataset.

    You can utilize the dataset stored as an Excel or CSV file to override the original dataset associated with the Suite, test, or schedule. For example, when you have associated a dataset in `.xlsx`, `.xls`, or `.csv` format with the test or schedule in desktop clients and if you have another set of data stored in an Excel or CSV file, then you can select that dataset from the **Override** list. If you want to run a test or schedule by using the schema created from the **Data Fabrication** page, see related links.

    **Remember:** You must have uploaded the dataset as an Excel or CSV file into the Git repository, and ensured that both the original dataset (from the test asset) and new datasets (added to the project) have the same column names.

    c. Select the dataset that you want to use in the test run from any of the following options:

    - Select the dataset that is displayed as the default dataset when the test asset contains a single dataset.

      **Note:** If there is only one dataset in the test asset, then that dataset is displayed as the default dataset.

    - Select the dataset from the list.

      **Note:** If there are multiple datasets in the test asset, the datasets are listed in their increasing alphabetical order.

    - Select the dataset from the **Override** list to override the dataset that was associated with the test in the desktop client.

> **Important:** If the test contains an encrypted dataset, the Project Owner must classify it in the **DATA SECURITY** tab on the Project page before you can select it. You must have added datasets to your project from the **Dataset** page for the datasets to be displayed in the **Override** list.

If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

10. Follow the instructions if the test requires a variable that must be passed to the test at the test run time.

    a. Click the **VARIABLES** tab, if it is not already open.

    b. Choose one of the following methods to add the variables:

    - To add new variables manually, click the **Add Variable** icon ⊕, enter the name, and value of the variable.
    - To add new variables from your local computer or from the Git repository that is associated with your server project, click the **Upload** icon ⬆ and select the **Upload from local system** or **Browse from server** to select the variable file.

      > **Note:** You must have created a file with the variables before you can select the file.

    The default value for the variables is null or an empty field.

    If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

11. Follow the instructions if you are running a test that has static agents configured:

    a. Click the **LOCATION** tab, if it is not already open.

    The static agents that are configured in the test asset are listed under the **Host** column. The information about the availability of the agent is displayed.

    > **Note:** You must have added agents to your project from the **Infrastructure** page for the agents to be displayed under the **Override** column.

    b. Select the agent where you want to run the test asset.

    You can select the same agent that is configured in the test asset. Alternatively, you can override the agent with any other agent added to the project by selecting it from the list in the **Override** column.

    The default value for the agents is null or an empty field if no agents were configured in the test asset. If the test asset contains agents that are configured, then the default agent is the first item to be displayed on the list of agents listed in the increasing alphabetical order.

    If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

12. Follow the instructions if you want to change the location for running the test:

    a. Click the **LOCATION** tab, if it is not already open.

       The **Default Cluster** is the default location where the test runs, and it is listed under the **Host** column. The information about the availability of the default location is displayed.

       **Important:** You must have added remote Docker hosts that are registered with HCL OneTest™ Server to your project from the **Infrastructure** page. The remote Docker hosts are then displayed under the **Override** column.

       **Notes:**

       - If remote Docker hosts are not added to your project, the option **No override options** is displayed as the default value and the test runs in the Kubernetes cluster of HCL OneTest™ Server.
       - If remote Docker hosts are added to your project, the added Docker hosts are displayed along with their availability status and ownership information.

    b. Select the location where you want to run the test asset from the following options:

       - Select the **Default Cluster** when no remote Docker hosts are available in your project.
       - Select the remote Docker host from the list when a remote Docker host is available in your project.
       - Select **No override options**, if you selected any remote Docker host and want to revert to the **Default Cluster** to run the test asset.

    If you want to run the test immediately or at the scheduled time, click **Execute**.

13. Click **Execute**.
    The test run is initiated.

**Results**

You have configured and either started or scheduled a test run of a Compound Test that contains performance tests.

**What to do next**

You can choose to perform any of the following tasks after you have initiated or scheduled a run:

- Stop the test run at any point after the test run is initiated, from the **Execution** page. See Stopping a test run on page 317.
- Cancel a scheduled test run, from the **Execution** page. See Canceling a scheduled test run on page 318.
- View the progress of the test from the **Progress** page. See Viewing the progress of running test assets on page 312.

- Monitor the test from the **Progress** page. See Monitoring a test run.
- View the results, reports, and logs of the test from the **Results** page after the test completes the run. See Test results on page 359.

Related information

Resetting the configuration settings for a test run on page 315

Tests configurations and test runs

Test run configurations on page 240

## Configuring a JMeter test run

After you added the test resources that you created in JMeter to the server project, you can configure a JMeter test run on HCL OneTest™ Server.

**Before you begin**

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to server projects on page 504.
- You must have enabled the JMeter microservice on HCL OneTest™ Server.
- You must have read and completed the tasks mentioned in Test run considerations for JMeter tests on page 213 before you configure a JMeter test run.

**About this task**

**Important:** You can run the JMeter tests only on the default cluster of HCL OneTest™ Server. You cannot run the JMeter tests on remote agents or remote Docker hosts that are registered with HCL OneTest™ Server.

1. Open the project that contains the test assets or resources that you added from the Git repository, and then click **Execution**.
2. Select the branch of the repository that contains the test assets or resources that you want to run.

   All test assets in the selected branch are displayed on the **Execution** page.
3. Identify the test asset or resource that you want to run by performing any of the following steps:

   a. Identify the test asset or resource by scrolling through the list.

      **Note:** You can also identify the type of the asset from the icon that represents the test type as shown in the following table:

| Icon | Represents the test asset or resource |
|---|---|
| 🌐 | AFT Suite |

| Icon | Represents the test asset or resource |
|---|---|
|  | API Suite |
|  | Compound Test |
| JMeter | JMeter Test |
| JU | JUnit Test |
|  | Postman resources |
|  | Rate Schedule |
|  | VU Schedule |

b. Search for the test asset or resource by entering any text contained in the test asset or resource name in the **Search** text box.

c. Create a filter query by using the **New filter** option by performing the following steps:

   i. Click **New filter**.

   ii. Select an operator, and add a rule, or a group of rules.

   iii. Add or enter the relevant parameters and either select or enter the condition and the criteria for the condition.

      You can select a parameter from the following list:

      - Type
      - Test Asset Name
      - Test Asset Path
      - Last Result
      - Next Run
      - Components

   iv. Apply the filter query to filter the assets based on the query.

      The test assets that match the filter criteria are displayed.

   v. Save the filter query by performing the following steps, if you want to reuse the filter query later:

      1. Click **Save**.
      2. Enter a name for the filter query.
      3. Click **Save**.

d. Retrieve a saved filter, if you have saved filter queries earlier by performing the following steps:

> **Note:** To open the filter query, you must have created and saved a filter query.

   i. Click the **Open filters** icon ☰.
   ii. Select the saved filter in the **Filters** dialog box.
   iii. Click **Apply** to apply the filter.

   The test assets that match the filter criteria are displayed.

4. Click the **Execute** icon ▶ in the row of the identified test asset.

   The **Execute test asset** dialog box is displayed.

5. Select the version of the test resources that you want to run by completing any of the following actions:

   - Expand the list in the **Version** field, find the version of the test resources, and then select the version.

     Use the following details about the version of the test resources that are displayed to identify the version that you want:
       - Commit message.
       - Tags labeled by the user for the version committed.
       - The user who committed the version to the repository.
       - Relative time of the commit. For example, *2 hours ago* or *3 days ago*.

     The list displays the versions of the test resources committed by all users to the branch in the repository. The versions are arranged with the latest version that is committed, and then followed by the versions committed previously.

   - Expand the list in the **Version** field, and then search for the version that you want to select by entering a partial or the complete commit message of that version.

     The version that matches the search criteria is displayed and it is selected for the test run.

6. Select the time for scheduling the test run from the following options:

   - Select **Now** to initiate the test run immediately after you click **Execute**.

     > **Important:** Click **Execute** only after you have configured the other settings in this dialog box.

   - Select **Later** and configure the date and time for scheduling a test to run at the scheduled date and time.

   The default time for scheduling a run is **Now**.

   > **Notes:**

- If you have configured some or all of the settings for the current test run, and you do not want to continue with those settings, you can reset the settings by clicking **Reset**.
- If you want to repeat a test run and do not want to use the saved settings from a previous run, you can reset all the saved settings to their default values by clicking **Reset**.

7. Enter a label for the test run that helps you to identify the test on the **Results** page.

   After the test run completes, the label that you entered is displayed for the test under the **Labels** column on the **Results** page. After you have created a label, any member of the project can use that label.

   The default value for the **Label** field is null or an empty field.

   **Important:** The configuration that you set for the test run in the **Execute test asset** dialog box is preserved when you run the same test again. Those changes are not visible when another user logs in to HCL OneTest™ Server. For example, if you created new variables on the server, those variables are available only for you when the same test is run again.

   If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.
8. Click **Advanced** to make the following advanced configurations:

   a. Enter any JVM arguments that must be passed to the test run at run time in the **JVM Arguments** field, if applicable for the test.

      For example, you can set a maximum Java heap size.

   b. Enter program arguments that must be passed to the test run at run time in the **Program Arguments**, if applicable for the test.

   For example, enter **-J** to override any JMeter property that is defined in the properties file.

   **Note:** You must separate the arguments or variables with a white space when you enter them in the same line or start each argument or variable on a new line.

   The default value for each of the fields for the advanced settings is null or an empty field.

   If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.
9. Click **Execute**.
   The test run is initiated.

**Results**
You have configured and either started or scheduled a test run of a JMeter test.

**What to do next**

You can choose to perform any of the following tasks after you have initiated or scheduled a run:

- Stop the test run at any point after the test run is initiated, from the **Execution** page. See Stopping a test run on page 317.
- Cancel a scheduled test run, from the **Execution** page. See Canceling a scheduled test run on page 318.
- View the progress of the test from the **Progress** page. See Viewing the progress of running test assets on page 312.
- View the results, reports, and logs of the test from the **Results** page after the test completes the run. See Test results on page 359.

Related information

Resetting the configuration settings for a test run on page 315

Tests configurations and test runs

Test run configurations on page 240

## Configuring a JUnit test run

After you added the test resources that contain the Maven project with the JUnit tests to the server project, you can configure a Junit test run on HCL OneTest™ Server.

### Before you begin

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to server projects on page 504.
- You must ensure that the JUnit extension on HCL OneTest™ Server is enabled when you configure a test run.
- You must have read and completed the tasks mentioned in Test run considerations for JUnit tests on page 215 before you configure a JUnit test run.

### About this task

**Important:** You can run the JUnit tests only on the default cluster of HCL OneTest™ Server. You cannot run the JUnit tests on remote agents or remote Docker hosts that are registered with HCL OneTest™ Server.

After the test runs, you can view the results of the JUnit test in the following formats:

- Surefire
- Jaeger trace
- JUnit XML Report

1. Open the project that contains the test assets or resources that you added from the Git repository, and then click **Execution**.
2. Select the branch of the repository that contains the test assets or resources that you want to run.

All test assets in the selected branch are displayed on the **Execution** page.

3. Identify the test asset or resource that you want to run by performing any of the following steps:

    a. Identify the test asset or resource by scrolling through the list.

    **Note:** You can also identify the type of the asset from the icon that represents the test type as shown in the following table:

    | Icon | Represents the test asset or resource |
    |---|---|
    | 🌐 | AFT Suite |
    | 🔌 | API Suite |
    | ▦ | Compound Test |
    | JMeter | JMeter Test |
    | JU | JUnit Test |
    | ⊘ | Postman resources |
    | 📶 | Rate Schedule |
    | 📶 | VU Schedule |

    b. Search for the test asset or resource by entering any text contained in the test asset or resource name in the **Search** text box.

    c. Create a filter query by using the **New filter** option by performing the following steps:

        i. Click **New filter**.

        ii. Select an operator, and add a rule, or a group of rules.

        iii. Add or enter the relevant parameters and either select or enter the condition and the criteria for the condition.

        You can select a parameter from the following list:

        - Type
        - Test Asset Name
        - Test Asset Path
        - Last Result
        - Next Run
        - Components

        iv. Apply the filter query to filter the assets based on the query.

        The test assets that match the filter criteria are displayed.

    v. Save the filter query by performing the following steps, if you want to reuse the filter query later:

        1. Click **Save**.
        2. Enter a name for the filter query.
        3. Click **Save**.

  d. Retrieve a saved filter, if you have saved filter queries earlier by performing the following steps:

    **Note:** To open the filter query, you must have created and saved a filter query.

    i. Click the **Open filters** icon ☰.

    ii. Select the saved filter in the **Filters** dialog box.

    iii. Click **Apply** to apply the filter.

        The test assets that match the filter criteria are displayed.

4. Click the **Execute** icon ▶ in the row of the identified test asset.

   The **Execute test asset** dialog box is displayed.

5. Select the version of the test resources that you want to run by completing any of the following actions:

    - Expand the list in the **Version** field, find the version of the test resources, and then select the version.

      Use the following details about the version of the test resources that are displayed to identify the version that you want:
        - Commit message.
        - Tags labeled by the user for the version committed.
        - The user who committed the version to the repository.
        - Relative time of the commit. For example, *2 hours ago* or *3 days ago*.

      The list displays the versions of the test resources committed by all users to the branch in the repository. The versions are arranged with the latest version that is committed, and then followed by the versions committed previously.

    - Expand the list in the **Version** field, and then search for the version that you want to select by entering a partial or the complete commit message of that version.

      The version that matches the search criteria is displayed and it is selected for the test run.

6. Select the time for scheduling the test run from the following options:

- Select **Now** to initiate the test run immediately after you click **Execute**.

  **Important:** Click **Execute** only after you have configured the other settings in this dialog box.

- Select **Later** and configure the date and time for scheduling a test to run at the scheduled date and time.

The default time for scheduling a run is **Now**.

**Notes:**

- If you have configured some or all of the settings for the current test run, and you do not want to continue with those settings, you can reset the settings by clicking **Reset**.
- If you want to repeat a test run and do not want to use the saved settings from a previous run, you can reset all the saved settings to their default values by clicking **Reset**.

7. Enter a label for the test run that helps you to identify the test on the **Results** page.

   After the test run completes, the label that you entered is displayed for the test under the **Labels** column on the **Results** page. After you have created a label, any member of the project can use that label.

   The default value for the **Label** field is null or an empty field.

   **Important:** The configuration that you set for the test run in the **Execute test asset** dialog box is preserved when you run the same test again. Those changes are not visible when another user logs in to HCL OneTest™ Server. For example, if you created new variables on the server, those variables are available only for you when the same test is run again.

   If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

8. Click **Advanced** to make the following advanced configurations:

   a. Enter any JVM arguments that must be passed to the test run at run time in the **JVM Arguments** field, if applicable for the test.

      You can enter a variable that points to the *MAVEN_OPTS* property.

   b. Enter the environment variables that must be passed to the test run at run time in the **Environment Variables** field, if applicable for the test.

      For example, if you have created the JUnit tests by using JDK V12, then you must enter the variable as **JDKOTHER** and the value as **jdkv12**, so that the test run uses the JDK in the path `/data/junit-ext/jdks/jdkv12`.

**Note:** You must separate the arguments or variables with a white space when you enter them in the same line or start each argument or variable on a new line.

The default value for each of the fields for the advanced settings is null or an empty field.

If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

9. Click **Execute**.
   The test run is initiated.

**Results**

You have configured and either started or scheduled a test run of a JUnit test.

**What to do next**

You can choose to perform any of the following tasks after you have initiated or scheduled a run:

- Stop the test run at any point after the test run is initiated, from the **Execution** page. See Stopping a test run on page 317.
- Cancel a scheduled test run, from the **Execution** page. See Canceling a scheduled test run on page 318.
- View the progress of the test from the **Progress** page. See Viewing the progress of running test assets on page 312.
- View the results, reports, and logs of the test from the **Results** page after the test completes the run. See Test results on page 359.

Related information

Resetting the configuration settings for a test run on page 315

Tests configurations and test runs

Test run configurations on page 240

## Configuring a Postman test run

After you added the test resources that you created and exported from Postman to the server project, you can configure a run for Postman collections on HCL OneTest™ Server.

**Before you begin**

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to server projects on page 504.
- Read and completed the tasks mentioned in Test run considerations for Postman tests on page 216.

1. Open the project that contains the test assets or resources that you added from the Git repository, and then click **Execution**.
2. Select the branch of the repository that contains the test assets or resources that you want to run.

   All test assets in the selected branch are displayed on the **Execution** page.

3. Identify the test asset or resource that you want to run by performing any of the following steps:

   **Note:** You can either run the entire Postman collection or any of the folders within the collection.

   a. Identify the test asset or resource by scrolling through the list.

      **Note:** You can also identify the type of the asset from the icon that represents the test type as shown in the following table:

      | Icon | Represents the test asset or resource |
      |---|---|
      | 🌐 | AFT Suite |
      | ⇄ | API Suite |
      | ▦ | Compound Test |
      | JMeter | JMeter Test |
      | Ju | JUnit Test |
      | ⊘ | Postman resources |
      | 📶 | Rate Schedule |
      | 📶 | VU Schedule |

   b. Search for the test asset or resource by entering any text contained in the test asset or resource name in the **Search** text box.
   c. Create a filter query by using the **New filter** option by performing the following steps:
      i. Click **New filter**.
      ii. Select an operator, and add a rule, or a group of rules.
      iii. Add or enter the relevant parameters and either select or enter the condition and the criteria for the condition.

         You can select a parameter from the following list:

         - Type
         - Test Asset Name

            - Test Asset Path
            - Last Result
            - Next Run
            - Components

        iv. Apply the filter query to filter the assets based on the query.

            The test assets that match the filter criteria are displayed.

        v. Save the filter query by performing the following steps, if you want to reuse the filter query later:

            1. Click **Save**.
            2. Enter a name for the filter query.
            3. Click **Save**.

    d. Retrieve a saved filter, if you have saved filter queries earlier by performing the following steps:

        **Note:** To open the filter query, you must have created and saved a filter query.

        i. Click the **Open filters** icon .
        ii. Select the saved filter in the **Filters** dialog box.
        iii. Click **Apply** to apply the filter.

            The test assets that match the filter criteria are displayed.

4. Click the **Execute** icon in the row of the identified test asset.

    The **Execute test asset** dialog box is displayed.

5. Select the version of the test resources that you want to run.

    **Restriction:** You can only select the latest version of the test asset in the selected branch in the repository.

6. Select the time for scheduling the test run from the following options:

    - Select **Now** to initiate the test run immediately after you click **Execute**.

        **Important:** Click **Execute** only after you have configured the other settings in this dialog box.

    - Select **Later** and configure the date and time for scheduling a test to run at the scheduled date and time.

    The default time for scheduling a run is **Now**.

    **Notes:**

- If you have configured some or all of the settings for the current test run, and you do not want to continue with those settings, you can reset the settings by clicking **Reset**.
- If you want to repeat a test run and do not want to use the saved settings from a previous run, you can reset all the saved settings to their default values by clicking **Reset**.

7. Enter a label for the test run that helps you to identify the test on the **Results** page.

   After the test run completes, the label that you entered is displayed for the test under the **Labels** column on the **Results** page. After you have created a label, any member of the project can use that label.

   The default value for the **Label** field is null or an empty field.

   **Important:** The configuration that you set for the test run in the **Execute test asset** dialog box is preserved when you run the same test again. Those changes are not visible when another user logs in to HCL OneTest™ Server. For example, if you created new variables on the server, those variables are available only for you when the same test is run again.

   If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.
8. Click **Advanced**, and then enter program arguments that must be passed to the test run at run time in the **Program Arguments**, if applicable for the test.

   You can enter the command options that you use when you run Postman tests by using Newman, as the program arguments.

   For example, enter **--verbose**, if you want the details of the test run and each request sent.

   **Note:** You must separate the arguments or variables with a white space when you enter them in the same line or start each argument or variable on a new line.

   The default value for each of the fields for the advanced settings is null or an empty field.

   If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.
9. Follow the instructions if you want to change the location for running the test:

   a. Click the **LOCATION** tab, if it is not already open.

      The **Default Cluster** is the default location where the test runs, and it is listed under the **Host** column. The information about the availability of the default location is displayed.

**Important:** You must have added remote Docker hosts that are registered with HCL OneTest™ Server to your project from the **Infrastructure** page. The remote Docker hosts are then displayed under the **Override** column.

**Notes:**

- If remote Docker hosts are not added to your project, the option **No override options** is displayed as the default value and the test runs in the Kubernetes cluster of HCL OneTest™ Server.
- If remote Docker hosts are added to your project, the added Docker hosts are displayed along with their availability status and ownership information.

b. Select the location where you want to run the test asset from the following options:

- Select the **Default Cluster** when no remote Docker hosts are available in your project.
- Select the remote Docker host from the list when a remote Docker host is available in your project.
- Select **No override options**, if you selected any remote Docker host and want to revert to the **Default Cluster** to run the test asset.

If you want to run the test immediately or at the scheduled time, click **Execute**.

10. Click **Execute**.
The test run is initiated.

**Results**

You have configured and either started or scheduled a test run of a Postman test.

**What to do next**

You can choose to perform any of the following tasks after you have initiated or scheduled a run:

- Stop the test run at any point after the test run is initiated, from the **Execution** page. See Stopping a test run on page 317.
- Cancel a scheduled test run, from the **Execution** page. See Canceling a scheduled test run on page 318.
- View the progress of the test from the **Progress** page. See Viewing the progress of running test assets on page 312.
- View the results, reports, and logs of the test from the **Results** page after the test completes the run. See Test results on page 359.

Related information

Resetting the configuration settings for a test run on page 315

Tests configurations and test runs

Test run considerations for running tests on remote Docker hosts on page 224

Test run configurations on page 240

## Configuring a run of a Rate Schedule or VU Schedule

After you added the test resources that you created in the desktop client to the project, you can configure a Rate Schedule or VU Schedule to be run on HCL OneTest™ Server.

**Before you begin**

- Ensured that you are assigned a role as a *Project Owner* or *Tester* in the project. See Managing access to server projects on page 504.
- Read Test run considerations for schedules on page 212, if you want to configure a run for Rate Schedules or VU Schedules that have Resource Monitoring sources configured.

1. Open the project that contains the test assets or resources that you added from the Git repository, and then click **Execution**.
2. Select the branch of the repository that contains the test assets or resources that you want to run.

   All test assets in the selected branch are displayed on the **Execution** page.
3. Identify the test asset or resource that you want to run by performing any of the following steps:

   a. Identify the test asset or resource by scrolling through the list.

   **Note:** You can also identify the type of the asset from the icon that represents the test type as shown in the following table:

   | Icon | Represents the test asset or resource |
   |---|---|
   | 🌐 | AFT Suite |
   | ⟨⟩ | API Suite |
   | ▦ | Compound Test |
   | JMeter | JMeter Test |
   | JU | JUnit Test |
   | ⊘ | Postman resources |
   | 📶 | Rate Schedule |

| Icon | Represents the test asset or resource |
|---|---|
| 📊 | VU Schedule |

b. Search for the test asset or resource by entering any text contained in the test asset or resource name in the **Search** text box.

c. Create a filter query by using the **New filter** option by performing the following steps:

   i. Click **New filter**.

   ii. Select an operator, and add a rule, or a group of rules.

   iii. Add or enter the relevant parameters and either select or enter the condition and the criteria for the condition.

      You can select a parameter from the following list:

      - Type
      - Test Asset Name
      - Test Asset Path
      - Last Result
      - Next Run
      - Components

   iv. Apply the filter query to filter the assets based on the query.

      The test assets that match the filter criteria are displayed.

   v. Save the filter query by performing the following steps, if you want to reuse the filter query later:

      1. Click **Save**.
      2. Enter a name for the filter query.
      3. Click **Save**.

d. Retrieve a saved filter, if you have saved filter queries earlier by performing the following steps:

   **Note:** To open the filter query, you must have created and saved a filter query.

   i. Click the **Open filters** icon ☰.

   ii. Select the saved filter in the **Filters** dialog box.

   iii. Click **Apply** to apply the filter.

      The test assets that match the filter criteria are displayed.

4. Click the **Execute** icon ▶ in the row of the identified test asset.

   The **Execute test asset** dialog box is displayed.

5. Select the version of the test resources that you want to run by completing any of the following actions:

**Note:** The test resources in the version can contain the test assets, datasets, `AFT XML` files, API environment tags, and other resources specific to projects created from any of the desktop clients.

- Expand the list in the **Version** field, find the version of the test resources, and then select the version.

  Use the following details about the version of the test resources that are displayed to identify the version that you want:

  - Commit message.
  - Tags labeled by the user for the version committed.
  - The user who committed the version to the repository.
  - Relative time of the commit. For example, *2 hours ago* or *3 days ago*.

  The list displays the versions of the test resources committed by all users to the branch in the repository. The versions are arranged with the latest version that is committed, and then followed by the versions committed previously.
- Expand the list in the **Version** field, and then search for the version that you want to select by entering a partial or the complete commit message of that version.

  The version that matches the search criteria is displayed and it is selected for the test run.

The default value for the version selected for the run is the latest version in the selected branch in the repository. If you do not select any version, then the latest version is selected for the test run.

**Notes:**

- If you selected a version but you do not want to use that version in the test run, you can remove the selected version by clicking the ✖ icon, then the default version is selected for the test run.
- If you repeated a test or ran the test again from the **Results** page, then the version of the test resources that you had selected for the earlier run is shown as selected. You can either retain this version or select any other version from the list. You can also remove the previous version by clicking the ✖ icon.

6. Select the time for scheduling the test run from the following options:

   - Select **Now** to initiate the test run immediately after you click **Execute**.

     **Important:** Click **Execute** only after you have configured the other settings in this dialog box.

   - Select **Later** and configure the date and time for scheduling a test to run at the scheduled date and time.

   The default time for scheduling a run is **Now**.

**Notes:**

- If you have configured some or all of the settings for the current test run, and you do not want to continue with those settings, you can reset the settings by clicking **Reset**.
- If you want to repeat a test run and do not want to use the saved settings from a previous run, you can reset all the saved settings to their default values by clicking **Reset**.

7. Enter a label for the test run that helps you to identify the test on the **Results** page.

   After the test run completes, the label that you entered is displayed for the test under the **Labels** column on the **Results** page. After you have created a label, any member of the project can use that label.

   The default value for the **Label** field is null or an empty field.

   **Important:** The configuration that you set for the test run in the **Execute test asset** dialog box is preserved when you run the same test again. Those changes are not visible when another user logs in to HCL OneTest™ Server. For example, if you created new variables on the server, those variables are available only for you when the same test is run again.

   If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

8. Click **Advanced** to make the following advanced configurations:

   a. Enter any JVM arguments that must be passed to the test run at run time in the **JVM Arguments** field, if applicable for the test.

      For example, you can set a maximum Java heap size.

   b. Enter program arguments that must be passed to the test run at run time in the **Program Arguments**, if applicable for the test.

      If the test that you want to configure supports Jaeger tracing, then do the required tasks in the following scenarios:

| If... | Then... |
|---|---|
| You want to obtain the test results as a Jaeger trace. | Enter `-history jaeger` in the **Program Arguments** field. |
| Jaeger is already set as a program argument and you want to remove the Jaeger trace for your test. | Delete the `-history jaeger` entry in the **Program Arguments** field. |

| If... | Then... |
|---|---|
| You want the report as a test log and as a Jaeger trace. | Enter `-history jaeger,testlog` in the **Program Arguments** field. |

**Note:** The default report format is the *test log* format for the test reports.

c. Enter the environment variables that must be passed to the test run at run time in the **Environment Variables** field, if applicable for the test.

For example, enter the environment variables when the third-party libraries that are used in the test run refer to the environment variables for configuration.

**Note:** You must separate the arguments or variables with a white space when you enter them in the same line or start each argument or variable on a new line.

The default value for each of the fields for the advanced settings is null or an empty field.

If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

9. Follow the instructions if you are running a test asset that contains datasets:

a. Click the **DATA SOURCES** tab, if it is not already open.

b. Consider the following information about datasets before you select a dataset:

The default value for the datasets in the **DATA SOURCES** tab is null if the test asset did not have an associated dataset. If the asset had an associated dataset, the default value is the associated dataset.

You can utilize the dataset stored as an Excel or CSV file to override the original dataset associated with the Suite, test, or schedule. For example, when you have associated a dataset in `.xlsx`, `.xls`, or `.csv` format with the test or schedule in desktop clients and if you have another set of data stored in an Excel or CSV file, then you can select that dataset from the **Override** list. If you want to run a test or schedule by using the schema created from the **Data Fabrication** page, see related links.

**Remember:** You must have uploaded the dataset as an Excel or CSV file into the Git repository, and ensured that both the original dataset (from the test asset) and new datasets (added to the project) have the same column names.

c. Select the dataset that you want to use in the test run from any of the following options:

- Select the dataset that is displayed as the default dataset when the test asset contains a single dataset.

  **Note:** If there is only one dataset in the test asset, then that dataset is displayed as the default dataset.

- Select the dataset from the list.

  **Note:** If there are multiple datasets in the test asset, the datasets are listed in their increasing alphabetical order.

- Select the dataset from the **Override** list to override the dataset that was associated with the test in the desktop client.

  **Important:** If the test contains an encrypted dataset, the Project Owner must classify it in the **DATA SECURITY** tab on the Project page before you can select it. You must have added datasets to your project from the **Dataset** page for the datasets to be displayed in the **Override** list.

If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

10. Follow the instructions if the test requires a variable that must be passed to the test at the test run time.

    a. Click the **VARIABLES** tab, if it is not already open.

    b. Choose one of the following methods to add the variables:

    - To add new variables manually, click the **Add Variable** icon ⊕, enter the name, and value of the variable.
    - To add new variables from your local computer or from the Git repository that is associated with your server project, click the **Upload** icon and select the **Upload from local system** or **Browse from server** to select the variable file.

      **Note:** You must have created a file with the variables before you can select the file.

    The default value for the variables is null or an empty field.

    If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

11. Follow the instructions if you are running a test that has static agents configured:

a. Click the **LOCATION** tab, if it is not already open.

The static agents that are configured in the test asset are listed under the **Host** column. The information about the availability of the agent is displayed.

> **Note:** You must have added agents to your project from the **Infrastructure** page for the agents to be displayed under the **Override** column.

b. Select the agent where you want to run the test asset.

You can select the same agent that is configured in the test asset. Alternatively, you can override the agent with any other agent added to the project by selecting it from the list in the **Override** column.

The default value for the agents is null or an empty field if no agents were configured in the test asset. If the test asset contains agents that are configured, then the default agent is the first item to be displayed on the list of agents listed in the increasing alphabetical order.

If you want to run the test immediately or at the scheduled time, click **Execute**, or continue with the next step.

12. Follow the instructions if you want to change the location for running the test:

a. Click the **LOCATION** tab, if it is not already open.

The **Default Cluster** is the default location where the test runs, and it is listed under the **Host** column. The information about the availability of the default location is displayed.

> **Important:** You must have added remote Docker hosts that are registered with HCL OneTest™ Server to your project from the **Infrastructure** page. The remote Docker hosts are then displayed under the **Override** column.

> **Notes:**
> - If remote Docker hosts are not added to your project, the option **No override options** is displayed as the default value and the test runs in the Kubernetes cluster of HCL OneTest™ Server.
> - If remote Docker hosts are added to your project, the added Docker hosts are displayed along with their availability status and ownership information.

b. Select the location where you want to run the test asset from the following options:

- Select the **Default Cluster** when no remote Docker hosts are available in your project.
- Select the remote Docker host from the list when a remote Docker host is available in your project.
- Select **No override options**, if you selected any remote Docker host and want to revert to the **Default Cluster** to run the test asset.

If you want to run the test immediately or at the scheduled time, click **Execute**.

13. Follow the instructions if you want to override the Resource Monitoring labels in the test asset with the Resource Monitoring labels that you created in HCL OneTest™ Server:

    a. Click the **Resource Monitoring** tab.

       **Note:**

       The **Resource Monitoring** tab displays only if you enabled the **Resource Monitoring from Service** option in HCL OneTest™ Performance when you created the test asset.

    b. Click +, and press the `Ctrl` + `Space bar` keys, or enter the initial letter of a label to select a label in the list.

       **Note:**

       You can select or add labels only if you added the labels in the **Resource Monitoring Sources** page in HCL OneTest™ Server.

14. Click **Execute**.
    The test run is initiated.

**Results**

You have configured and either started or scheduled a test run of a Rate Schedule or VU Schedule.

**What to do next**

You can choose to perform any of the following tasks after you have initiated or scheduled a run:

- Stop the test run at any point after the test run is initiated, from the **Execution** page. See Stopping a test run on page 317.
- Cancel a scheduled test run, from the **Execution** page. See Canceling a scheduled test run on page 318.
- View the progress of the test from the **Progress** page. See Viewing the progress of running test assets on page 312.
- Monitor the test from the **Progress** page. See Monitoring a test run.
- View the results, reports, and logs of the test from the **Results** page after the test completes the run. See Test results on page 359.

Related information

Resetting the configuration settings for a test run on page 315

Tests configurations and test runs

Test run configurations on page 240

## Running tests by using Data Fabrication

Starting from V10.0.2, if a test or schedule is associated with a dataset, you can replace the dataset at run time with the schema created from the **Data Fabrication** page.

**Before you begin**

You must have completed the following tasks:

- Created a dataset and associated it with a test in desktop clients and added to the Git repository.
- Created a project in HCL OneTest™ Server. See Test assets and a server project on page 496.
- Configured the repository that contains the test assets in your project. See Adding repositories to a server project on page 499.
- Created a schema from the **Data Fabrication** page. See Schema fabrication on page 120.

**About this task**

In HCL OneTest™ Server, from the **Data Fabrication** page, you can create a schema that can be used while running a test or schedule that is associated with a dataset. For example, after the recording, a test is generated that captures interactions between the client and the server. When you run this test, it uses the same data that you used during recording. For example, if you want to test the application with random test data, you can override the dataset being used in the test or schedule with a schema that you created from the **Data Fabrication** page of HCL OneTest™ Server.

Whenever you generate test data to perform application testing, the generated data is different. You can produce the same set of data multiple times by setting the seed value. For example, you have used seed value as 1 to generate test data. When you run the test or schedule by providing the seed value as 1, the same set of data is used for testing instead of different data.

**Note:** You must ensure that the original dataset that is associated with a test or schedule and a schema that you have created have the same column names.

1. Initiate the test run on the **Execution** page.
2. Create or pick a label for the test run to identify the run.

   After the run completes, the label is displayed against the run on the **Results** page.
3. Click the **DATA SOURCES** tab and perform the following sub-steps:

a. Select a schema to override the dataset that was associated with a test or schedule in the desktop client.

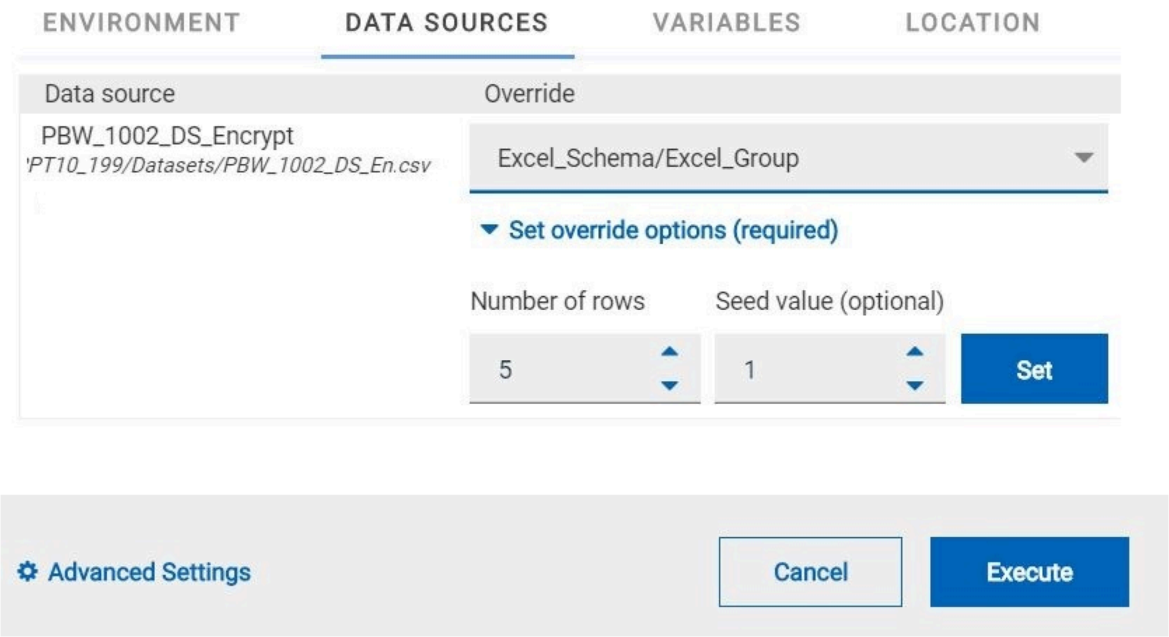
   **Note:** If the dataset and schema have the same column names, only then the **Override** drop-down list shows the existing schema names.

b. Enter the number of records that you want to generate in the **Number of rows** field.

c. Optional. Enter a seed value in the **Seed value (optional)** field.

   The seed value acts as an instance of random data that can be repeated if required. By default, the seed value is blank. Alternatively, you can use the up-down control button to increment or decrement the **Number of rows** and **Seed value**.

   **Note:** You must provide a value in the **Number of rows** field. Otherwise, you cannot run the test or schedule.



| Data source | Override |
|---|---|
| PBW_1002_DS_Encrypt<br>'PT10_199/Datasets/PBW_1002_DS_En.csv | Excel_Schema/Excel_Group |

▾ Set override options (required)

Number of rows: 5

Seed value (optional): 1

4. Click **Set**, and then **Execute**.

**What to do next**

You can monitor a test run and check the logs to ensure that the data is taken from the schema instead of the dataset when running the test.

Related information

Test assets and a server project on page 496

# Management of running tests

Find information about the tasks that you can perform on a test that you configured for a run either while it runs or after it completes the run.

## Viewing the progress of running test assets

After you initiate or schedule a test run on the **Execution** page, you can view the progress of the test assets that are running or scheduled to run, from the **Progress** page. You can also view the test assets that have completed their run, are stopped or canceled, during the past hour.

**Before you begin**

You must have initiated runs of the test assets in your project from the **Execution** page.

1. Open your project and click **Progress**.

   Any one of the following views is displayed:

   - No test assets are displayed in the **Progress** page if there are no test assets that are running, scheduled to run, or have completed their run in the past hour.
   - Test assets in your project that are in any of the following states are listed:
     - Running
     - Scheduled
     - Completed
     - Stopped
     - Canceled
   - No test assets are displayed but the options **Hide inactive** and **Show all** are present. Click **Show all** to display all the test assets on the **Progress** page. Test assets that were hidden from the display are displayed.

   **Note:** Test assets are automatically removed from the display after *60 minutes* of being added to the **Progress** page. Therefore, you can only view the test assets that were added to the **Progress** page in the past hour. You cannot use the **Progress** page to view the history of test assets that were run from the **Execution** page.

2. Identify the test asset for which you want to view the progress from the test assets listed. You can also identify the test asset by completing any of the following actions:

- Search for the test asset by entering any text contained in the test asset name in the **Search** text box.
- Sort the test assets by the user who started the test by clicking the **Started by** column header, and find the test asset listed against your name.
- Sort the test assets by the type of the test assets by clicking the **Started by** column header, and find the test asset type you ran or scheduled.
- Click **Hide inactive** to only display the test assets that are running or scheduled to run. This action removes all completed, canceled, stopped, or failed test assets from the display.

You can view the following details of your test asset on the Progress page:

| **Column header** | **Description** |
|---|---|
| Type | Displays the icon for the type of the test asset. |
| Name | Displays the name of the test asset added to the project from the Git repository. |
| Started by | Displays the name of the user who started the run of the test asset. |
| Start Time | Displays the start time of the test asset run. |
| End Time | Displays the end time of the test asset run. |
| Status | Displays the state of the test asset progressively.<br>For example, after the run is in the `Initiated` state, it moves on to the `Running` state and ends in its final state depending on the verdict of the test run as either `Completed`, `Failed`, or `Inconclusive`.<br>If the test run is stopped or a scheduled run is canceled, the status displayed is `Stopped by User` for a stopped run or `Canceled` for a canceled test. |

From the **Actions** column, you can perform the following operations on your test asset:

- Stop a run.
- Cancel a scheduled run.
- Monitor a VU Schedule, Rate Schedule, Compound Test, or an AFT Suite.
- View the execution log.
- View the result of the test asset on the **Results** page.

**Results**

You have viewed the progress of the test assets that you ran from the **Execution** page.

### What to do next

You can choose to perform any of the following tasks from the **Progress** page:

- Stop a test asset that is running.
- Cancel a scheduled run.
- Monitor tests (such as the Schedules, Compound Test, and AFT Suites).
- View the results or the execution log of the test asset from the **Actions** column by clicking the **Open action menu** icon ⋮.
- View the results, reports, and logs of the completed test run from the **Results** page.

Related information

[Test run configurations on page 240](#)

Monitoring a test run

[Checking logs on page 314](#)

[Canceling a scheduled test run on page 318](#)

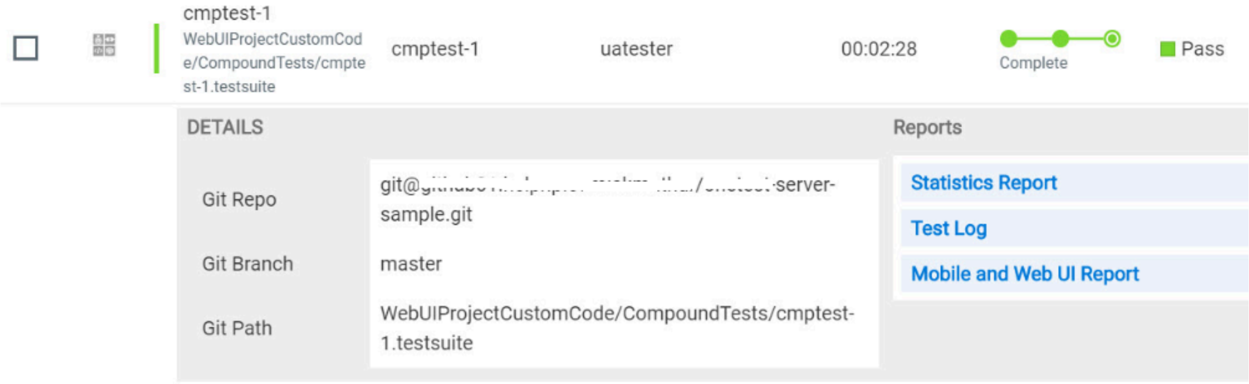
[Stopping a test run on page 317](#)

## Checking logs

To verify how the test ran or to debug test run failure, you can check the `Test Log` and `Execution log`.

### About this task

The `Test Log` displays the interaction between HCL OneTest™ Server and the application or system under test. After the test run completes, the verdict of the run can be *Pass*, *Fail*, or *Inconclusive*. If the verdict of the run is Pass, the `Test Log` is available in the **Results** page.



| | | cmptest-1<br>WebUIProjectCustomCode/CompoundTests/cmptest-1.testsuite | cmptest-1 | uatester | 00:02:28 | Complete | 🟩 Pass |
|---|---|---|---|---|---|---|---|

DETAILS

| | |
|---|---|
| Git Repo | git@[redacted]server-sample.git |
| Git Branch | master |
| Git Path | WebUIProjectCustomCode/CompoundTests/cmptest-1.testsuite |

The `Execution log` displays the console messages of the run time process that runs the test. This log is useful in determining the cause of the failure if the verdict of the run is *Fail* or *Inconclusive*. You can view the `Execution log` from the **Progress** page.

Viewing the `Test Log`

1. Go to the **Results** page and identify the test that you ran.
2. Click the test so that the **Reports** panel is displayed.
3. Click the **Test Log** in the **Reports** panel.

   The `Test Log` is displayed in a browser window.

Viewing the `Execution` log

4. Go to the **Progress** page and identify the test that is in the *Running* state.
5. Click the **Open action menu** icon.
6. Click **Execution log**.

   The `Execution log` is displayed in a browser window.

Related information

Test run configurations on page 240

Monitoring a test run

## Resetting the configuration settings for a test run

When you are configuring a test run either as a first-run or when repeating the run, and you do not want to proceed with the settings configured or saved for the test run, you can reset the configuration settings. Resetting the configuration reverts all the settings to their default values for the test run.

**Before you begin**

You must be a member of the project with the *Owner* or *Tester* role to run the tests.

You must ensure that you have the **Execute test asset** dialog box is displayed before you proceed with resetting the configuration settings for the test run.

**About this task**

You can reset the configuration settings when you are configuring a test run either for its first run or when you are repeating the test run.

1. Click **Reset** in the **Execute test asset** dialog box at any point when you want to reset the configuration settings for the test run.

   **Notes:**

- If you have configured some or all of the settings for the current test run and you do not want to continue with those settings, clicking **Reset** resets the settings to their default values.
- If you are repeating a test run and do not want to use the saved settings from a previous run, clicking **Reset** reverts all the saved settings to their default values.

The settings in the **Execute test asset** dialog box are reset to their default values.

2. Use the following table to find the default values for each of the settings in the **Execute test asset** dialog box:

| Window or Tab | Field or Setting | Default value is |
|---|---|---|
| **Execute test asset** | Scheduling the test run | **Now** is selected. |
| **Execute test asset** | Label for settings | Null or empty field. |
| **Advanced** | **JVM Arguments** | Null or empty field. |
| | **Program Arguments** | Null or empty field. |
| | **Environment Variables** | Null or empty field. |
| **ENVIRONMENT** tab | API test environment | The environment configured in the test asset. |
| | Secrets collection | Null or empty field. |
| **DATA SOURCES** tab | **Override** list | Null or empty field if the asset has no dataset. |
| | | The dataset in the test asset if the asset has one dataset. |
| | | The first dataset on the list of datasets, which are listed in the increasing alphabetical order, if the asset has multiple datasets. |
| **VARIABLES** tab | | Null or empty field. |
| **LOCATION** tab | Agents | Null or empty field if the asset has no agents configured. |
| | | The first agent on the list of agents, which are listed in the increasing alphabetical order, if the asset has multiple agents configured. |
| | Docker host | Internal Docker host if no remote Docker hosts are added. |
| | | **No override options** if other Docker hosts are saved in previous runs. |

**What to do next**

You can complete configuring the settings that you want for the test run. See Test run configurations on page 240.

## Stopping a test run

You might want to stop a test run when you realized that you did not configure all the settings or you want to change a few settings for the test run.

### Before you begin

You must have initiated a test run from the **Execution** page.

### About this task

You can stop a running test from the Progress page by using the **Stop execution** icon ⊗. You cannot stop the test run if the test has already completed its run.

1. Go to the **Progress** page and identify the test that you want to stop.

   **Note:** You can stop a running test if it is in the `In Transition` or `Running` state.

2. Click the **Stop execution** icon ⊗ in the **Actions** column of the selected test.
   **Result**

   The **Stop execution** dialog box is displayed.

3. Set the timeout period for stopping the test run. Enter a numeric value and select the unit from the options available as *Seconds*, *Minutes*, or *Hours*.

   The time out period is the time during which the test run is allowed to stop on its own and after the timeout period, the test is forced to stop abruptly.

   **Important:** If the timeout period is not set, the default of *30 seconds* is considered as the default timeout period.

4. Keep the option **Capture results** selected, if you want the results to be captured for the test. The captured results for the test are available to view from the Results page. Clear this option if you do not want to capture the results.

   **Note:** The option **Capture results** is selected by default.

   **Important:** The results are always captured for API suites even if you clear the **Capture results** option.

5. Keep the option **Execute 'finally-block', if present** selected, if you want the *finally-block* code in the test script, if present, to be run before the test run is stopped. Clear this option if you do not want to run the *finally-block* code in the test script.

**Note:** The option **Execute 'finally-block', if present** is selected by default.

**Restriction:** The *tear-down* steps equivalent to the **'finally-block'** code, if present in API suites are not controllable at runtime. The **Execute 'finally-block', if present** option has no impact on API suites.

6. Click **Stop execution**.

   A notification message is displayed that the test run is stopped successfully.

   **Note:** If the test run completes before you can configure the options for stopping the test run, a notification is displayed in the **Stop execution** dialog box that the test asset has completed its run.

**Results**

You have successfully stopped a running test. The stopped test runs are displayed on the Progress page with the status as *Stopped by User*. Stopped tests are not considered in the count of tests executed that are displayed on the Overview page.

**What to do next**

You can re-initiate the test run from the Execution page by completing the configurations that you want for the test.

You can see the results or the execution log of the stopped test from the **Actions** column on the Progress page by clicking the **Open action menu** icon ⋮.

Related information

Test run configurations on page 240

Viewing the progress of running test assets on page 312

Canceling a scheduled test run on page 318

## Canceling a scheduled test run

You might want to cancel a scheduled test run when you realized that you did not configure all the settings for a test, want to change a few settings for the test, or do not want that test to run.

**Before you begin**

You must have scheduled a test run from the **Execution** page.

**About this task**

You can cancel a scheduled test by using the **Cancel** icon ⊗ available on the **Progress** page.

1. Go to the **Progress** page and identify the scheduled test that you want to cancel.
2. Click the **Cancel** icon ⊗ in the **Actions** column of the selected test.

   **Result**

   The **Cancel scheduled execution** dialog box is displayed.
3. Click **Yes**.

   **Result**

   A notification is displayed that the scheduled test run is canceled.

**Results**

You have successfully canceled a scheduled test. The canceled test runs are displayed on the **Progress** page with the status as *Canceled*. The canceled test runs are not displayed on the **Results** page and are not considered in the count of tests executed that are displayed on the **Overview** page.

**What to do next**

You can re-initiate the test run from the **Execution** page by completing the configurations that you want for the test.

Related information

Test run configurations on page 240

Viewing the progress of running test assets on page 312

Stopping a test run on page 317

## Management of virtualized services

**Disclaimer:** This release contains access to the Virtual services that virtualize the Istio based services feature in HCL OneTest™ Server as a Tech Preview. The Tech Preview is intended for you to view the capabilities for virtualized services offered by HCL OneTest™ Server, and to provide your feedback to the product team. You are permitted to use the information only for evaluation purposes and not for use in a production environment. HCL provides the information without obligation of support and "as is" without warranty of any kind.

You can find information about the tasks that you can perform on and manage the virtualized services that run on HCL OneTest™ Server. You can start or stop the virtual services that are connected to HCL OneTest™ Server. You can view the routing rules and usage statistics of virtualized services, agents, or intercepts that are connected to HCL OneTest™ Server.

You must create stubs or virtual services in HCL OneTest™ API for the following types of services. You must commit the virtual service resources to a remote repository and then add the repository to your project on HCL OneTest™ Server before you can run the virtual services on HCL OneTest™ Server:

- Virtual services that utilize the WebSphere® MQ transport.
- Virtual services that utilize the HTTP transport.
- Virtual services that virtualize the Istio based services. You can run virtual services for the following types of requests received or sent by the Istio service mesh:
  - Requests received by services in the Istio service mesh.
  - Requests sent from namespaces in the Istio service mesh to external services that are not in the Istio service mesh.

**Important:** You can run virtual services only in the **Default Cluster** location of HCL OneTest™ Server. You cannot run virtual services on a remote Docker host.

## Working with virtual services

You can perform the following tasks on virtual services on HCL OneTest™ Server:

- Read about the considerations before you configure a run of the virtual services. See Prerequisites for running virtual services on page 321.
- Read about the considerations before you configure a run of the Istio based services that are virtualized. See Prerequisites for running virtualized Istio based services in HCL OneTest Server on page 322.
- Set up the HTTP proxy, if you have configured stubs in the test assets to use an HTTP proxy to route requests. See Setting up the HTTP proxy on page 323.
- View all intercepts that are registered with HCL OneTest™ Server including those that you have not configured. See Viewing intercepts that are registered with HCL OneTest Server on page 325.
- View virtual service resources in the test assets that are in the repositories added to a project on HCL OneTest™ Server. Viewing virtual service resources on page 325.
- Configure runs of virtual services. See Configuring a run of a virtual service on page 328.
- Configure runs of HTTP virtual services to run without using proxies. See Running HTTP virtual services without using proxies on page 340.
- Run Istio stubs that were created in HCL OneTest™ API to virtualize services in the Kubernetes cluster on which HCL OneTest™ Server is installed.
- View running instances of virtual services. See Viewing running instances of virtual services on page 345.
- View configurations of a running virtual service instance. See Viewing configurations of running instances of virtual services on page 348.
- Modify the configurations of a running virtual service instance. See Modifying configurations of running instances of virtual services on page 353.
- View details of the usage statistics of the virtual services. See Viewing usage statistics of virtual services on page 356.
- View routing rules of the intercepts. See Viewing routing rules of the virtual services on page 355.
- Stop a running virtual service instance. See Stopping a virtual service on page 359.

## Prerequisites for running virtual services

You can find information about the prerequisite tasks that you must complete before you configure a run of the virtual services on HCL OneTest™ Server.

## Prerequisites for running HTTP virtual services

You can find information about the tasks that you must complete before you configure a run of the HTTP virtual services on HCL OneTest™ Server.

### Conditions for running of HTTP virtual services

Before you run the HTTP virtual services from HCL OneTest™ Server, you must check for the following conditions and perform the actions indicated:

| If... | Then... |
|---|---|
| You want to route the HTTP traffic via the HTTP proxy either to the virtual service or live systems, based on routing rules. | You must set up the HTTP proxy. See Setting up the HTTP proxy on page 323. |
| You are working with HCL OneTest™ Server V10.1.3, which is installed on Ubuntu. | HTTP virtual services are exposed via host names are of the following form:<br>`in-<unique_id>.<INGRESS_DOMAIN>`<br>**Note:** The DNS used by a computer that runs a client application or the HTTP proxy that routes traffic to the virtual service needs to resolve such host names to the IP address of the ingress domain.<br>You must perform the following actions that depend on the method you have used to set up the ingress domain:<br>• If you used `nip.io` to form the ingress domain, then that virtual services host names are automatically resolved to the IP address of the ingress domain.<br>For example, if you used `nip.io` to form the ingress domain as follows:<br>`10.1.2.3.nip.io` |

| If... | Then... |
|---|---|
| | The virtual services host names are resolved by `nip.io` to the IP address as `10.1.2.3`.<br>• If your ingress domain does not use `nip.io` then you must ensure that the DNS in use by the client application or the HTTP proxy can resolve host names of the virtual services to the IP address of the ingress domain. For example, if the ingress domain is as follows:<br>`myhost.mycom.com`<br>The DNS must resolve `*.myhost.mycom.com` to the IP address of `myhost.mycom.com`. |

Related information

Setting up the HTTP proxy on page 323

Running HTTP virtual services without using proxies on page 340

## Prerequisites for running virtualized Istio based services in HCL OneTest™ Server

You can find information about the tasks that you must complete before you run virtualized Istio based services in HCL OneTest™ Server.

### Configuring virtualized Istio based services in HCL OneTest™ Server

When you intend to virtualize services in the Istio mesh in HCL OneTest™ Server, you can create stubs in HCL OneTest™ API.

When you create the Istio stubs in HCL OneTest™ API, you must ensure that you are aware of the fully qualified domain name (FQDN) or the short name of the service that you want to virtualize in the Kubernetes cluster that hosts HCL OneTest™ Server.

The FQDN of the service is as follows:

```
<service-name>.<namespace>.<svc>.<cluster-name>
```

The short name of the service is as follows:

```
<service-name>
```

For example, if you want to virtualize the HTTP traffic in the service named as *reviews* that runs in a namespace named as *bookinfo*, and in the resource named as *svc* in the Kubernetes cluster named as *mycluster*, then the FQDN of the service is as follows:

```
reviews.bookinfo.svc.mycluster
```

The short name of the service is as follows:

```
reviews
```

You must enter the host of the service that you want to virtualize, in the **Host** field of the physical transport in HCL OneTest™ API. You can provide either the FQDN or the short name of the service.

**Note:** The pass-through action in the stubs is not supported.

### Considerations for running of stubs that virtualize Istio services

Before you run the stubs that virtualize services in namespaces contained in the HCL OneTest™ Server Kubernetes cluster, you must install HCL OneTest™ Server with the *demo* configuration.

When you install HCL OneTest™ Server by using the *demo* configuration, then HCL OneTest™ Server is preconfigured to use the **bookinfo** namespace as the virtualization namespace.

**Note:** You can change or add virtualization namespaces by adding the helm installation argument execution.istio.namespaces along with a list of namespaces in a helm command.

After you author the virtualized Istio based services, you must commit the virtual service resources to a repository and add the repository to the project on HCL OneTest™ Server. You can configure a run of the virtualized Istio service. See Configuring a run of a virtual service on page 328.

Related reference

Installation of the server software on page 35

## Setting up the HTTP proxy

When you want to run HTTP virtual services on HCL OneTest™ Server, you can set up an HTTP proxy to route requests to the virtual service. You must configure the HTTP proxy with the HCL OneTest™ Server URL and offline user token generated from HCL OneTest™ Server so that the HTTP proxy can register with HCL OneTest™ Server.

### Before you begin

You must have installed the proxy component from HCL® Quality Server of the same version as that of HCL OneTest™ Server.

### About this task

You must edit the `registration.xml` file that is located in the `<QS_install_directory>\httptcp` directory for the following attributes:

- **server base-url**
- **security-token**

You must enter the HCL OneTest™ Server URL and offline user token as the values for the attributes. After you restart the proxy, the HTTP proxy registers with HCL OneTest™ Server. You can view the registered intercepts from the **Infrastructure > Agents and Intercepts** page.

1. Perform the following steps in HCL OneTest™ Server:
   a. Log in to HCL OneTest™ Server.
   b. Generate the offline user token.
   c. Copy the offline user token to a text file.
2. Perform the following steps to edit the `registration.xml` file that is located in the `<QS_install_directory>\httptcp` directory:

   a. Use a text editor to open the `registration.xml` file.

   b. Enter the HCL OneTest™ Server URL and offline user token generated from HCL OneTest™ Server in the following setting:

   ```
   <server base-url="<HCL OneTest™ Server_URL>" security-token="<Offline_User_Token>" />
   ```

   **Note:** If an entry exists, you must either comment out that entry and add a new one or replace the contents.

   c. Save and close the file.

   d. Restart the HTTP proxy.

3. Perform the following steps in HCL OneTest™ Server:

   a. Log in to HCL OneTest™ Server.

   b. Click **Infrastructure > Agents and Intercepts** to view the registered intercepts.

   The HTTP proxies that you configured with your offline user token are displayed.

### Results

You have successfully configured the HTTP proxy and viewed that it is registered with HCL OneTest™ Server.

### What to do next

You can perform the following tasks:

- Read for the prerequisite tasks that you must perform before you can run the HTTP stubs. See Prerequisites for running HTTP virtual services on page 321.
- Run the virtual service resource. See Configuring a run of a virtual service on page 328.
- View the routing rules for the intercept. See Viewing routing rules of the virtual services on page 355.
- View the routing rules for the intercept in a running instance of the virtual service. See Viewing configurations of running instances of virtual services on page 348.

## Viewing intercepts that are registered with HCL OneTest™ Server

After you install the HTTP proxies and configure them with your offline user token, the proxies register with HCL OneTest™ Server as intercepts and you can view the registered intercepts on the **Agents and Intercepts** page.

### Before you begin

You must have completed the following tasks:

- Installed the HTTP proxies of the same version as the version of HCL OneTest™ Server.
- Configured the HTTP proxy with your offline user token so that the proxy can register as an intercept with HCL OneTest™ Server. See Setting up the HTTP proxy on page 323.

1. Log in to HCL OneTest™ Server.
2. Click **Infrastructure > Agents and Intercepts** in the navigation pane.

   The **Agents and Intercepts** page is displayed.

   You can view the agents, intercepts, or Docker hosts that are registered with HCL OneTest™ Server.
3. You can view the intercepts that you own in any of the following ways:
   - Search for the intercept by entering the name of the intercept in the **Search** field.

     **Note:** You can enter either the full name or any text that is in the name. The search is enabled for case-sensitive text that you can enter.
   - Sort the **Type** column to show the items sorted with the intercepts in the rows at the top of the table.
   - Sort the **Agents** column and identify the intercepts by their names or by their owner.

   You can view the following details about the intercepts that are registered with HCL OneTest™ Server:
   - `Any project` is displayed in the **Projects** column indicating that the intercept can be used by virtual services in any project.
   - The status of the intercept is displayed in the **Status** column.

### Results

You have viewed the registered intercepts from the **Agents and Intercepts** page.

### What to do next

You can start the virtual services that use the registered intercepts. See Configuring a run of a virtual service on page 328.

## Viewing virtual service resources

When project repositories contain virtual service resources and you want to view the resources so that you can run them, you can view the virtual services from the **Resources** page in HCL OneTest™ Server.

**Before you begin**

You must have completed the following tasks:

- Created virtual services that use the HTTP or IBM® WebSphere® MQ transport for tests in HCL OneTest™ API, committed the test resources to the remote repository, and added the repository to your project.
- Been assigned the *Viewer*, *Tester*, or *Owner* role in the project so that you can view the virtual service resources that are in the project repositories.

**About this task**

After you have committed the test assets and resources to the remote repositories and added the repositories to projects on HCL OneTest™ Server, you can go to the **Resources** page and perform the following tasks:

- View the virtual service resources that are contained in any branch of the repositories that are added to your project.
- Create or use saved filters for viewing virtual service resources.
- Run or start an instance of the virtual service resource.
- View the number of instances of the virtual service resources that are running.
- View all instances of a running virtual service on the **Instances** page by clicking the **Show in instances page** icon ↗.

1. Log in to HCL OneTest™ Server.
2. Open the project that contains the virtual service resources in the test assets by clicking **Projects > My Projects > *project_name***.
3. Click **Virtualization > Resources** in the navigation pane.

   The **Resources** page is displayed.
4. Complete the steps for the task that you want to perform as listed in the following table:

| Task | Action |
|---|---|
| Viewing the virtual service resources that are contained in a branch of any of the repositories that are added to your project. | Select the branch of the repository that contains the virtual services that you want to run from the list in the **Branch** field.<br>All virtual services in the selected branch are displayed on the **Resources** page. |
| Creating or using saved filters for viewing specific instances of virtual service resources. | To create a query, perform the following actions:<br>a. Click **New filter**.<br>b. Create a rule with an appropriate operator.<br>c. Select criteria such as *Name*, *Path*, *Instance number*, or *Instance state*.<br>d. Select the condition, and then enter the value for the criteria. |

| Task | Action |
|---|---|
| | e. Apply the filter query.<br>The virtual services that match the filter criteria are displayed.<br>f. Save the filter query for retrieving it from the saved filters list.<br>To use a saved filter query, perform the following actions:<br>**Note:** To open the filter query, you must have created and saved a filter query.<br>a. Click the **Open filters** icon<br>b. Select the saved filter.<br>c. Apply the filter.<br>The virtual services that match the filter criteria in the filter that is applied are displayed. |
| Running an instance of the virtual service resource. | After you identify the virtual service that you want to run, perform the following actions:<br>a. Click the **Execute** icon in the row of the identified virtual service.<br>The **Execute virtual service** dialog box is displayed.<br>b. Run the virtual service with the settings that are configured in the resource by clicking **Execute** or modify any of the settings, and then click **Execute**.<br>An instance of the virtual service starts to run. The virtual service is displayed with the state as `Running` along with the number of instances that are running in the **Active instances** column. |
| Viewing the number of instances of the virtual service resources that are running. | If any of the virtual services in the project repositories are started by any project member with the *Tester* or *Owner* role, you can view the number of instances that are running or active from the **Resources** page.<br>The virtual services are displayed with the state as `Running` along with the number of instances that are running, in the **Active instances** column. |
| Viewing all instances of a running virtual service. | Click the **Show in instances page** icon in the row of the virtual service on the **Resources** page. |

| Task | Action |
|---|---|
| | The **Instances** page is displayed with all the running instances of a particular virtual service.<br>You can also view the number of requests that are received by the virtual service. The number of requests is displayed as the number of hits when you hover the cursor over the text in the **Activity** column. |

### Results

You have viewed virtual services from the **Resources** page on HCL OneTest™ Server.

Related information

Configuring a run of a virtual service on page 328

## Configuring a run of a virtual service

When you have virtual service resources that are in the repositories added to your project, you must configure a run of the virtual service to start an instance of the virtual service on HCL OneTest™ Server.

### Before you begin

You must have completed the following tasks:

- Been assigned the *Tester* or *Owner* role in the project to run virtual services.
- Created virtual services that use the HTTP or IBM® WebSphere® MQ transport for tests in HCL OneTest™ API, committed the test resources to the remote repository, and added the repository to your project.
- Completed the prerequisite tasks before you configure a run of the HTTP stubs. See Prerequisites for running HTTP virtual services on page 321.
- Read the prerequisite information about virtualizing the Istio based services before you can run the virtualized service on HCL OneTest™ Server. See Prerequisites for running virtualized Istio based services in HCL OneTest Server on page 322.

### About this task

You can start the following types of stubs contained in API Suites from the **Resources** page on HCL OneTest™ Server:

- Virtual services that utilize the WebSphere® MQ transport.
- Virtual services that utilize the HTTP transport.
- Virtual services that virtualize the Istio based services. You can run virtual services for the following types of requests received or sent by the Istio service mesh:

  - Requests received by services in the Istio service mesh.
  - Requests sent from namespaces in the Istio service mesh to external services that are not in the Istio service mesh.

1. Log in to HCL OneTest™ Server.
2. Open the project that contains the virtual service resources in the test assets by clicking **Projects > My Projects > *project_name***.
   The **Overview** page is displayed.
3. Click **Virtualization > Resources** in the navigation pane.

   The **Resources** page is displayed.
4. Select the branch of the repository that contains the virtual services that you want to run from the list in the **Branch** field.

   All virtual services in the selected branch are displayed on the **Resources** page.
5. Identify the virtual service that you want to run by completing any of the following steps:
   - Search for the virtual service by entering the name or the path of the virtual service in the repository in the **Search** field box.
   - Create a filter query by using the **New filter** option and complete the following steps:
     a. Create a rule with an appropriate operator.
     b. Select criteria such as *Name*, *Path*, *Instance number*, or *Instance state*. Select the condition and enter the value for the criteria. Apply the filter query.

        The virtual services that match the filter criteria are displayed.
     c. Save the filter query for retrieving it from the saved filters list.
   - Retrieve a saved filter by using the **Open filters** icon by completing the following steps:

     **Note:** To open the filter query, you must have created and saved a filter query.

     a. Select the saved filter.
     b. Apply the filter.

        The virtual services that match the filter criteria in the filter that is applied are displayed.
6. Click the **Execute** icon in the row of the identified virtual service.

   The **Execute virtual service** dialog box is displayed.
7. Select the version of the virtual service that is in the repository that you want to start by performing any of the following actions:

- Expand the list in the **Version** field, find the version of the test resources, and then select the version.

  Use the following details about the version of the test resources that are displayed to identify the version that you want:

  - Commit message.
  - Tags labeled for the version committed.
  - The user who committed the version to the repository.
  - Relative time of the commit. For example, *2 hours ago* or *3 days ago*.

  The list displays the versions of the test resources committed by all users to the branch in the repository. The versions are arranged with the latest version committed followed by the versions committed previously.

- Expand the list in the **Version** field, and then search for the version that you want to select by entering a partial or the complete commit message of that version.

  The version that matches the search criteria is displayed and it is selected for the test run.

8. Select the environment that was used to bind the physical and logical resource in the API project, in the **ENVIRONMENT** tab.

   **Important:** The configuration that you set for the run in the **Execute virtual service** dialog box is preserved when you run the same virtual service again. The configurations that you set are not available to other members when they want to run the virtual service. For example, if you selected an environment, the same environment is selected when you run the virtual service again.

9. Enter a label, if required.

   A label that you enter for the test run that helps you to identify the virtual service instance on the **Instances** page. The label that you entered is displayed for the virtual service under the **Labels** column on the **Instances** page. After you have created a label, any member of the project can use that label.

10. Follow the instructions if you want to modify the configurations for the behavior of the virtual service:

    a. Click the **BEHAVIOR** tab, if it is not already open.

    b. Configure or change the settings for the following options:

    **Note:** The settings displayed are the settings that were configured for the virtual service when it was authored in HCL OneTest™ API.

| Option | Description |
|---|---|
| Performance | The following settings are available for handling of requests by the virtual service: |

<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td><b>Optimize performance</b></td><td>If enabled, attempts are made to reduce the amount of processing between the time the virtual service receives a request and the time it sends a response. Specific optimizations depend on the message contents, as in the following examples:
<ul>
<li>When the virtual service receives requests, all validations are disabled, and for all XML payloads, the store and filter actions are converted to use XPath expressions.</li>
<li>When the virtualization sends responses, any store actions set on a message are disabled, and any XML content is collapsed when the virtual service is compiled instead of being collapsed every time that a response is sent.</li>
</ul>
The optimization is disabled as the default action for this setting and you can enable the performance optimization if you fully understand the implications of optimizing the performance.</td></tr>
<tr><td><b>Threads</b></td><td>The maximum number of threads used in processing requests received by the virtual service. The default number of threads is <i>10</i>.</td></tr>
</table>
</td></tr>
<tr><td>Operation</td><td>Specifies an operation referenced by the virtual service.</td></tr>
<tr><td>Response time</td><td>
<p>Specifies the response time behavior for responses sent by the virtual service for the selected operation.</p>
<p>You can modify the value by selecting a value from the following options:</p>
<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td>No delay</td><td>Select this option for a response with no delay.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td>Minimum delay</td><td>Select this option for a delay in the response by entering the required delay in milliseconds (ms).</td></tr>
<tr><td>Uniform distribution</td><td>Select this option for a uniformly distributed response time by specifying the minimum and maximum delay in milliseconds (ms).</td></tr>
<tr><td>Gaussian distribution</td><td>Select this option for a response time with Gaussian distribution by specifying the minimum and maximum delay in milliseconds (ms).</td></tr>
</table>
</td></tr>
<tr><td>Passthrough</td><td>
Specifies the pass through behavior for the selected operation when requests are not handled within the virtual service.<br>
You can modify the value by selecting a value from the following options:
<table>
<tr><td>Discard</td><td>This option stops the system under test from receiving the intercepted message. This option can disrupt the calling system. For example, the system might time out while it waits for a reply.</td></tr>
<tr><td>Pass Through</td><td>This option passes the intercepted message to the system under test, with an optional delay.</td></tr>
<tr><td>Simulate Error</td><td>This option returns an error to the calling system. The message is not passed to the system under test.</td></tr>
</table>
</td></tr>
</table>

11. Follow the instructions, if the virtual service references datasets.
    - You can use the dataset referenced in the asset.
    - You can choose to override the dataset with another data source. If alternative data sources are available, select from the set of overrides available.
12. Follow the instructions if the virtual service requires a variable that must be passed at run time.

    a. Click the **VARIABLES** tab, if it is not already open.

    b. Choose one of the following methods to add the variables:

- To add new variables manually, click the **Add Variable** icon ⊕, enter the name, and value of the variable.
- To add new variables from your local computer or from the Git repository that is associated with your server project, click the **Upload** icon ⬆ and select the **Upload from local system** or **Browse from server** to select the variable file.

  **Note:** You must have created a file with the variables before you can select the file.

13. Click **Advanced** to make the following advanced configurations:

    a. Enter any JVM arguments that must be passed at run time in the **JVM Arguments** field.

    **Note:** Each JVM argument should be separated with white space.

    For example, you can set a maximum Java heap size.

    b. Enter the environment variables that must be passed at run time in the **Environment Variables** field, if applicable.

    For example, enter the environment variables when the third-party libraries that are used in the run refer to the environment variables for configuration.

    c. Select the stub logging level for the virtual service from the following options in the **Logging** list:

| Option | Description |
|---|---|
| None | Specifies that the virtual service does not write log messages. |
| Normal | Specifies that the virtual service writes informational messages. |
| Debug | Specifies that the virtual service writes informational and debugging messages. |

    d. Enter other configuration options as parameters and their values in the **Additional Configuration Parameters** fields, if applicable.

    You can refer to the additional parameters that you can use for virtual services from the topic in the related links. For example, if you want to start the virtual service to run in a new container, you can specify the following parameter and its value:

| Parameter name | Value |
|---|---|
| **execution.stub.dedicated.container** | **true** |

**Note:** Click **Add** to add additional parameters.

**Note:** You must separate the arguments or variables with a white space when you enter them in the same line or start each argument or variable on a new line.

The default value for the fields for the advanced settings is null or an empty field.

**Notes:**

- If you have configured some or all of the settings for the current run, and you do not want to continue with those settings, you can reset the settings by clicking **Reset**.
- If you want to repeat a run and do not want to use the saved settings from a previous run, you can reset all the saved settings to their default values by clicking **Reset**.

14. Click **Execute**.

**Results**

You have started a virtual service from the **Resources** page on HCL OneTest™ Server.

**What to do next**

After you start a run of a virtual service, you can view the status of the virtual service in any of the following ways:

- On the **Resources** page, the virtual services are displayed with the state as `Running` along with the number of instances that are running in the **Active instances** column.
- On the **Resources** page, you must click the **Show in instances page** icon ↗ in the row of the virtual service to view all the running instances of a particular virtual service on the **Instances** page.
- Go to the **Instances** page by clicking **Virtualization > Instances**. You can view instances of all the virtual services that are running and are displayed with the state as `Running` under the **State** column. You can also view the requests that are received by the virtual service. The number of requests is displayed as number of hits when you hover the cursor over the text in the **Activity** column.

You can view the details, configuration settings, routing rules, usage statistics, or logs of a specific running instance of the virtual service. See Viewing configurations of running instances of virtual services on page 348.

You can modify the behavior or the logging level of a running instance of the virtual service that you started before running tests on them. See Modifying configurations of running instances of virtual services on page 353.

When you have completed testing the virtual service, you must stop the running virtual service. See Stopping a virtual service on page 359.

Related reference

Additional configuration parameters for virtual services on page 335

## Additional configuration parameters for virtual services

You can find information about the additional configuration parameters that you can set when you want to run virtual services on HCL OneTest™ Server.

When you use the server UI to start the virtual service, some of the parameters are handled by fields in the **Execute virtual service** dialog box.

You can enter the other parameters as a *name-value* pair in the **Additional configuration parameters** field that is displayed in the **Advanced** settings panel of the **Execute virtual service** dialog box.

The following additional configuration parameters are supported when you configure a run for a virtual service.

| Requirement | Configuration parameter name | Supported values | An example value | Result of using the example value |
|---|---|---|---|---|
| To specify a unique identifier for use in the generated host name | `stub.ingress.host.identifier` | Any <custom_id> that you specify. | *mystub* | See Running HTTP virtual services without using proxies on page 340, to know how the identifier you specified is used. |
| To specify whether a virtual service instance should be run in a dedicated container. | `stub.dedicated.container` | • *true*<br>• *false*<br>**Note:** The default value is *false*. | *true* | A dedicated container is used to run a single virtual service instance. |
| To specify whether performance optimization is enabled | `stub.performance.optimize` | • *true*<br>• *false* | *true* | The performance optimization is enabled for the virtual service. |

| Requirement | Configuration parameter name | Supported values | An example value | Result of using the example value |
|---|---|---|---|---|
| for the virtual service instance. The value set overrides the setting from the stub definition.<br>This option can be set in the **Behavior** tab from the UI. | | **Note:** The default value is *false*. | | |
| To specify the number of threads to be used for processing requests for the virtual service instance. The value set overrides the setting from the stub definition.<br>This option can be set in the **Behavior** tab from the UI. | `stub.worker.thread-.count` | Any number that you specify as number of threads to use. | *15* | 15 threads are used for processing requests for the virtual service instance. |
| To specify the stub logging level for the virtual service instance. The value set overrides the setting from the stub definition.<br>This option can be set in the **Advanced** settings from the UI. | `stub.logging.level` | • *none*<br>• *normal*<br>• *debug* | *debug* | The virtual service writes informational and debugging messages to the container log. |
| To specify the pass-through behaviour type of the virtual service instance. The value set overrides | `stub.pass.through-.behavior` | Valid values depend on the stub operation transport type but can be as follows: | *simulate_error* | This option returns an error to the calling system. The message is not passed to the system under test. |

| Requirement | Configuration parameter name | Supported values | An example value | Result of using the example value |
|---|---|---|---|---|
| the setting from the stub definition.<br>This option can be set in the **Behavior** tab from the UI. | | • *simulate_error*<br>• *discard*<br>• *pass_through* | | |
| To specify the delay in passing through from the virtual service instance. The value set overrides the setting from the stub definition. | `stub.pass.through.delay.period` | Any number of seconds in milliseconds. | *10000* | A delay of 10 seconds in passing through from the virtual service instance. |
| To specify the status code to use in the virtual service instance response when simulating an error. The value set overrides the setting from the stub definition.<br>**Note:** Applies to HTTP stubs.<br>This option can be set in the **Behavior** tab from the UI. | `stub.pass.through.status.code` | You must specify a valid HTTP status code as the value. For example, `503`. | *500* | The HTTP error code of `500` is sent in the response if the virtual service simulates an error when passing through. |
| To specify the reason phrase to use in the virtual service instance response when simulating an error. The value set overrides the setting from the stub definition. | `stub.pass.through.reason.phrase` | You must specify the text of the status message as the value. | *my custom error msg* | The text that you specify is set as the HTTP reason phrase in the response if the virtual service simulates an error when passing through. |

| Requirement | Configuration parameter name | Supported values | An example value | Result of using the example value |
|---|---|---|---|---|
| **Note:** Applies to HTTP stubs.<br>This option can be set in the **Behavior** tab from the UI. | | | | |
| To specify the type of response-time behavior that is required for the virtual service instance. The value set overrides the setting from the stub definition.<br>This option can be set in the **Behavior** tab from the UI. | `stub.response.time.type` | • *no_delay*<br>• *minimum_delay*<br>• *gaussian*<br>• *uniform*<br>• *performance_profile*<br>**Note:** *performance_profile* is only valid if the stub operation in the stub definition is defined to use a performance profile. | *no_delay* | The virtual service sends a response without any delay. |
| To specify the minimum time for the response time behaviour required for the virtual service instance. The value set overrides the setting from the stub definition. | `stub.response.time.minimum` | Any number of seconds in milliseconds. | *5000* | The virtual service sends a response after a minimum delay of 5 seconds. |

| Requirement | Configuration parameter name | Supported values | An example value | Result of using the example value |
|---|---|---|---|---|
| This option can be set in the **Behavior** tab from the UI. | | | | |
| To specify the maximum time for the response time behaviour required for the virtual service instance. The value set overrides the setting from the stub definition.<br>This option can be set in the **Behavior** tab from the UI. | `stub.response.time.maximum` | Any number of seconds in milliseconds. | *20000* | The virtual service must send a response after a maximum delay of 20 seconds. |

### Virtual services where the stub definition refers to multiple operations

When a stub definition has two or more events that refer to different operations and each operation has a different behavior, you must specify the *<operation_id>* suffixed to the configuration parameter name in the **Additional configuration parameters** field.

**Note:** The *<operation_id>* is the internal ID of the operation that is found in the **Documentation** tab when you open the operation in HCL OneTest™ API.

When the stub definition refers to multiple operations and you want to start a run of a virtual service, you must append the *<operation_id>* to the *<parameter_name>* in the **Additional configuration parameters** field.

The configuration parameter names that you must use for the virtual service, which is associated with a particular operation are as follows:

- `stub.pass.through.behavior.<operation_id>`
- `stub.pass.through.delay.period.<operation_id>`
- `stub.pass.through.status.code.<operation_id>`
- `stub.pass.through.reason.phrase.<operation_id>`
- `stub.response.time.type.<operation_id>`
- `stub.response.time.<operation_id>`
- `stub.response.time.maximum.<operation_id>`

For example, if the *<operation_id>* is *7281b750:162483a09f1:-7e68*, the parameter name for the `stub.response.time.type` option that you must specify is `stub.response.time.type.7281b750:162483a09f1:-7e68`.

Related information

Configuring a run of a virtual service on page 328

## Running HTTP virtual services without using proxies

When you run HTTP virtual services on HCL OneTest™ Server, the endpoint on which the virtual service is exposed externally uses a host name that includes a generated ID. If a client application is required to call the virtual service directly via its external host name and not via the HTTP proxy, then you might want to determine the host name to configure the client first before you start the virtual service.

### Before you begin

You must have completed the following tasks:

- Been assigned the *Tester* or *Owner* role in the project to run the virtual services.
- Created a project on HCL OneTest™ Server and added the repository that contains the HTTP virtual services.
- Completed the prerequisite tasks before you start HTTP virtual services. See Prerequisites for running HTTP virtual services on page 321.
- Selected the branch of the Git repository to view test resources on the **Execution** page.

### About this task

When you installed HCL OneTest™ Server on Ubuntu, the default HTTP stub endpoints are of the following form:

```
in-<unique_id>.<ingress_domain>
```

For example, with an ingress domain as **10.1.2.3.nip.io**, a stub endpoint might be as follows:

```
in-cd3a755635574c6fa7e264911301821a.10.1.2.3.nip.io
```

When you installed HCL OneTest™ Server on OpenShift, the default HTTP stub endpoints are of the following form:

```
<ingress_domain_first_label>-<unique_id>.<remainder_of_ingress_domain>
```

For example, with an ingress domain as **ots.mycluster-123456-0000.region.containers.appdomain.cloud**, a stub endpoint might be as follows:

```
ots-cd3a755635574c6fa7e264911301821a.mycluster-123456-0000.region.containers.appdomain.cloud
```

The <unique_id> is generated when the stub is started. The full host name can be viewed in the **Routing to** field of the routing rule created for the stub on the **Intercept Rules** page. When traffic is routed to the stubs via the HTTP proxy, a client application is unaware of this host name. If the client application is required to call the stubs directly, it must be aware of the stub host name. To allow a host name to be known before you start the stub, you can replace the generated ID with a value that you specify.

**Notes:**

- The value of the **<unique_id>** that you specify is used as the **<unique_id>** for the stub.
- The **<unique_id>** must start and end with an alphanumeric character. The characters supported include a-z, 0-9, and a hyphen. No other characters can be used in the **<unique_id>**.
- The **<unique_id>** must be unique to a stub that is running.

  **Restriction:** You must not assign the same **<unique_id>** to another stub that is running simultaneously. Doing so might result in an undefined behavior of the stub.

For example, on Ubuntu, if you specify the value of the *<custom_id>* for the stub as `stub123-test` and HCL OneTest™ Server has an ingress domain as **10.1.2.3.nip.io**. Then the host name of the stub endpoint is as follows:

```
in-stub123-test.10.1.2.3.nip.io
```

1. Log in to HCL OneTest™ Server.
2. Open the project that contains the virtual service resources in the test assets by clicking **Projects > My Projects > *project_name***.
   The **Overview** page is displayed.
3. Click **Virtualization > Resources** in the navigation pane.

   The **Resources** page is displayed.
4. Select the branch of the repository that contains the virtual services that you want to run from the list in the **Branch** field.

   All virtual services in the selected branch are displayed on the **Resources** page.
5. Identify the virtual service that you want to run.
6. Click the **Execute** icon ▶ in the row of the identified virtual service.

   The **Execute virtual service** dialog box is displayed.
7. Select the version of the virtual service that is in the repository that you want to start by performing any of the following actions:

   - Expand the list in the **Version** field, find the version of the test resources, and then select the version.

     Use the following details about the version of the test resources that are displayed to identify the version that you want:
     - Commit message.
     - Tags labeled for the version committed.
     - The user who committed the version to the repository.
     - Relative time of the commit. For example, *2 hours ago* or *3 days ago*.

The list displays the versions of the test resources committed by all users to the branch in the repository. The versions are arranged with the latest version committed followed by the versions committed previously.

- Expand the list in the **Version** field, and then search for the version that you want to select by entering a partial or the complete commit message of that version.

  The version that matches the search criteria is displayed and it is selected for the test run.

8. Select the environment that was used to bind the physical and logical resource in the API project, in the **ENVIRONMENT** tab.

   **Important:** The configuration that you set for the run in the **Execute virtual service** dialog box is preserved when you run the same virtual service again. The configurations that you set are not available to other members when they want to run the virtual service. For example, if you selected an environment, the same environment is selected when you run the virtual service again.

9. Enter a label, if required.

   A label that you enter for the test run that helps you to identify the virtual service instance on the **Instances** page. The label that you entered is displayed for the virtual service under the **Labels** column on the **Instances** page. After you have created a label, any member of the project can use that label.

10. Follow the instructions if you want to modify the configurations for the behavior of the virtual service:

    a. Click the **BEHAVIOR** tab, if it is not already open.

    b. Configure or change the settings for the following options:

    **Note:** The settings displayed are the settings that were configured for the virtual service when it was authored in HCL OneTest™ API.

<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td>Performance</td><td>The following settings are available for handling of requests by the virtual service:
<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td><b>Optimize performance</b></td><td>If enabled, attempts are made to reduce the amount of processing between the time the virtual service receives a request and the time it sends a response. Specific optimizations depend on the message contents, as in the following examples:</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td></td><td><ul><li>When the virtual service receives requests, all validations are disabled, and for all XML payloads, the store and filter actions are converted to use XPath expressions.</li><li>When the virtualization sends responses, any store actions set on a message are disabled, and any XML content is collapsed when the virtual service is compiled instead of being collapsed every time that a response is sent.</li></ul>The optimization is disabled as the default action for this setting and you can enable the performance optimization if you fully understand the implications of optimizing the performance.</td></tr>
<tr><td><b>Threads</b></td><td>The maximum number of threads used in processing requests received by the virtual service. The default number of threads is <i>10</i>.</td></tr>
</table>
</td></tr>
<tr><td>Operation</td><td>Specifies an operation referenced by the virtual service.</td></tr>
<tr><td>Response time</td><td><p>Specifies the response time behavior for responses sent by the virtual service for the selected operation.</p><p>You can modify the value by selecting a value from the following options:</p>
<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td>No delay</td><td>Select this option for a response with no delay.</td></tr>
<tr><td>Minimum delay</td><td>Select this option for a delay in the response by entering the required delay in milliseconds (ms).</td></tr>
<tr><td>Uniform distribution</td><td>Select this option for a uniformly distributed response time by specifying the minimum and maximum delay in milliseconds (ms).</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td></td><td><table><tr><th>Option</th><th>Description</th></tr><tr><td>Gaussian distribution</td><td>Select this option for a response time with Gaussian distribution by specifying the minimum and maximum delay in milliseconds (ms).</td></tr></table></td></tr>
<tr><td>Passthrough</td><td>Specifies the pass through behavior for the selected operation when requests are not handled within the virtual service.<br>You can modify the value by selecting a value from the following options:<table><tr><td>Discard</td><td>This option stops the system under test from receiving the intercepted message. This option can disrupt the calling system. For example, the system might time out while it waits for a reply.</td></tr><tr><td>Pass Through</td><td>This option passes the intercepted message to the system under test, with an optional delay.</td></tr><tr><td>Simulate Error</td><td>This option returns an error to the calling system. The message is not passed to the system under test.</td></tr></table></td></tr>
</table>

11. Click **Advanced** to make the following advanced configurations:

    a. Enter the following configuration options as parameters and their values in the **Additional Configuration Parameters** fields.

| Parameter name | Value |
|---|---|
| **execution.ingress.host.identifier** | **<custom_id>** |

12. Click **Execute**.

**Results**

You have started an HTTP virtual service that does not use a proxy. The virtual service exposes the host name that contains a custom ID that you specified.

**What to do next**

After you start a run of a virtual service, you can view the status of the virtual service in any of the following ways:

- On the **Resources** page, the virtual services are displayed with the state as `Running` along with the number of instances that are running in the **Active instances** column.
- On the **Resources** page, you must click the **Show in instances page** icon ↗ in the row of the virtual service to view all the running instances of a particular virtual service on the **Instances** page.

- Go to the **Instances** page by clicking **Virtualization > Instances**. You can view instances of all the virtual services that are running and are displayed with the state as `Running` under the **State** column. You can also view the requests that are received by the virtual service. The number of requests is displayed as number of hits when you hover the cursor over the text in the **Activity** column.

## Viewing running instances of virtual services

After you start virtual services from the **Resources** page, you can view the instances of the virtual services that are running from the **Instances** page in HCL OneTest™ Server. You can view the details, configuration settings, routing rules, usage statistics, or logs of a specific running instance of the virtual service. You can also stop the running instance.

### Before you begin

You must have completed the following tasks:

- Started the virtual service resources as a *Tester* or *Owner* from the **Resources** page.
- Been assigned the *Viewer*, *Tester*, or *Owner* role in the project so that you can view the virtual services that are running.

### About this task

After you started the virtual services from the **Resources** page on HCL OneTest™ Server, you can go to the **Instances** page and you can perform the following tasks:

- View all running instances of the virtual services that are contained in the project repositories.
- Search for a specific instance of the virtual service for which you want to view the details.
- Create or use saved filters for viewing specific running instances of the virtual service.
- View the details, configuration settings, routing rules, usage statistics, or logs of a specific running instance of the virtual service.
- Change the behavior or logging level of a running instance of the virtual service before you run tests on the virtual service.
- Stop a specific running instance of the virtual service.

1. Log in to HCL OneTest™ Server.
2. Open the project that contains the virtual services in the test assets by clicking **Projects > My Projects > *project_name***.
   The **Overview** page is displayed.
3. Click **Virtualization > Instances** in the navigation pane.

   The **Instances** page is displayed.

   You can view all running instances of the virtual services that you started and the instances that were started by other members of the project. You can find the following information about the running instances that are displayed on the **Instances** page.

- The name and path of the virtual service in the project repository are displayed in the **Name** column.
- The tags or labels added to the instance when the run was configured is displayed in the **Labels** column.
- The name of the environment associated with the operation in which the virtual service was created in HCL OneTest™ API is displayed in the **Environment** column.
- The state of the running instance is displayed as `Running` in the **State** column.
- The activity of the running instance of the virtual service based on the number of requests received in the previous hour is displayed in the **Activity** column. The activities are classified and displayed as indicated in the following table:

| **Activity load** | **Activity displayed as** |
|---|---|
| Received *0* requests | `None` |
| Received *1 - 10* requests | `Low` |
| Received *11 - 100* requests | `Mid` |
| Received *101 - 1000* requests | `High` |
| Received more than *1000* requests | `Very High` |

- The **Stop** icon ⏹ is displayed in the **Actions** column and stops the running instance of the virtual service, when clicked.

4. Complete the steps for the task that you want to perform as listed in the following table:

| **Task** | **Action** |
|---|---|
| Searching for a specific instance of the virtual service for which you want to view the details or want to stop the instance. | Enter either the complete text or part of the text in the name or the path of the virtual service that is in the repository in the **Search** field.<br>The instances of the virtual services that match the text searched are displayed. |
| Creating or using saved filters for viewing specific instances of virtual service resources. | To create a query, perform the following actions:<br>a. Click **New filter**.<br>b. Create a rule with an appropriate operator.<br>c. Select criteria such as *Name*, *Path*, *Instance number*, or *Instance state*.<br>d. Select the condition, and then enter the value for the criteria.<br>e. Apply the filter query.<br>The virtual services that match the filter criteria are displayed.<br>f. Save the filter query for retrieving it from the saved filters list.<br>To use a saved filter query, perform the following actions: |

| Task | Action |
|---|---|
| | **Note:** To open the filter query, you must have created and saved a filter query.<br>a. Click the **Open filters** icon ☰<br>b. Select the saved filter.<br>c. Apply the filter.<br>The virtual services that match the filter criteria in the filter that is applied are displayed. |
| Viewing the details, configuration settings, routing rules, usage statistics, or logs of a specific running instance of the virtual service. | After you identify the virtual service instance for which you want to view the details, perform the following actions:<br>a. Click the **Expand** icon › in the row of the identified instance of the virtual service.<br>The **Details** panel of the instance is displayed. You can find the following details about the instance displayed:<br>▪ Name<br>▪ Path<br>▪ Labels<br>▪ API Environment<br>▪ Version<br>▪ Started at<br>▪ Started by<br>▪ Associated components<br>▪ Documentation<br>b. Click **Behavior** to expand the panel and view the behavior that was configured for the virtual service.<br>c. Click **Routing Rules** to expand the panel and view the routing rules that were configured in the virtual service.<br>d. Click **Statistics** to expand the panel and view the graphical representation of the activities performed by the virtual service.<br>e. Click **Diagnostics** to expand the panel and view the logging configuration of the virtual service and the links to the logs. |
| Changing the behavior or logging level of a running instance of the virtual service. | After you identify the virtual service instance for which you want to view the details, perform the following actions:<br>a. Click the **Expand** icon › in the row of the identified instance of the virtual service.<br>The **Details** panel of the instance is displayed. |

| Task | Action |
|---|---|
| | b. Click **Behavior** to expand the panel.<br>Perform the following steps to change the behavior configurations:<br>i. Click the **Edit** icon ✎.<br>ii. Select a different operation for the virtual service, if available.<br>iii. Select a different response time and enter the values for the delay you want to set.<br>iv. Click **apply**.<br>c. Click **Diagnostics** to expand the panel.<br>Perform the following steps to change the logging level configuration:<br>i. Click the **Edit** icon ✎.<br>ii. Select a different logging level.<br>iii. Click **apply**. |
| Stop a specific running instance of the virtual service. | Click the **Stop** icon ⏹ in the row of the virtual service instance. |

### Results

You have viewed running instances of the virtual services from the **Instances** page on HCL OneTest™ Server.

### What to do next

You can perform the following tasks:

- View the configurations of a running instance of the virtual service. See Viewing configurations of running instances of virtual services on page 348.
- Modify the behavior or logging level of a running instance of the virtual service before you run tests on the virtual service. See Modifying configurations of running instances of virtual services on page 353.
- Stop a running instance of the virtual service. See Stopping a virtual service on page 359.

## Viewing configurations of running instances of virtual services

After you start virtual services from the **Resources** page, you can view the details, configuration settings, routing rules, usage statistics, or logs of a specific running virtual service instance from the **Instances** page on HCL OneTest™ Server.

### Before you begin

You must have completed the following tasks:

- Been assigned the *Viewer*, *Tester*, or *Owner* role in the project so that you can view the virtual services that are running.
- Started the virtual service resources as a *Tester* or *Owner* from the **Resources** page.

**About this task**

After you have started the virtual services from the **Resources** page on HCL OneTest™ Server, you can go to the **Instances** page and view the information about a specific running virtual service instance.

1. Log in to HCL OneTest™ Server.
2. Open the project that contains the virtual services in the test assets by clicking **Projects > My Projects > *project_name***.

   The **Overview** page is displayed.
3. Click **Virtualization > Instances** in the navigation pane.

   The **Instances** page is displayed with all running instances of the virtual services that are in your project.
4. Identify the running instance of the virtual service of which you want to view the details, configuration settings, and other information.
5. Click the **Expand** icon > of the instance of the virtual service.

   The **Details** panel of the instance is displayed with the following details about the virtual service instance:

   **Note:** The information displayed pertains to the configuration settings that were configured in the virtual service at the time it was created or run.

| Option | Information displayed |
|---|---|
| Name | Displays the name of the virtual service. |
| Path | Displays the path of the virtual service resource in the project repository. |
| Labels | Displays the labels or tags that were entered for the virtual service when a run of the virtual service was configured. |
| API Environment | Displays the name of the environment associated with the operation in which the virtual service was created in HCL OneTest™ API or the environment selected for the run. |
| Version | Displays the version of the virtual service resource that is in the project repository that was started as an instance. |
| Started at | Displays the date and time when the virtual service instance started. |
| Started by | Displays the name of the project member who started the virtual service instance. |

| Option | Information displayed |
|---|---|
| Associated components | Displays the components in the system model that are associated with the virtual service. |
| Documentation | Displays the text that was entered on the **Documentation** tab of the **Stub Editor** when the virtual service was created in HCL OneTest™ API. |

The other panels that you can view are as follows:

- Behavior
- Routing Rules
- Statistics
- Diagnostics

6. Expand each panel by clicking the panel to view the information in that panel.

You can view the following information in the different panels:

<table>
<tr><th>Panel</th><th>Details</th></tr>
<tr><td>Behavior</td><td>The <b>Behavior</b> panel of the instance is displayed when expanded with the following details about the virtual service instance:<br><b>Note:</b> The information displayed pertains to the configuration settings that were configured in the virtual service at the time it was created or run.
<table>
<tr><th>Option</th><th>Information displayed</th></tr>
<tr><td>Performance</td><td>Displays the optimization setting and the maximum number of threads in use by the virtual service.</td></tr>
<tr><td>Operation</td><td>Displays the operations referenced by the virtual service. You can select each operation to view the related response time and the pass through settings.</td></tr>
<tr><td>Response time</td><td>Displays the response time option that is in use for the selected operation within the virtual service.</td></tr>
<tr><td>Passthrough</td><td>Displays the pass through option that is in use for the selected operation within the virtual service.</td></tr>
</table></td></tr>
<tr><td>Routing Rules</td><td>The <b>Routing Rules</b> panel of the instance is displayed when expanded.<br>The routing rules that relate to the virtual service are displayed.</td></tr>
</table>

<table>
<tr><th>Panel</th><th>Details</th></tr>
<tr><td></td><td>
<table>
<tr><th>Column</th><th>Description</th></tr>
<tr><td>Activity</td><td>Indicates the routing type.</td></tr>
<tr><td>Target</td><td>Displays the endpoint of the target service for which requests are being routed.</td></tr>
<tr><td>Recipient</td><td>Displays the endpoint of the virtual service that receives the routed requests</td></tr>
</table>
<p>You can expand the panel for each intercept by clicking on the expand card icon ›. You can then find the following information:</p>
<table>
<tr><th>Tab</th><th>Description</th></tr>
<tr><td>Details</td><td>The <b>Details</b> panel provides the following information:
<table>
<tr><th>Option</th><th>Information displayed</th></tr>
<tr><td>Condition</td><td><p>Lists the condition that causes the traffic to be routed to a virtual service.</p><p>You can view the detailed condition of the rule defined for the intercept by clicking the <b>View</b> icon 👁.</p><p>The detailed condition of the rule is displayed in the <b>Full condition</b> window.</p><p><b>Note:</b> You can copy the detailed condition if you intend to use it elsewhere.</p></td></tr>
<tr><td>Routing to</td><td>Lists the virtual service to which traffic is routed.</td></tr>
<tr><td>Created by</td><td>Lists the user who created the rule.</td></tr>
<tr><td>Created at</td><td>Lists the date and time when the rule was created.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Panel</th><th>Details</th></tr>
<tr><td></td><td>
<table>
<tr><th>Tab</th><th>Description</th></tr>
<tr><td>Activity</td><td>The <b>Activity</b> panel lists the log of the activities performed in relation to the routing rule. This panel also displays the details of the activity, date and time, and message for that activity.</td></tr>
</table>
</td></tr>
<tr><td>Statistics</td><td>The <b>Statistics</b> panel of the instance is displayed when expanded.<br>The requests received by the instance of the virtual service in the previous hour are displayed as a graph. The total requests received is indicated in the <b>Total hits</b> field and the activity load in the previous hour are displayed in the <b>Activity</b> field.<br>The view can be refreshed by clicking the <b>Refresh</b> icon 🔄.</td></tr>
<tr><td>Diagnostics</td><td>The <b>Diagnostics</b> panel of the instance is displayed when expanded with the following details:
<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td>Logging</td><td>Displays the logging level that was configured for the virtual service.</td></tr>
<tr><td>Init container log</td><td>Displays the link to the log of the Init container.<br><b>Note:</b> This logs contains messages that relate to configuring the container which is ready to run the virtual service.</td></tr>
<tr><td>Main container log</td><td>Displays the link to the log of the main container. This log contains messages logged by the virtual service based on the selected logging level.</td></tr>
</table>
</td></tr>
</table>

### Results

You have viewed the information about the configuration, routing rules, usage, or logs of the virtual service instance.

### What to do next

You can modify the behavior or logging level configurations of the running virtual service instance before you run tests on the instance. See Modifying configurations of running instances of virtual services on page 353.

Related information

Configuring a run of a virtual service on page 328

## Modifying configurations of running instances of virtual services

After you start virtual services from the **Resources** page and before you run tests on the virtual service, you might want to modify the configurations for the behavior or the logging level of the running instance.

### Before you begin

You must have completed the following tasks:

- Been assigned the *Viewer*, *Tester*, or *Owner* role in the project so that you can view the virtual services that are running.
- Started the virtual service resources as a *Tester* or *Owner* from the **Resources** page.

### About this task

After you have started the virtual services from the **Resources** page on HCL OneTest™ Server, you can go to the **Instances** page and identify the running instance of the virtual service for which you want to modify the configurations. You can modify the configurations for the behavior or logging levels of the running instance of the virtual service.

1. Log in to HCL OneTest™ Server.
2. Open the project that contains the virtual services in the test assets by clicking **Projects > My Projects > *project_name***.
   The **Overview** page is displayed.
3. Click **Virtualization > Instances** in the navigation pane.

   The **Instances** page is displayed with all running instances of the virtual services that are in your project.
4. Identify the running instance of the virtual service for which you want to modify the configurations.
5. Click the **Expand** icon > of the instance that you want to modify the configurations.

   The **Details** panel of the instance is displayed.
6. Click **Behavior** to expand the panel and perform the following steps to change the configurations:

   a. Click the **Edit** icon ✎.

      The options that you can modify are enabled for editing.

   b. Select a different operation for the virtual service, if available, from the list for the **Operation** option.

   c. Select a different response time from the list for the **Response time** option, and then enter the values for the delay that you want to set.

      You can modify the value by selecting a value from the following options:

| Option | Description |
|---|---|
| No delay | Select this option for a response with no delay. |
| Minimum delay | Select this option for a delay in the response by entering the required delay in milliseconds (ms). |
| Uniform distribution | Select this option for a uniformly distributed response time by specifying the minimum and maximum delay in milliseconds (ms). |
| Gaussian distribution | Select this option for a response time with Gaussian distribution by specifying the minimum and maximum delay in milliseconds (ms). |

d. Click **apply** to apply the modified settings.

7. Click **Diagnostics** to expand the panel and perform the following steps to change the logging level that is configured:

   a. Click the **Edit** icon ✎.

   The **Logging level** option is enabled for editing.

   b. Select a different logging level from the following options.

| Option | Description |
|---|---|
| None | Specifies that the virtual service does not write log messages. |
| Normal | Specifies that the virtual service writes informational messages. |
| Debug | Specifies that the virtual service writes informational and debugging messages. |

   c. Click **apply** to apply the modified settings.

**Results**

You have viewed and modified the configurations of a running virtual service instance from the **Instances** page on HCL OneTest™ Server.

**What to do next**

You can run tests that are in an API Suite in your project, which then run on the virtual service instance. See Configuring an API Suite run on page 249.

## Viewing routing rules of the virtual services

When you run virtual services that are created to run on proxies or intercepts and have routing rules defined, you can view the proxies or intercepts and the routing rules from the **Intercept Rules** page.

### Before you begin

You must have completed the following tasks:

- Started the virtual service resources as a *Tester* or *Owner* from the **Resources** page.
- Been assigned the *Viewer*, *Tester*, or *Owner* role so that you can view the routing rules defined for the virtual services from the **Intercept Rules** page.

### About this task

After you run the virtual service resources that are in your project on HCL OneTest™ Server, you can view the routing rules for the running instances of the virtual service resources, from the **Intercept Rules** page.

1. Log in to HCL OneTest™ Server.
2. Open the project that contains the stubs in the test assets by clicking **Projects > My Projects > *project_name***.

   The **Overview** page is displayed.
3. Click **Virtualization > Intercept Rules** in the navigation pane.

   The **Intercept Rules** page is displayed.

   The agents, proxies, or intercepts that are configured in the virtual service resources that are in your project are displayed.

   You can find the following information that is displayed on the **Intercept Rules** page.

   | Column | Description |
   |---|---|
   | Activity | Lists the agent or proxy. |
   | Target | Lists a summary view of the endpoint that was recorded or virtualized. |
   | Recipient | Lists the destination where messages sent to are captured. |

4. Expand the proxy or intercept panel by clicking the **Expand** icon ›.

   The details of the proxy or intercept are displayed in the **Details** tab in the expanded panel.
5. Click each tab to view the following information:

   | Tab | Description |
   |---|---|
   | Details | The **Details** panel provides the following information: |

<table>
<tr><th>Tab</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td>Condition</td><td>Lists the condition that causes the traffic to be recorded or routed to a virtual service.<br>You can view the detailed condition of the rule defined for the proxy or intercept by clicking the <b>View</b> icon 👁.<br>The detailed condition of the rule is displayed in the <b>Full condition</b> window.<br><b>Note:</b> You can copy the detailed condition if you intend to use it elsewhere.</td></tr>
<tr><td>Routing to</td><td>Lists the virtual service to which traffic is routed.</td></tr>
<tr><td>Created by</td><td>Lists the user who created the rule.</td></tr>
<tr><td>Created at</td><td>Lists the date and time when the rule was created.</td></tr>
</table>
</td></tr>
<tr><td>Activity</td><td>The <b>Activity</b> panel lists the log of the activities performed by the proxy or intercept. This panel also displays the details of the activity, date and time, and message for that activity.</td></tr>
</table>

**Results**

You have viewed the routing rules for the virtual service, proxy, or intercept that are running or connected to HCL OneTest™ Server.

**What to do next**

You can perform any of the following tasks:

- You can view the usage statistics of the virtual services that are running. See Viewing usage statistics of virtual services on page 356.
- You can stop the stubs that are running on HCL OneTest™ Server. See Stopping a virtual service on page 359.

Related information

Management of virtualized services on page 319

## Viewing usage statistics of virtual services

After you start stubs or virtual service resources that are in the project repositories, and then run tests on the virtual service, you can view the usage statistics of the virtual services from the **Usage** page on HCL OneTest™ Server.

### Before you begin

You must have completed the following tasks:

- Started the virtual service resources as a *Tester* or *Owner* from the **Resources** page.
- Been assigned the *Viewer*, *Tester*, or *Owner* role in the project so that you can view the usage details of virtual services.
- Ran API Suites with tests that run on the virtual service resources that are configured.

### About this task

After you run the virtual services that are in a project, you can view details of the usage statistics of the virtual services from the **Usage** page on HCL OneTest™ Server.

**Important:** The settings such as the **Period**, or **From** and **To** that you configure to view the usage details are retained for you during your ongoing session and the settings are retained even when you perform the following activities:

- Log out of the server and log in again.
- Log in after a server restart.

1. Log in to HCL OneTest™ Server.
2. Open the project that contains the stubs in the test assets by clicking **Projects > My Projects > *project_name***.

   The **Overview** page is displayed.
3. Click **Virtualization > Usage** in the navigation pane.

   The **Virtual Service Usage** page is displayed.

   The total requests that were received by all virtual services on the server are displayed as a graph.

   The default view displays the usage data of the virtual services that are gathered during the previous week and grouped in an hourly interval. The total requests received from when the virtual service was started are displayed.

   The metric data that is reported is for the number of requests that are received by a virtual service. The requests can be requests, messages, or other calls that are received by the virtual service.
4. Change the default settings to view the graph for the period for which you want to view the usage.

   You can change the default view by selecting an option from the following options:

| Option | Description |
|---|---|
| *1 Hour* | Displays the usage data of the virtual service that are gathered during the previous hour and grouped in a 20-second interval. |
| *1 Day* | Displays the usage data of the virtual service that are gathered during the previous day and grouped in an eight-minute interval. |
| *1 Week* | Displays the usage data of the virtual service that are gathered during the previous week and grouped in an hourly interval.<br>**Note:** This is the default view. |
| *Range* | Displays the usage data of the virtual service that are gathered during the date range specified in the **From** and **To** fields. You can select the date and time for both the **From** and **To** fields. The data is dynamically grouped in intervals that depend on the range selected. |

**Note:** You can refresh the view at any time to present the graph for the latest data by clicking **Refresh**.

5. View details of the virtual service by hovering the mouse over any of the data-points to view the details at that point.

   The following details about the virtual service are displayed when you hover the mouse over the data-point:

   - The date and time of the request received.
   - The number of requests received.

**Results**

You have viewed the usage details of the virtual services in your project that ran on HCL OneTest™ Server.

**What to do next**

You can perform any of the following tasks:

- You can view the routing rules defined in the virtual service, agent, proxy, or intercept. See Viewing routing rules of the virtual services on page 355.
- You can stop the virtual service instances that are running on HCL OneTest™ Server. See Stopping a virtual service on page 359.

Related information

Management of virtualized services on page 319

## Stopping a virtual service

When you want to stop a virtual service instance that is running in HCL OneTest™ Server, you can stop the running virtual service from the **Instances** page.

### Before you begin

You must have started the virtual service from the **Resources** page. See Configuring a run of a virtual service on page 328.

### About this task

You can stop a running virtual service instance from the **Instances** page by using the **Stop** icon ⏹.

1. Log in to HCL OneTest™ Server.
2. Open the project that contains the stubs in the test assets by clicking **Projects > My Projects >** ***project_name***.
3. Go to the **Instances** page and identify the instance of the virtual service that you want to stop.

   **Note:** You can stop a running stub if it is in the `In Transition` or `Running` state.

4. Click the **Stop** icon ⏹ in the **Actions** column of the virtual service.

   **Result**

   The **Stop stub instance** dialog box is displayed.
5. Click **Ok**.

   A notification message is displayed that the running virtual service instance is being stopped.

### Results

You have stopped a running virtual service instance. The stopped virtual service instance on the **Instances** page is displayed with the status as *Stopped by User* in the **State** column. Stopped stubs are not considered in the count of test assets run that is displayed on the **Overview** page.

### What to do next

You can restart the virtual service by completing the configurations that you want from the **Resources** page.

You can open and view the `Main container log` of the stopped virtual service.

# Test results

After the tests or schedules are run and completed, you can view the results and reports in HCL OneTest™ Server to analyze the verdict, the performance, and statistics. You can also re-execute tests and schedules from the **Result** page with the same commit id.

From the Results page, you can perform the following tasks:

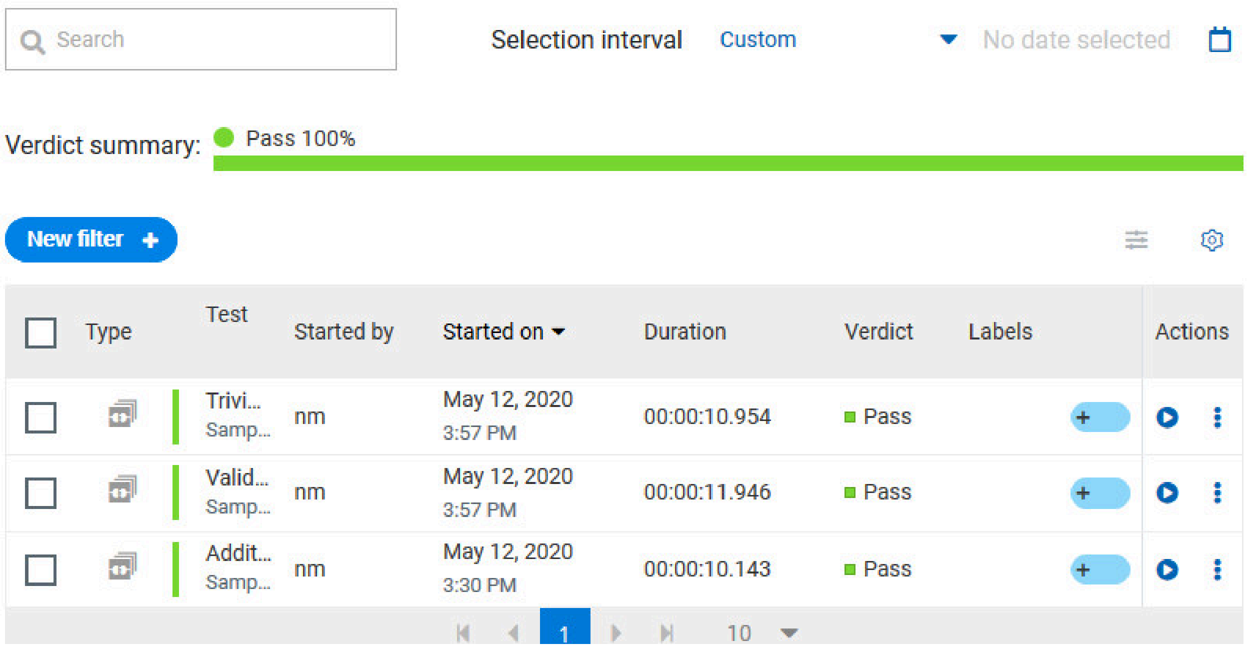
- Searching for test results. You can filter test results by using any of the following ways:
    - By using the **Search** field to search for results by name.
    - By selecting predefined time interval from the **Selection interval** drop-down list.
    - By selecting **To** and **From date** from the **Date Interval** option.
    - By clicking verdict from the **Verdict summary** slider (*Pass*, *Fail*, *Inconclusive*, or *Error*).
    - By creating filter queries.
- Adding labels if you have a tester or an owner role.
- Locking test results: Locked results can be unlocked by the project owner or the project member who locked the results.
- Deleting test results.
- Comparing performance reports.
- Viewing trending reports.
- Viewing multiple reports depending on the test types: Statistics reports, Mobile and Web UI reports, Functional reports, Unified Reports, or HTML reports generated from Postman or JMeter test runs.
- Re-executing tests with the same commit id.

  For more details, see the links in the next section of this page.

## Test results and reports overview

HCL OneTest™ Server is a single location for hosting the results and reports of all tests run on different desktop clients and for tests run from the server.

### Test results list



Verdict summary: Pass 100%

| ☐ | Type | Test | Started by | Started on | Duration | Verdict | Labels | Actions |
|---|---|---|---|---|---|---|---|---|
| ☐ | | Trivi...<br>Samp... | nm | May 12, 2020<br>3:57 PM | 00:00:10.954 | 🟩 Pass | + | |
| ☐ | | Valid...<br>Samp... | nm | May 12, 2020<br>3:57 PM | 00:00:11.946 | 🟩 Pass | + | |
| ☐ | | Addit...<br>Samp... | nm | May 12, 2020<br>3:30 PM | 00:00:10.143 | 🟩 Pass | + | |

When you expand the results, the results details and reports cards are displayed.

Results, reports, and logs are generated for tests run from HCL OneTest™ Server, or from desktop clients such as HCL OneTest™ Performance, HCL OneTest™ UI, or HCL OneTest™ API.

You must have configured the desktop clients to publish reports of tests that are run from the desktop client to the HCL OneTest™ Server. For more information about the publishing procedure, refer to the links at the end of this page.

## Test result details

By using the default settings, you can view the following details about the test assets used in the test run in the Details card:

- The status of the test results, the date and time it was executed and completed.
- The details about the Git repository that contains the test assets.
- The execution location that uses the test asset for a test run.

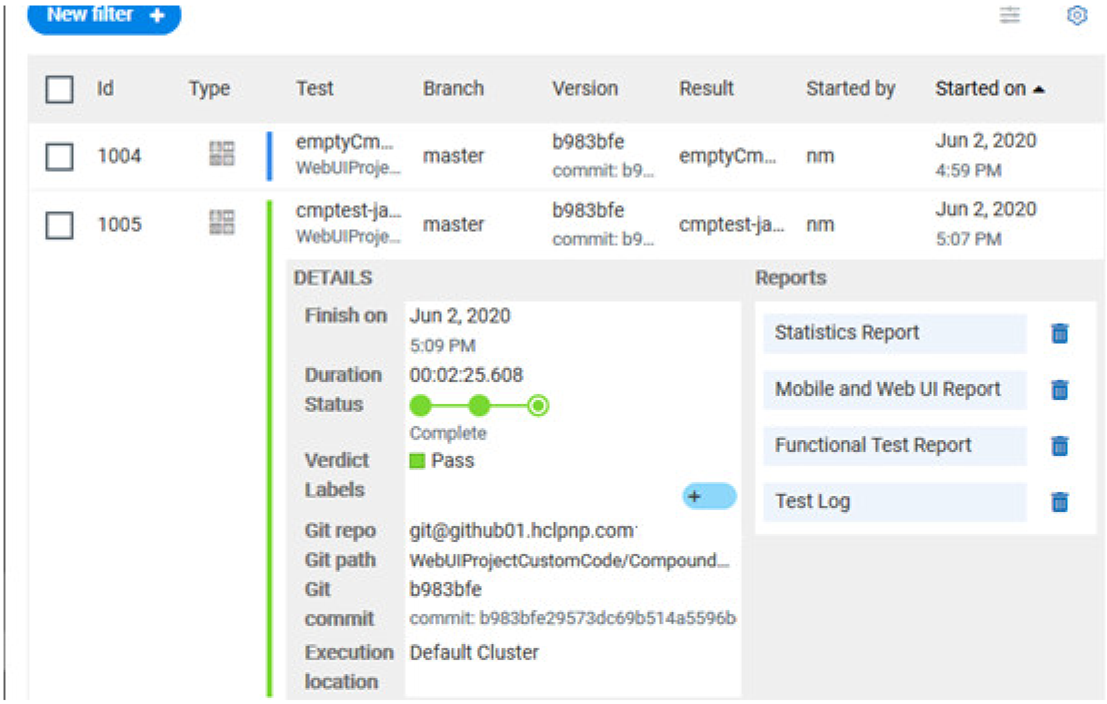
  > **Note:** The execution location can be the default cluster or the location that you indicate to override a docker host or the agent location when you execute a test. For more information about test configuration, see the link at the end of this page.

- The Resource Monitoring labels (override) used in a performance schedule to control Resource Monitoring sources. For more information about controlling Resource Monitoring sources, see the link at the end of this page.

You can configure the information displayed in the **Details** card and the **Results** columns from the **View Settings** window. You access this window by clicking the **Settings** icon ⚙.

Depending on the attribute you cleared in the **View Settings** window, the Details card displays additional information about the test results.

The selected attributes are displayed in columns of the Result view.



| □ | Id | Type | Test | Branch | Version | Result | Started by | Started on ▲ |
|---|---|---|---|---|---|---|---|---|
| □ | 1004 | | emptyCm...<br>WebUIProje... | master | b983bfe<br>commit: b9... | emptyCm... | nm | Jun 2, 2020<br>4:59 PM |
| □ | 1005 | | cmptest-ja...<br>WebUIProje... | master | b983bfe<br>commit: b9... | cmptest-ja... | nm | Jun 2, 2020<br>5:07 PM |

DETAILS

| | |
|---|---|
| Finish on | Jun 2, 2020<br>5:09 PM |
| Duration | 00:02:25.608 |
| Status | Complete |
| Verdict | 🟩 Pass |
| Labels | + |
| Git repo | git@github01.hclpnp.com |
| Git path | WebUIProjectCustomCode/Compound... |
| Git commit | b983bfe<br>commit: b983bfe29573dc69b514a5596b |
| Execution location | Default Cluster |

Reports

- Statistics Report
- Mobile and Web UI Report
- Functional Test Report
- Test Log

## Reports

The Reports card contains the links to the test reports and the logs that are displayed in a web browser.

As a default configuration, test logs are delivered in a traditional format for the executed compound tests and schedules. You can still set the `-history jaeger` Program Argument to produce Jaeger traces when you run the tests.

The reports are generated for Compound Tests, Schedules, and Test Suites, or for other tests (Postman or JMeter tests for example). They can be run from a desktop client or from HCL OneTest™ Server.

You can run Web UI tests and SAP tests from HCL OneTest™ UI and publish them to the server from V10.1.3. You can also run and publish these tests from HCL OneTest™ Server.

The following table lists the reports that are generated for each type of test and the links to help pages.

**Table 8. Analyzing Reports**

| Report | Tests type | Product | More information |
|---|---|---|---|
| Functional Test Report | API Suite | HCL OneTest™ API | Viewing reports |
| | Compound Test | HCL OneTest™ UI | Unified Report |
| | AFT Suite | | |
| | Functional Test (for SAP test) | | |

**Table 8. Analyzing Reports (continued)**

| Report | Tests type | Product | More information |
|---|---|---|---|
| | Functional test (For Web UI tests) | | |
| Mobile and Web UI Report | Compound Test | HCL OneTest™ UI and HCL OneTest™ Performance | Mobile and web UI test result reports<br>Unified Report |
| | AFT Suite | HCL OneTest™ UI | |
| Statistics Report | Compound Test | HCL OneTest™ UI and HCL OneTest™ Performance | Reports and counters<br>Service Performance report<br>From the Statistic reports, you can add counters. Refer to Adding additional counters on a separate page.<br>You can also display the statistics of counters on graphs. Refer to Displaying counter data in tables or as graphs. |
| | AFT Suite | HCL OneTest™ UI | |
| | Functional test (For SAP and Web UI tests) | HCL OneTest™ UI | |
| | VU Schedule | HCL OneTest™ Performance | |
| | Rate Schedule | | |
| Test Log | Compound Test | HCL OneTest™ UI and HCL OneTest™ Performance | |
| | AFT Suite | HCL OneTest™ UI | |
| | Functional Test (Web UI and SAP tests) | HCL OneTest™ UI | |
| | VU Schedule | HCL OneTest™ Performance | |
| | Rate Schedule | | |

Related information

Publishing test results to the server from HCL OneTest API

Publishing test results to the server from HCL OneTest Performance

Publishing test results to the server from HCL OneTest UI

Controlling Resource Monitoring sources in a schedule

Test run configurations on page 240

## Creating search queries to filter test results

To filter the test results list, you can create a search query that you apply as a filter in HCL OneTest™ Server.

**About this task**

You can use rules and group of rules in your query. This procedure explains how to create a query that contains two rules and a group of rules as an example, and how to select other filters.

1. Click **New filter** on the Result page.

Follow these steps to add a rule in your query:

2. Select **Add a rule** in your query, and enter or select the required attribute in the fields to define the rule:
    a. Select the first attribute that you want to query in the drop-down list.
    **Example**
    Select **Verdict**.

    b. Select the condition as *equal to (=)* or *not equal to (!=)*, *Contain* or *Not contain*.
    **Example**
    Select **=**.

    c. Select a value for the first attribute you selected.
    The content of this field depends on the option you selected as first attribute. Enter a value for the **id**, **TypeTest**, **Version**, **Result**, **Labels**, **Branch**, **Test path**, and **Text** attributes. For the other attributes, select a value in the drop-down list.
    **Example**
    Select **Fail**.

    d. Select either the **AND** or the **OR** operator in the first field to add a condition to compare the first rule with another rule.

    e. Select **Add rule** in the drop-down list.

    f. Select an attribute, a condition, and select or enter a value.
    **Example**
    Select **Test** as an attribute, select **=** as a condition operator, and enter *mytest* as a value.

Follow these steps to add a group in your query:

3. Select **Add group** in the drop-down list.

   **Note:** A group is a collection of rules and can also contain other groups within a query. You can add a rule or a group within this group.

4. Select a condition operator.

   **Example**

   Select **Or**.
5. Select **Add rule** in the drop-down list to add a rule in the group.
6. Select the required an attribute, an operator, and enter a value for the rule.

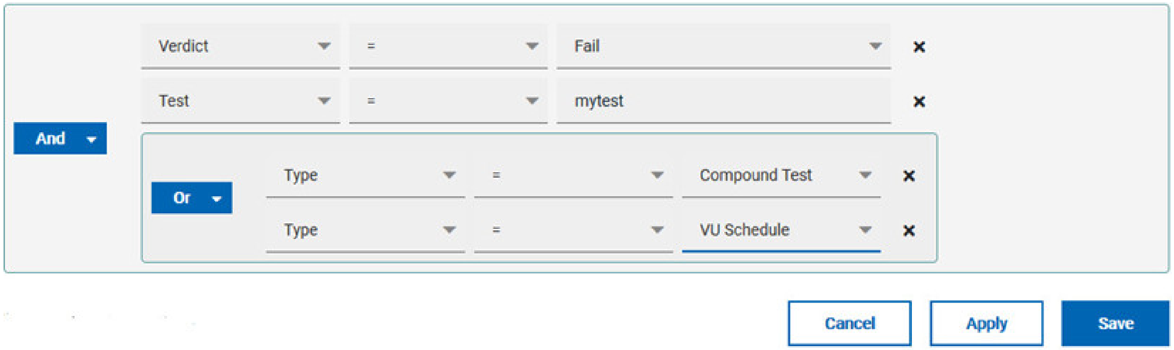
   **Example**

   Select the **Type** attribute, **=** as an operator, and **Compound Test** as a value.
7. Select **Add rule** to add another rule in the group.
8. Select and enter the appropriate parameters as you did for the other rules.

   **Example**

   Select **Type** as an attribute, **=** as a condition, and **VU Schedule** as a value.
9. Click **Apply** if you want to apply the filter now.

   You can cancel the query or save the query when the results are found.



And
- Verdict = Fail
- Test = mytest
- Or
  - Type = Compound Test
  - Type = VU Schedule

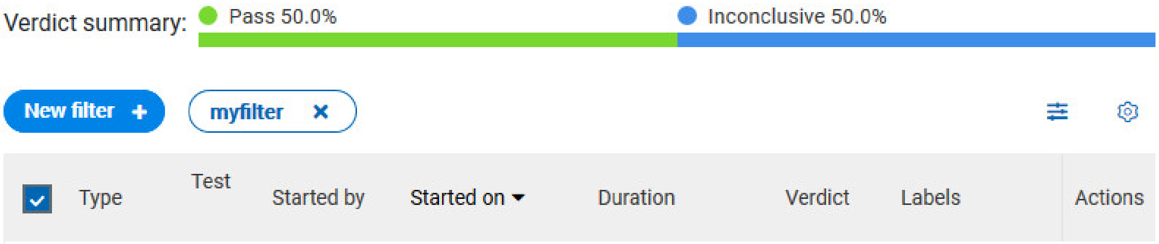
10. Click **Save**.

Proceed as follows to select your filters:

11. Enter a name for the filter and save in the dialog box that is displayed.

    **Result**

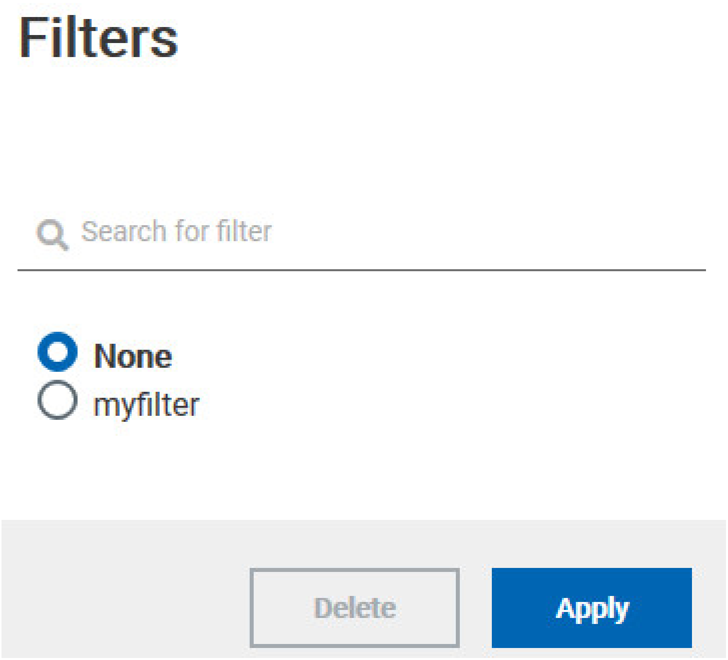
    The filter is displayed as a favorite filter.



Verdict summary: ● Pass 50.0% ● Inconclusive 50.0%

| ☑ | Type | Test | Started by | Started on ▾ | Duration | Verdict | Labels | Actions |
|---|---|---|---|---|---|---|---|---|

12. Click the icon ☰ to select the filters you created from the list of saved filters.



13. Click a filter in the list of saved filters. If a filter does not display in the list, enter the name of a filter in the search field and select it.

    **Note:** You can delete a filter from the list or disable filters by selecting **none** in the list.

14. Apply the selected filter.
    **Result**
    The filter is displayed as a new favorite filter in the **Results** page.

## Comparing test reports

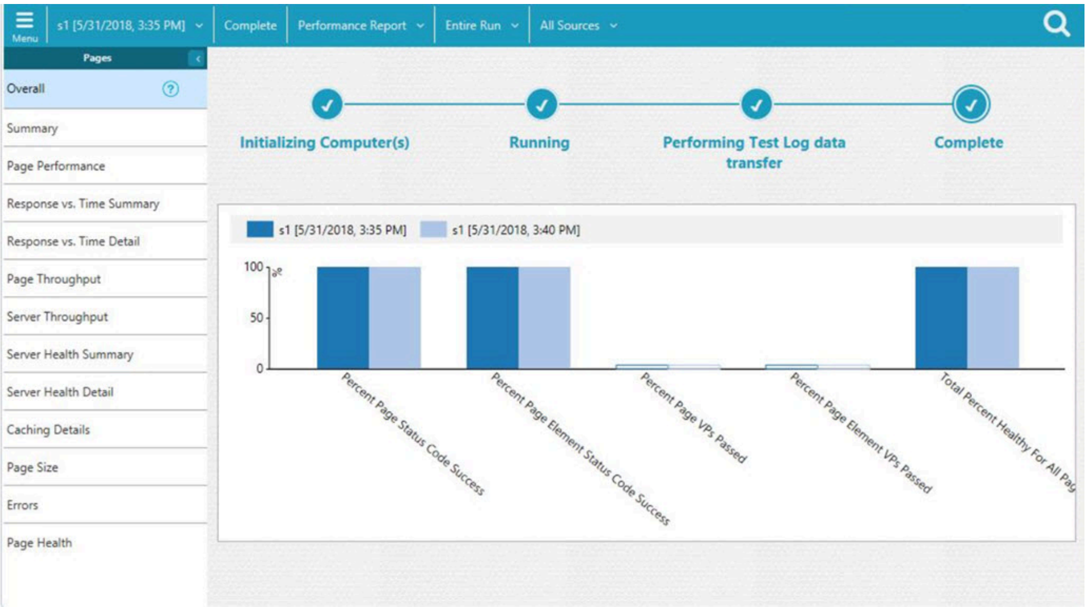
After you run performance tests, you can analyze the difference between two or multiple test results in HCL OneTest™ Server. For example, you can compare the performance of an application at different time slots or different milestone builds between two test results.

**About this task**

You can compare the test runs for performance tests that are in the same project or in different projects.

1. Select results in the **Results** page by checking the boxes in the first column.
2. Click the ⟷ icon

    **Note:** When comparing multiple runs, you cannot compare multiple time-ranges or stages.



**Result**

The two or multiple test results are displayed in a browser window in the same report.

## Viewing trending reports

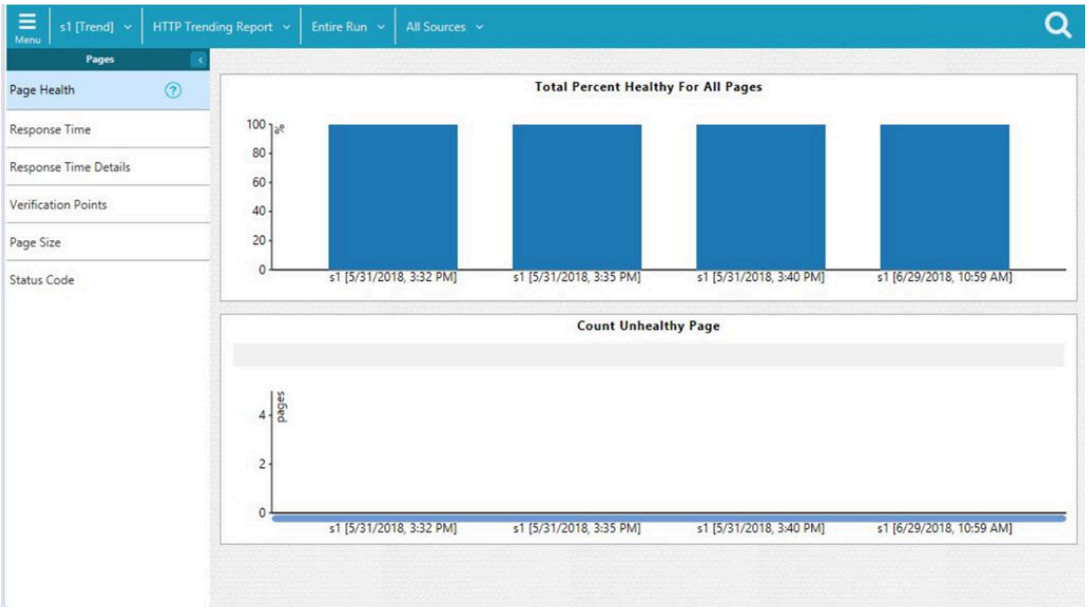
When the performance test runs are complete, you can view the trend of response time for an application over a period of time in HCL OneTest™ Server from trending reports. In addition to the response time, you can view the trend for the loops, transactions, and performance requirements for the application.

### About this task

The trending reports are available for performance tests only.

1. Select results in the first column in the **Results** page.
2. Click the 📈 icon to view the trending reports.

**Note:** When comparing multiple runs, you cannot compare multiple time-ranges or stages.



**Results**

You can analyze the trend information for multiple test results in the same report from a browser window.

## Re-executing tests from results

You can re-execute a test, a schedule, a collection of tests or schedules from HCL OneTest™ Server if you want to keep the same commit id.

### Before you begin

You must have executed a test or multiple tests from the **Execution** page in HCL OneTest™ Server.

Re-executing a test from the **Results** view requires a Tester or an Owner role.

### About this task

The following procedure describes how to re-execute a single test or multiple tests at a time.

Perform all the following steps from the Results page.

- Proceed as follows to re-execute one test at a time:

  1. Identify the test that you want to re-execute and click the **Re-execute** button in the Actions column.

     **Result**

     The **Execute test asset** dialog box displays the same parameters as the ones that were set to execute the initial test from the **Execution** page.

2. Select another **Version** of the test.

   **Note:** If you select another version of the test, the settings configured in the dialog box and in the **Environment**, **Data Sources** and **Locations** tabs might change.

3. Modify the **Schedule** parameters.

   **Note:** If you don't want to execute the test now, schedule the time when you want the test to be re-executed.

4. Click the **Environment** tab and select another **API test environment** if multiple environments were initially set for a test that is running an API Suite.
5. Click the **Data Sources** tab and select an override option if multiple data sources were defined for the initial test run.
6. Click the **Variables** tab and add a new variable if you want to re-execute a test asset with a different variable from the one configured in the asset.
7. Click the **Location** tab to override the default cluster location if the test asset has a docker host or static agents configured.
8. Click **Advanced** to modify the advanced parameters that were set for the initial test run.
9. Enter a new **Program Argument** if applicable to the test.

   **Note:** If Jaeger is installed with HCL OneTest™ Server, you can enter the -history jaeger Program Argument to produce Jaeger traces when you run the tests.

10. Enter a new instance of **Java arguments** if applicable to the test.
11. Enter the environment variables that must be passed to the test run at runtime in the **Environment Variables** field.
12. In the **Resource Monitoring** tab, click the +, and press the Ctrl + Space keys, or enter the initial letter of a label to select a label in the list and override the current sources in the schedule.

    **Note:** The **Resource Monitoring** tab is displayed for performance schedules and only if you have enabled the **Resource Monitoring from Service** option from HCL OneTest™ Performance.

    The label field is grayed if you don't have any label entered in your current project. To collect data during the test execution, you must first enter labels in the Resource Monitoring sources page from HCL OneTest™ Server.

For more details about controlling Resource Monitoring sources and labels, see the links at the end of this page.

13. Click the **Re-execute** ▶ icon to re-execute the test now or at a scheduled time.

- Proceed as follows to re-execute multiple tests at a time:

    1. Click the checkboxes for the tests that you want to re-execute.
       You can select them one by one or select all of them.
       **Result**
       A toolbar is displayed.

    2. Click the **Re-execute** icon ▶ in the toolbar.
       **Result**
       A message indicates the number of selected tests and the number of tests to be re-executed. The tests are re-executed with the same initial commit ids and parameters.

    3. Click **Execute**.

**Result**
You can view the results, logs, and reports of the re-executed tests in the **Results** page.

Related information

Adding a project on page 499

Test results and reports overview on page 360

Controlling Resource Monitoring sources in a schedule

Tests configurations and test runs

# Resource Monitoring service

When you apply load to a system under test, the amount of resources consumed by your system is increasing. If the capacity of the resources does not match the load, you can see performance issues in the results. The Resource Monitoring service in HCL OneTest™ Server helps you monitoring the resources of a system and establish the performance metrics of the system. Thus, you can observe the health of these resources while a schedule is running.

You can use the Resource Monitoring service while running a schedule to capture data, such as processor or memory usage, or to monitor the availability of hosts and services by using counters. The Resource Monitoring service can provide a comprehensive view of a system under test to help determine problems. Hosts and services can be servers, virtual machines, or any local host or network services. You can also monitor remote hosts and services with agents.

### Monitoring local host and service

You can use the Resource Monitoring service to monitor any local host and network service. In this case, the resources of the hosts are locally monitored. This method is used for host targets that the service can reach directly. With HCL OneTest™ Server, you can also monitor performance metrics that are collected and stored by a monitoring system that is monitoring a host. You can use Prometheus server or a data collector through an exporter, OpenMetrics exporter, for example.

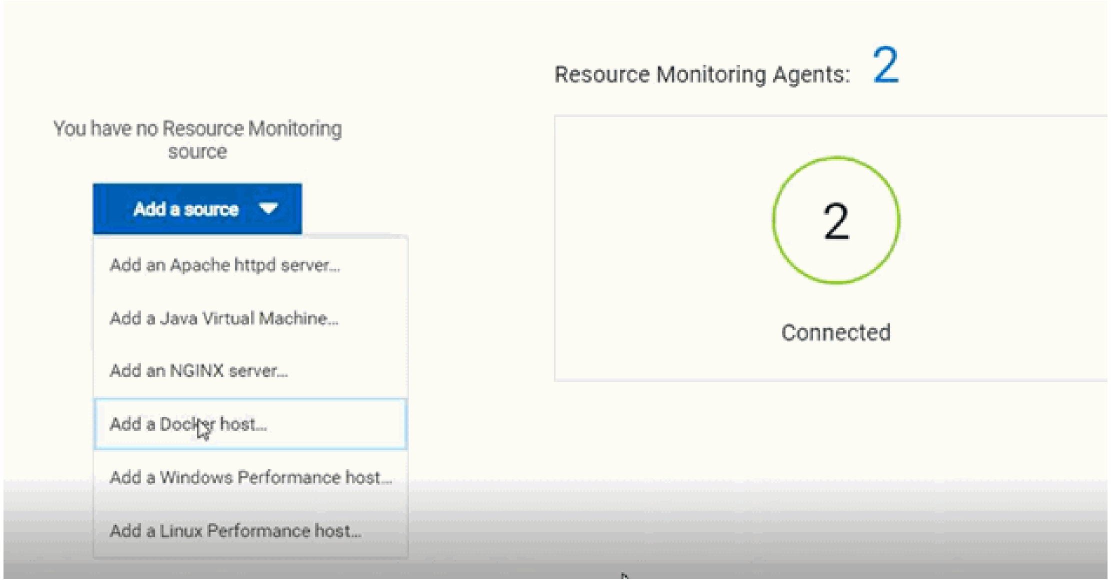
To monitor resources on a local system, you must add the source of the resource monitoring data from the Resource Monitoring page. Then you have to enter some connection settings and select the performance counters to monitor the sources. The metrics are exposed in a graph for each selected counter.

### Monitoring remote host with monitoring agents

You can use the Resource Monitoring service to monitor remote hosts and services with agents through wider sets of data collectors, computers, and networks. The purpose is to capture CPU load, disk space, memory, and the running process for example.

Agent-based monitoring is useful when remote services are not directly accessible through the network. Agents are closer to the target that you want to monitor. You can set up the agent on an authorized host, for example, when access to an **Apache httpd** or **NGINX** server status page or to a **JVM** JMX port is restricted to one or few client hosts only. The configuration task is simplified, and no security changes are required.

To monitor a host across monitoring agents, you must first set up the Resource Monitoring agents on the target host for which you want to collect the performance statistics. The agents establish a connection with the Resource Monitoring service. When the connection is set, the agent is showing up in the list of sources in the Resource Monitoring main page. You can select it, choose performance counters, and view the metrics statistics in a graph for each selected counter.


Resource Monitoring Agents: 2
You have no Resource Monitoring source
Add a source
Add an Apache httpd server...
Add a Java Virtual Machine...
Add an NGINX server...
Add a Docker host...
Add a Windows Performance host...
Add a Linux Performance host...
2
Connected

**Resource Monitoring sources**

With HCL OneTest™ Server, you can monitor resources for the following sources:

- **Apache httpd server**
- **NGINX and NGINX Plus**
- **Java Virtual Machine**: You can monitor JVM resources from a local or from a remote system. Some parameters must be set in the command before running the Java Virtual Machine. For more details, see the link to the page about starting a Java Virtual Machine on this page.
- **Windows Performance host**: To monitor the performance of a Windows host, you must have installed an agent on the target Windows host.

  If HCL OneTest™ Server is installed on a Linux system, and you want to check the performance of the Windows host, you must have installed an agent on the target Windows host to start monitoring its performance.
- **Linux Performance host**: To monitor the performance of a Linux host, you must have installed an agent on the target Linux host.

  If HCL OneTest™ Server is installed on a Windows system and you want to check the performance of the Linux host, you must have installed the Resource Monitoring agent on the Linux system.
- **Docker host**: A Resource Monitoring agent is mandatory to get a Docker Data Collector. The Docker host source is added to the list of resource monitoring sources when the agent is installed.
- **Prometheus server**: You can monitor metrics of a host under test that are collected by a Prometheus server. Prometheus collects metrics from monitored targets by regularly requesting appropriate HTTP endpoints on these targets (called *scraping*).

The metrics data are collected through data collectors sets (groups of performance counters) that are tracking the system performance. The performance data results are stored under a Prometheus REST API so that they can be consumed by external systems.

With the Resource Monitoring service in HCL OneTest™ Server, you can query these performance metrics collected by Prometheus servers. 'PromQL' is the language for querying Prometheus metric data.

In the Resource Monitoring service, you can select default queries, or create additional ones.

A Prometheus server is required and it must be configured to scrap a set of exporters, every 15 seconds by default. Pushgateway, service discovery, and Alertmanager are optional.

- **OpenMetrics exporter**:

  You can monitor the metrics of a host under test that are collected and exposed through an OpenMetrics format or a Prometheus exporter format.

  The Resource Monitoring service in HCL OneTest™ Server is given the ability to scrap metrics exposed by an OpenMetrics or Prometheus exporter through metric counters. No Prometheus server installation is required, you only need to set up one or multiple OpenMetrics, or Prometheus exporters that fit your software or host target. Refer to the link at the end of this page to see the list of exporters.

Click the links in the next section of this page for more details on the Resource Monitoring tasks that you can perform in HCL OneTest™ Server.

Related information

Exporters and Integrations

Starting a Java Virtual Machine on page 396

## Resource monitoring capabilities

With HCL OneTest™ Server, you can monitor a host or a service that you add to a project as a source. You can select the statistics counters that are used to collect metrics and the resource usage data throughout the monitoring system. You can visualize metrics and data that are gathered during the monitoring process in HCL OneTest™ Server.

**Requirements:**

- You must be a *Team Space Owner* to create, modify and remove resource monitoring sources in a team space. Any team space *Member* (not owner) can only view resource monitoring sources.
- You must be a project *Owner* to create, modify and remove resource monitoring sources in a project. Any project *Member* (not the owner) can only view resource monitoring sources.

### Supported sources

With HCL OneTest™ Server, you can monitor resources for the following sources:

- **Apache httpd server**: You can optionally use agents.
- **NGINX and NGINX Plus**: You can optionally use agents.
- **Java Virtual Machine**: You can optionally use agents.

  You can monitor JVM resources from a local or a remote system. Some parameters must be set in the command before running the Java Virtual Machine. For more details, see the link to the page about starting a Java Virtual Machine on this page.
- **Windows Performance host**:

  To monitor the performance of a Windows host, you must have installed an agent on the target Windows host

  or on a Windows host that is configured to access the performance data of the Windows host target.
- **Linux Performance host**:

  To monitor the performance of a Linux host, you must have installed an agent on the target Linux host.
- **Docker host**:

  A resource monitoring agent is mandatory to get a Docker data collector. The Docker host source is added to the list of resource monitoring data collectors when the agent is installed.
- **Prometheus server**:

  You can monitor metrics of a host under test that are collected by a Prometheus server. Prometheus collects metrics from monitored targets by regularly requesting appropriate HTTP endpoints on these targets (called *scraping*).

  The metrics data are collected through data collectors sets (groups of monitoring counters) that are tracking the system performance. The performance data results are stored in the Prometheus database so that they can be consumed by external systems through a REST API.

  In HCL OneTest™ Server, you can query these performance metrics collected by Prometheus servers. 'PromQL' is the language for querying Prometheus metric data. Refer to Prometheus PromQL documentation for information about this query language.

  You can select default queries, or create additional ones.

  A Prometheus server is required. It must be configured to scrap a set of exporters, every 15 seconds by default. Pushgateway, service discovery, and Alertmanager are optional.

If HCL OneTest™ Server cannot directly reach the Prometheus server, you can consider setting up an agent on an authorized host or on the same host that is running Prometheus.

- **OpenMetrics exporter**:

  You can monitor the metrics of a host under test that are collected and exposed through an OpenMetrics format or a Prometheus exporter format.

  HCL OneTest™ Server can scrap metrics exposed by an OpenMetrics or Prometheus exporter through metric counters. No Prometheus server installation is required, you only need to set up one or multiple OpenMetrics, or Prometheus exporters that fit your software or host target.

  If HCL OneTest™ Server cannot directly reach the Prometheus server, you can consider setting up an agent on an authorized host or on the same host that is running Prometheus.

## Monitoring host resources

To monitor host resources, you must add the monitoring sources to the Resource Monitoring service, enter connection settings and select performance counters that are used to capture the performance statistics.

**Before you begin**

You must be logged in HCL OneTest™ Server and be the owner of an existing new project.

If you monitor a remote host, it must be connected with the computer that you use to access Resource Monitoring service. See Installing Resource Monitoring agents on remote hosts on page 388.

**About this task**

This task applies to NGINX and NGINX Plus, Apache httpd server, Java Virtual Machine, Windows Performance host, Linux Performance host, and Docker host sources.

1. In the Resource Monitoring page, click **Add a source**, for example: **Add an NGINX server**.
2. In the **New server** dialog box, fill in the following connection settings:

   a. In **Target host**, enter the IP address or host name and port number of the host where the server to monitor is installed.

   b. In **Server edition**, select the appropriate server from the drop-down list.

   c. In **Path to the status page of the server**, enter the name of the page to view the status of the server.

   If you select NGINX Plus (with API version 3) as a source, you must specify the name of the path to view the API root of the NGINX Plus server.

   d. In **Security**, select the following options:

   **Secured with TLS/SSL**

   If the application server is secured with TLS/SSL.

**Trust self-signed certificate**

To accept the server certificate.

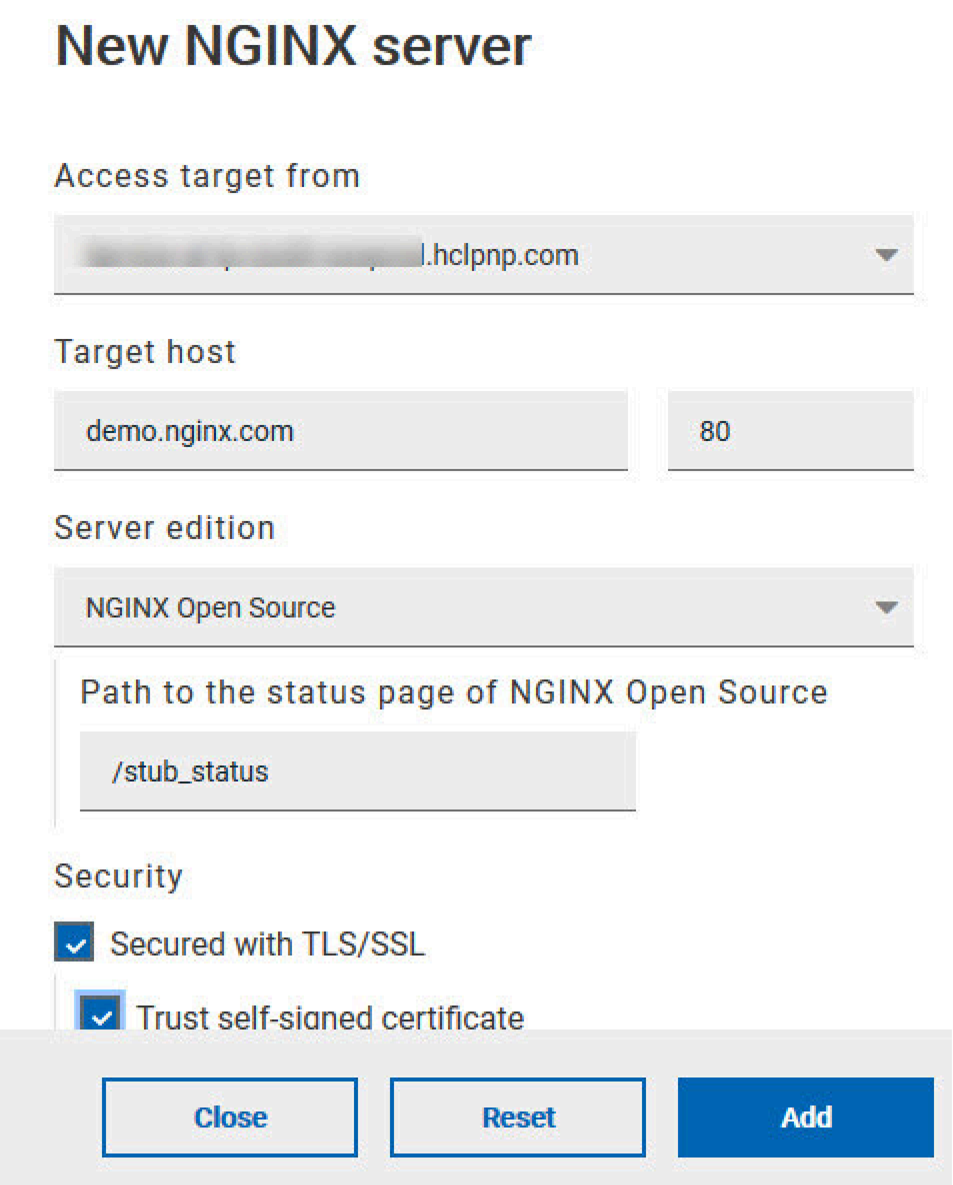
**Do not verify host name**

To ignore verification of the host name in the certificate.

**Require credentials**

If the server requires log-in credentials, enter a **User name** and **Password**.

**Note:** For remote hosts that are already connected to HCL OneTest™ Server through an agent, you have only the **Access target from** field enabled.



New NGINX server

Access target from: [redacted].hclpnp.com

Target host: demo.nginx.com 80

Server edition: NGINX Open Source

Path to the status page of NGINX Open Source: /stub_status

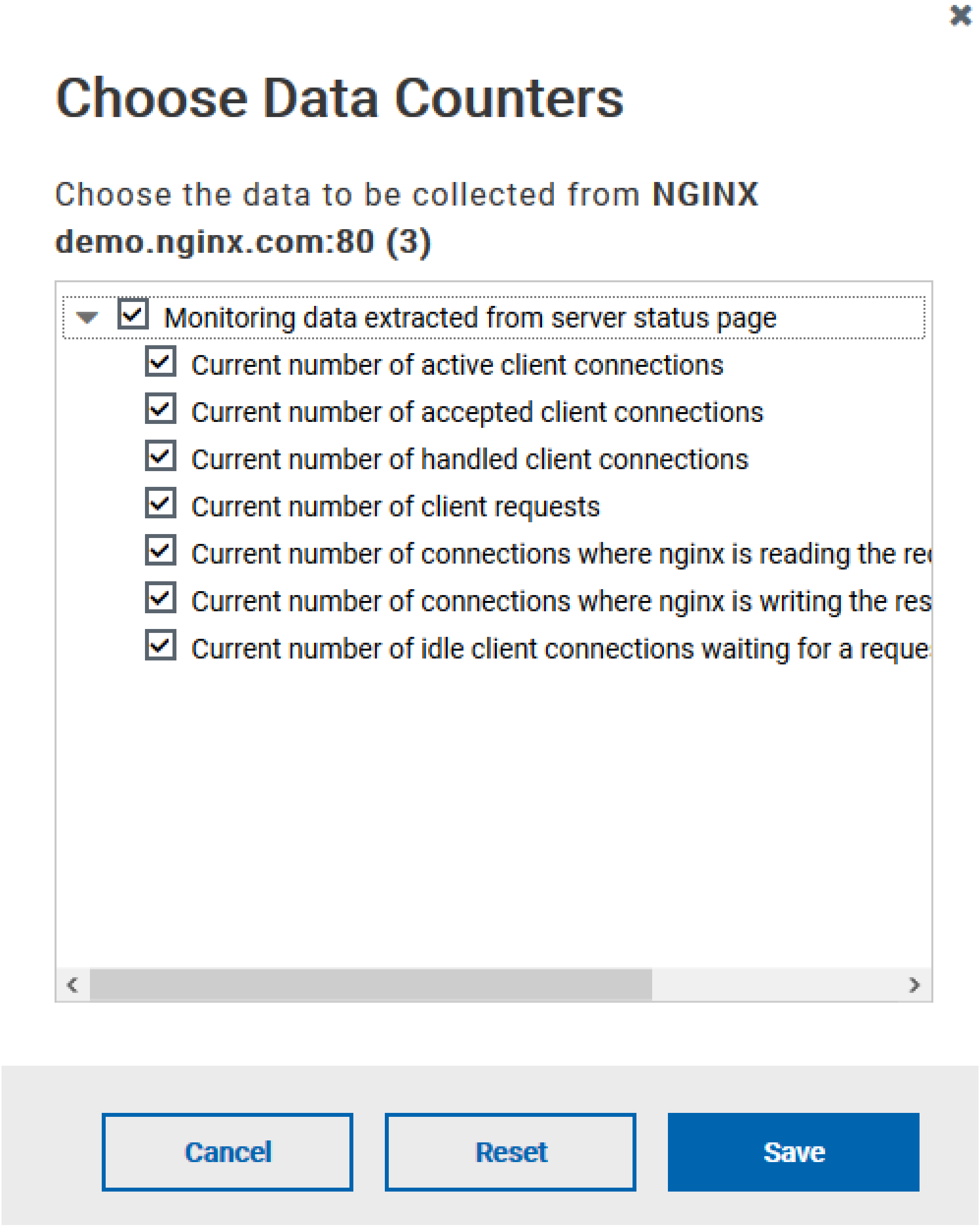
Security

☑ Secured with TLS/SSL

☑ Trust self-signed certificate

e. Click **Add**.

3. Select and save the resource counters to monitor. You can select them from the list where they follow the server logical organization.

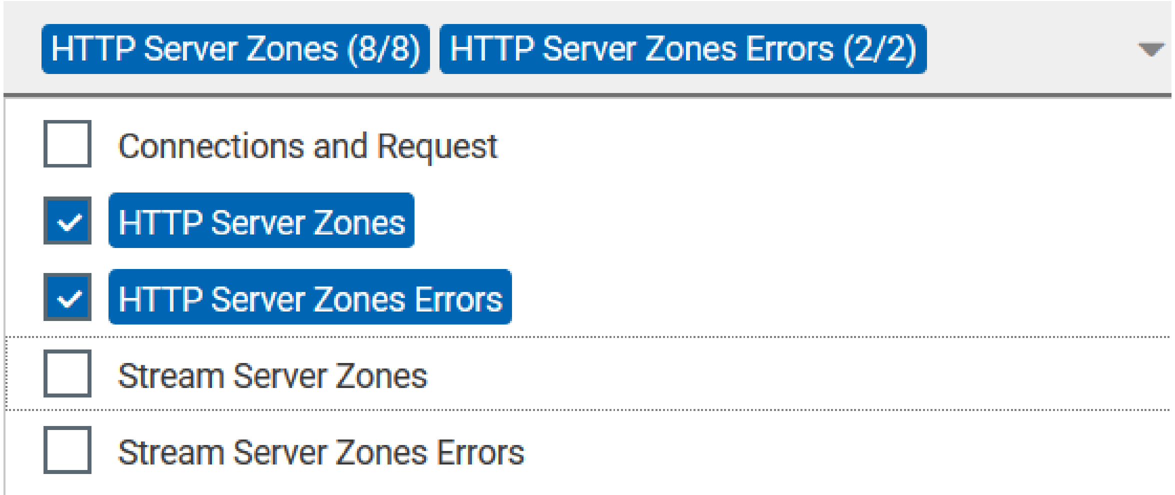


Choose Data Counters

Choose the data to be collected from **NGINX demo.nginx.com:80 (3)**

- ☑ Monitoring data extracted from server status page
  - ☑ Current number of active client connections
  - ☑ Current number of accepted client connections
  - ☑ Current number of handled client connections
  - ☑ Current number of client requests
  - ☑ Current number of connections where nginx is reading the re
  - ☑ Current number of connections where nginx is writing the res
  - ☑ Current number of idle client connections waiting for a reque

For a faster selection, select the counters from the built-in sets drop-down list where they are organized by theme and save your selection.



HTTP Server Zones (8/8) HTTP Server Zones Errors (2/2)

- □ Connections and Request
- ☑ HTTP Server Zones
- ☑ HTTP Server Zones Errors
- □ Stream Server Zones
- □ Stream Server Zones Errors

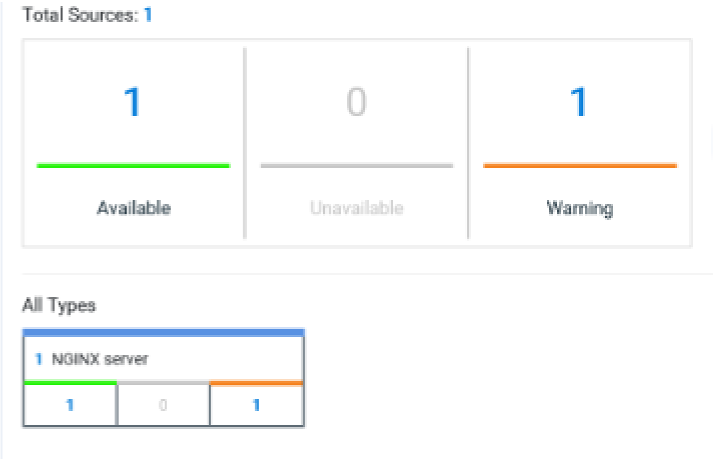
**Note:**

For Docker host, you can select different sets of counters.

- The first level of counters: Generic counters that are related to a Docker Image and all its running Docker Containers (in an exit, running or paused, created, restarted, removing, dead or transitive state).
- The second level of counters: Specific counters that are related to the existing Docker Containers.

### Result

When the selected counters are saved, two tables are displayed in the resource monitoring main page. They contain the total number of sources you have added and the number of sources ordered by type.



Total Sources: 1

| 1 | 0 | 1 |
|---|---|---|
| Available | Unavailable | Warning |

All Types

| 1 NGINX server | | |
|---|---|---|
| 1 | 0 | 1 |

### What to do next

You can click the links in the tables to view the performance metrics of your monitored system.

## Monitoring metrics collected by a Prometheus server

You can monitor your system resources by using metrics data that are collected by a Prometheus server while executing a schedule.

### Before you begin

You must have created a project in HCL OneTest™ Server.

You must have installed and configured a Prometheus server.

Read the concept information about Resource Monitoring and monitoring Prometheus server source. See Resource Monitoring service on page 370

### About this task

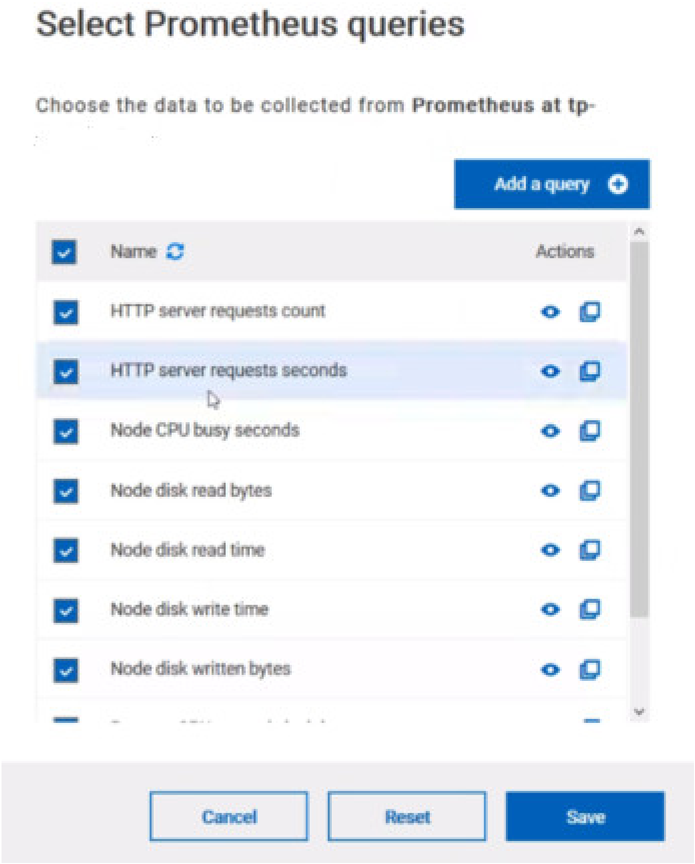
The following procedure describes how to select and create Prometheus queries in HCL OneTest™ Server. You must set a connection between HCL OneTest™ Server and the Prometheus server first to access to the Prometheus queries.

Enter the connection settings, then select queries.

1. Click **Add a source** and select **Add a Prometheus Server**.
2. Enter the connection settings in the **New server** dialog box,.
    a. In **Target host**, enter the IP address or host name and port number of the host where the Prometheus server is installed.
    b. Enter the **Path to the API root of Prometheus** to point to the Prometheus API where the monitoring data are saved.
    c. Enter the security parameters if required. For information about this section, see related links.
    d. Click **Add** to close the dialog box.

    **Result**

    The Prometheus server source is created successfully in the background and a dialog box opens to select the Prometheus data queries.
3. In the **Select Prometheus queries** dialog box, select queries that are available in the list to collect Prometheus metrics data by clicking the checkboxes.
    You can reset the selection.



Select Prometheus queries

Choose the data to be collected from **Prometheus at tp-**

| ☑ | Name | Actions |
|---|---|---|
| ☑ | HTTP server requests count | |
| ☑ | HTTP server requests seconds | |
| ☑ | Node CPU busy seconds | |
| ☑ | Node disk read bytes | |
| ☑ | Node disk read time | |
| ☑ | Node disk write time | |
| ☑ | Node disk written bytes | |

4. Follow these steps to create a query:

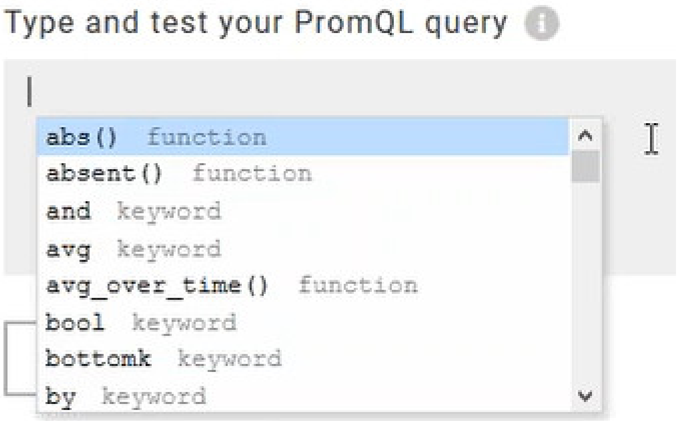
   a. Click **Add a query**.

   b. Enter a name for the new query in the **Create a Prometheus query** dialog box.

   c. Enter a PromQL query.any keyword and press Ctrl+Space in the **Type and test your PromQL query.**

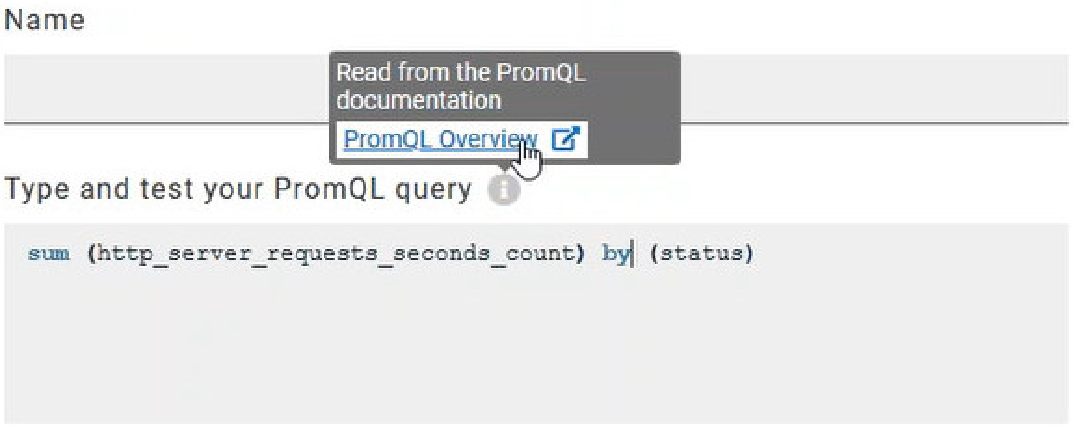
      **Result**

      All the possible combinations of keywords and metrics related to your searched keyword are listed.



d. Select a query in the list.

e. Click the link that points to the 'Querying Prometheus documentation' page in the information tool tip to find other examples of queries in a web page that you can use as samples.


Name
Read from the PromQL documentation
PromQL Overview
Type and test your PromQL query

```
sum (http_server_requests_seconds_count) by (status)
```

f. Click **Test query.**
If the test is successful, a green check is displayed, otherwise you get a red check with an error message.

g. Click **Create**.

5. Proceed as follows to duplicate a query:

   a. Click the **Duplicate** icon ⧉ to duplicate a standard query.

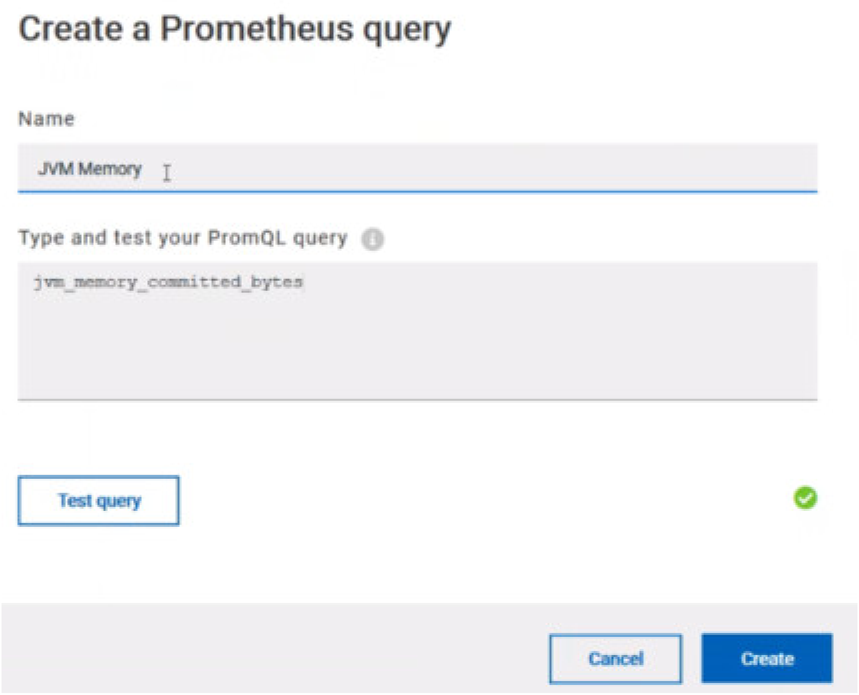
   b. Rename the duplicated query in the **View the Prometheus query** dialog box.
   Make sure that the name you give to the query is not already used.

   c. Modify the query type.

d. Test the query .

e. Click **Create**.

**Note:** You can click Create or Duplicate only if the test of the query is successful and the name of the query is not already in use.



Create a Prometheus query

Name

JVM Memory

Type and test your PromQL query

```
jvm_memory_committed_bytes
```

6. Proceed as follows to modify a query:
   a. Click the **Modify** icon ✎ .
   b. Modify the name and type of the query.
   c. Test the query.
   d. Click **Update**.
   e. Delete duplicated or created queries.
   f. Click **Save** to close the **Select Prometheus queries** dialog box.
7. Click Save and close the **Select Prometheus queries** dialog box.

### Results

The Resource Monitoring page displays the total number of sources you have added to your project and the number of sources ordered by type in cards.

### What to do next

Click the links in the cards to go to the sources page and see the performance metrics for each source you added to your project. For more details, see Viewing the performance metrics

Related information

Resource Monitoring service on page 370

Monitoring host resources on page 375

Viewing the performance metrics

## Monitoring metrics exposed by an OpenMetrics exporter

With HCL OneTest™ Server, you can monitor your system resources by using metrics data that are exposed by an OpenMetrics or a Prometheus exporter through metric counters while executing a schedule.

**Before you begin**

You must have at least one exporter or software to expose monitoring metrics with an OpenMetrics or Prometheus format.

You must have created a project in HCL OneTest™ Server.

**About this task**

The following procedure describes how to select and create OpenMetrics counters in HCL OneTest™ Server. You must set a connection between HCL OneTest™ Server and the OpenMetrics exporter to select counters.

1. Click **Add a source** and select **Add an OpenMetrics exporter** .
2. In the **New OpenMetrics exporter** dialog box that opens, enter the OpenMetrics exporter details to access its exposition endpoint as follows:

    a. In **Target host**, enter the IP address or host name and port number of the host where the server is installed.

    b. Enter the **Path to the exposition endpoint** to point to the application where the data performance results must be stored. As the path depends on the type of exporter used, refer to the exporter's documentation and configuration to indicate the right path.

    c. In **Security**, enter the security parameters if required.

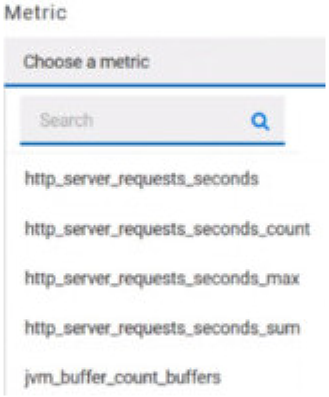
    d. Click **Add** to close the dialog box.
    **Result**
    The source is created successfully in the background. A dialog box opens to select the OpenMetrics counters.

3. Click the checkboxes to select counters in the **Select OpenMetrics counters** dialog box.
4. Follow these steps to create a counter:

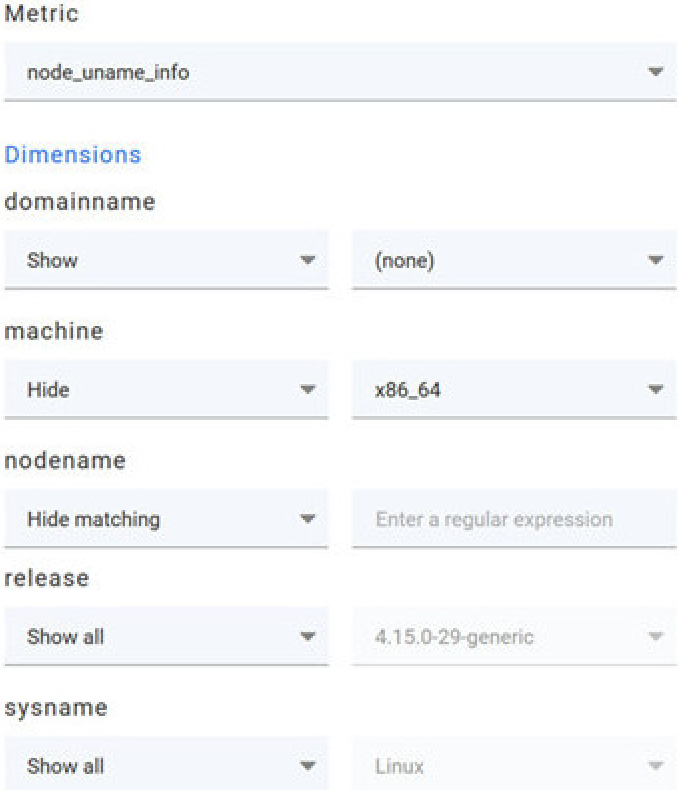
a. Click **Add a counter**.

b. Enter a name for the new counter, in the **OpenMetrics counter**.

c. Select metrics in the drop-down list or enter a keyword in the **Dimensions** field to enable the dynamic input help. You can also use the search field that appears in the list of metrics.


Metric
Choose a metric
Search
http_server_requests_seconds
http_server_requests_seconds_count
http_server_requests_seconds_max
http_server_requests_seconds_sum
jvm_buffer_count_buffers

d. Select **Dimensions** from the dynamic fields when they are required.



Metric
node_uname_info

Dimensions

domainname
Show
(none)

machine
Hide
x86_64

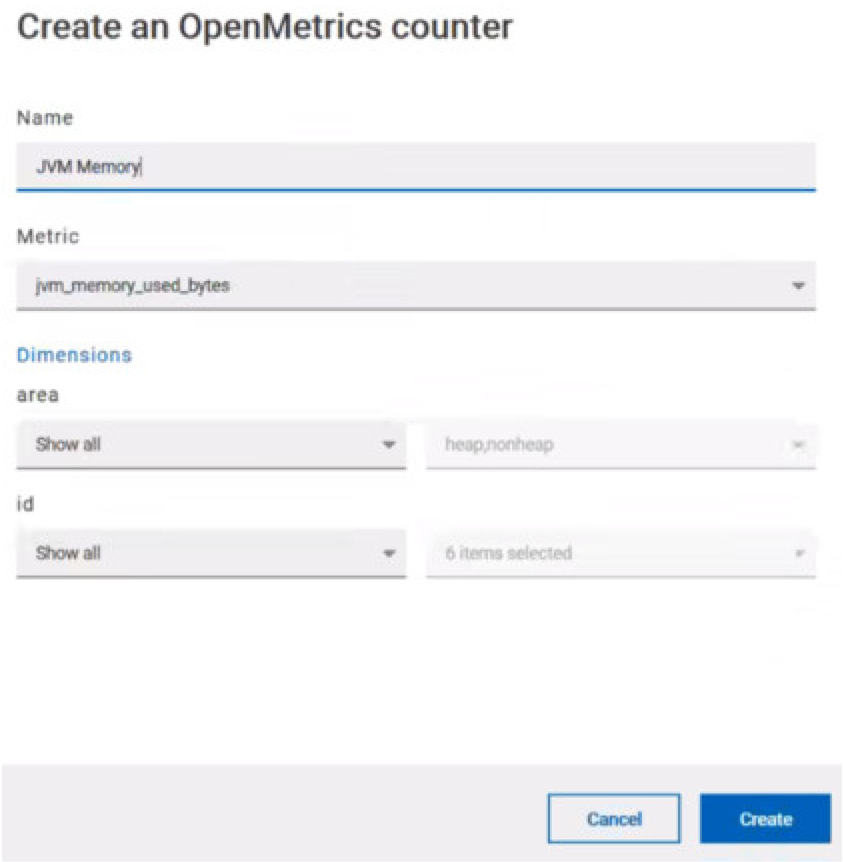
nodename
Hide matching
Enter a regular expression

release
Show all
4.15.0-29-generic

sysname
Show all
Linux

e. Select the appropriate OpenMetrics counter that you are looking for.

f. Click **Create**.



Create an OpenMetrics counter

Name: JVM Memory

Metric: jvm_memory_used_bytes

Dimensions

area: Show all — heap,nonheap

id: Show all — 6 items selected

5. Follow these steps to duplicate a counter:
    a. Select a counter in the list, and click the **Duplicate** icon ⧉.
    b. Modify the name of the counter and the Dimensions.
    c. Click **Create**.
6. Proceed as follows to modify a new counter:
    a. Click the **Modify** icon.
    b. Modify the name, Metrics, and Dimensions of the counter in the **Modify the OpenMetrics counter**.
    c. Click **Update**.

    **Note:** You can create, duplicate, or update a counter only if the name of the counter is not already used.

7. Click **Save** and close the **Select OpenMetrics counters** dialog box.

    **Note:** You can delete counters, but only the ones that you created or duplicated.

**Results**

The Resource Monitoring page displays the total number of sources you have added to your project and the number of sources ordered by type in cards.

**What to do next**

Click the links in the cards to go to the sources page and see the performance metrics for each source you added to your project. For more details, see Viewing the performance metrics.

Related information

Resource Monitoring service on page 370

## Resource Monitoring agents

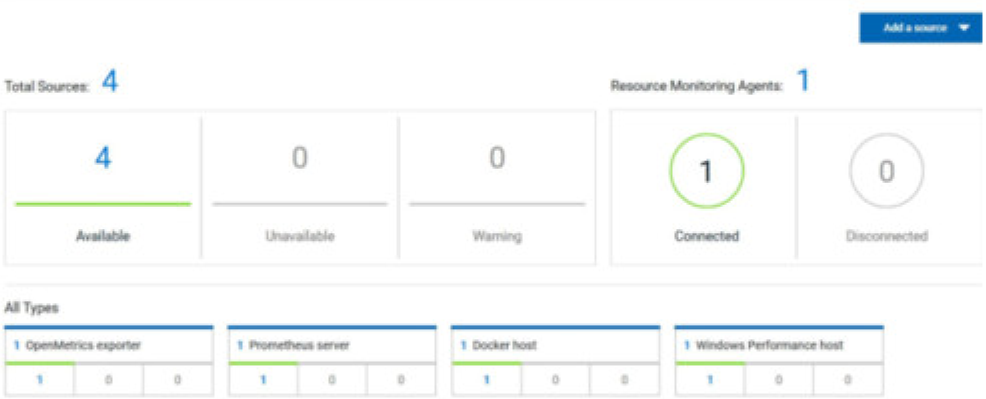
You can connect agents to the Resource Monitoring service from other hosts to monitor a larger set of data collectors, computers, servers, or systems, or if a server to monitor requires authorization access. To monitor these remote servers, you must configure and run the agents so that a connection is set between HCL OneTest™ Server Resource Monitoring service and the agents.

For testing, you would need many hosts. For example, you might have one host with the application server, another host with the database server, and some hosts to apply the user load. Due to network or firewall issues, sometimes, it becomes difficult for multiple hosts to connect to each other. Resource Monitoring agents are installed on the target hosts so that they can establish a connection with HCL OneTest™ Server to gather resource statistics of the target host.

The agents always try to connect with the server through the HTTPS protocol. You must install the agent and start it. From the Resource Monitoring service page in HCL OneTest™ Server, you can copy the command lines to download files and to run the agents. When you stop the agent, the monitoring sources that you have already added persist but the live data will not be available.

You can configure your Resource Monitoring agent as a service so that the agent automatically runs when the host is restarted. For details, see the documentation pages about starting Resource Monitoring agents on Windows or Linux.

When the agent is connected to HCL OneTest™ Server, it is added to the main page of the Resource Monitoring service.



To monitor your host across the agent, you must add the agent as a monitoring source and select performance counters to monitor as described in Monitoring host resources help page.

---

Related information

Resource Monitoring service on page 370

Monitoring host resources on page 375

Starting a Resource Monitoring agent as a service on Windows on page 390

Starting a Resource Monitoring agent as a service on Linux on page 393

## Installing Resource Monitoring agents on remote hosts

You must install the Resource Monitoring agents on the target host for which you want to monitor the performance statistics. You need to run the agents to establish a connection with the Resource Monitoring service.

### Before you begin

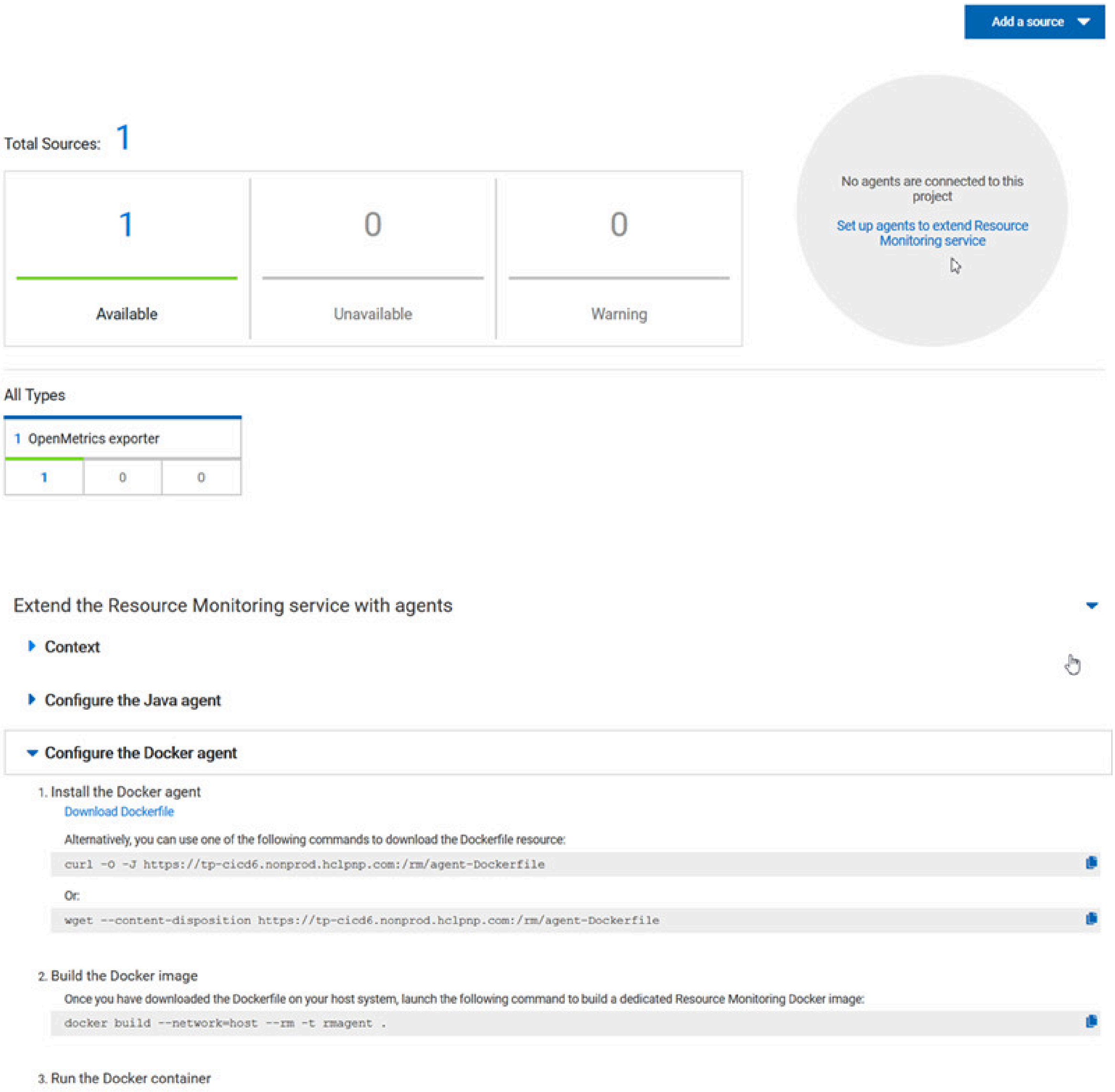
- The Resource Monitoring service does not require access to the agent host but the agent must have reached the service host over HTTPS.
- The Java agent must have been launched from a jar file and requires a Java 8 virtual machine.
- The Docker agent must have been launched in a Docker container and requires Docker 19.03.

### About this task

This task applies to Java and Docker agents. However, the commands used to install agents are different from Java and Docker agents. You can find the commands and instructions in HCL OneTest™ Server in the Resource Monitoring agents page.

Before you start the Docker agents, you must build a Docker image.

1. Click **Set up Agents to extend Resource Monitoring service** link in HCL OneTest™ Server to access the agents page where you can find the instructions and commands that are to be used to install and run the agents.



Add a source
Total Sources: 1
1
Available
0
Unavailable
0
Warning
No agents are connected to this project
Set up agents to extend Resource Monitoring service
All Types
1 OpenMetrics exporter
1
0
0
Extend the Resource Monitoring service with agents
Context
Configure the Java agent
Configure the Docker agent

1. Install the Docker agent

   Download Dockerfile

   Alternatively, you can use one of the following commands to download the Dockerfile resource:

   ```
   curl -O -J https://tp-cicd6.nonprod.hclpnp.com:/rm/agent-Dockerfile
   ```

   Or:

   ```
   wget --content-disposition https://tp-cicd6.nonprod.hclpnp.com:/rm/agent-Dockerfile
   ```

2. Build the Docker image

   Once you have downloaded the Dockerfile on your host system, launch the following command to build a dedicated Resource Monitoring Docker image:

   ```
   docker build --network=host --rm -t rmagent .
   ```

3. Run the Docker container

2. Expand **Configure the Java agent** and click **Download jar file** to download the Java agent, or expand **Configure the Docker agent** and click **Download Docker file** to download a Docker agent.
   You can also use the *curl* and *wget* commands to download the agent without accessing the Resource Monitoring web UI. For more facility, use the code snippets to copy and fill in the commands with the valid offline token and the jar or docker file name.

   ```
   curl -O -J https://hostName/rm/Agent-jar
   wget --content-disposition https://hostName/rm/Agent-jar
   ```

3. Copy the build command that is under **Build the Docker image**, paste it in your console and run it to build a Resource Monitoring Docker image.

```
docker build --network=host --rm -t rmagent
```

**Note:** This step applies to Docker agents only.

4. Configure the command as follows to run your agent:
    a. Copy the appropriate command for Windows or Linux with the code snippets.
    b. Enter the path to the directory that contains the agent .jar file, and paste the command.

**Example**

On Windows:

```
set HCL_ONETEST_OFFLINE_TOKEN=(Enter your offline token here)
java -jar (Enter the name of the downloaded jar file here) --ServiceUrl=https://hostName/rm
 --projectId=<project_id>
```

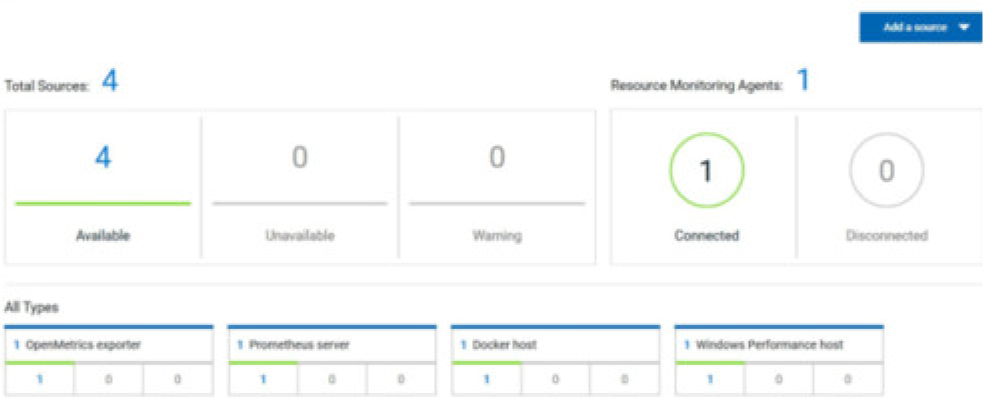
**Example**

On Linux:

```
sudo HCL_ONETEST_OFFLINE_TOKEN=(Enter your offline token here) java -jar (Enter the name of the
 downloaded jar file here) --ServiceUrl=portNumber/rm --projectId=<project_id>
```

**Results**

When the agent is started, the agent is connected to HCL OneTest™ Server. The agent is displayed in the list of connected Resource Monitoring agents with the host name and status of the agent. The agent is also added to the main page of the Resource Monitoring service in HCL OneTest™ Server.



**What to do next**

You can add a new Resource Monitoring source. It can be collected from a service or from the named agents, depending on the capabilities supported by the environment.

## Starting a Resource Monitoring agent as a service on Windows

To ensure that the Resource Monitoring agent starts by itself when the host is running, you can set up the environment in such a way that the Resource Monitoring agent can be started as a service.

**Before you begin**

You must have completed the following tasks:

- Installed Java 8 on the host.
- Added HCL OneTest™ Server to the PATH environment variable.
- Created an offline token to connect the agent securely with appropriate permissions.

  > **Note:** You can create an offline token from the User menu of the Resource Monitoring page or you can re-use your active offline token. The token expires if it is not used for a month.

  To allow remote access to the performance data on Windows 10, you must have considered the following information:

  - The user name and password are the same on the local and remote servers. Otherwise you must have provided remote server credentials.
  - The remote user must be a member of the Performance Monitor Users group (start `lusrmgr.msc` and add the user to this group).
  - The Remote Registry service must be running on the remote host (start `services.msc`) and must have verified the Remote Registry status).
  - File and printer sharing must be enabled on the Network Interface of the remote host that is implied in the communication with the local host.
  - You must have activated the following Windows Firewall rules so that the remote firewall does not block the access:
    - File and Printer Sharing (NB-Name-In)
    - File and Printer Sharing (NB-Session-In)
    - File and Printer Sharing (*)

1. Download the latest release of *winsw*.
   You can choose to download `WinSW.Net2.exe` or `WinSW.Net4.exe` depending on the version of .Net framework that you already have on the host Windows.
2. Create a folder on your local hard drive like `RMAgent-winservice`.
3. Copy the downloaded executable to this new folder and rename the file to `RMAgent-winservice.exe`.
4. Create a new text file in the same folder and name it `RMAgent-winservice.xml`.
5. Copy and adapt the following content to this new RMAgent-winservice.xml file to set up the offline token:

```
<service>
      <id>RMAgent-winservice</id>
      <name>Resource Monitoring Agent</name>
      <description>This service runs the Resource Monitoring Agent.</description>
      <executable>java</executable>
      <env name="HCL_ONETEST_OFFLINE_TOKEN" value="(Enter your offline token here"/>

      <arguments>-jar %BASE%\RMAgent.jar --serviceUrl=https://<service-host>/rm
 --projectId=<projet-id>
 --autoUpgradeDownloadThen=execute:cmd,/c,start,%BASE%\auto-upgrade.bat</arguments>
      <onfailure action="restart" delay="10 sec"/>
      <logmode>rotate</logmode>
</service>
```

**Notes:**

- Replace <*service-host*> by the host name of the host that runs HCL OneTest™ Server.
- Replace <*project-id*> which is the number that you find after /projects/ in the browser's URL when you browse to this project.

6. Create a new text file in the same folder and name it `auto-upgrade.bat`.
7. Copy the following content to this new file to upgrade the agent .jar file automatically:

```
@echo off
for /f "tokens=*" %%a in ('dir /b /od %BASE%\com.hcl.test.rm.agent-*.jar') do set newest=%%a
%BASE%\RMAgent-winservice.exe stop
del %BASE%\RMAgent.jar
mklink %BASE%\RMAgent.jar %BASE%\%newest%
%BASE%\RMAgent-winservice.exe start
```

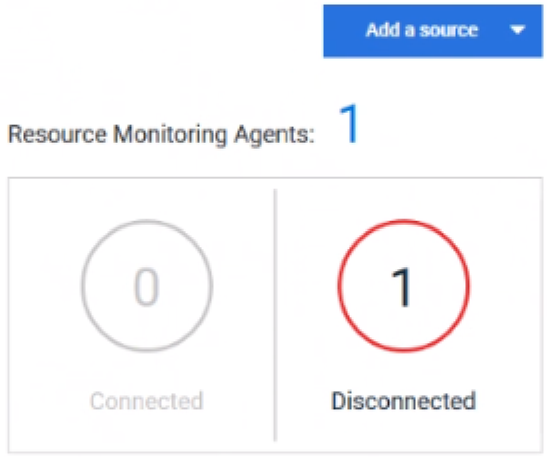
8. Start the command prompt as an administrator and change the directory to the newly created directory: `RMAgent-winservice`.
9. Download the agent .jar file that is available from the agents page in the Resource Monitoring service, under the **Extend the Resource Monitoring service with agents** section and save it to the same directory.
10. Enter the following command in the command prompt to create a symbolic link named `RMAgent.jar` to the agent jar file:

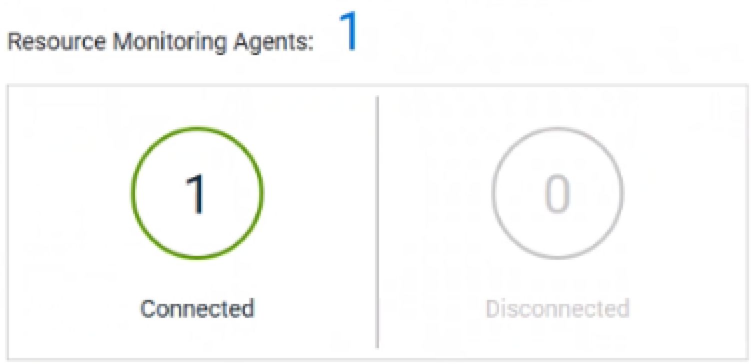
```
mklink RMAgent.jar com.hcl.test.rm.agent-<version-and-datetime>.jar
```

11. Enter the following commands:

```
RMAgent-winservice install
```

```
RMAgent-winservice start
```





**Notes:**

In the Services Windows application, you can check whether the service is up and running. You can see in the Resource Monitoring page whether the Resource Monitoring agent is disconnected or connected.

The Resource Monitoring agents write the logs to the same folder in files named `RMAgent-winservice.out.log`, `RMAgent-winservice.err.log` and `RMAgent-winservice.wrapper.log`. If the Windows service for the agent is not started or if it is not connected to the Resource Monitoring service, verify the log files.

**Results**

The Windows agent is added to the main page of the Resource Monitoring service in HCL OneTest™ Server.

**What to do next**

You can monitor the Windows agent by selecting counters. See Monitoring host resources on page 375.

Related information

Other commands

winsw file

## Starting a Resource Monitoring agent as a service on Linux

To ensure that the Resource Monitoring agent starts by itself when the host is restarted, you can set up the environment in such a way that the Resource Monitoring agent can be started as a service.

**Before you begin**

You must have completed the following tasks:

- Installed Java 8 on the host.
- Added HCL OneTest™ Server to the PATH environment variable.
- Created an offline token to connect the agent securely with appropriate permissions.

> **Note:** You can create an offline token from the User menu of the Resource Monitoring page or you can re-use your active offline token. The token expires if it is not used for a month.

### About this task

This topic relies on systemd services that are the default on most modern Linux distributions. Other ways mi require adaptations of the instructions but the provided script will be a good basis in most cases.

1. Create a folder on your local hard drive like `/opt/RMAgent-linuxservice`.
2. Download the agent .jar file that is available from the Agents page in the Resource Monitoring service, below the **Extend the Resource Monitoring service with agents** section and save it to the same directory.
3. Create a new file `/etc/systemd/system/RMAgent-linuxservice.service`.
4. Add the following content to this new file:

```
[Unit]
Description = Resource Monitoring Agent
After = network.target

[Service]
Type = forking
ExecStart = /opt/RMAgent-linuxservice/RMAgent-linuxservice.sh start
ExecStop = /opt/RMAgent-linuxservice/RMAgent-linuxservice.sh stop
Restart = on-failure
RestartSec = 10

[Install]
WantedBy = multi-user.target
```

5. Create another file `/opt/RMAgent-linuxservice/RMAgent-linuxservice.sh`.

```
#!/bin/sh

#!/bin/sh

# Update the 3 following variables with the Server's host name, project id and offline token:
SERVICE_URL=https://<hostname>/rm
PROJECT_ID=<project-id>
export HCL_ONETEST_OFFLINE_TOKEN=<offline-token>


ARGS="--serviceUrl=$SERVICE_URL --projectId=$PROJECT_ID --autoUpgradeDownloadThen=exitFailure"
SCRIPT=$(readlink -f "$0")
RMAGENT_HOME=$(dirname "$SCRIPT")

# Ensure we're using the latest downloaded jar file
PATH_TO_JAR=`ls -t $RMAGENT_HOME/com.hcl.test.rm.agent-*.jar | head -1`
if [ -z "$PATH_TO_JAR" ]
```

```
then
  cd $RMAGENT_HOME && { curl -k -O -J $SERVICE_URL/agent-jar; cd -; }
  PATH_TO_JAR=`ls -t $RMAGENT_HOME/com.hcl.test.rm.agent-*.jar | head -1`
  if [ -z "$PATH_TO_JAR" ]
  then
    echo "Start the server at $SERVICE_URL to allow download of the latest agent jar file"
    echo "Exiting..."
    exit 1
  fi
fi
SERVICE_NAME="Resource Monitoring Agent"
#Pid file will reside in this script’s folder
PATH_TO_PID=$RMAGENT_HOME/RMAgent-pid
#Log file will reside in this script’s folder
PATH_TO_LOG=$RMAGENT_HOME/RMAgent.log

case $1 in
    start)
        echo "Starting $SERVICE_NAME ..."
        if [ ! -f $PATH_TO_PID ]; then
            nohup java -jar $PATH_TO_JAR $ARGS >> $PATH_TO_LOG 2>&1 &
                        echo $! > $PATH_TO_PID
            echo "$SERVICE_NAME started ..."
        else
            echo "$SERVICE_NAME is already running ..."
        fi
    ;;
    stop)
        if [ -f $PATH_TO_PID ]; then
            PID=$(cat $PATH_TO_PID);
            echo "$SERVICE_NAME stopping ..."
            kill $PID;
            echo "$SERVICE_NAME stopped ..."
            rm $PATH_TO_PID
        else
            echo "$SERVICE_NAME is not running ..."
        fi
    ;;
    restart)
        if [ -f $PATH_TO_PID ]; then
            PID=$(cat $PATH_TO_PID);
            echo "$SERVICE_NAME stopping ...";
            kill $PID;
            echo "$SERVICE_NAME stopped ...";
            rm $PATH_TO_PID
            echo "$SERVICE_NAME starting ..."
            nohup java -jar $PATH_TO_JAR $ARGS >> $PATH_TO_LOG 2>&1 &
                        echo $! > $PATH_TO_PID
            echo "$SERVICE_NAME started ..."
        else
            echo "$SERVICE_NAME is not running ..."
        fi
    ;;
esac
```

**Notes:**

- Replace <*service-hostname*> by the host name of the host that runs HCL OneTest™ Server.
- Replace <*project-id*> which is the number you'll find after /projects/ in the browser's URL when browsing to this project.

**Results**

The Resource Monitoring agent starts automatically when the host restarts. The Linux agent is added to the main page of the Resource Monitoring service in HCL OneTest™ Server.

**What to do next**

You can monitor the Linux Performance host source by adding counters. See Monitoring host resources on page 375.

Related information

systemd services

## Starting a Java Virtual Machine

To monitor the Resource Monitoring data from a Java Virtual Machine (JVM), you must first start the JVM. You must enter the IP address of the JVM, the IP port, and the security parameters in the command before running the Java Virtual Machine.

1. Enter The IP address of the JVM (local or remote host) and the IP port in the command that is used to run the Java Virtual Machine.
2. **Optional:** You can use the authentication security data to start the virtual machine. In this case, you must enter the name of the password file. You can also enter the name of an access file that might be needed if user privileges are required.

**Exemple**

**Parameters used to launch a JVM without security:**

```
java
-Dcom.sun.management.jmxremote
-Dcom.sun.management.jmxremote.port=9010
-Dcom.sun.management.jmxremote.local.only=false
-Dcom.sun.management.jmxremote.authenticate=false
-Dcom.sun.management.jmxremote.ssl=false
-jar  MyapplicationFile.jar
```

**Parameters used to launch a JVM with authentication security:**

```
java
-Dcom.sun.management.jmxremote
-Dcom.sun.management.jmxremote.port=9010
-Dcom.sun.management.jmxremote.local.only=false
-Dcom.sun.management.jmxremote.authenticate=true
-Dcom.sun.management.jmxremote.ssl=false
-Dcom.sun.management.jmxremote.password.file=jmxremote.password
-Dcom.sun.management.jmxremote.access.file=jmxremote.access (this command is optional)
```

```
-jar MyapplicationFile.jar
```

Related information

Resource Monitoring agents on page 387

Monitoring host resources on page 375

# Configuration of a change management system

When you use any application to create and track defects (bugs), issues, or other work items, you can configure the application as a change management system on HCL OneTest™ Server. You can then create defects (bugs), issues, or other work items without the need to open your application.

You use HCL OneTest™ Server to run tests for the software that you develop and view their results. When you discover that you might want to raise defects (bugs), issues, or other types of work items for these tests, you are required to open the application that you use. You can only then, create defects (bugs), issues, or other work items in your application.

When you configure your application as a change management system on HCL OneTest™ Server, you can create defects (bugs), issues, or other work items without the need to open your application.

You can find details in the following table about the usage of the applications that you can configure as a change management system on HCL OneTest™ Server:

| Issue tracking application | Usage as a change management system | Go to... |
|---|---|---|
| Atlassian Jira Software | You can create stories, defects, or tasks as issues in your project in Jira from HCL OneTest™ Server and also launch your project in Jira from HCL OneTest™ Server to track and manage these issues. | Configuration of Jira as a change management system on page 397 |

## Configuration of Jira as a change management system

You can integrate HCL OneTest™ Server with Jira to manage and create defects after the test run is complete.

You can create multiple defects for a test result from the **Results** page of HCL OneTest™ Server, if required. After you create a defect, you can track the defect from the **Change Management System** card on the **Results** page.

When you do not configure the Jira server in your project and if you click the **Open action menu** icon ⋮ from the **Actions** column in the **Results** page, the following message is displayed:

```
The change management system is not configured for the project. You must contact your project owner to configure Jira for the project. If you are a project owner click here.
```

If you are a project *Owner*, you can click the link provided in the message to open the **CHANGE MANAGEMENT SYSTEM** tab. You can then configure the Jira server for your project.

The following topics provide more details about integrating HCL OneTest™ Server with Jira and creating defects.

## Task flow for integrating Jira

You must perform these tasks in sequence as listed in the following table. The table also provides you the links to the information about the tasks.

| | Tasks | More information |
|---|---|---|
| 1 | Install HCL OneTest™ Server. | Installation of the server software on page 35 |
| 2 | Create the type of tests that are supported for running on HCL OneTest™ Server. | Tests supported on HCL OneTest Server on page 20 |
| 3 | Create a project on HCL OneTest™ Server and add repositories to the project to access the test resources available in the respective repository. | Test assets and a server project on page 496 |
| 4 | Configure Jira to your project on HCL OneTest™ Server where the test resources are available.<br>**Note:** You must have the project *Owner* access to configure Jira. | Configuration of Jira as a change management system on page 397 |
| 5 | Create defects for the tests available in your project. | Creating Jira defects on page 401 |

## Generating public and consumer keys

You must first generate a public key and a consumer key for your project that is available on HCL OneTest™ Server to configure Jira.

### Before you begin

You must have completed the following tasks:

- You must have installed Jira on your computer. For more information about installing Jira, refer to Installing Jira software.
- You must be able to access the Jira server from HCL OneTest™ Server.

### About this task

After you add the URL of the Jira server to your project, you can generate both the public key and the consumer key from HCL OneTest™ Server. You can then share both the keys with the administrator of Jira server.

1. Log in to HCL OneTest™ Server, if you are not already logged in.
2. Open your project.
3. Expand **Configure**, and then click **Project**.
4. Click the **Change Management System** tab, and then click **New System**.

   **Result**

   The **Configure Change Management System** page is displayed.
5. Enter the URL of Jira server in the **Change Management System** link field, and then click **Add**.

   **Result**

   A public key and a consumer key are generated and displayed on the **Configure Tracking System** page.

**Results**

You have successfully generated both the public and consumer keys.

**What to do next**

You can then click **Copy** to copy both the keys and save the keys in your local file. You can then share both the keys with the administrator of the Jira server to setup an application link in Jira.

The administrator of the Jira server then sets up the application link in Jira by using both the keys that you shared. After the application link set up is complete, you can then configure the Jira server for your project on HCL OneTest™ Server. See Configuring the Jira server on page 399.

## Configuring the Jira server

After the administrator of the Jira server sets up the application link in Jira by using both the public and consumer keys, you can configure the Jira server for your project on HCL OneTest™ Server where the tests are available.

**Before you begin**

You must have completed the following tasks:

- Generated a public key and a consumer key on HCL OneTest™ Server. See Generating public and consumer keys on page 398.
- Shared both the keys with the administrator of the Jira server to set up an application link in Jira.

**About this task**

When you share both the keys with the administrator of the Jira server, the administrator of the Jira server sets up the application link in Jira.

After the completion of setting up of the application link by the administrator of the Jira server, you can then open the **Change Management System** tab of your project and complete the authorization process to configure the Jira server for your project on HCL OneTest™ Server.

**Tasks performed by the administrator of the Jira server**

As an administrator of the Jira server, you must perform the following actions to set up an application link on the Jira server:

1. Log in to the Jira server and open the **Configure Application Links** page.
2. Enter the URL of the application, and then click **Create new link**.
3. Verify the URL on the **Configure Application URL** page, and then click **Continue** to open the **Link applications** page.
4. Enter the information for the fields specified as follows on the **Link applications** page, and then click **Continue**:
    - **Application Name**: Enter an application name. The application name must be unique. For example, `HCL`.
    - **Application Type**: Select the type of application from the **Application Type** drop-down list. The default selection is **General Application**.
    - Select the **Create incoming link** check box.

    **Note:** You need not set other fields when you set up an application link to establish a connection between the Jira server and HCL OneTest™ Server.
5. Enter the information on the **Link applications** page for the fields specified as follows, and then click **Continue** to set up the application link:
    - **Consumer Key**: Enter the consumer key shared by the administrator of HCL OneTest™ Server.
    - **Consumer Name**: Enter an application name. For example, `HCL_1`.
    - **Public Key**: Enter the public key shared by the administrator of HCL OneTest™ Server.

You have successfully set up the application link on the Jira server to establish a connection between the Jira server and HCL OneTest™ Server.

To know more about the application link, refer to the Configuring the client application as a consumer in Jira section.

1. Open your project.
2. Expand **Configure**, and then click **Project**.
3. Open the **Change Management System** tab.
4. Click the **Open action menu** button from **Jira Integration Details**, and then click **Configure**.
    **Result**
    The **Authorize User** page is displayed.
5. Complete the authorization process on the **Authorize User** page by performing the following actions:
    a. Click the authorization URL of the Jira server to open the **Jira authorization** page.

    b. Log in to your Jira server account with your valid credentials.

    c. Click **Allow** to authorize the Jira server account to generate an access token.
    **Result**

    The authorization is successful.

    The **Access Approved** page is displayed with a verification code.

    d. Copy the verification code, and then paste the verification code in the **Enter Verification Code** field on the **Authorize User** page.

    e. Click **Save**.

   **Result**
   The **Configure Change Management System** page is displayed.

6. Select a Jira server project that is available from the **Project** drop-down list, and then click **Update**.

   **Note:** The change management system name and the change management system link are already displayed on the **Configure Change Management System** page.

#### Results

You have successfully configured the Jira server for your project on HCL OneTest™ Server.

You can see a tick icon ✓ next to the URL of the Jira server on the successful configuration. You can also expand the **Jira Integration Details** section to see the project name of the Jira that you selected.

**Note:** You can delete the URL of the Jira server that you no longer require in your test environment.

#### What to do next

You can create defects for the tests available in your project and track the defects from the **Results** page on HCL OneTest™ Server after the test run is complete. See Creating Jira defects on page 401.

## Creating Jira defects

You can create a Jira defect for a test available in your project and track the defect from HCL OneTest™ Server after the test run is complete.

#### Before you begin

You must have completed the following tasks:

- Been a member of the project with the *Owner* or *Tester* role to create a defect for a test.
- Configured Jira in HCL OneTest™ Server for your project. See Configuration of Jira as a change management system on page 397.

**Note:** You must have the project *Owner* access to configure Jira.

- Initiated a test run of a test available in your branch from the **Execution** page.

### About this task

After you configure Jira for your project on HCL OneTest™ Server, you can create a Jira defect for a test available in your project and track the defect from the **Results** page after the test run is complete.

**Note:** You can create a Jira defect for a test result from the **Results** page only if you have executed the test on HCL OneTest™ Server.

1. Open your project on HCL OneTest™ Server.
2. Run a test from the **Execution** page.
3. Click **Results** from the left pane to open the **Results** page.
4. Select the test result from the **Results** page.
5. Click the **Jira: Raise Defect** icon ◆ from **Actions**.

   **Important:** You can create a Jira defect for a test result only if Jira is configured to your server project. If Jira is not configured with your server project and you click the **Open action menu** button from **Actions** on the **Results** page for a test result, you can see a message is displayed on the **Results** page that states that there is no change management system configured for the project. You must then contact your project *Owner* to configure Jira for the project. You project *Owner* can then click the link provided in the message to open the project to configure Jira for your project.

   **Result**

   The **Raise defect in Jira** page is displayed with the fields available for your project.
6. Enter the information required for all the fields, and then click **Create**.

   **Important:** You can create a story, defect, or task. Creating other issue types are not supported.

   **Result**

   A Jira defect is created.
7. Expand the test result to see the Jira tracking ID details from the **Results** page.

   **Note:** You can raise multiple defects for a test, if required.

### Results

You have created a defect for a test result from the **Results** page on HCL OneTest™ Server.

### What to do next

You can now click the Jira tracking ID from the **Results** page to open the defect and view the details. The **Reporter** field on the Jira server page displays the name of the person who has created the defect. To know more about Jira server defects and how it works, refer to the documentation.

# Integrating with other applications

You can integrate certain applications with HCL OneTest™ Server to run tests, view the test results, and create defects.

## Integration plugin compatibility matrix

You can find information about the versions of the integration plugin that are compatible with HCL OneTest™ Server.

The following table lists the versions of the integration plugin that are required to integrate IBM® Engineering Test Management, HCL Launch, Jenkins, and UrbanCode™ Deploy with HCL OneTest™ Server.

**Note:** You must download the required version of the integration plugin from the HCL® License & Delivery portal based on the existing version of HCL OneTest™ Server. You can then integrate Engineering Test Management, Jenkins, HCL Launch, and UrbanCode™ Deploy with HCL OneTest™ Server.

| HCL OneTest™ Server | 10.1.0 | 10.1.1 | 10.1.2 | 10.1.3 |
|---|---|---|---|---|
| Jenkins plugin | HOT-SERVER-Jenkins-2.0 | HOT-SERVER-Jenkins-3.0 | HOT-SERVER-Jenkins-4.0 | HOT-SERVER-Jenkins-4.0 |
| UrbanCode™ Deploy plugin | HOT-SERVER-UCD-1.0 | HOT-SERVER-UCD-1.0 | HOT-SERVER-UCD-1.0 | HOT-SERVER-UCD-2.0 |
| HCL™ Launch plugin | NA | HOT-SERVER-LAUNCH-1.0 | HOT-SERVER-LAUNCH-2.0 | HOT-SERVER-LAUNCH-2.0 |

## Integration with Azure DevOps

When you use Azure DevOps for continuous integration and continuous development of your application, you can run tests created for your application and available in a project on HCL OneTest™ Server, in Azure DevOps pipelines by using the *HCL OneTest Studio* extension.

### Overview

You can use the *HCL OneTest Studio* extension to integrate HCL OneTest™ Server with Azure DevOps.

The *HCL OneTest Studio* extension enables you to select any type of test available for your project in HCL OneTest™ Server that you can add to your task for the job in the Azure DevOps pipelines.

You must have created the tests in the desktop clients and committed the test assets and test resources to a remote repository. The remote repository must be added to the project.

Depending on the type of test you want to perform on your application, you must have created any or all of the following types of tests in the desktop clients for the application you are testing:

| Type of test | Desktop client |
|---|---|
| • Accelerated Functional Testing suites<br>• Compound tests | HCL OneTest™ UI |
| • Test Suite | HCL OneTest™ API |
| • Compound tests<br>• VU Schedule<br>• Rate Schedule | HCL OneTest™ Performance |

You can now follow the tasks listed in the task flow table to integrate HCL OneTest™ Server with Azure DevOps. See Task flow for integrating Azure DevOps on page 404.

## Task flow for integrating HCL OneTest™ Server with Azure DevOps

The table shows the task flow for integrating HCL OneTest™ Server with Azure DevOps by using the *HCL OneTest Studio* extension. You must perform these tasks in sequence as listed in the following table. The table also provides you the links to the information about the tasks.

| | Tasks | More information |
|---|---|---|
| 1 | Install HCL OneTest™ Server. | Installation of the server software on page 35 |
| 2 | Create tests in the desktop clients for the application you are testing. | • HCL OneTest™ Performance documentation<br>• HCL OneTest™ UI documentation<br>• HCL OneTest™ API documentation |

| | Tasks | More information |
|---|---|---|
| 3 | Create a project on HCL OneTest™ Server and add the test assets that are created in the desktop clients to your project. | Test assets and a server project on page 496 |
| 4 | Create an organization and a project in Azure DevOps for running jobs in Azure DevOps pipelines. | Creating an organization |
| 5 | Access the **Visual Studio Marketplace** portal and search for the latest version of the *HCL OneTest Studio* extension. | Visual Studio Marketplace |
| 6 | Install the latest version of the *HCL OneTest Studio* extension. | Installing the *HCL OneTest Studio* extension on page 405 |
| 7 | Extend the trust certificate if you have used an internal CA certificate. | Extending the trusted CA list |
| 8 | Run tests in an Azure DevOps pipeline. | Running tests in an Azure DevOps Pipeline on page 406 |

Related information

Integration with Azure DevOps on page 403

## Installing the *HCL OneTest Studio* extension

You must install the latest version of the *HCL OneTest Studio* extension in your Azure DevOps organization before you can use the extension for running your application tests in an Azure DevOps pipeline.

**Before you begin**

You must have access to the **Visual Studio Marketplace** portal to install the latest version of the *HCL OneTest Studio* extension.

**About this task**

After you install the latest version of the *HCL OneTest Studio* extension from the **Visual Studio Marketplace** portal in your Azure DevOps organization, you can select the tests that you want to run for your application in an Azure DevOps pipeline by using the *HCL OneTest Studio* extension.

1. Log in to the **Visual Studio Marketplace** portal, if you are not already logged in.
2. Click the **Azure DevOps** tab.
3. Search for the *HCL OneTest Studio* extension.
4. Click the *HCL OneTest Studio* extension.
5. Click **Get it free**.

   **Result**

The **Visual Studio Marketplace** portal for the *HCL OneTest Studio* extension is displayed.

6. Select the organization where you want to run your test from the **Select an Azure DevOps Organization** list.
7. Click **Install**.

   **Result**

   The installation is completed.
8. Click **Proceed to organization**.

   **Result**

   The **Organization** page in Azure DevOps is displayed.
9. Click **Organization settings > Extensions**.

   **Result**

   The *HCL OneTest Studio* extension is displayed as an installed extension.

**Results**

You have installed the *HCL OneTest Studio* extension in your Azure DevOps organization.

**What to do next**

You can run tests that are available in HCL OneTest™ Server as a job in an Azure DevOps pipeline. See Running tests in an Azure DevOps Pipeline on page 406.

## Running HCL OneTest™ Server tests in an Azure DevOps Pipeline

After you install the *HCL OneTest Studio* extension in your organization, you can run tests that are available in HCL OneTest™ Server as a job in Azure DevOps pipelines.

**Before you begin**

You must have completed the following tasks:

- Added the tests that you created in the desktop clients for your application to the project on HCL OneTest™ Server.
- Installed the latest version of the *HCL OneTest Studio* extension in your organization. See Installing the *HCL OneTest Studio* extension on page 405.
- Installed an agent in your pipeline. See Azure Pipelines agents.
- Generated an offline user token from HCL OneTest™ Server. See Generating an offline token on page 482.

**About this task**

After you add the *HCL OneTest Studio* extension in your Azure DevOps organization, you can use an existing pipeline or create a new one to add HCL OneTest™ Server test tasks. You can install an agent or use the one that you installed in your default agent pool. You can add HCL OneTest™ Server tests as tasks to your 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** checkbox 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.

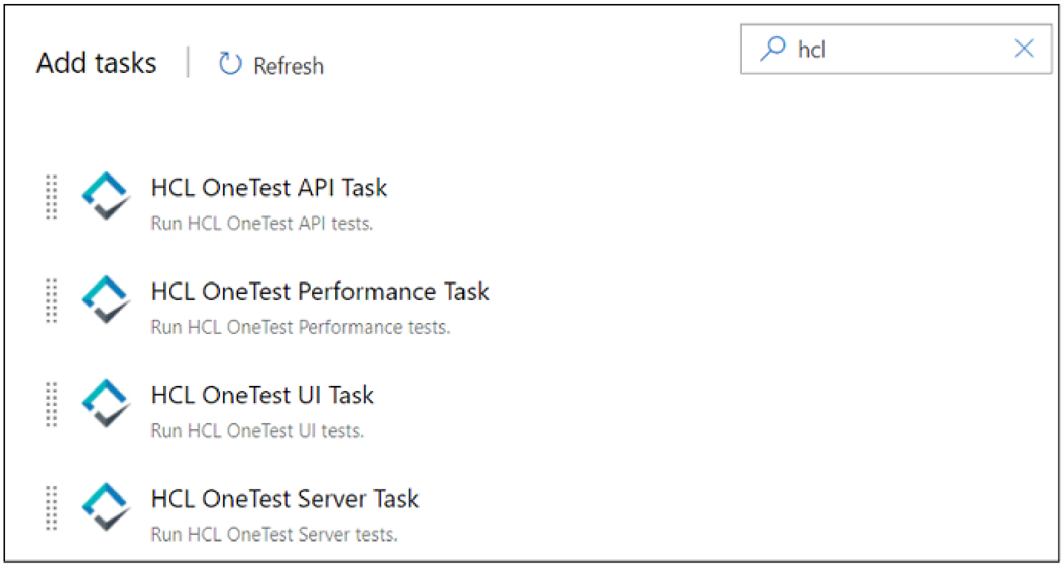
      **Result**

      The **Add tasks** pane is displayed.

   b. Search for the HCL tasks defined in the *HCL OneTest Studio* extension.

      **Result**

      The tasks that you can select are displayed.



Add tasks | Refresh

hcl

HCL OneTest API Task
Run HCL OneTest API tests.

HCL OneTest Performance Task
Run HCL OneTest Performance tests.

HCL OneTest UI Task
Run HCL OneTest UI tests.

HCL OneTest Server Task
Run HCL OneTest Server tests.

c. Select the **HCL OneTest™ Server Task**, 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 step 4 on page 408.

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 task by referring to the following table:

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

| Field | Action |
|---|---|
| Display name | Enter a name of the task. |
| HCL OneTest™ Server service connection | Select the service connection from the drop-down list.<br>If you are selecting the HCL OneTest™ Server service connection for the first time, you must click **New** to add the following details to add an HCL OneTest™ Server service connection, and then save the connection details: |

| Field | Action |
|---|---|
| | ▪ **Server URL** - Enter the URL of HCL OneTest™ Server. The format of the URL is as follows: `https://hostname.`<br>▪ **Offline Token** - Enter the offline token that you generated from HCL OneTest™ Server.<br>▪ **Service connection name** - Enter a service connection name.<br>▪ **Description (optional)** - Enter the details of the service connection if required.<br>▪ Optionally, select the **Grant access permission to all pipelines** checkbox if required.<br>You can save the service connection details. The service connection is available for selection from the **HCL OneTest™ Server service connection** drop-down list.<br>**Note:**<br>You can edit or delete the service connection that you added if required. Click **Manage**, and then select the service connection from the **Service connections** list to open the service connection. You can click **Edit** to edit the service connection details. If you want to delete an existing service connection details, click the **Vertical ellipsis** icon, and then click **Delete** to delete the service connection details.<br>Optionally, you can also create, edit, and delete a service connection from your Azure DevOps project dashboard. Open your project, click **Project settings > Service connections**, and then perform any of the following tasks:<br>▪ Create a new service connection.<br>▪ Edit or delete an existing service connection that you already added. |
| Project | Enter the name of the project containing the test. |
| Branch | Enter the branch where the test assets are stored. |
| Repository Link | Enter the repository path that is configured for the project that contains the test to run. |
| File Path | Enter the path of the test in HCL OneTest™ Server that you want to run. |
| API Test En- | Enter the name of the environment that is configured in the test asset for the test created in HCL OneTest™ API. |

| Field | Action |
|---|---|
| vironment | **Note:** This setting is applicable only if you are running an API suite test. |

   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 from 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 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** checkbox 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 you are testing, 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. The **Reports information** section on the **Task** page displays the names of the report along with its corresponding URLs. The report URLs are the HCL OneTest™ Server URLs where the reports are stored. You can access the report URLs to view the test execution information at any point of time.

You can also view the test reports and logs of the test that was run on the Azure DevOps from the **Results** page on HCL OneTest™ Server. See Test results and reports overview on page 360.

## Integration with HCL Launch

When you use HCL Launch for automating application deployments of your application, you can run tests created for your application and available in a project on HCL OneTest™ Server from HCL Launch by using the HCL OneTest™ Server Launch plugin.

### Overview

You can use the HCL OneTest™ Server Launch plugin to integrate HCL OneTest™ Server with HCL Launch.

You must have created the tests in the desktop clients and committed the test assets and test resources to a remote repository. The remote repository must be added to the project.

Depending on the type of tests you want to perform on your application, you must have created any or all of the following types of tests in the desktop clients for the application you are testing:

| Type of test | Desktop client |
|---|---|
| • Accelerated Functional Testing suites<br>• Compound tests | HCL OneTest™ UI |
| • Test Suite | HCL OneTest™ API |
| • Compound tests<br>• VU Schedule<br>• Rate Schedule | HCL OneTest™ Performance |

You can now follow the tasks listed in the task flow table to integrate HCL OneTest™ Server with HCL Launch. See Task flow for integrating HCL Launch.

## Installing the HCL OneTest™ Server Launch plugin

You must install the latest version of the HCL OneTest™ Server Launch plugin on the HCL Launch server to integrate HCL OneTest™ Server with HCL Launch and run tests on the HCL Launch server.

### Before you begin

You must have downloaded the latest version of the HCL OneTest™ Server Launch plugin from HCL License and Delivery Portal. See HCL® License & Delivery portal.

1. Open the HCL Launch dashboard.
2. Click **Settings**.
3. Click **Automation Plugins** from the **Automation** pane.
4. Click **Load Plugin**.
5. Click **Choose File** to locate and **Open** the compressed HCL OneTest™ Server Launch plugin file.

   **Note:** Do not extract the HCL OneTest™ Server Launch plugin compressed file contents.

6. Click **Submit**.

   **Result**

   The HCL OneTest™ Server Launch plugin is displayed in the **Automation Plugins** tab.

### What to do next

You can run the tests for the application that are available in your HCL OneTest™ Server project on the HCL Launch server. See Running tests on the HCL Launch server on page 425.

## Using a custom trust store for HCL Launch integration

You can use a custom trust store in the HCL OneTest™ Server Launch plugin file to establish a trusted and secure connection between the HCL Launch server and HCL OneTest™ Server.

### Before you begin

You must have configured the certificate that is used by HCL OneTest™ Server as a trusted CA, and then install HCL OneTest™ Server. See Installation of the server software on page 35.

### About this task

If the SSL certificate assigned to HCL OneTest™ Server is signed by an internal Certified Authority (CA), then you must download and import the CA certificate to a custom trust store. You can then use the custom trust store in the **HCL OneTest™ Server test** process step to establish a trusted and secure connection between the HCL Launch server and HCL OneTest™ Server.

**Note:** If the internal CA certificate is already imported to the default trust store that is used by the HCL Launch server, you need not use a custom trust store.

1. Locate the default trust store file (`cacerts` file) in your JRE directory from your computer, and then copy the file to a location of your choice on your computer.
2. Run the following command from the command prompt or terminal to import the CA certificate to your custom trust store:

   ```
   keytool -import -trustcacerts -file <path to the downloaded CA certificate with the file
    extension> -alias <custom label for the certificate> -keystore <path to the trust store>
   ```

   For example,

   `keytool -import -trustcacerts -file C:\Users\ca file.crt -alias alias1 -keystore D:\cert\cacerts`

   **Note:** The default password of the trust store is *changeit*. It remains the same for the custom trust store. If you want to change the password, you can run the following command, and then enter the new password:

   ```
   keytool -storepasswd -keystore <path to the trust store>
   ```

   For example, `keytool -storepasswd -keystore D:\cert\cacerts`

**Results**

You have successfully imported the downloaded CA certificate to the custom trust store.

**What to do next**

You can then run the tests that are available in your HCL OneTest™ Server project from the HCL Launch server.

## Creating a component in HCL® Launch

You must create a component to include artifacts and processes. The artifacts include runnable files, images, databases, configuration instructions. Whereas the processes define the activities that components can perform.

**Before you begin**

- You must be familiar with working with HCL® Launch.
- You must have been granted access to HCL® Launch.

1. Log in to HCL® Launch, if you are not already logged in.

   **Result**

   The HCL® Launch dashboard is displayed.
2. Click **Components**, and then click **Create Component**.
3. Enter a name for the component in the **Name** field.
4. Enter the details in the other optional fields based on your requirement, and then click **Save**.

**Result**

The component that you created is displayed.

**Results**

You have created the component in HCL® Launch.

**What to do next**

You must create a process for the component in HCL® Launch. See Creating a process in HCL Launch on page 414.

Related information

HCL Launch Documentation

## Creating a process in HCL® Launch

You must create a process for the component to include step properties for the test that you want to run from HCL® Launch.

**Before you begin**

- You must be familiar with working with HCL® Launch.
- You must have performed the following tasks:
  - Been granted access to HCL® Launch.
  - Created a component in HCL® Launch. See Creating a component in HCL Launch on page 413.

1. Log in to HCL® Launch, if you are not already logged in.

   **Result**

   The HCL® Launch dashboard is displayed.
2. Click **Components**.

   **Result**

   A list of components that are available in HCL® Launch is displayed.
3. Select the component from the list for which you want to create a process.
4. Click the **Processes** tab, and then click **Create Process**.

   **Result**

   The **Create Process** dialog is displayed.
5. Enter a name for the process in the **Name** field.
6. Select **Operational (No Version Needed)** from **Process Type** drop-down list.
7. Verify the **Default Working Directory** field.

   The **Default Working Directory** field defines the location that the agent uses to run the process. The default value is `${p:resource/work.dir}/${p:component.name}`.

Where `${p:resource/work.dir}` is the default working directory for the agent and `${p:component.name}` is the name of the component.

8. Click **Save**.

   **Result**

   The process that you created is listed in the **Processes** tab and the **Design** tab for the process is displayed.

**Results**

You have created the process for the component in HCL® Launch.

**What to do next**

You must configure the process in HCL® Launch. See Configuring the process on page 415.

Related information

HCL Launch Documentation

## Configuring the process

You must configure the process that you created for the component to organize the steps in the process, specify the properties of the steps, and connect them.

**Before you begin**

- You must be familiar with working with HCL® Launch.
- You must have performed the following tasks:
  - Been granted access to HCL® Launch.
  - Created a component in HCL® Launch. See Creating a component in HCL Launch on page 413.
  - Created a process for the component in HCL® Launch. See Creating a process in HCL Launch on page 414.

**About this task**

When you open any process to configure, the process is displayed in the process editor. The process editor lists the plugins and steps. The required **Start** and **Finish** steps represent the beginning and the end of the process and steps are automatically placed on the design area.

You must provide the values for certain fields in the properties for the selected test step to run tests from HCL® Launch.

1. Log in to HCL® Launch, if you are not already logged in.

   **Result**

The HCL® Launch dashboard is displayed.

2. Click **Components**.

   **Result**

   A list of components that are available in HCL® Launch is displayed.
3. Select the component from the list in which you created the process.
4. Click the **Processes** tab.

   **Result**

   A list of processes that are available for the component is displayed.
5. Select the process from the list that you want to configure.

   **Result**

   The **Design** tab for the process is displayed.
6. Click **HCL OneTest Studio**, and then **HCL OneTest Server** from the left menu.
7. Drag the **Run HCL OneTest Server test** step, and then drop it into the design area.

   **Note:** The selected test must be placed between **Start** and **Finish** steps.

8. Specify the properties for the selected test by performing the following steps:

   a. Click the **Edit** icon.

      **Result**

      The **Edit Properties for Run an HCL OneTest Server test** dialog is displayed.

   b. Specify the properties for the selected test step by referring to the following table:

      The following table lists the required fields that you must provide to run the test from HCL® Launch:

| Fields | Action |
|---|---|
| **Name** | Enter the name for the test step. |
| **HCL OneTest Server URL** | Enter the URL of HCL OneTest™ Server. |
| **Offline Token** | Enter the offline token that you generated from HCL OneTest™ Server. |
| **Team Space Name** | Enter the name of the team space that contains the project.<br>**Note:** The license for the team space must be configured and you must be a member of that team space. |
| **Project Name** | Enter the name of the project that are available in the team space. |

| Fields | Action |
|---|---|
| **Repository Link** | Enter the URL of the Git repository that you added to your project in HCL OneTest™ Server. |
| **File Path** | Enter the path of the test assets that you want to run.<br>You can find the path of the test assets from the **Execution** page in HCL OneTest™ Server |

The following table lists the optional fields that you can provide to run the test from HCL® Launch:

| Fields | Description |
|---|---|
| **Branch Name** | Use this field to enter the name of the branch in the Git repository where you stored test assets. |
| **Custom Trust Store Password** | Use this field to enter the trust store password if you have modified the password while creating the custom trust store.<br>**Note:** If you have not modified the trust store password, you can retain the **Custom Trust Store Password** field blank. |
| **Custom Trust Store Path** | Use this field to enter the file path of your trust store followed by the file name if HCL OneTest™ Server uses an internal CA certificate and you have imported the certificate to a custom trust store. |
| **Datasets** | Use this field to enter the path to the dataset if you want to replace the values of the dataset during a test run.<br>You must ensure that both original and new datasets are in the same workspace and have the same column names. When you enter a value for the **Datasets** field, you must also include the path to the dataset. The path must be in the following format:<br>`/project_name/ds_path/original_ds.csv:/project_name/ds_path/new_ds.csv` |

| Fields | Description |
|---|---|
| | **Note:** You can override multiple datasets that are saved in a different project by adding multiple paths to the dataset separated by a semicolon. |
| **Environment** | Use this field to enter the environment details for the following types of test assets:<br>▪ APISUITE<br>▪ APITEST<br>▪ APISTUB |
| **Labels** | Use this field to add labels to test results when the test run is complete.<br>You can add multiple labels to a test result separated by a comma.<br>For example, *label1, label2*<br>After the test run is complete, then the values provided in the **Labels** field are displayed on the **Results** page of HCL OneTest™ Server. |
| **Secrets Collection Name** | Use this field to enter the name of the Secret if you created secrets collections for your project and to use Secrets at test run time.<br>This field is mandatory only if you want to run the following types of test assets:<br>▪ APISUITE<br>▪ APITEST<br>▪ APISTUB |
| **Show Hidden Properties** | Select this checkbox if you want to use an HTTP Proxy to connect to HCL OneTest™ Server.<br>After you select the **Show Hidden Properties** checkbox, you can configure the HTTP Proxy by using the following fields:<br>▪ **HTTP Proxy Host**: Use this field to enter the hostname of the HTTP proxy that you want to connect to HCL OneTest™ Server.<br>▪ **HTTP Proxy Port**: Use this field to enter the port number where the HTTP proxy is used to listen to both HTTP and HTTPS traffic. |

| Fields | Description |
|---|---|
| | ▪ **HTTP Proxy User name**: Use this field to provide the username for the HTTP proxy.<br>▪ **HTTP Proxy Password**: Use this field to enter a valid password for the user that you specified in the **HTTP Proxy User name** field.<br>**Note:** You can use the **Show Hidden Properties** field only when you want to run the following types of test assets:<br>▪ APISUITE<br>▪ APITEST<br>▪ APISTUB |
| **Start Date** | Use this field to enter the date and time in the following format for running tests at the scheduled date and time:<br>`yyyy/MM/dd/HH:mm`<br>When you enter a value in the **Start date** field, the status of the test displays as `Scheduled` on the **Overview** and **Progress** pages of HCL OneTest™ Server. |
| **Variables** | Use this field to enter the name of the variable and its value if your test requires variables during the test run time.<br>You must enter the variables in the following format:<br>`name_of_the_variable=value_of_the_varibale`<br>You can add multiple variables to the test run separated by a semicolon.<br>For example, *varname1=value1;varname2=value2* |

**Note:** In addition to optional fields, you can also use the following fields from HCL® Launch to configure your test run:

- **Working Directory**
- **Precondition**
- **Post Processing Script**
- **Use Impersonation**
- **Auth Token Restriction**

> You can accept the default values or change the values based on your requirements. For more information about these fields, see the related links.

c. Click **OK** to save the properties for the test.

9. Click **Save** in the design area.

**Results**

You have configured the process for the component in HCL® Launch.

**What to do next**

You must create a resource in HCL® Launch. See Creating a resource in HCL Launch on page 420.

Related information

HCL Launch Documentation

Process step preconditions

Post-processing scripts

User impersonation for process steps

Restricting authentication tokens

## Creating a resource in HCL® Launch

You must create a resource to associate agents with components that you created in HCL® Launch.

**Before you begin**

- You must be familiar with working with HCL® Launch.
- You must have been granted access to HCL® Launch.

1. Log in to HCL® Launch, if you are not already logged in.

   **Result**

   The HCL® Launch dashboard is displayed.
2. Click **Resources**, and then click **Create Top-Level Group**.

   **Result**

   The **Create Resource** dialog is displayed.
3. Enter a name for the resource in the **Name** field.
4. Click **Save**.

**Results**

You have created the resource in HCL® Launch.

**What to do next**

You must configure the resource. See Configuring the resource on page 421.

Related information

HCL Launch Documentation

## Configuring the resource

You must configure the resource to add an agent and associate the agent with the component.

**Before you begin**

- You must be familiar with working with HCL® Launch.
- You must have performed the following tasks:
  - Been granted access to HCL® Launch.
  - Created a component in HCL® Launch. See Creating a component in HCL Launch on page 413.
  - Created a resource in HCL® Launch. See Creating a resource in HCL Launch on page 420.

1. Log in to HCL® Launch, if you are not already logged in.

   **Result**

   The HCL® Launch dashboard is displayed.
2. Click **Resources**.

   **Result**

   A list of resources that are available in HCL® Launch is displayed.
3. Perform the following steps to add an agent to the resource:
   a. Click the resource from the list for which you want to add an agent.
   b. Click the **Actions** icon from the last column, and then click **Add Agent**.
   c. Select the agent from the drop-down list.

      **Note:** The **Name** field is auto populated with the name of the agent.

   d. Click **Save**.

      **Result**

      The selected agent is added to the resource and you can view the status of the agent in the **Status** column.
4. Perform the following steps to add a component to the agent:

a. Click the agent from the list for which you want to add a component.

b. Click the **Actions** icon from the last column, and then click **Add Component**.

c. Select the component from the drop-down list.

**Note:** The **Name** field is auto populated with the name of the component.

d. Click **Save**.

**Result**

The selected component is added to the agent.

**Results**

You have configured the resource in HCL® Launch.

**What to do next**

You must create an application. See Creating an application in HCL Launch on page 422.

Related information

HCL Launch Documentation

## Creating an application in HCL® Launch

You must create an application to fetch all the components together that you want to deploy.

**Before you begin**

- You must be familiar with working with HCL® Launch.
- You must have been granted access to HCL® Launch.

1. Log in to HCL® Launch, if you are not already logged in.

   **Result**

   The HCL® Launch dashboard is displayed.
2. Click **Applications**.
3. Click **Create Applications**, and then **New Applications**.
4. Enter a name for the application in the **Name** field.

   **Result**

   The **Environments** page for the application that you created is displayed.

**Results**

You have created the application in HCL® Launch.

**What to do next**

You must configure the application. See Configuring the application on page 423.

Related information

HCL Launch Documentation

## Configuring the application

You must configure the application to associate resources with environments and define processes to run test assets.

**Before you begin**

- You must be familiar with working with HCL® Launch.
- You must have performed the following tasks:
  - Created a process for the component in HCL® Launch. See Creating a process in HCL Launch on page 414.
  - Created a component in HCL® Launch. See Creating a component in HCL Launch on page 413.
  - Created a process for the component in HCL® Launch. See Creating a process in HCL Launch on page 414.
  - Configure the process for the component in HCL® Launch. See Configuring the process on page 415.
  - Created a resource in HCL® Launch. See Creating a resource in HCL Launch on page 420.
  - Configured the resource in HCL® Launch. See Configuring the resource on page 421.
  - Created an application in HCL® Launch. See Creating an application in HCL Launch on page 422.

1. Log in to HCL® Launch, if you are not already logged in.

   **Result**

   The HCL® Launch dashboard is displayed.
2. Click **Applications**.

   **Result**

   A list of applications that are available in HCL® Launch is displayed.
3. Click the application that you want to configure from the **Name** column.

   **Result**

   The **Environments** page for the selected application is displayed.
4. Perform the following steps to create an environment for the application that you selected:

   a. Click **Create Environment**.
   b. Enter a name for the environment in the **Name** field.
   c. Click **Save**.
5. Perform the following steps to configure resources to the environment:
   a. Click the environment that you created.
   b. Click **Add Base Resources**.

      **Result**

      A list of resources that are available in HCL® Launch is displayed.
   c. Select the checkbox to add resources to the environment.
   d. Click **Save**.

      **Result**

      You can view the corresponding agent and the component that you added for the resource by using the **Expand** icon.
6. Perform the following steps to add the component to the application:
   a. Click **Applications**, and then select your application from the list.
   b. Click the **Components** tab, and then **Add Components**.
   c. Select the checkbox from the drop-down list to add components to the application.
   d. Click **Save**.
7. Perform the following steps to create a process for the application:
   a. Click the **Processes** tab, and then **Create Process**.
   b. Enter a name for the process in the **Name** field.
   c. Click **Save**.

      **Result**

      The **Design** tab for the process that you created is displayed.
8. Drag the component process listed under the **Component Process Steps** option from the left navigation pane and drop it into the design area.
9. Select the component process from the drop-down list in the **Operational (No Version Needed) Process** field.
10. Click **Save**.
11. Click the **Edit** icon, and then change the name of the properties.
12. Click **OK**, and then click **Save**.

**Results**

You have configured the application to run test assets from HCL® Launch.

**What to do next**

You can run test assets from HCL® Launch. See Running HCL OneTest Server tests on the HCL Launch server on page 425.

Related information

HCL Launch Documentation

## Running HCL OneTest™ Server tests on the HCL Launch server

After you install the HCL OneTest™ Server Launch plugin on the HCL Launch server, you can create a `process request` that contains the test for your application, and then run the test on the HCL Launch server.

**Before you begin**

You must have completed the following tasks:

- Added the tests that you created in the desktop clients for your application to the project on HCL OneTest™ Server.
- Installed the latest version of the HCL OneTest™ Server Launch plugin. See Installing the HCL OneTest™ Server Launch plugin on page 412.
- Generated an offline user token from HCL OneTest™ Server. See Generating an offline token on page 482.

**About this task**

After you have installed the HCL OneTest™ Server Launch plugin on the HCL Launch server, you can either use an existing component in your project or create a component. You can create a component process and select the **HCL OneTest™ Server test** step to edit the *step properties* for the test you want to run. After selecting the agent, you can create an application. You can then create an application process for the application, and then submit the application process for a run.

1. Log in to the HCL Launch server, if you are not already logged in.
2. Click **Components** from the HCL Launch dashboard, and then click **Create Component** to create a component.

   **Note:** You can either use an existing component or create a component.

3. Create a component process in the component by performing the following steps:

   a. Open the component that you created.

   b. Click the **Processes** tab from the component dashboard, and then click **Create Process**.
   **Result**

   The **Create Process** dialog box is displayed.

   c. Enter the required values to create a component process and click **Save**.

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

i. Enter the process name in the **Name** field.
ii. Select **Operational (No Version Needed)** from the **Process Type** list.

**Note:** The **Default Working Directory** field displays the folder path where the agent can download the artifacts and create temporary files.

The process that you created is listed in the **Processes** list and the **Design** tab for the process is displayed.

**Note:** The process opens in the process editor. The process editor lists the plugins and steps. The required **Start** and **Finish** steps represent the beginning and the end of the process and are automatically placed on the design area.

4. Select the process step you want to run by completing the following steps:

   a. Search for the **HCL OneTest™ test** process step from the left design pane.

   b. Select the **Run HCL OneTest™ Server test** process step and drag the test into the design area.
   **Result**

   The selected test is placed in between the **Start** and **Finish** steps.

5. Specify the properties for the selected test by performing the following steps:

   a. Click the **Edit** icon ✎.
   **Result**

   The **Edit Properties for Run HCL OneTest™ Server test** dialog box of the selected test is displayed.

   b. Specify the properties for the selected test step by following the action in the table that follows.

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

| Field | Action required for a HCL OneTest™ Server test |
|---|---|
| Name | Enter the name of the step. |
| HCL OneTest™ Server URL | Enter the URL for HCL OneTest™ Server. |
| Offline Token | Enter the offline token that you generated from HCL OneTest™ Server. |

| Field | Action required for a HCL OneTest™ Server test |
|---|---|
| File Path | Enter the file path of the HCL OneTest™ Server test that you want to run. |
| Branch Name | Enter the name of the branch where the test assets are stored. |
| Repository Link | Enter the repository path for the test to run. |
| Project Name | Enter the name of the project that contains the test to run. |
| RIT Environment | Enter the HCL OneTest™ API environment details for the test.<br>**Note:** This field is applicable only for an API test. |
| **Field** | **Optional action for a HCL OneTest™ Server test** |
| Working Directory | Specify an alternative path to the working directory for this step. [1] |
| Post Processing Script | Specify if you want to run any scripts after the completion of the test run. You can click **New** to add new scripts. [1] |
| Precondition | Specify any conditions that are to be completed before the test runs. You can edit the script by clicking the script displayed. [1] |
| **Use Impersonation** checkbox | Select this checkbox to run the test as a different user. [1] |
| Auth Token Restriction | Set the authentication token actions by applying token restrictions.<br>The **System Default** is selected by default. You can add a new token restriction or edit the one already added. [1] |
| Custom Trust Store Path | Enter the file path of your trust store followed by the file name in the **Custom Trust Store Path** field if the HCL OneTest™ Server uses an internal CA certificate and you have imported the certificate to a custom trust store. |
| Custom Trust Store Password | Enter the trust store password if you have modified the password while creating the custom trust store.<br>**Note:** If you have not modified the trust store password, you can keep the **Custom Trust Store Password** field blank. |

c. Click **OK** to save the properties for the test.

6. Click **Save** in the design area.

[1] You need not set this property for a step when you are running an HCL OneTest™ Server test.

7. Click the **Resources** tab from the HCL Launch dashboard and create a resource by clicking **Create Top-Level Group**.
   **Result**

   The created resource is displayed on the **Resource Tree** tab page.
8. Select the agent that runs the test by completing the following steps:

   **Important:** You must have already installed the agent on the HCL Launch server that you want to use.

   a. Select the resource displayed on the **Resource Tree** tab page.

   b. Click the **Horizontal ellipsis** icon ••• for the selected resource.

   c. Click **Add Agent**.

   d. Select the agent to add to the resource, and then click **Save**.
      **Result**

      The selected agent is added to the resource in the **Resource Tree** pane and the status of the agent can also be viewed.

      **Important:** The agent must be **Online** for the test to run.
9. Add the component to the agent by performing the following steps:

   a. Click the **Horizontal ellipsis** icon ••• for the agent.

   b. Click **Add Component** on the list.

   c. Select the component to add to the resource, and then click **Save**.
      **Result**

      The selected component is added to the agent for the resource in the **Resource Tree** pane.
10. Create an application by completing the following steps:

    a. Click the **Applications** tab from the HCL Launch dashboard.

    b. Click **Create Application**.

    c. Complete the details in the **Create Application** dialog box, and then click **Save**.
       **Result**

       The **Environments** tab page is displayed for the created application.

    d. Click **Create Environment** to create an environment for the application that you created.

e. Complete the details in the **Create Environment** dialog box, and then click **Save**.

   **Result**

   The environment that you created is displayed.

f. Click the environment to open, and then click **Add Base Resources**.

g. Select the resource from the list in the **Add Resource to Environment** dialog box, and then click **Save** to add the resource to the environment.

   **Result**

   The resource added to the environment is displayed.

   > **Note:** When you add a resource to an environment, the corresponding agent and the component are displayed for the resource.

h. Click the application from the `breadcrumbs`.

   **Result**

   The **Environments** tab page is displayed for your application.

i. Add the component to the application by performing the following steps:

   i. Click the **Components** tab.
   ii. Click **Add Component**.
   iii. Select the component from the list in the **Add a Component** dialog box, and then click **Save**.

   **Result**

   The selected component is displayed on the **Components** tab page.

j. Create a process for the application by performing the following steps:

   i. Click the **Processes** tab.
   ii. Click **Create Process**.
   iii. Complete the details in the **Create an Application Process** dialog box, and then click **Save**.

   **Result**

   The **Design** tab page for the application process that you created is displayed.

k. Select the component process from the left pane and drag it into the design area.

   > **Note:** You can click the **Edit** icon ✎ to add the properties, if required.

l. Click **Save** in the design area.

11. Select the application process to run the test by completing the following steps:

a. Click **Applications** from the HCL Launch dashboard.

b. Click the application that you configured for a test run.

c. Click **Request Process**.

   **Result**

   The **Create Deployment** page is displayed.

d. Specify the process request by performing the following steps:
   i. Select the application from the **Application** list.
   ii. Select the environment from the **Environment** list.
   iii. Select the application process from the **Process** list.
   iv. Optionally, enter a description in the **Description** field.
   v. Click **Next**.
   vi. Select the component versions as required, and then click **Next**.
   vii. Enable **Run Now** to run the `application process request`.

   > **Note:** You can also schedule the `application process request` at a later point of time. To do this, you can disable **Run Now**, and then specify the date, time, and the recurrence pattern to run the `application process request` at a stipulated time.

   viii. Click **Next**.
   ix. Verify the process request details, and then click **Submit Deployment**.

**Result**

The HCL Launch dashboard shows the progress of the application process request.

**Results**

You have used the HCL OneTest™ Server Launch plugin to integrate HCL OneTest™ Server with HCL Launch and run the test from your project on the HCL Launch server.

After the HCL Launch process request runs successfully, you can view the status of the completed process request displayed as follows:

- `Success`: When the test run is successful
- `Failed`: When the test run is failed

**What to do next**

- You can view the details of the test run as a process from the HCL Launch dashboard by performing the following action:
    - Expand the *step*. You can then expand the application process. You can then hover over the process, and then click the **Output Log** icon . The output log is displayed. You can verify the log details.
- You can also view the test reports and logs of the test that was run on the HCL Launch server from the **Results** page on HCL OneTest™ Server. See Test results and reports overview on page 360.

## Integration with Jenkins

When you integrate HCL OneTest™ Server with Jenkins, you can run tests created for your application and available in a project on HCL OneTest™ Server from a Jenkins server by using the HCL OneTest™ Server Jenkins plugin.

### Prerequisites

You must have installed Jenkins on your computer. For more information about installing Jenkins, refer to Installing Jenkins.

You can then run the following command to start the Jenkins server to support UTF-8 character sets:

```
java -Dfile.encoding=UTF8 -jar jenkins.war
```

### Overview

You can use the HCL OneTest™ Server Jenkins plugin to integrate HCL OneTest™ Server with Jenkins.

You must have created the tests in the desktop clients and committed the test assets and test resources to a remote repository. The remote repository must be added to the project.

Depending on the type of tests you want to perform on your application, you must have created any or all of the following types of tests in the desktop clients for the application you are testing:

| Type of test | Desktop client |
|---|---|
| • Accelerated Functional Testing suites<br>• Compound tests | HCL OneTest™ UI |
| • Test Suite | HCL OneTest™ API |

| Type of test | Desktop client |
|---|---|
| • Compound tests<br>• VU Schedule<br>• Rate Schedule | HCL OneTest™ Performance |

You can now follow the tasks listed in the task flow table to integrate HCL OneTest™ Server with Jenkins. See Task flow for integrating Jenkins.

## Task flows for running test assets from Jenkins

You can perform certain tasks to run test assets from the Jenkins **Freestyle** project.

The following table lists the task flows for running test assets from the Jenkins **Freestyle** project:

| Tasks | More information |
|---|---|
| Install the HCL OneTest™ Server Jenkins plugin. | Installing the HCL OneTest Server Jenkins Plugin on page 432 |
| Import the CA certificate to your custom trust store. | Using a custom trust store on page 433 |
| Configure the **Freestyle** project. | Configuring the Freestyle project on page 434 |
| Run HCL OneTest™ Server test assets on Jenkins. | Running HCL OneTest Server tests on the Jenkins server on page 437 |

## Installing the HCL OneTest™ Server Jenkins Plugin

You must install the latest version of the HCL OneTest™ Server plugin on the Jenkins server before you can use the plugin for running tests that are available in your HCL OneTest™ Server project on the Jenkins server.

**Before you begin**

You must have downloaded the HCL OneTest™ Server Jenkins plugin `4.0` from the HCL License and Delivery portal.

1. Open the Jenkins dashboard.
2. Click **Manage Jenkins > Manage Plugins**.
3. Click **Advanced**.
4. Click **Choose File** to locate and **Open** the HCL OneTest™ Server Jenkins plugin file from the **Upload Plugin** section.
5. Click **Upload**.

   **Result**

   The HCL OneTest™ Server Jenkins plugin is displayed in the **Installed** tab.

**What to do next**

You can run the tests that are available in HCL OneTest™ Server on the Jenkins server. See Running HCL OneTest™ Server tests on the Jenkins server on page 437.

## Using a custom trust store

You can use a custom trust store in the Jenkins build step of a HCL OneTest™ Server Jenkins plugin to establish a trusted and secure connection between the Jenkins server and HCL OneTest™ Server.

**Before you begin**

You must have configured the certificate that is used by HCL OneTest™ Server as a trusted CA, and then install HCL OneTest™ Server. See Installation of the server software on page 35.

**About this task**

If the SSL certificate assigned to HCL OneTest™ Server is signed by an internal Certified Authority (CA), then you must download and import the CA certificate to a custom trust store. You can then use the custom trust store in the Jenkins plugin build step to establish a trusted and secure connection between the Jenkins server and HCL OneTest™ Server.

**Note:** If the internal CA certificate is already imported to the default trust store that is used by the Jenkins server, you need not use a custom trust store.

**Restriction:**

When you use Red Hat Enterprise Linux (RHEL) operating systems for Jenkins, you must run the Jenkins service with a user who has access to the custom trust store path to utilize the custom trust store feature. To change the Jenkins user, you must open the `/etc/sysconfig/jenkins` file and set the JENKINS_USER to the user who has access to the custom trust store path.

You can then run the following commands to set JENKINS_USER to another user who has access to the path of the custom trust store:

```
$JENKINS_USER= <username>
```

For example, `$JENKINS_USER= <user1>`

**Note:** You must ensure that the user account is available in the `/etc/passwd` file.

You can then run the following commands to change the ownership of the Jenkins folder:

```
chown -R username:username /var/lib/jenkins
```

```
chown -R username:username /var/cache/jenkins
```

```
chown -R username:username /var/log/jenkins
```

🚫 For example,

`chown -R user1:user1 /var/lib/jenkins`

`chown -R user1:user1 /var/cache/jenkins`

`chown -R user1:user1 /var/log/jenkins`

After the change of ownership is complete, run the following command to restart the Jenkins server:

```
/etc/init.d/jenkins restart
```

1. Locate the default trust store file (`cacerts` file) in your JRE directory from your computer, and then copy the file to a location of your choice on your computer.
2. Run the following command from the command prompt or terminal to import the CA certificate to your custom trust store:

   ```
   keytool -import -trustcacerts -file <path to the downloaded CA certificate with the file
    extension> -alias <custom label for the certificate> -keystore <path to the trust store>
   ```

   For example,

   `keytool -import -trustcacerts -file C:\Users\ca file.crt -alias alias1 -keystore D:\cert\cacerts`

   **Note:** The default password of the trust store is *changeit*. It remains the same for the custom trust store. If you want to change the password, you can run the following command, and then enter the new password:

   ```
   keytool -storepasswd -keystore <path to the trust store>
   ```

   For example, `keytool -storepasswd -keystore D:\cert\cacerts`

#### Results

You have successfully imported the downloaded CA certificate to the custom trust store.

#### What to do next

You can add the HCL OneTest™ Server tests to the Jenkins build step, and then run the tests from the Jenkins server.

## Configuring the Freestyle project

You must configure a **Freestyle** project to add a build step, and then run tests that are available in your HCL OneTest™ Server project from Jenkins.

#### Before you begin

You must have completed the following tasks:

- Installed the HCL OneTest™ Server Jenkins plugin on the Jenkins server. See Installing the HCL OneTest Server Jenkins Plugin on page 432.
- Created a Jenkins **Freestyle** project.
- Generated an offline user token from HCL OneTest™ Server. See Managing access to HCL OneTest Server on page 482.

1. Click the **Build** tab, and then click **Add build step**.
2. Select the **Run HCL OneTest Server test** option from the drop-down list.
3. Provide the details about the test run for the fields in the following table:

| Field | Description |
|---|---|
| Name | Enter a name for the Jenkins build step. |
| Use Custom Trust Store | Select the **Use Custom Trust Store** checkbox if you have used an internal CA certificate and have imported the certificate to the custom trust store. You can then enter the file path of your trust store followed by the file name in the **Custom Trust Store Path** field.<br>Select the **Use Custom Password for the Trust Store** checkbox if you have modified the trust store password, and then enter the new password in the **Custom Trust Store Password** field.<br>**Note:** If you have not modified the trust store password, you must clear the **Use Custom Password for the Trust Store** checkbox. |
| Server URL | Enter the URL of HCL OneTest™ Server.<br>The format of the URL is as follows: *https://hostname*. |
| Offline Token | Enter the offline user token that you generated from HCL OneTest™ Server. |
| Team Space | Select the name of the team space from the drop-down list.<br>The **Team Space** drop-down list displays the names of the team spaces if you are an *Owner* or a *member* of the team space. |
| Project | Select a project from the drop-down list.<br>The **Project** drop-down list displays the projects that are available in the corresponding team space of HCL OneTest™ Server. |

| Field | Description |
|---|---|
| | **Note:** The **Project** drop-down list displays the projects where you are an *Owner* or a *member* of HCL OneTest™ Server. You must be an *Owner* or a *Tester* of the project that is available in the team spaces to run the tests from the Jenkins server. |
| Branch | Select the branch from the drop-down list.<br>The **Branch** drop-down list displays the branches available in the corresponding project of HCL OneTest™ Server.<br>**Note:** After you select the branch from the **Branch** drop-down list, if you want to change the URL, offline user token, or the project in the **Build**, then you must close the build step. Then you must click **Add build step**, select **Run HCL OneTest™ Server test**, and then enter the details. |
| Asset Type | Select the test asset type that you want to run from the **Asset Type** drop-down list.<br>The available options for the **Asset Type** field are as follows:<br>◦ AFT Suite<br>◦ API Suite<br>◦ Compound Test<br>◦ Rate Schedule<br>◦ VU Schedule<br>**Notes:**<br>◦ You cannot run the following test assets from the Jenkins server:<br>▪ API test<br>▪ Functional test<br>▪ HCL AppScan CodeSweep<br>▪ JMeter Test<br>▪ JUnit Test<br>▪ Performance test<br>▪ Postman resources<br>◦ The **Test Environment** field is mandatory if you select **APISUITE** as an asset type to run an API suite. |
| Test | Select the required test from the drop-down list. |

| Field | Description |
|---|---|
| | The **Tests** drop-down list displays the available test assets from the corresponding branch in the following format:<br>`Project_name [Path: the path of the test assets] [Repo: URL of the Git repository that the test belongs to]`<br>**Note:** After you select the test details from the **Tests** drop-down list if you want to change the URL, offline user token, or the project in the **Build step**, then you must close the build step. You must then click **Add build step**, select **Run HCL OneTest™ Server test**, and then enter the details. |
| Test Environment | This field is mandatory only if you select **Asset Type** as **APISUITE**.<br>The **Test Environment** list displays the available test environments from HCL OneTest™ Server.<br>**Note:** The following message is displayed when you select any other asset type apart from **APISUITE** in the **Asset Type** field:<br>`You can select Test Environment only if you are running tests of type APISUITE.` |

**Note:** You can run multiple tests sequentially in the same job by adding multiple build steps and providing details for the tests that you want to run.

4. Click **Save** to save the build step.

### Results

You have configured the **Freestyle** project by adding the build step.

### What to do next

You can run test assets from the Jenkins server. See Running HCL OneTest Server tests on the Jenkins server on page 437.

## Running HCL OneTest™ Server tests on the Jenkins server

After you install the HCL OneTest™ Server Jenkins plugin on the Jenkins server, you can run tests that are available in your HCL OneTest™ Server project on the Jenkins server.

### Before you begin

You must have completed the following tasks:

- Added the tests that you created in the desktop clients for your application to the project on HCL OneTest™ Server.
- Install the latest version of the HCL OneTest™ Server Jenkins plugin on the Jenkins server. See Installing the HCL OneTest™ Server Jenkins Plugin on page 432.
- Generated an offline user token from HCL OneTest™ Server. See Generating an offline token on page 482.
- Created a Jenkins free-style software project. See Building a software project in Jenkins.

1. Login to the Jenkins server, if you are not already logged in.
2. Open your Jenkins free-style software project.
3. Click **Configure**.
4. Click the **Build** tab.
5. Click **Add build step**, and then click **Run HCL OneTest™ Server test**.

   **Result**

   The **Run HCL OneTest™ Server test** pane is displayed.

   a. Provide details about the test run by referring to the following table.

| Field | Description |
|---|---|
| Name | Enter a name for the Jenkins build step. |
| Use Custom Trust Store | Select the **Use Custom Trust Store** checkbox if you have used an internal CA certificate and have imported the certificate to the custom trust store.<br>You can then enter the file path of your trust store followed by the file name in the **Custom Trust Store Path** field.<br>You can select the **Use Custom Password for the Trust Store** checkbox if you have modified the trust store password. Enter the new password.<br>**Note:** If you have not modified the trust store password, you can keep the **Use Custom Password for the Trust Store** checkbox unselected. |
| Server URL | Enter the URL of HCL OneTest™ Server. The format of the URL is as follows: *https://hostname*. |
| Offline Token | Enter the offline user token that you generated from HCL OneTest™ Server. |
| Project | Select a project from the **Project** list. The **Project** list displays the projects where you are the owner or member of HCL OneTest™ Server. |
| Branch | Select the branch from the **Branch** list. The **Branch** list displays the branches available in the corresponding project of HCL OneTest™ Server. |

| Field | Description |
|---|---|
| | **Note:** After you select the branch from the **Branch** list, if you want to change the URL, offline user token, or the project in the **Build**, then you must close the build step. Then you must click **Add build step**, select **Run HCL OneTest™ Server test**, and then enter the details. |
| Asset Type | Select the test asset type of the test that you want to run from the **Asset Type** list. The available asset types are as follows:<br>▪ AFTSUITE<br>▪ COMPOUND<br>▪ VUSCHEDULE<br>▪ RATESCHEDULE<br>▪ APISUITE<br>**Note:** The test environment is mandatory if you select **APISUITE** as an asset type to run an API suite. |
| Test | Select the required test from the **Tests** list. The **Tests** list displays the available test assets from the corresponding branch in the selected project, test asset path, and the repository (that the test belongs to) from HCL OneTest™ Server based on the type of the test asset you selected from the **Asset Type** list.<br>**Note:** After you select the test details from the **Tests** list, if you want to change the URL, offline user token, or the project in the **Build**, then you must close the build step. You must then click **Add build step**, select **Run HCL OneTest™ Server test**, and then enter the details. |
| Test Environment | This field is mandatory only if you are running an API suite test. Based on the asset type as **APISUITE** and test from the **Tests** list that you select, the **Test Environment** list displays the available test environments from HCL OneTest™ Server. |

| Field | Description |
|---|---|
| | **Note:** The following message is displayed when you select any other asset type apart from **APISUITE** in the **Asset Type** field: `You can select Test Environment only if you are running tests of type APISUITE`. |

b. Click **Save** to save the build step.

c. Optionally, you can run multiple tests sequentially in the same job by adding multiple build steps and provide details for the tests that you want to run.

6. Click **Build Now** from the left pane to run the test on the Jenkins server.

**Results**

The Jenkins dashboard shows the progress of the test run.

**What to do next**

After the Jenkins build completes, you can view the test results. You can click the build number from the **Build History** pane on the Jenkins dashboard. You can then click **Console Output** to view a detailed log of the build from the console output.

The `Test Result` displays the status of the completed test that you ran.

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

The **Reports information** section displays the names of the report along with its corresponding URLs. The report URLs are the HCL OneTest™ Server URLs where the reports are stored. You can access the report URLs to view the test execution information at any point of time.

You can also view the test reports and logs of the test that was run on the Jenkins server from the **Results** page on HCL OneTest™ Server. See Test results and reports overview on page 360.

## Integration with UrbanCode Deploy

When you use UrbanCode™ Deploy (UCD) for automating application deployments of your application, you can run tests created for your application and available in a project on HCL OneTest™ Server from UCD by using the HCL OneTest™ Server UCD plugin.

### Overview

You can use the HCL OneTest™ Server UCD plugin to integrate HCL OneTest™ Server with UCD.

You must have created the tests in the desktop clients and committed the test assets and test resources to a remote repository. The remote repository must be added to the project.

Depending on the type of tests you want to perform on your application, you must have created any or all of the following types of tests in the desktop clients for the application you are testing:

| Type of test | Desktop client |
|---|---|
| • Accelerated Functional Testing suites<br>• Compound tests | HCL OneTest™ UI |
| • Test Suite | HCL OneTest™ API |
| • Compound tests<br>• VU Schedule<br>• Rate Schedule | HCL OneTest™ Performance |

You can now follow the tasks listed in the task flow table to integrate HCL OneTest™ Server with UCD. See Task flow for integrating UrbanCode Deploy.

## Installing the HCL OneTest™ Server UCD plugin

You must install the latest version of the HCL OneTest™ Server UCD plugin to integrate HCL OneTest™ Server with UCD and run tests on the UCD server.

**Before you begin**

You must have downloaded the HCL OneTest™ Server UCD plugin `2.0` from the HCL® License & Delivery portal.

1. Open the UCD dashboard.
2. Click **Settings**.
3. Click **Automation Plugins** from the **Automation** pane.
4. Click **Load Plugin**.
5. Click **Choose File** to locate and **Open** the compressed HCL OneTest™ Server UCD plugin file.

   **Note:** Do not extract the HCL OneTest™ Server UCD plugin compressed file contents.

6. Click **Submit**.

   **Result**

   The HCL OneTest™ Server UCD plugin is displayed in the **Automation Plugins** tab.

**What to do next**

You can run the tests for the application that are available in your HCL OneTest™ Server project on the UCD server. See Running tests on the UCD server on page 454.

## Using a custom trust store for UCD integration

You can use a custom trust store in the HCL OneTest™ Server UCD plugin file to establish a trusted and secure connection between the UCD server and HCL OneTest™ Server.

### Before you begin

You must have configured the certificate that is used by HCL OneTest™ Server as a trusted CA, and then install HCL OneTest™ Server. See Installation of the server software on page 35.

### About this task

If the SSL certificate assigned to HCL OneTest™ Server is signed by an internal Certified Authority (CA), then you must download and import the CA certificate to a custom trust store. You can then use the custom trust store in the **HCL OneTest™ Server test** process step to establish a trusted and secure connection between the UCD server and HCL OneTest™ Server.

**Note:** If the internal CA certificate is already imported to the default trust store that is used by the UCD server, you need not use a custom trust store.

1. Locate the default trust store file (`cacerts` file) in your JRE directory from your computer, and then copy the file to a location of your choice on your computer.
2. Run the following command from the command prompt or terminal to import the CA certificate to your custom trust store:

   ```
   keytool -import -trustcacerts -file <path to the downloaded CA certificate with the file
    extension> -alias <custom label for the certificate> -keystore <path to the trust store>
   ```

   For example,

   ```
   keytool -import -trustcacerts -file C:\Users\ca file.crt -alias alias1 -keystore D:\cert\cacerts
   ```

   **Note:** The default password of the trust store is *changeit*. It remains the same for the custom trust store. If you want to change the password, you can run the following command, and then enter the new password:

   ```
   keytool -storepasswd -keystore <path to the trust store>
   ```

   For example, `keytool -storepasswd -keystore D:\cert\cacerts`

### Results

You have successfully imported the downloaded CA certificate to the custom trust store.

### What to do next

You can then run the tests that are available in your HCL OneTest™ Server project from the UCD server.

## Creating a component in UrbanCode™ Deploy

You must create a component to include artifacts and processes. The artifacts include runnable files, images, databases, configuration instructions. Whereas the processes define the activities that components can perform.

#### Before you begin

- You must be familiar with working with UrbanCode™ Deploy.
- You must have been granted access to UrbanCode™ Deploy.

1. Log in to UrbanCode™ Deploy, if you are not already logged in.

   **Result**

   The UrbanCode™ Deploy dashboard is displayed.
2. Click **Components**, and then click **Create Component**.
3. Enter a name for the component in the **Name** field.
4. Enter the details in the other optional fields based on your requirement, and then click **Save**.

   **Result**

   The component that you created is displayed.

#### Results

You have created the component in UrbanCode™ Deploy.

#### What to do next

You must create a process for the component in UrbanCode™ Deploy. See Creating a process in UrbanCode Deploy on page 443.

Related information

IBM UrbanCode Deploy Documentation

## Creating a process in UrbanCode™ Deploy

You must create a process for the component to include step properties for the test that you want to run from UrbanCode™ Deploy.

#### Before you begin

- You must be familiar with working with UrbanCode™ Deploy.
- You must have performed the following tasks:

  - Been granted access to UrbanCode™ Deploy.
  - Created a component in UrbanCode™ Deploy. See Creating a component in UrbanCode Deploy on page 443.

1. Log in to UrbanCode™ Deploy, if you are not already logged in.

   **Result**

   The UrbanCode™ Deploy dashboard is displayed.
2. Click **Components**.

   **Result**

   A list of components that are available in UrbanCode™ Deploy is displayed.
3. Select the component from the list for which you want to create a process.
4. Click the **Processes** tab, and then click **Create Process**.

   **Result**

   The **Create Process** dialog is displayed.
5. Enter a name for the process in the **Name** field.
6. Select **Operational (No Version Needed)** from **Process Type** drop-down list.
7. Verify the **Default Working Directory** field.

   The **Default Working Directory** field defines the location that the agent uses to run the process. The default value is `${p:resource/work.dir}/${p:component.name}`.

   Where `${p:resource/work.dir}` is the default working directory for the agent and `${p:component.name}` is the name of the component.
8. Click **Save**.

   **Result**

   The process that you created is listed in the **Processes** tab and the **Design** tab for the process is displayed.

**Results**

You have created the process for the component in UrbanCode™ Deploy.

**What to do next**

You must configure the process in UrbanCode™ Deploy. See Configuring the process on page 444.

Related information

IBM UrbanCode Deploy Documentation

## Configuring the process

You must configure the process that you created for the component to organize the steps in the process, specify the properties of the steps, and connect them.

**Before you begin**

- You must be familiar with working with UrbanCode™ Deploy.
- You must have performed the following tasks:
  - Been granted access to UrbanCode™ Deploy.
  - Created a component in UrbanCode™ Deploy. See Creating a component in UrbanCode Deploy on page 443.
  - Created a process for the component in UrbanCode™ Deploy. See Creating a process in UrbanCode Deploy on page 443.
  - Generated an offline user token from HCL OneTest™ Server. See Generating an offline token on page 482.
  - Added the tests assets to the project on HCL OneTest™ Server.

**About this task**

When you open any process to configure, the process is displayed in the process editor. The process editor lists the plugins and steps. The required **Start** and **Finish** steps represent the beginning and the end of the process and steps are automatically placed on the design area.

1. Log in to UrbanCode™ Deploy, if you are not already logged in.

   **Result**

   The UrbanCode™ Deploy dashboard is displayed.
2. Click **Components**.

   **Result**

   A list of components that are available in UrbanCode™ Deploy is displayed.
3. Select the component from the list in which you created the process.
4. Click the **Processes** tab.

   **Result**

   A list of processes that are available for the component is displayed.
5. Select the process from the list that you want to configure.

   **Result**

   The **Design** tab for the process is displayed.
6. Click **HCL OneTest Studio**, and then **HCL OneTest Server** from the left menu.
7. Drag the **Run HCL OneTest Server test** step, and then drop it into the design area.

   **Note:** The selected test must be placed between **Start** and **Finish** steps.
8. Specify the properties for the selected test by performing the following steps:

   a. Click the **Edit** icon.

      **Result**

The **Edit Properties for Run HCL OneTest Server test** dialog is displayed.

b. Specify the properties for the selected test step by referring to the following tables:

The following table lists the required fields that you must provide to run the test from UrbanCode™ Deploy:

| Fields | Action |
|---|---|
| **Name** | Enter the name for the test step. |
| **HCL OneTest Server URL** | Enter the URL of HCL OneTest™ Server. |
| **Offline Token** | Enter the offline token that you generated from HCL OneTest™ Server. |
| **Team Space Name** | Enter the name of the team space that contains the project.<br>**Note:** The license for the team space must be configured and you must be a member of that team space. |
| **Project Name** | Enter the name of the project that are available in the team space. |
| **Repository Link** | Enter the URL of the Git repository that you added to your project in HCL OneTest™ Server. |
| **File Path** | Enter the path of the test assets that you want to run.<br>You can find the path of the test assets from the **Execution** page in HCL OneTest™ Server |

The following table lists the optional fields that you can provide to run the test from UrbanCode™ Deploy:

| Fields | Description |
|---|---|
| **Branch Name** | Use this field to enter the name of the branch in the Git repository where you stored test assets. |

| Fields | Description |
|---|---|
| **Custom Trust Store Password** | Use this field to enter the trust store password if you have modified the password while creating the custom trust store.<br>**Note:** If you have not modified the trust store password, you can retain the **Custom Trust Store Password** field blank. |
| **Custom Trust Store Path** | Use this field to enter the file path of your trust store followed by the file name if HCL OneTest™ Server uses an internal CA certificate and you have imported the certificate to a custom trust store. |
| **Datasets** | Use this field to enter the path to the dataset if you want to replace the values of the dataset during a test run.<br>You must ensure that both original and new datasets are in the same workspace and have the same column names. When you enter a value for the **Datasets** field, you must also include the path to the dataset. The path must be in the following format:<br>`/project_name/ds_path/original_ds.csv:/project_name/ds_path/new_ds.csv`<br>**Note:** You can override multiple datasets that are saved in a different project by adding multiple paths to the dataset separated by a semicolon. |
| **Environment** | Use this field to enter the environment details for the following types of test assets:<br>▪ APISUITE<br>▪ APITEST<br>▪ APISTUB |
| **Labels** | Use this field to add labels to test results when the test run is complete.<br>You can add multiple labels to a test result separated by a comma.<br>For example, *label1*, *label2*<br>After the test run is complete, then the values provided in the **Labels** field are displayed on the **Results** page of HCL OneTest™ Server. |

| Fields | Description |
|---|---|
| **Secrets Collection Name** | Use this field to enter the name of the Secret if you created secrets collections for your project and to use Secrets at test run time.<br>This field is mandatory only if you want to run the following types of test assets:<br>▪ APISUITE<br>▪ APITEST<br>▪ APISTUB |
| **Show Hidden Properties** | Select this checkbox if you want to use an HTTP Proxy to connect to HCL OneTest™ Server.<br>After you select the **Show Hidden Properties** checkbox, you can configure the HTTP Proxy by using the following fields:<br>▪ **HTTP Proxy Host**: Use this field to enter the hostname of the HTTP proxy that you want to connect to HCL OneTest™ Server.<br>▪ **HTTP Proxy Port**: Use this field to enter the port number where the HTTP proxy is used to listen to both HTTP and HTTPS traffic.<br>▪ **HTTP Proxy User name**: Use this field to provide the username for the HTTP proxy.<br>▪ **HTTP Proxy Password**: Use this field to enter a valid password for the user that you specified in the **HTTP Proxy User name** field.<br>**Note:** You can use the **Show Hidden Properties** field only when you want to run the following types of test assets:<br>▪ APISUITE<br>▪ APITEST<br>▪ APISTUB |
| **Start Date** | Use this field to enter the date and time in the following format for running tests at the scheduled date and time:<br>`yyyy/MM/dd/HH:mm`<br>When you enter a value in the **Start date** field, the status of the test displays as `Scheduled` on the **Overview** and **Progress** pages of HCL OneTest™ Server. |

| Fields | Description |
|---|---|
| **Variables** | Use this field to enter the name of the variable and its value if your test requires variables during the test run time.<br>You must enter the variables in the following format:<br>`name_of_the_variable=value_of_the_varibale`<br>You can add multiple variables to the test run separated by a semicolon.<br>For example, *varname1=value1;varname2=value2* |

**Note:** In addition to optional fields, you can also use the following fields from UrbanCode™ Deploy to configure your test run:

- **Working Directory**
- **Precondition**
- **Post Processing Script**
- **Use Impersonation**
- **Auth Token Restriction**

You can accept the default values or change the values based on your requirements. For more information about these fields, see the related links.

c. Click **OK** to save the properties for the test.

9. Click **Save** in the design area.

### Results

You have configured the process for the component in UrbanCode™ Deploy.

### What to do next

You must create a resource in UrbanCode™ Deploy. See Creating a resource in UrbanCode Deploy on page 450.

Related information

IBM UrbanCode Deploy Documentation

Process step preconditions

Post-processing scripts

User impersonation for process steps

Restricting authentication tokens

## Creating a resource in UrbanCode™ Deploy

You must create a resource to associate agents with components that you created in UrbanCode™ Deploy.

**Before you begin**

- You must be familiar with working with UrbanCode™ Deploy.
- You must have been granted access to UrbanCode™ Deploy.

1. Log in to UrbanCode™ Deploy, if you are not already logged in.

   **Result**

   The UrbanCode™ Deploy dashboard is displayed.
2. Click **Resources**, and then click **Create Top-Level Group**.

   **Result**

   The **Create Resource** dialog is displayed.
3. Enter a name for the resource in the **Name** field.
4. Click **Save**.

**Results**

You have created the resource in UrbanCode™ Deploy.

**What to do next**

You must configure the resource. See Configuring the resource on page 450.

Related information

IBM UrbanCode Deploy Documentation

## Configuring the resource

You must configure the resource to add an agent and associate the agent with the component.

**Before you begin**

- You must be familiar with working with UrbanCode™ Deploy.
- You must have performed the following tasks:
  - Been granted access to UrbanCode™ Deploy.
  - Created a component in UrbanCode™ Deploy. See Creating a component in UrbanCode Deploy on page 443.
  - Created a resource in UrbanCode™ Deploy. See Creating a resource in UrbanCode Deploy on page 450.

1. Log in to UrbanCode™ Deploy, if you are not already logged in.

   **Result**

   The UrbanCode™ Deploy dashboard is displayed.
2. Click **Resources**.

   **Result**

   A list of resources that are available in UrbanCode™ Deploy is displayed.
3. Perform the following steps to add an agent to the resource:

   a. Click the resource from the list for which you want to add an agent.

   b. Click the **Actions** icon from the last column, and then click **Add Agent**.

   c. Select the agent from the drop-down list.

      **Note:** The **Name** field is auto populated with the name of the agent.

   d. Click **Save**.

      **Result**

      The selected agent is added to the resource and you can view the status of the agent in the **Status** column.
4. Perform the following steps to add a component to the agent:

   a. Click the agent from the list for which you want to add a component.

   b. Click the **Actions** icon from the last column, and then click **Add Component**.

   c. Select the component from the drop-down list.

      **Note:** The **Name** field is auto populated with the name of the component.

   d. Click **Save**.

      **Result**

      The selected component is added to the agent.

**Results**

You have configured the resource in UrbanCode™ Deploy.

**What to do next**

You must create an application. See Creating an application in UrbanCode Deploy on page 452.

Related information

IBM UrbanCode Deploy Documentation

## Creating an application in UrbanCode™ Deploy

You must create an application to fetch all the components together that you want to deploy.

### Before you begin

- You must be familiar with working with UrbanCode™ Deploy.
- You must have been granted access to UrbanCode™ Deploy.

1. Log in to UrbanCode™ Deploy, if you are not already logged in.

   **Result**

   The UrbanCode™ Deploy dashboard is displayed.
2. Click **Applications**.
3. Click **Create Applications**, and then **New Applications**.
4. Enter a name for the application in the **Name** field.

   **Result**

   The **Environments** page for the application that you created is displayed.

### Results

You have created the application in UrbanCode™ Deploy.

### What to do next

You must configure the application. See Configuring the application on page 452.

Related information

IBM UrbanCode Deploy Documentation

## Configuring the application

You must configure the application to associate resources with environments and define processes to run test assets.

### Before you begin

- You must be familiar with working with UrbanCode™ Deploy.
- You must have performed the following tasks:
  - Been granted access to UrbanCode™ Deploy.
  - Created a component in UrbanCode™ Deploy. See Creating a component in UrbanCode Deploy on page 443.
  - Created a process for the component in UrbanCode™ Deploy. See Creating a process in UrbanCode Deploy on page 443.

- Configure the process for the component in UrbanCode™ Deploy. See Configuring the process on page 444.
- Created a resource in UrbanCode™ Deploy. See Creating a resource in UrbanCode Deploy on page 450.
- Configured the resource in UrbanCode™ Deploy. See Configuring the resource on page 450.
- Created an application in UrbanCode™ Deploy. See Creating an application in UrbanCode Deploy on page 452.

1. Log in to UrbanCode™ Deploy, if you are not already logged in.

   **Result**

   The UrbanCode™ Deploy dashboard is displayed.
2. Click **Applications**.

   **Result**

   A list of applications that are available in UrbanCode™ Deploy is displayed.
3. Click the application that you want to configure from the **Name** column.

   **Result**

   The **Environments** page for the selected application is displayed.
4. Perform the following steps to create an environment for the application that you selected:
   a. Click **Create Environment**.
   b. Enter a name for the environment in the **Name** field.
   c. Click **Save**.
5. Perform the following steps to configure resources to the environment:
   a. Click the environment that you created.
   b. Click **Add Base Resources**.

      **Result**

      A list of resources that are available in UrbanCode™ Deploy is displayed.
   c. Select the checkbox to add resources to the environment.
   d. Click **Save**.

      **Result**

      You can view the corresponding agent and the component that you added for the resource by using the **Expand** icon.
6. Perform the following steps to add the component to the application:
   a. Click **Applications**, and then select your application from the list.
   b. Click the **Components** tab, and then **Add Components**.
   c. Select the checkbox from the drop-down list to add components to the application.
   d. Click **Save**.

7. Perform the following steps to create a process for the application:

    a. Click the **Processes** tab, and then **Create Process**.

    b. Enter a name for the process in the **Name** field.

    c. Click **Save**.

       **Result**

       The **Design** tab for the process that you created is displayed.

8. Drag the component process listed under the **Component Process Steps** option from the left navigation pane and drop it into the design area.
9. Select the component process from the drop-down list in the **Operational (No Version Needed) Process** field.
10. Click **Save**.
11. Click the **Edit** icon, and then change the name of the properties.
12. Click **OK**, and then click **Save**.

**Results**

You have configured the application to run test assets from UrbanCode™ Deploy.

**What to do next**

You can run test assets from UrbanCode™ Deploy. See Running HCL OneTest Server tests on the UCD server on page 454.

Related information

IBM UrbanCode Deploy Documentation

## Running HCL OneTest™ Server tests on the UCD server

After you install the HCL OneTest™ Server UCD plugin on the UCD server, you can create a `process request` that contains the test for your application, and then run the test on the UCD server.

**Before you begin**

You must have completed the following tasks:

- Added the tests that you created in the desktop clients for your application to the project on HCL OneTest™ Server.
- Installed the latest version of the HCL OneTest™ Server UCD plugin. See Installing the plugin on page 441.
- Generated an offline user token from HCL OneTest™ Server. See Generating an offline token on page 482.

**About this task**

After you have installed the HCL OneTest™ Server UCD plugin on the UCD server, you can either use an existing component in your project or create a component. You can create a component process and select the **HCL**

**OneTest™ Server test** step to edit the *step properties* for the test you want to run. After selecting the agent, you can create an application. You can then create an application process for the application, and then submit the application process for a run.

1. Log in to the UrbanCode Deploy (UCD) server, if you are not already logged in.
2. Click **Components** from the UCD dashboard, and then click **Create Component** to create a component.

   **Note:** You can either use an existing component or create a component.

3. Create a component process in the component by performing the following steps:

   a. Open the component that you created.

   b. Click the **Processes** tab from the component dashboard, and then click **Create Process**.
   **Result**

   The **Create Process** dialog box is displayed.

   c. Enter the required values to create a component process and click **Save**.

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

   i. Enter the process name in the **Name** field.
   ii. Select **Operational (No Version Needed)** from the **Process Type** list.

   **Note:** The **Default Working Directory** field displays the folder path where the agent can download the artifacts and create temporary files.

   The process that you created is listed in the **Processes** list and the **Design** tab for the process is displayed.

   **Note:** The process opens in the process editor. The process editor lists the plugins and steps. The required **Start** and **Finish** steps represent the beginning and the end of the process and are automatically placed on the design area.

4. Select the process step you want to run by completing the following steps:

   a. Search for the **HCL OneTest™ Server test** process step from the left design pane.

   b. Select the **Run HCL OneTest™ Server test** process step and drag the test into the design area.
   **Result**

   The selected test is placed in between the **Start** and **Finish** steps.

5. Specify the properties for the selected test by performing the following steps:

a. Click the **Edit** icon .

**Result**

The **Edit Properties for Run HCL OneTest™ Server test** dialog box of the selected test is displayed.

b. Specify the properties for the selected test step by following the action in the table that follows.

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

| Field | Action required for a HCL OneTest™ Server test |
|---|---|
| Name | Enter the name of the step. |
| HCL OneTest™ Server URL | Enter the URL for HCL OneTest™ Server. |
| Offline Token | Enter the offline token that you generated from HCL OneTest™ Server. |
| File Path | Enter the file path of the HCL OneTest™ Server test that you want to run. |
| Branch Name | Select the branch where the test assets are stored. |
| Repository Link | Enter the repository path for the test to run. |
| Project Name | Enter the name of the project that contains the test to run. |
| RIT Environment | Enter the HCL OneTest™ API environment details for the test.<br>**Note:** This field is applicable only for an API test. |
| **Field** | **Optional action for a HCL OneTest™ Server test** |
| Working Directory | Specify an alternative path to the working directory for this step. [2] |
| Post Processing Script | Specify if you want to run any scripts after the completion of the test run. You can click **New** to add new scripts. [2] |
| Precondition | Specify any conditions that are to be completed before the test runs. You can edit the script by clicking the script displayed. [2] |
| **Use Impersonation** checkbox | Select this checkbox to run the test as a different user. [2] |
| Auth Token Restriction | Set the authentication token actions by applying token restrictions.<br>The **System Default** is selected by default. You can add a new token restriction or edit the one already added. [2] |

[2] You need not set this property for a step when you are running an HCL OneTest™ Server test.

| Field | Action required for a HCL OneTest™ Server test |
|---|---|
| Custom Trust Store Path | Enter the file path of your trust store followed by the file name in the **Custom Trust Store Path** field if the HCL OneTest™ Server uses an internal CA certificate and you have imported the certificate to a custom trust store. |
| Custom Trust Store Password | Enter the trust store password if you have modified the password while creating the custom trust store.<br>**Note:** If you have not modified the trust store password, you can keep the **Custom Trust Store Password** field blank. |

c. Click **OK** to save the properties for the test.

6. Click **Save** in the design area.
7. Click the **Resources** tab from the UCD dashboard and create a resource by clicking **Create Top-Level Group**.

   **Result**

   The created resource is displayed on the **Resource Tree** tab page.
8. Select the agent that runs the test by completing the following steps:

   **Important:** You must have already installed the agent on the UCD server that you want to use.

   a. Select the resource displayed on the **Resource Tree** tab page.

   b. Click the **Horizontal ellipsis** icon ••• for the selected resource.

   c. Click **Add Agent**.

   d. Select the agent to add to the resource, and then click **Save**.

   **Result**

   The selected agent is added to the resource in the **Resource Tree** pane and the status of the agent can also be viewed.

   **Important:** The agent must be **Online** for the test to run.
9. Add the component to the agent by performing the following steps:

   a. Click the **Horizontal ellipsis** icon ••• for the agent.

   b. Click **Add Component** on the list.

   c. Select the component to add to the resource, and then click **Save**.

**Result**

The selected component is added to the agent for the resource in the **Resource Tree** pane.

10. Create an application by completing the following steps:

    a. Click the **Applications** tab from the UCD dashboard.

    b. Click **Create Application**.

    c. Complete the details in the **Create Application** dialog box, and then click **Save**.

    **Result**

    The **Environments** tab page is displayed for the created application.

    d. Click **Create Environment** to create an environment for the application that you created.

    e. Complete the details in the **Create Environment** dialog box, and then click **Save**.

    **Result**

    The environment that you created is displayed.

    f. Click the environment to open, and then click **Add Base Resources**.

    g. Select the resource from the list in the **Add Resource to Environment** dialog box, and then click **Save** to add the resource to the environment.

    **Result**

    The resource added to the environment is displayed.

    **Note:** When you add a resource to an environment, the corresponding agent and the component are displayed for the resource.

    h. Click the application from the `breadcrumbs`.

    **Result**

    The **Environments** tab page is displayed for your application.

    i. Add the component to the application by performing the following steps:

        i. Click the **Components** tab.

        ii. Click **Add Component**.

        iii. Select the component from the list in the **Add a Component** dialog box, and then click **Save**.

    **Result**

    The selected component is displayed on the **Components** tab page.

    j. Create a process for the application by performing the following steps:

i. Click the **Processes** tab.
ii. Click **Create Process**.
iii. Complete the details in the **Create an Application Process** dialog box, and then click **Save**.

**Result**

The **Design** tab page for the application process that you created is displayed.

k. Select the component process from the left pane and drag it into the design area.

**Note:** You can click the **Edit** icon ✎ to add the properties, if required.

l. Click **Save** in the design area.

11. Select the application process to run the test by completing the following steps:

a. Click **Applications** from the UCD dashboard.

b. Click the application that you configured for a test run.

c. Click the **Request Process** icon ▶.

**Result**

The **Run Process on <environment name>** window is displayed.

d. Select the application process that contains the test from the **Process** list.

e. Click **Submit**.

**Result**

The UCD dashboard shows the progress of the application process request.

**Results**

You have used the HCL OneTest™ Server UCD plugin to integrate HCL OneTest™ Server with UCD and run the test from your project on the UCD server.

After the UCD process request runs successfully, you can view the status of the completed process request displayed as follows:

- `Success`: When the test run is successful
- `Failed`: When the test run is failed

**What to do next**

- You can view the details of the test run as a process from the UCD dashboard by performing the following action:
    - Expand the *step*. You can then expand the application process. You can then hover over the process, and then click the **Output Log** icon . The output log is displayed. You can verify the log details.
- You can also view the test reports and logs of the test that was run on the UCD server from the **Results** page on HCL OneTest™ Server. See Test results and reports overview on page 360.

## Integration with other applications

You can integrate certain applications with HCL® OneTest™ Data to generate the test data.

You can find instructions to integrate other applications with HCL® OneTest™ Data.

## Generating the test data by using Jenkins

When you perform a test during the continuous integration and continuous deployment process on Jenkins, you might want to generate the test data. You can generate random test data to test your application by integrating Jenkins with HCL® OneTest™ Data by using the HCL® OneTest™ Data Jenkins Plugin.

### Before you begin

You must have completed the following tasks:

- Installed the latest version of Jenkins.
- You must have an account in the Jenkins application.
- Downloaded the latest version of **HCL OneTest Data Jenkins Plugin** from the HCL License & Delivery portal.
- Installed **HCL OneTest Data Jenkins Plugin** in Jenkins.

> **Note:** You must restart the Jenkins application after installing the **HCL OneTest Data Jenkins Plugin**.

- Logged in to HCL OneTest™ Server.
- Created a project and a schema in HCL® OneTest™ Data.
- Established a connection between HCL® OneTest™ Data and a supported database. See
    - Establishing a JDBC connection with HCL OneTest Data on page 149
    - Establishing a MongoDB connection with HCL OneTest Data on page 160

### About this task

You can write the test data generated by integrating Jenkins with HCL® OneTest™ Data into a file or any supported database. After integration with Jenkins, **HCL OneTest Data Jenkins Plugin** supports the insertion of the generated test data in both JDBC supported database and MongoDB. To write the generated test data in the database, you must establish a connection between HCL® OneTest™ Data and the supported database.

1. Log in to the Jenkins application.
2. Create a Jenkins free-style software project.

    For more details about how to create a project in Jenkins, refer to the related link.

The project dashboard is displayed.

3. From the project dashboard, perform the following steps:

    a. Click the **Add build step** list under **Build** and select **Run an HCL OneTest Data Generation**.

    b. Set the properties for the **HCL OneTest Data Jenkins Plugin** for HCL OneTest Data by referring to the following table:

| Field | Action | Required/Optional |
|---|---|---|
| Name | Enter the name of the build. | Required |
| Use Custom Trust Store | Select the checkbox if the SSL certificate is available in the custom trust store.<br>**Note:** If you do not find details of the custom trust store in the application plugin, then the application navigates to the default trust store to access the certificate.<br>The default trust store is at the following location: `$JAVA_HOME/jre/lib/security/cacerts` | Optional |
| Custom Trust Store Path | Enter the path of the custom trust store where the certificate is stored. | Required, if **Use Custom Trust Store** is selected. |
| Use Custom Password for the Trust Store | Select the checkbox if the default password for the custom trust store is changed when you place the certificate in the custom trust store.<br>**Note:** The default password to access the custom trust store is `changeit`. | Optional |
| Custom Trust Store Password | Enter the password to access the custom trust store. | Required, if **Use Custom Password for the Trust Store** is selected. |
| Server URL | Enter the URL of HCL OneTest™ Server. | Required |

| Field | Action | Required/Optional |
|---|---|---|
| | The format for the URL is as follows: https://<fully-qualified-dns-name>/ | |
| Offline Token | Enter the offline token that is generated in HCL OneTest™ Server. | Required |
| Project | Select the name of the project from the list of your projects. | Required |
| Schema | Select the name of the schema from the list of schemas associated with the project you selected. | Required |
| Root Element | Specify the root path of the element for which you want to generate the test data.<br>For example, Root:NewType1 | Required |
| Number of Records | Enter the number of records you want to generate.<br>This field accepts only numbers. | Required |
| Numeric Seed Value | Enter the seed value that acts as an instance of random data when you generate the test data.<br>This field accepts both positive and negative numbers. | Optional |
| Output Data Storage | Select the data storage type. The data storage is a location where you want the generated test data to be written.<br>You can select `FILE`, `JDBC`, or `MONGODB` as a data storage type.<br>`FILE`: The generated test data is written into a file and you can download it in your local file system.<br>`JDBC`: The generated test data is written in the selected JDBC supported database.<br>`MONGODB`: The generated test data is written into MongoDB. | Required |

| Field | Action | Required/Optional |
|---|---|---|
| Connection Names | **Note:** This field is enabled only when you select `JDBC` as the data storage type.<br>Select the connection name from the populated list of connection names. | Required |
| Output Format | Select the output file format of the generated test data from the populated list of the output formats. The output file format is based on the schema you selected. | Required |
| Data File Location | Specify the location of the output file. If the specified location is invalid, by default, the output file is saved in the HCL® OneTest™ Data server.<br>**Notes:**<br>▪ You can find the output file in the HCL® OneTest™ Data pod at the following location:<br>`/opt/hcl/hip-rest/output/<accountId>/<userID>/<projectId>/<schemaId>/<genMapPath>`<br>▪ This field is not applicable if you select the data storage type as `JDBC`. | Optional |

c. Click **Save**.

4. From the Jenkins dashboard, select the project and click **Build Now**.

### Results

You have successfully generated the test data by using the **HCL OneTest Data Jenkins Plugin**.

**Note:** If the test data generation request fails, you can view the test data generation logs. See Viewing the test data generation logs on page 464.

### What to do next

After the build completes, you can perform the following tasks:

- If you selected `FILE` as a data storage type, the generated test data is downloaded in the local file system at the specified location.
- If you selected `JDBC` as the data storage type, then you can use the generated test data from the database.
- If you selected `MONGODB` as the data storage type, then you can use the generated test data from the database.

Related information

Building a software project

## Viewing the test data generation logs

After the Jenkins build generates the test data, you can view the details of the test data generation from the logs.

1. Click the build number from **Build History**.
2. Click **Console Output** to open the console for the project.

**Exemple**

The following sample shows the logs of the generated test data when you select `FILE` as the data storage type:

```
Started by user
Running as SYSTEM
Building in workspace C:\Program Files (x86)\Jenkins\workspace\HelloWorld

--------------------- START Build Step --------------------------
Master/Slave details
System Information : LP1-AP-51837142/10.115.94.185
Windows OS

Data Generation information:
Server URL: https://otd-build.nonprod.hclpnp.com/
Project: testing
Schema: schema
Data Location:
Seed Value: 10
Root Element: Root:NewType1

Include Header: true
No Of Records: 10
Data Storage: FILE
Output Format: Excel

Status: Starting the data generation..

Data Generation Response is
 {"code":200,"project_id":"4400","schema_id":"5ebd2915612cae00fe8e48dd","connectionURL":null,"
dbSchemaName":null,"message":"Data generation
 succeeded.","data_location":"5ebd2906612cae00fe8e48db_e584d146-db2e-4b5b-bb15-495b1a53a18a_GL
zXjhMqlX_1","timestamp":"2020-05-14T13:14:52.623Z","tableName":null}
```

```
Status: Test data generation completed successfully


Status: Downloading the Generated Test Data

Not able to save the file as the location was invalid/blank
File is available
 in /
opt/hcl/hip-rest/output/5ebbc065612cae002d82b141/5ebd2906612cae00fe8e48db/4400/5ebd2915612cae0
0fe8e48dd/e584d146-db2e-4b5b-bb15-495b1a53a18a/GLzXjhMqlX.xlsx location of OTD Server
--------------------- END Build Step ----------------------------
Finished: SUCCESS
```

The following sample shows the logs of the generated test data when you select `JDBC` as the data storage type:

```
Started by user1
Running as SYSTEM
Building in workspace C:\Program Files (x86)\Jenkins\workspace\HelloWorld6
--------------------- START Build Step --------------------------
Master/Slave details
System Information : LP1-AP-51837142/10.115.94.185
Windows OS

Test data generation information:
Server URL: https://otd-build.nonprod.hclpnp.com/
Project: otdproject
Schema: anonymous_schema
Data Location:
Seed Value: 10
Root Element: DB:Table:Row

Include Header: false
No Of Records: 10
Data Storage: JDBC
Connection Name: SampleJDBC
Output Format: Native

Status: Starting the test data generation..

Test data generation response:
 ["code":200,"project_id":"1050","schema_id":"5e830314e4332f0192e3487e","connectionURL":"jdbc:
mysql:\/\/10.134.198.10:8081","dbSchemaName":"sampleSchema","message":"Data generation
 succeeded.","data_location":null,"timestamp":"2020-03-31T09:06:48.005Z","table name":"tasks"]

Status: Test data generation completed successfully
Data is written into the database
Connection URL: jdbc:mysql://10.134.198.10:8081
Database schema name: sampleSchema
Table name: tasks
```

The following sample shows the logs of the generated test data when you select `MONGODB` as the data storage type:

```
Started by user
Running as SYSTEM
Building in workspace C:\Program Files (x86)\Jenkins\workspace\HCLOneTestData Integration

--------------------- START Build Step --------------------------
Master/Slave details
System Information : LP1-AP-51837142/10.115.94.185
Windows OS

Data Generation information:
Server URL: https://otd-fvt1.nonprod.hclpnp.com/
Project: testproject
Schema: EmployeeDetails
Data Location:
Seed Value: 1
Root Element: Root:Employee

Include Header: false
No Of Records: 1
Data Storage: MONGODB
Connection Name: TestMongo
Output Format: Json

Status: Starting the data generation..

Data Generation Response is
 {"code":200,"databaseName":"oneTestDataDB","project_id":"2850","schema_id":"5efedcb33d358500a
1662993","connectionURL":"{my-ots}-mongodb:27017","dbSchemaName":null,"message":"Data
 generation
 succeeded.","data_location":null,"timestamp":"2020-07-03T07:29:06.055Z","tableName":null,"col
lectionName":"employee"}

Data is written into Mongo Database
Connection URL: {my-ots}-mongodb:27017
Database Name: oneTestDataDB
DB Collection Name: employee

Status: Test data generation completed successfully
--------------------- END Build Step ----------------------------
Finished: SUCCESS
```

## Generating the test data by using UrbanCode™ Deploy

When you perform a test while you deploy an application on UrbanCode™ Deploy, you might want to generate test data. You can generate a random test data to test your application by integrating HCL® OneTest™ Data UCD Plugin with HCL® OneTest™ Data.

### Before you begin

You must have completed the following tasks:

- Installed the latest version of the UrbanCode™ Deploy server and agent on the target system.
- Configured the agent and verified that the agent is running on the target system. For information about how to configure the agent, refer to Configure the agent and target system.
- Downloaded the latest version of HCL® OneTest™ Data UCD Plugin from the HCL License & Delivery portal.
- Installed HCL® OneTest™ Data UCD Plugin on the UrbanCode™ Deploy server. See Installing the HCL OneTest Data UCD Plugin on page 470.
- Created an account on the UrbanCode™ Deploy server.
- Logged in to HCL OneTest™ Server.
- Created a project and a schema in HCL® OneTest™ Data.
- Established a connection between HCL® OneTest™ Data and a supported database. See
    - Establishing a JDBC connection with HCL® OneTest™ Data on page 149
    - Establishing a MongoDB connection with HCL OneTest Data on page 160

## About this task

You can write the test data generated by integrating HCL® OneTest™ Data UCD Plugin with HCL® OneTest™ Data into a file or any supported database. After integration with HCL® OneTest™ Data UCD Plugin, HCL® OneTest™ Data supports the insertion of the generated test data in both JDBC supported database and MongoDB. To write the generated test data in the database, you must establish a connection between HCL® OneTest™ Data and the supported database.

## Procedure

1. Log in to the HCL UrbanCode Deploy server.
2. Create a component.
3. Create a component process, and then set the properties for the component by referring to the following table:

| Field | Action | Required/Optional |
|---|---|---|
| Name | Enter a name for the HCL UrbanCode Deploy application process. | Required |
| Server URL | Enter the URL of HCL OneTest™ Server.<br>The format for the URL is as follows: https://<fully-qualified-dns-name>/ | Required |
| Offline Token | Enter the offline token that is generated in HCL OneTest™ Server. | Required |
| Project | Enter the name of your project. | Required |
| Schema | Enter the name of the schema associated with the project you selected. | Required |
| Root Element | Specify the root path of the element for which you want to generate the test data. | Required |

| Field | Action | Required/Optional |
|---|---|---|
| | For example, Root:NewType1 | |
| Number of Records | Enter the number of records you want to generate. | Required |
| Numeric Seed Value | Enter the seed value that acts as an instance of random data when you generate the test data. | Optional |
| Data Storage | Select the data storage type. The data storage is a location where you want the generated test data to be written.<br>You can select `FILE`, `JDBC`, or `MONGODB` as a data storage type.<br>`FILE`: The generated test data is written into a file and you can download it in your local file system.<br>`JDBC`: The generated test data is written in the selected JDBC supported database.<br>`MONGODB`: The generated test data is written into MongoDB. | Required |
| Connection Name | **Note:** This field is applicable only when you select `JDBC` or `MONGODB` as the data storage type.<br>Enter the name of the `JDBC` or `MONGODB` connection. | Required |
| Output Format | Specify the file format of the generated test data. | Required |
| Data File Location | Specify the location for the output file. If the specified location is invalid, by default, the output file is saved in the HCL® OneTest™ Data server.<br>**Notes:** | Optional |

| Field | Action | Required/Optional |
|---|---|---|
| | ◦ You can find the output file in the HCL® OneTest™ Data pod at the following location:<br>`/opt/hcl/hip-rest/output/<accountId>/<userID>/<projectId>/<schemaId>/<genMapPath>`<br>◦ This field is not applicable if you select the data storage type as `JDBC` or `MONGODB`. | |

**Note:** You can ignore the following property fields while you set up the integration of HCL UrbanCode Deploy with HCL® OneTest™ Data:

- Working Directory
- Post Processing Script
- Precondition
- Use Impersonation

4. Create a resource and select the agent.
5. Add the component that you created to the agent.
6. Create an application.
7. Create an environment from the **Applications** dashboard.
8. Add the resource and the component that you created to the environment.
9. Create a process for the application by clicking the **Processes** tab from the **Applications** dashboard.

   The page of the application process is displayed.

10. Click the component process that you created in step 3 from the **Component Process Steps** on the left navigation pane and drag it into the design area.
11. Click **Save**.
12. Go to the **Applications** dashboard, and then click **Request Process** for the environment of the application process that you want to execute.

    The **Run Process on environment name** dialog box is displayed.

13. Select the application process that you want to execute and click **Submit**.

    The HCL UrbanCode Deploy dashboard shows the progress of the application process request to generate the test data.

### Result

You have successfully generated the test data by using the HCL® OneTest™ Data UCD Plugin for HCL® OneTest™ Data.

You can view the completed request process with the status displayed as **Success** or **Failed**.

**Note:**

If the test data generation request fails, you can view the logs of the process. See Viewing the UrbanCode Deploy logs on page 470

### What to do next

After the successful completion of process, you can perform the following tasks:

- If you selected `FILE` as a data storage type, the generated test data is downloaded in the local file system at the specified location.
- If you selected `JDBC` as the data storage type, then you can use the generated test data from the database.
- If you selected `MONGODB` as the data storage type, then you can use the generated test data from the database.

## Installing the HCL® OneTest™ Data UCD Plugin

When you want to generate the test data by using the HCL® OneTest™ Data UCD Plugin for HCL® OneTest™ Data, you must install the HCL® OneTest™ Data UCD Plugin on the UrbanCode™ Deploy server.

1. From **Settings**, click **Automation Plugins**.
2. Click **Load Plugin**.
3. Enter the path of the compressed plug-in file, and then click **Submit**.

### Results

The plug-in is listed on the **Automation Plugins** pane.

### What to do next

After you successfully installed the HCL® OneTest™ Data UCD Plugin on the server, you must configure the HCL® OneTest™ Data UCD Plugin and generate the test data. See Generating the test data by using UrbanCode Deploy on page 466.

## Viewing the UrbanCode™ Deploy logs

After completion of the generation of test data, you can view the details of the process in the UrbanCode™ Deploy console log.

You can click the **Output Log** icon from the console log of the UrbanCode Deploy dashboard to view the output log.

The following sample shows the logs of the generated test data when you select `FILE` as the data storage type:

```
================================================================================
plugin: HCL OneTest Data UCD Plugin, id: com.urbancode.air.plugin.otd, version: 1
plugin command: 'cmd' '/C' '"D:\UCD_Agent\opt\groovy-1.8.8\bin\groovy.bat -cp
 "D:
\UCD_Agent\var\plugins\com.urbancode.air.plugin.otd_1_200b8a9ae7959125d32be1cde7aa20472d2b47c4
de3754dbd7627bd6778d33d0\classes;D:\UCD_Agent\var\plugins\com.urbancode.air.plugin.otd_1_200b8
a9ae7959125d32be1cde7aa20472d2b47c4de3754dbd7627bd6778d33d0\lib\commons-codec-1.10.jar;D:\UCD_
Agent\var\plugins\com.urbancode.air.plugin.otd_1_200b8a9ae7959125d32be1cde7aa20472d2b47c4de375
4dbd7627bd6778d33d0\lib\commons-logging-1.2.jar;D:\UCD_Agent\var\plugins\com.urbancode.air.plu
gin.otd_1_200b8a9ae7959125d32be1cde7aa20472d2b47c4de3754dbd7627bd6778d33d0\lib\groovy-all-1.8.
4.jar;D:\UCD_Agent\var\plugins\com.urbancode.air.plugin.otd_1_200b8a9ae7959125d32be1cde7aa2047
2d2b47c4de3754dbd7627bd6778d33d0\lib\groovy-plugin-utils-1.2.jar;D:\UCD_Agent\var\plugins\com.
urbancode.air.plugin.otd_1_200b8a9ae7959125d32be1cde7aa20472d2b47c4de3754dbd7627bd6778d33d0\
lib\gson-2.8.6.jar;D:\UCD_Agent\var\plugins\com.urbancode.air.plugin.otd_1_200b8a9ae7959125d32
be1cde7aa20472d2b47c4de3754dbd7627bd6778d33d0\lib\hamcrest-core-1.1.jar;D:\UCD_Agent\var\plug
ins\com.urbancode.air.plugin.otd_1_200b8a9ae7959125d32be1cde7aa20472d2b47c4de3754dbd7627bd6778
d33d0\lib\httpclient-4.5.6.jar;D:\UCD_Agent\var\plugins\com.urbancode.air.plugin.otd_1_200b8a9
ae7959125d32be1cde7aa20472d2b47c4de3754dbd7627bd6778d33d0\lib\httpcore-4.4.10.jar;D:\UCD_Ag
ent\var\plugins\com.urbancode.air.plugin.otd_1_200b8a9ae7959125d32be1cde7aa20472d2b47c4de3754d
bd7627bd6778d33d0\lib\json-simple-1.1.1.jar;D:\UCD_Agent\var\plugins\com.urbancode.air.plugin.
otd_1_200b8a9ae7959125d32be1cde7aa20472d2b47c4de3754dbd7627bd6778d33d0\lib\jsoup-1.11.3.jar
;D:\UCD_Agent\var\plugins\com.urbancode.air.plugin.otd_1_200b8a9ae7959125d32be1cde7aa20472d2b4
7c4de3754dbd7627bd6778d33d0\lib\junit-4.10.jar"
 D:
\UCD_Agent\var\plugins\com.urbancode.air.plugin.otd_1_200b8a9ae7959125d32be1cde7aa20472d2b47c4
de3754dbd7627bd6778d33d0\OneTestDataGenerationUCD.groovy
 D:\UCD_Agent\var\temp\logs-63ec6300-a796-49c2-b3ad-794d0e28b1be\input.props
 D:\UCD_Agent\var\temp\logs-63ec6300-a796-49c2-b3ad-794d0e28b1be\output.props"'
working directory: D:\UCD_Agent\var\work\UCDComponent
properties:

 PLUGIN_INPUT_PROPS=D:\UCD_Agent\var\temp\logs-63ec6300-a796-49c2-b3ad-794d0e28b1be\input.pr
ops

 PLUGIN_OUTPUT_PROPS=D:\UCD_Agent\var\temp\logs-63ec6300-a796-49c2-b3ad-794d0e28b1be\output.pr
ops
  connectionName=
  dataFileLocation=C:\Users\user\Downloads
  dataStorage=FILE
  offlineToken=****
  outputFormat=CSV
  project=testing
  records=10
  rootElement=Root:NewType1
  schema=schema
  seedValue=10
  serverUrl=https://otd-build.nonprod.hclpnp.com/
environment:
  AGENT_HOME=D:\UCD_Agent
  AH_AUTH_TOKEN=****
  AH_WEB_URL=https://LP1-AP-51837142.PROD.HCLPNP.COM:8443
  AUTH_TOKEN=****
  DS_AUTH_TOKEN=****
  DS_SYSTEM_ENCODING=Cp1252
  JAVA_OPTS=-Dfile.encoding=Cp1252 -Dconsole.encoding=Cp1252

 PLUGIN_HOME=D:\UCD_Agent\var\plugins\com.urbancode.air.plugin.otd_1_200b8a9ae7959125d32be1cde
7aa20472d2b47c4de3754dbd7627bd6778d33d0
  UD_DIALOGUE_ID=63ec6300-a796-49c2-b3ad-794d0e28b1be
  WE_ACTIVITY_ID=172134c9-a5ec-22cd-3c8f-4d3d9789c96d
================================================================================
Data Generation information:
Server URL: https://otd-build.nonprod.hclpnp.com/
Project: testing
schema: schema
Root Element: Root:NewType1
Output Format: CSV
Data Location: C:\Users\user\Downloads
Seed Value: 10
Data Storage: FILE
Connection Name:
Include Header: true
No Of Records 10
Validating: server URL , offlineToken
Successfully validated server URL and offlineToken
Status: Test Data Generation Started...
Data Generation Response is [code:200, project_id:4400, schema_id:5ebd2915612cae00fe8e48dd,
 connectionURL:null, dbSchemaName:null, message:Data generation succeeded.,
 data_location:5ebd2906612cae00fe8e48db_4876c2b6-1424-442f-a1f1-5c3cad77576e_biWSum9yW8_1,
 timestamp:2020-05-14T13:08:15.840Z, tableName:null]
Status: Test data generation completed successfully
Status: Downloading the Generated Test Data
File saved to
 C:
\Users\user\Downloads\5ebd2906612cae00fe8e48db_4876c2b6-1424-442f-a1f1-5c3cad77576e_biWSum9yW8
_1.csv location
Status: Completed the download of Generated Test Data
```

The following sample shows the logs of the generated test data when you select `MONGODB` as the data storage type:

```
================================================================================
plugin: HCL OneTest Data UCD Plugin, id: com.urbancode.air.plugin.otd, version: 1
plugin command: 'cmd' '/C' '"D:\UCD_Agent\opt\groovy-1.8.8\bin\groovy.bat -cp
 "D:
\UCD_Agent\var\plugins\com.urbancode.air.plugin.otd_1_73a2c474971db8d52bfdb80c0335e047f8f6cc65
6e1784133370b8d253310f6a\classes;D:\UCD_Agent\var\plugins\com.urbancode.air.plugin.otd_1_73a2c
474971db8d52bfdb80c0335e047f8f6cc656e1784133370b8d253310f6a\lib\commons-codec-1.10.jar;D:\UCD_
Agent\var\plugins\com.urbancode.air.plugin.otd_1_73a2c474971db8d52bfdb80c0335e047f8f6cc656e178
4133370b8d253310f6a\lib\commons-logging-1.2.jar;D:\UCD_Agent\var\plugins\com.urbancode.air.plu
gin.otd_1_73a2c474971db8d52bfdb80c0335e047f8f6cc656e1784133370b8d253310f6a\lib\groovy-all-1.8.
4.jar;D:\UCD_Agent\var\plugins\com.urbancode.air.plugin.otd_1_73a2c474971db8d52bfdb80c0335e047
f8f6cc656e1784133370b8d253310f6a\lib\groovy-plugin-utils-1.2.jar;D:\UCD_Agent\var\plugins\com.
urbancode.air.plugin.otd_1_73a2c474971db8d52bfdb80c0335e047f8f6cc656e1784133370b8d253310f6a\
lib\gson-2.8.6.jar;D:\UCD_Agent\var\plugins\com.urbancode.air.plugin.otd_1_73a2c474971db8d52bf
```

```
db80c0335e047f8f6cc656e1784133370b8d253310f6a\lib\hamcrest-core-1.1.jar;D:\UCD_Agent\var\plug
ins\com.urbancode.air.plugin.otd_1_73a2c474971db8d52bfdb80c0335e047f8f6cc656e1784133370b8d2533
10f6a\lib\httpclient-4.5.6.jar;D:\UCD_Agent\var\plugins\com.urbancode.air.plugin.otd_1_73a2c47
4971db8d52bfdb80c0335e047f8f6cc656e1784133370b8d253310f6a\lib\httpcore-4.4.10.jar;D:\UCD_Ag
ent\var\plugins\com.urbancode.air.plugin.otd_1_73a2c474971db8d52bfdb80c0335e047f8f6cc656e17841
33370b8d253310f6a\lib\json-simple-1.1.1.jar;D:\UCD_Agent\var\plugins\com.urbancode.air.plugin.
otd_1_73a2c474971db8d52bfdb80c0335e047f8f6cc656e1784133370b8d253310f6a\lib\jsoup-1.11.3.jar
;D:\UCD_Agent\var\plugins\com.urbancode.air.plugin.otd_1_73a2c474971db8d52bfdb80c0335e047f8f6c
c656e1784133370b8d253310f6a\lib\junit-4.10.jar"
 D:
\UCD_Agent\var\plugins\com.urbancode.air.plugin.otd_1_73a2c474971db8d52bfdb80c0335e047f8f6cc65
6e1784133370b8d253310f6a\OneTestDataGenerationUCD.groovy
 D:\UCD_Agent\var\temp\logs-48a47aee-3889-4a3b-ba91-298cbc33a99e\input.props
 D:\UCD_Agent\var\temp\logs-48a47aee-3889-4a3b-ba91-298cbc33a99e\output.props"'
working directory: D:\UCD_Agent\var\work\OTDComponent
properties:

 PLUGIN_INPUT_PROPS=D:\UCD_Agent\var\temp\logs-48a47aee-3889-4a3b-ba91-298cbc33a99e\input.pr
ops

 PLUGIN_OUTPUT_PROPS=D:\UCD_Agent\var\temp\logs-48a47aee-3889-4a3b-ba91-298cbc33a99e\output.pr
ops
  connectionName=TestMongo
  dataFileLocation=
  dataStorage=MONGODB
  offlineToken=****
  outputFormat=JSON
  project=testproject
  records=1
  rootElement=Root:Employee
  schema=EmployeeDetails
  seedValue=1
  serverUrl=https://otd-fvt1.nonprod.hclpnp.com/
environment:
  AGENT_HOME=D:\UCD_Agent
  AH_AUTH_TOKEN=****
  AH_WEB_URL=https://LP1-AP-51837142.PROD.HCLPNP.COM:8443
  AUTH_TOKEN=****
  DS_AUTH_TOKEN=****
  DS_SYSTEM_ENCODING=Cp1252
  JAVA_OPTS=-Dfile.encoding=Cp1252 -Dconsole.encoding=Cp1252

 PLUGIN_HOME=D:\UCD_Agent\var\plugins\com.urbancode.air.plugin.otd_1_73a2c474971db8d52bfdb80c0
335e047f8f6cc656e1784133370b8d253310f6a
  UD_DIALOGUE_ID=48a47aee-3889-4a3b-ba91-298cbc33a99e
  WE_ACTIVITY_ID=1731397b-b82e-1d8a-dc50-153fd3010ce0
================================================================================
Data Generation information:
Server URL: https://otd-fvt1.nonprod.hclpnp.com/
Project: testproject
schema: EmployeeDetails
Root Element: Root:Employee
```

```
Output Format: JSON
Data Location:
Seed Value: 1
Data Storage: MONGODB
Connection Name: TestMongo
Include Header: false
No Of Records 1
Validating: server URL , offlineToken
Successfully validated server URL and offlineToken
Status: Test Data Generation Started...
Data Generation Response is [code:200, databaseName:oneTestDataDB, project_id:2850,
 schema_id:5efedcb33d358500a1662993, connectionURL:{my-ots}-mongodb:27017, dbSchemaName:null,
 message:Data generation succeeded., data_location:null, timestamp:2020-07-03T07:33:11.497Z,
 tableName:null, collectionName:employee]
Status: Test data generation completed successfully
Data is written into Mongo database
Connection URL: {my-ots}-mongodb:27017
Database Name: oneTestDataDB
Collection name: employee
```

## Generating the test data by using HCL Launch

When you perform a test while you deploy an application on HCL Launch, you might want to generate test data. You can generate a random test data to test your application by integrating HCL OneTest Data Launch Plugin with HCL® OneTest™ Data.

### Before you begin

You must have completed the following tasks:

- Installed the latest version of the HCL Launch server and agent on the target system.
- Configured the agent and verified that the agent is running on the target system. For information about how to configure the agent, refer to Configure the agent and target system.
- Downloaded the latest version of HCL OneTest Data Launch Plugin from the HCL License & Delivery portal.
- Installed HCL OneTest Data Launch Plugin on the HCL Launch server. See Installing the HCL OneTest Data Launch Plugin on page 478.
- Created an account on the HCL Launch server.
- Logged in to HCL OneTest™ Server.
- Created a project and a schema in HCL® OneTest™ Data.
- Established a connection between HCL® OneTest™ Data and a supported database. See
    - Establishing a JDBC connection with HCL OneTest Data on page 149
    - Establishing a MongoDB connection with HCL OneTest Data on page 160

### About this task

You can write the test data generated by integrating HCL OneTest Data Launch Plugin with HCL® OneTest™ Data into a file or any supported database. After integration with HCL OneTest Data Launch Plugin, HCL® OneTest™ Data supports the insertion of the generated test data in both JDBC supported database and MongoDB. To write the generated test data in the database, you must establish a connection between HCL® OneTest™ Data and the supported database.

## Procedure

1. Log in to the HCL Launch server.
2. Create a component.
3. Create a component process, and then set the properties for the component by referring to the following table:

| Field | Action | Required/Optional |
|---|---|---|
| Name | Enter a name for the HCL Launch application process. | Required |
| Server URL | Enter the URL of HCL OneTest™ Server.<br>The format for the URL is as follows: https://<fully-qualified-dns-name>/ | Required |
| Custom Trust Store Path | Enter the path of the custom trust store where the certificate is stored.<br>**Note:** If you do not find details of the custom trust store in the application plugin, then the application navigates to the default trust store to access the certificate.<br>The default trust store is at the following location: `$JAVA_HOME/jre/lib/security/cacerts` | Optional |
| Custom Trust Store Password | Enter the password to access the custom trust store. | Optional |
| Offline Token | Enter the offline token that is generated in HCL OneTest™ Server. | Required |
| Project | Enter the name of your project. | Required |
| Schema | Enter the name of the schema associated with the project you selected. | Required |
| Root Element | Specify the root path of the element for which you want to generate the test data.<br>For example, Root:NewType1 | Required |
| Number of Records | Enter the number of records you want to generate. | Required |

| Field | Action | Required/Optional |
|---|---|---|
| Numeric Seed Value | Enter the seed value that acts as an instance of random data when you generate the test data. | Optional |
| Data Storage | Select the data storage type. The data storage is a location where you want the generated test data to be written.<br>You can select `FILE`, `JDBC`, or `MONGODB` as a data storage type.<br>`FILE`: The generated test data is written into a file and you can download it in your local file system.<br>`JDBC`: The generated test data is written in the selected JDBC supported database.<br>`MONGODB`: The generated test data is written into MongoDB. | Required |
| Connection Name | **Note:** This field is applicable only when you select `JDBC` or `MONGODB` as the data storage type.<br>Enter the name of the `JDBC` or `MONGODB` connection. | Required |
| Output Format | Specify the file format of the generated test data. | Required |
| Data File Location | Specify the location for the output file. If the specified location is invalid, by default, the output file is saved in the HCL® OneTest™ Data server.<br>**Notes:**<br>◦ You can find the output file in the HCL® OneTest™ Data pod at the following location:<br>*`/opt/hcl/hip-rest/output/<accountId>/<userID>/<projectId>/<schemaId>/<genMapPath>`*<br>◦ This field is not applicable if you select the data storage type as `JDBC` or `MongoDB`. | Optional |

**Note:** You can ignore the following property fields while you set up the integration of HCL Launch with HCL® OneTest™ Data:

- Working Directory
- Post Processing Script
- Precondition
- Use Impersonation

4. Create a resource and select the agent.
5. Add the component that you created to the agent.
6. Create an application.
7. Create an environment from the **Applications** dashboard.
8. Add the resource and the component that you created to the environment.
9. Create a process for the application by clicking the **Processes** tab from the **Applications** dashboard.

   The page of the application process is displayed.
10. Click the component process that you created in step 3 from the **Component Process Steps** on the left navigation pane and drag it into the design area.
11. Click **Save**.
12. Go to the **Applications** dashboard, and then click **Request Process** for the environment of the application process that you want to execute.
13. Select the name of the application, environment, and process from the drop-down list, and then click **Next**.
14. Verify that the default settings of the component versions are retained, and then click **Next**.
15. Toggle the **Run Now** switch to the on position to set the **Schedule Deployment**, and then click **Next**.
16. Check the deployment details, and then click **Submit Deployment**.

## Result

You have successfully generated the test data by using the HCL OneTest Data Launch Plugin for HCL® OneTest™ Data.

You can view the completed request process with the status displayed as **Success** or **Failed**.

**Note:**

If the test data generation request fails, you can view the logs of the process. See Viewing the HCL Launch logs on page 478.

## What to do next

After the successful completion of the process, you can perform the following tasks:

- If you selected `FILE` as a data storage type, then you can download the generated test data in the local file system at the specified location.
- If you selected `JDBC` as the data storage type, then you can use the generated test data from the database.
- If you selected `MONGODB` as the data storage type, then you can use the generated test data from the database.

## Installing the HCL OneTest Data Launch Plugin

When you want to generate the test data by using the HCL® OneTest™ Data for HCL® OneTest™ Data, you must install the HCL® OneTest™ Data on the HCL Launch server.

1. From **Settings**, click **Automation Plugins**.
2. Click **Load Plugin**.
3. Enter the path of the compressed plug-in file, and then click **Submit**.

**Results**

The plug-in is listed on the **Automation Plugins** pane.

**What to do next**

After you successfully installed the HCL® OneTest™ Data on the server, you must configure the HCL® OneTest™ Data and generate the test data. See Generating the test data by using HCL Launch on page 474.

## Viewing the HCL Launch logs

After completion of the generation of test data, you can view the details of the process in the HCL Launch console log.

You can click the **Output Log** icon from the console log of the HCL Launch dashboard to view the output log.

The following sample shows the logs of the generated test data when you select `FILE` as the data storage type:

```
================================================================================
plugin: HCL OneTest Data Launch Plugin, id: com.hcllaunch.air.plugin.otd, version: 2
plugin command: 'cmd' '/C' '""D:\HCL Launch\LaunchAgent\opt\groovy-2.4.15\bin\groovy.bat" -cp
 "D:\HCL
 Launch\LaunchAgent\var\plugins\com.hcllaunch.air.plugin.otd_2_ebae55c6a5da97650f3890760d07778
5e4fa877523edeadf3cfda916b293b4a1\classes;D:\HCL
 Launch\LaunchAgent\var\plugins\com.hcllaunch.air.plugin.otd_2_ebae55c6a5da97650f3890760d07778
5e4fa877523edeadf3cfda916b293b4a1\lib\commons-codec-1.10.jar;D:\HCL
 Launch\LaunchAgent\var\plugins\com.hcllaunch.air.plugin.otd_2_ebae55c6a5da97650f3890760d07778
5e4fa877523edeadf3cfda916b293b4a1\lib\commons-logging-1.2.jar;D:\HCL
 Launch\LaunchAgent\var\plugins\com.hcllaunch.air.plugin.otd_2_ebae55c6a5da97650f3890760d07778
5e4fa877523edeadf3cfda916b293b4a1\lib\groovy-all-1.8.4.jar;D:\HCL
 Launch\LaunchAgent\var\plugins\com.hcllaunch.air.plugin.otd_2_ebae55c6a5da97650f3890760d07778
5e4fa877523edeadf3cfda916b293b4a1\lib\groovy-plugin-utils-1.2.jar;D:\HCL
 Launch\LaunchAgent\var\plugins\com.hcllaunch.air.plugin.otd_2_ebae55c6a5da97650f3890760d07778
5e4fa877523edeadf3cfda916b293b4a1\lib\gson-2.8.6.jar;D:\HCL
 Launch\LaunchAgent\var\plugins\com.hcllaunch.air.plugin.otd_2_ebae55c6a5da97650f3890760d07778
5e4fa877523edeadf3cfda916b293b4a1\lib\hamcrest-core-1.1.jar;D:\HCL
 Launch\LaunchAgent\var\plugins\com.hcllaunch.air.plugin.otd_2_ebae55c6a5da97650f3890760d07778
5e4fa877523edeadf3cfda916b293b4a1\lib\httpclient-4.5.6.jar;D:\HCL
 Launch\LaunchAgent\var\plugins\com.hcllaunch.air.plugin.otd_2_ebae55c6a5da97650f3890760d07778
5e4fa877523edeadf3cfda916b293b4a1\lib\httpcore-4.4.10.jar;D:\HCL
 Launch\LaunchAgent\var\plugins\com.hcllaunch.air.plugin.otd_2_ebae55c6a5da97650f3890760d07778
5e4fa877523edeadf3cfda916b293b4a1\lib\json-simple-1.1.1.jar;D:\HCL
 Launch\LaunchAgent\var\plugins\com.hcllaunch.air.plugin.otd_2_ebae55c6a5da97650f3890760d07778
```

```
5e4fa877523edeadf3cfda916b293b4a1\lib\jsoup-1.11.3.jar;D:\HCL
 Launch\LaunchAgent\var\plugins\com.hcllaunch.air.plugin.otd_2_ebae55c6a5da97650f3890760d07778
5e4fa877523edeadf3cfda916b293b4a1\lib\junit-4.10.jar" "D:\HCL
 Launch\LaunchAgent\var\plugins\com.hcllaunch.air.plugin.otd_2_ebae55c6a5da97650f3890760d07778
5e4fa877523edeadf3cfda916b293b4a1\OneTestDataGenerationHCLLaunch.groovy" "D:\HCL
 Launch\LaunchAgent\var\temp\logs-89c29385-395a-4c98-b695-de7eb9dbdee5\input.props" "D:\HCL
 Launch\LaunchAgent\var\temp\logs-89c29385-395a-4c98-b695-de7eb9dbdee5\output.props""'
working directory: D:\HCL Launch\LaunchAgent\var\work\OTDComponent1
properties:
  PLUGIN_INPUT_PROPS=D:\HCL
 Launch\LaunchAgent\var\temp\logs-89c29385-395a-4c98-b695-de7eb9dbdee5\input.props
  PLUGIN_OUTPUT_PROPS=D:\HCL
 Launch\LaunchAgent\var\temp\logs-89c29385-395a-4c98-b695-de7eb9dbdee5\output.props
  connectionName=
  dataFileLocation=C:\Users\user1\Downloads
  dataStorage=FILE
  offlineToken=****
  outputFormat=CSV
  project=otdproject
  records=10
  rootElement=Root:NewType1
  schema=otdschema
  seedValue=10
  serverUrl=https://otd-fvt2.nonprod.hclpnp.com/
environment:
  AGENT_HOME=D:\HCL Launch\LaunchAgent
  AH_AUTH_TOKEN=****
  AH_WEB_URL=https://LP1-AP-51837142.PROD.HCLPNP.COM:8443
  AUTH_TOKEN=****
  DS_AUTH_TOKEN=****
  DS_SYSTEM_ENCODING=Cp1252
  JAVA_OPTS=-Dfile.encoding=Cp1252 -Dconsole.encoding=Cp1252
  PLUGIN_HOME=D:\HCL
 Launch\LaunchAgent\var\plugins\com.hcllaunch.air.plugin.otd_2_ebae55c6a5da97650f3890760d07778
5e4fa877523edeadf3cfda916b293b4a1
  UD_DIALOGUE_ID=89c29385-395a-4c98-b695-de7eb9dbdee5
  WE_ACTIVITY_ID=17443a96-551a-d05a-6041-0d9b77073106
================================================================================
Data Generation information:
Server URL: https://otd-fvt2.nonprod.hclpnp.com/
Project: otdproject
schema: otdschema
Root Element: Root:NewType1
Output Format: CSV
Data Location: C:\Users\user1\Downloads
Seed Value: 10
Data Storage: FILE
Connection Name:
Include Header: true
No Of Records 10
Validating: server URL, offlineToken
Successfully validated server URL and offlineToken
```

```
Status: Test Data Generation Started...
Data Generation Response is [code:200, databaseName:null,
 project_id:3200, schema_id:5f48ad2b1491d2009db126fd, connectionURL:null,
 dbSchemaName:null, message:Data generation succeeded.,
 data_location:5f48ad1c1491d2009db126fb_077ae85b-5527-4aba-a64f-5b58d15df899_6dH9eZZfoq_1,
 timestamp:2020-08-31T08:36:56.526Z, tableName:null, collectionName:null]
Status: Test data generation completed successfully
Status: Downloading the Generated Test Data
File saved to
 C:
\Users\user1\Downloads\5f48ad1c1491d2009db126fb_077ae85b-5527-4aba-a64f-5b58d15df899_6dH9eZZfo
q_1.csv location
Status: Completed the download of Generated Test Data
```

The following sample shows the logs of the generated test data when you select `MONGODB` as the data storage type:

```
================================================================================
plugin: HCL OneTest Data Launch Plugin, id: com.hcllaunch.air.plugin.otd, version: 2
plugin command: 'cmd' '/C' '""D:\HCL Launch\LaunchAgent\opt\groovy-2.4.15\bin\groovy.bat" -cp
 "D:\HCL
 Launch\LaunchAgent\var\plugins\com.hcllaunch.air.plugin.otd_2_fcffff6fc25d3aad57596e97b05f621
f6a850ad06d6b0495e2954cbf5e1be0e0\classes;D:\HCL
 Launch\LaunchAgent\var\plugins\com.hcllaunch.air.plugin.otd_2_fcffff6fc25d3aad57596e97b05f621
f6a850ad06d6b0495e2954cbf5e1be0e0\lib\commons-codec-1.10.jar;D:\HCL
 Launch\LaunchAgent\var\plugins\com.hcllaunch.air.plugin.otd_2_fcffff6fc25d3aad57596e97b05f621
f6a850ad06d6b0495e2954cbf5e1be0e0\lib\commons-logging-1.2.jar;D:\HCL
 Launch\LaunchAgent\var\plugins\com.hcllaunch.air.plugin.otd_2_fcffff6fc25d3aad57596e97b05f621
f6a850ad06d6b0495e2954cbf5e1be0e0\lib\groovy-all-1.8.4.jar;D:\HCL
 Launch\LaunchAgent\var\plugins\com.hcllaunch.air.plugin.otd_2_fcffff6fc25d3aad57596e97b05f621
f6a850ad06d6b0495e2954cbf5e1be0e0\lib\groovy-plugin-utils-1.2.jar;D:\HCL
 Launch\LaunchAgent\var\plugins\com.hcllaunch.air.plugin.otd_2_fcffff6fc25d3aad57596e97b05f621
f6a850ad06d6b0495e2954cbf5e1be0e0\lib\gson-2.8.6.jar;D:\HCL
 Launch\LaunchAgent\var\plugins\com.hcllaunch.air.plugin.otd_2_fcffff6fc25d3aad57596e97b05f621
f6a850ad06d6b0495e2954cbf5e1be0e0\lib\hamcrest-core-1.1.jar;D:\HCL
 Launch\LaunchAgent\var\plugins\com.hcllaunch.air.plugin.otd_2_fcffff6fc25d3aad57596e97b05f621
f6a850ad06d6b0495e2954cbf5e1be0e0\lib\httpclient-4.5.6.jar;D:\HCL
 Launch\LaunchAgent\var\plugins\com.hcllaunch.air.plugin.otd_2_fcffff6fc25d3aad57596e97b05f621
f6a850ad06d6b0495e2954cbf5e1be0e0\lib\httpcore-4.4.10.jar;D:\HCL
 Launch\LaunchAgent\var\plugins\com.hcllaunch.air.plugin.otd_2_fcffff6fc25d3aad57596e97b05f621
f6a850ad06d6b0495e2954cbf5e1be0e0\lib\json-simple-1.1.1.jar;D:\HCL
 Launch\LaunchAgent\var\plugins\com.hcllaunch.air.plugin.otd_2_fcffff6fc25d3aad57596e97b05f621
f6a850ad06d6b0495e2954cbf5e1be0e0\lib\jsoup-1.11.3.jar;D:\HCL
 Launch\LaunchAgent\var\plugins\com.hcllaunch.air.plugin.otd_2_fcffff6fc25d3aad57596e97b05f621
f6a850ad06d6b0495e2954cbf5e1be0e0\lib\junit-4.10.jar" "D:\HCL
 Launch\LaunchAgent\var\plugins\com.hcllaunch.air.plugin.otd_2_fcffff6fc25d3aad57596e97b05f621
f6a850ad06d6b0495e2954cbf5e1be0e0\OneTestDataGenerationHCLLaunch.groovy" "D:\HCL
 Launch\LaunchAgent\var\temp\logs-49ec0c81-7f25-4494-accd-4bf9bd384d93\input.props" "D:\HCL
 Launch\LaunchAgent\var\temp\logs-49ec0c81-7f25-4494-accd-4bf9bd384d93\output.props""'
working directory: D:\HCL Launch\LaunchAgent\var\work\OTDComponent1
properties:
  PLUGIN_INPUT_PROPS=D:\HCL
 Launch\LaunchAgent\var\temp\logs-49ec0c81-7f25-4494-accd-4bf9bd384d93\input.props
```

```
  PLUGIN_OUTPUT_PROPS=D:\HCL
 Launch\LaunchAgent\var\temp\logs-49ec0c81-7f25-4494-accd-4bf9bd384d93\output.props
  connectionName=MongoConnection
  dataFileLocation=
  dataStorage=MONGODB
  offlineToken=****
  outputFormat=JSON
  project=otdproject
  records=1
  rootElement=Root:NewType1
  schema=otdschema
  seedValue=
  serverUrl=https://otd-fvt1.nonprod.hclpnp.com/
environment:
  AGENT_HOME=D:\HCL Launch\LaunchAgent
  AH_AUTH_TOKEN=****
  AH_WEB_URL=https://LP1-AP-51837142.PROD.HCLPNP.COM:8443
  AUTH_TOKEN=****
  DS_AUTH_TOKEN=****
  DS_SYSTEM_ENCODING=Cp1252
  JAVA_OPTS=-Dfile.encoding=Cp1252 -Dconsole.encoding=Cp1252
  PLUGIN_HOME=D:\HCL
 Launch\LaunchAgent\var\plugins\com.hcllaunch.air.plugin.otd_2_fcffff6fc25d3aad57596e97b05f621
f6a850ad06d6b0495e2954cbf5e1be0e0
  UD_DIALOGUE_ID=49ec0c81-7f25-4494-accd-4bf9bd384d93
  WE_ACTIVITY_ID=174d8e29-f157-0a93-07a4-5a90df84fb41
================================================================================
Data Generation information:
Server URL: https://otd-fvt1.nonprod.hclpnp.com/
Project: otdproject
schema: otdschema
Root Element: Root:NewType1
Output Format: JSON
Data Location:
Seed Value:
Data Storage: MONGODB
Connection Name: MongoConnection
Include Header: false
No Of Records 1
Validating: server URL , offlineToken
Successfully validated server URL and offlineToken
Status: Test Data Generation Started...
Data Generation Response is [code:200, databaseName:mdb, project_id:1150,
 schema_id:5f72c22171ff1a00a009ae50, connectionURL:{my-ots}-mongodb:27017, message:Data
 generation succeeded., timestamp:2020-09-29T08:02:59.141Z, collectionName:users]
Status: Test data generation completed successfully
Data is written into Mongo database
Connection URL: {my-ots}-mongodb:27017
Database Name: mdb
Collection name: users
```

# Managing access to HCL OneTest™ Server

Desktop clients and third-party integrations use offline user tokens to connect to HCL OneTest™ Server.

You can configure the desktop clients and third-party integrations to access HCL OneTest™ Server by using the offline user tokens created from the server.

Offline user tokens are used in the following cases:

- To retrieve secrets configured on the server to be used in certain tests in HCL OneTest™ API
- To enable publishing of test results and reports from desktop clients to HCL OneTest™ Server
- To enable the Resource Monitoring Service in HCL OneTest™ Performance to monitor a data source during a run on HCL OneTest™ Server
- To enable third-party integrations.

### About this task

From HCL OneTest™ Server, you can create an offline user token, copy it, and then use that token in each of the desktop clients and third-party integrations.

You can also delete all the tokens, when you no longer need access or you want to prevent access to the server.

Creating a token

- Follow these steps:

  1. Click the **User** icon 👤 from the menu bar and select **Create Token**.

     The Create Offline User Token dialog box is displayed.

     **Note:** This token is not accessible after you copy it as the dialog box closes.

  2. Copy the token and paste it into a private location for ongoing reference.

     The dialog box closes.

  3. Paste the token into the desktop client UI or the third-party integrations UI where this token is required.

     See the following table for a list of where you can use the tokens and where you can find more information.

     **Table 9. Using the offline user tokens to access HCL OneTest™ Server**

| Use the offline user token in the following places | More information |
|---|---|
| HCL OneTest™ API | See HCL OneTest™ API and HCL OneTest™ Server. |
| HCL OneTest™ Performance | See Publishing test results to the server. |

| Use the offline user token in the following places | More information |
|---|---|
| | See Enabling Enablement of Resource Monitoring services for a schedule. |
| HCL OneTest™ UI | See Publishing test results to the server. |
| Third-party integrations | See Integrating with other applications on page 403. |

Deleting tokens

- Follow these steps:

  1. Click the **User** icon 👤 from the menu bar and select **Delete Tokens**.

     The Delete All Offline User Tokens dialog box is displayed.

  2. Inform the desktop client users that the tokens were deleted. Fix any test automation scripts that used tokens.

# Chapter 7. Test Manager Guide

This guide describes how to manage and track your overall test effort. This guide is intended for test managers.

Related information

Managing access to server projects on page 504

## Team space management

You might want to become a team space member, add members to your team space, change a user role in a team space, modify the team space, or delete a team space.

Each team space is associated with at least one *Team Space Owner*. You can assign multiple owners to your team space. Any user who becomes a member of a team space and is assigned the role of a *Member*, *Project Creator*, *Architect*, or *Team Space Owner* of a team space, can access the team space.

As an owner of a team space, after you create a team space, you must configure the team space license. You can then add members to the team space and can assign different roles to each member of the team space. You can update or delete a team space. You can also add a repository to a team space to store a system model and edit the repository settings. Apart from adding a repository, you can add, update, or delete the Resource Monitoring agent if no longer required.

As an *Architect* of a team space, you can manage a team space repository and design a system model.

As a *Project Creator* of a team space, you can create a project in the team space.

As a licensed user who is assigned with a *Member*, *Project Creator*, *Architect*, or *Team Space Owner* role of a team space, you can add a Docker host to your team space. You can view the configured Docker, agents, or intercepts of your team space. You can edit only those Docker hosts that you created.

### Viewing team spaces

After you log in to HCL OneTest™ Server for the first time, you can view all the existing team spaces on the **Team Space Dashboard** page.

**About this task**

As a member of a team space, you can view your team spaces under **My Team Spaces**. You can also view the team spaces that are created by other users as the public team space and the Permissive team spaces under **Other Team Spaces**.

If the initial team space does not exist, then you can view the **Team Space Dashboard** without any navigation pane.

You as a member of a team space, when you want to use that team space immediately after you log in to HCL OneTest™ Server, you can suffix the alias in the application URL.

For example, if the alias of the team space is `first`, then you must use the following HCL OneTest™ Server URL in the address bar:

```
<server url>/#/first
```

Log in to HCL OneTest™ Server.
**Result**
The **Projects** page of the initial team space is displayed.

You can view the team spaces in the following panels:

- **My Team Spaces**: You can view all the team spaces that you created. You can also view the team spaces in which you are a member.
- **Other Team Spaces**: You can view all the team spaces that are created by others.

  **Note:** You cannot view the Restrictive private type team spaces.

### Results

You have viewed the team spaces that are created by you and other users on the **Team Space Dashboard**.

### What to do next

You can select a team space and can send a request to become a member of that team space. See Becoming a team space member on page 492.

## Adding members to a team space

When you create a team space to manage projects, you must add users to your team space so that they can create and access the projects.

### Before you begin

- As an administrator, you are assigned the default role of a *Team Space Owner*. If you log in to HCL OneTest™ Server as another user, then you must have been assigned the role of a *Team Space Owner*.
- As an owner of a team space, you must have configured a license for the team space. See Configuring named user licenses on page 101 .

You must have been assigned a role as a *Team Space Owner*. If you are a server administrator, then you are assigned with the default role of a *Team Space Owner*.

### About this task

As an owner of a team space, you can add multiple users to your **Permissive** or **Restrictive** team space.

1. Log in to HCL OneTest™ Server.
   **Result**

The **Projects** page of the initial team space is displayed.

2. Click **Dashboard**.

   **Result**

   The **Team Space Dashboard** page of the initial team space is displayed.

   **Note:** If the initial team space does not exist, then you can view the **Team Space Dashboard** page without any navigation pane.

3. Identify the team space for which you want to add the members.
4. Click the **Settings** icon ⚙.

   Alternatively, click **Manage > Configuration** in the navigation pane.

   **Result**

   The **Team Space Configuration** page is displayed.
5. Click the **Details** tab.
6. Enter a user name in the **Add People** field on the to add users.

   Alternatively, you can find the user name by using type search.

   **Result**

   The list of users is displayed.
7. Select the user.
8. Assign a role to the selected user.

   **Note:** All members are assigned the default roles of a *Project Creator* and *Architect*.

   You can choose the role for the user based on its associated actions:

| Roles | Responsibilities |
|---|---|
| *Team Space Owner* | ◦ Create a team space.<br>◦ Update and delete a team space.<br>◦ Add members to a team space.<br>◦ Remove members from the team space.<br>◦ Assign role to the members.<br>◦ Modify roles of the members.<br>◦ Create and modify a project.<br>◦ Add or update the system model.<br>◦ Add a repository to store a system model.<br>◦ Update and delete a repository in a team space.<br>◦ Add, update, or delete Docker.<br>◦ Configure or update agent and intercepts.<br>◦ Register, update, delete, and list Resource Monitoring sources in a team space. |

| Roles | Responsibilities |
|---|---|
| *Project Creator* | ◦ Create, update, and view a project<br>◦ Add, update, or delete Docker.<br>◦ Configure or update agent and intercepts. |
| *Architect* | ◦ Create, update, delete, and view a repository in a team space.<br>◦ Create, update, and delete components in the system model.<br>◦ Add, update, or delete Docker.<br>◦ Configure or update agent and intercepts. |
| *Member* | ◦ View the list of team spaces in the **Team Space Dashboard.**<br>◦ View the configuration of the team space.<br>◦ Add, update, or delete Docker.<br>◦ Configure or update agent and intercepts.<br>◦ View the registered Dockers, agents, or intercepts.<br>◦ View the Resource Monitoring agents in a team space.<br>◦ View the registered Resource Monitoring agents.<br>◦ View components in the system model. |

**Notes:**

- The members with the assigned role of a *Team Space Owner*, *Project Creator*, *Architect*, or *Member* of a team space can be assigned with any of the following project roles:
    - *Project Owner*
    - *Tester*
    - *Viewer*
- You can add or remove the role of a member by using the **Menu** icon ⋮ next to each user in the **Members** list.

9. Repeat the applicable steps to add another user.

**Results**

You have added members with assigned roles to your team space. You can view the list of all added users on the **Team Space Configuration** page.

**What to do next**

You can reassign or remove the roles of users to one or more team spaces that you own. See Managing members and their roles in a team space on page 493.

All users can request to be a member of another team space. See Becoming a team space member on page 492.

Users added as members to the team space can view the team space in **My Team Spaces**.

## Modifying the configuration of a team space

After you create a team space, if you want to modify the details of the team space, add a repository to your team space, or delete a team space, then you can perform these tasks on the **Team Space Configuration** page.

**Before you begin**
You must have been assigned the role of a *Team Space Owner*.

**About this task**

The **Team Space Configuration** page displays details of the team space.

As an owner of a team space, you can modify details of the team space, add members to your team space, and assign roles to the members of the team space on the **Team Space Configuration** page.

The *Team Space Owner* and *Architect* can add or remove a repository from the team space.

If the team space is no longer required, then the *Team Space Owner* can also delete that team space.

As a *Member*, *Project Creator*, or *Architect*, you can only view the configuration details of a team space on the **Details** tab of the **Team Space Configuration** page.

As a *Member*, and *Project Creator* you can only view the configuration details of a team space on the **Repository** tab of the **Team Space Configuration** page.

1. Log in to HCL OneTest™ Server.
   **Result**
   The **Projects** page of the initial team space is displayed.
2. Click **Dashboard**.
   **Result**
   The **Team Space Dashboard** page is displayed.
3. Select the team space that you want to modify from **My Team Spaces**.
4. Click the **Settings** icon ⚙ of the selected team space.
   **Result**
   The **Team Space Configuration** page is displayed.
5. Perform the modifications on the following details of the **Details** tabs of the team space:

| Fields | More information |
|---|---|
| Name | Name of the team space. |
| URL alias | An alias of the team space. Suffix the alias in the application URL to access a specific team space after you log into the application. |
| Available at | Displays the application URL by using an alias. You can copy this URL and use it when you want to ac- |

| Fields | More information |
|---|---|
| | cess the team space immediately after you log in to the application. |
| Member Access | Access to a team space based on the type of the team space. The team space can be of the following types:<br>◦ **Permissive**: All licensed users can access this team space without the approvals of a *Team Space Owner*.<br>◦ **Restrictive**: All licensed users must get approvals from the *Team Space Owner* to access this team space. |
| Visibility | The visibility mode of a team space can be of the following types:<br>◦ Public: The team spaces are visible to all licensed users in the **Other Team Spaces** panel.<br>◦ Private: The team spaces are not visible to all the licensed users in the **Other Team Spaces** panel.<br>**Note:** The visibility mode is available only when the **Restrictive** type of team space is selected. |
| Description | Brief description of the team space. This is an optional field. |
| Add People | List of members in a team space. |

6. Click the **Repository** tab to modify the details of the repository of the team space.

### Results

You have modified the configuration details of the selected team space.

### What to do next

You can add members and assign roles to each member of your team space.

Related information

Adding members to a team space on page 485

Managing members and their roles in a team space on page 493

## Viewing the configuration of a team space

You can view the configuration of a team space as a member of any team space. You can view the configuration details on the **Team Space Configuration** page.

**Before you begin**

You must be a member of a team space.

**About this task**

The **Team Space Configuration** page displays the configuration details of the team space.

As a member of a team space, if you want to access your team space immediately after logging into HCL OneTest™ Server, then you must suffix the alias in the application URL. You can find details of the alias on the **Team Space Configuration** page. You can also view the list of members and their roles in that team space on the **Team Space Configuration** page.

1. Log in to HCL OneTest™ Server.
2. Click **Dashboard**.

   **Result**

   The **Team Space Dashboard** page is displayed.
3. Identify the team space for which you want to view the configuration from **My Team Spaces**.
4. Click the **Settings** icon ⚙ of the team space.

   The **Team Space Configuration** page is displayed.

   You can view the following details about the team space:

| Fields | Tabs | More information |
|---|---|---|
| Name | **Details** | Displays the name of the team space. |
| URL alias | **Details** | Displays an alias of the team space. You must suffix this alias in the application URL to access a specific team space after you log in to the application. |
| Available at | **Details** | Displays the application URL by using an alias. You can copy this URL and use it when you want to ac- |

| Fields | Tabs | More information |
|---|---|---|
| | | cess the team space immediately after you log in to the application. |
| Member Access | **Details** | Displays the type of team space that defines the permissions to access the team space. A team space can be of the following types:<br>◦ **Permissive**: All licensed users can access this team space without the approvals of a *Team Space Owner*.<br>◦ **Restrictive**: All licensed users must get approvals from the *Team Space Owner* to access this team space. |
| Visibility | **Details** | The visibility mode of a team space can be of the following types:<br>◦ Public: The team spaces are visible to all licensed users in the **Other Team Spaces** panel.<br>◦ Private: The team spaces are not visible to all the licensed users in the **Other Team Spaces** panel.<br>**Note:** The visibility mode is available only when the **Restrictive** type of team space is selected. |
| Members | **Details** | Displays a list of members of the team space. |
| Repository | **Repository** | Displays the repository details that host the system model. |

**Results**

You have viewed the configuration details of the selected team space.

## Becoming a team space member

After logging into HCL OneTest™ Server, the licensed users must be a member of any team space to perform test activities in a project. As a licensed user, if you are not assigned the role of a member in any team space, then you can request the *Team Space Owner*to become a member of any existing team space.

**Before you begin**

You must be a licensed user.

**About this task**

If you want to become a member of a **Restrictive** team space, then only the owner of that team space can grant you the role of a member of a team space. If you request to become a member of a **Permissive** team space, then you automatically become a member of that team space.

**Note:** The server administrator can join any team space automatically without any approval and is assigned with a default role of *Team Space Owner*.

1. Log in to HCL OneTest™ Server.

   **Result**

   The **Projects** page of the initial team space is displayed.
2. Click **Dashboard**.

   **Result**

   The **Team Space Dashboard** page is displayed.
3. Select the team space that you want to join from the list of other team spaces in the **Team Space Dashboard**.

   **Note:**

   You can view only the public type of **Restrictive** team spaces and the **Permissive** team spaces.
4. Click the **Key** icon 🔑 of the selected team space.
5. Confirm your request when prompted.

   If you agree, then the *Team Space Owner* sees a notification about the request.

   After opening the notification, the *Team Space Owner* can view the user who is requesting the access and can accept or decline the request. If the *Team Space Owner* accepts your request, you are added and assigned a *Member* role.

   **Notes:**

- You can become a member of any team space only when the license of the selected team space is configured.
- If you are a server administrator, then the *Team Space Owner* role is the default role assigned.
- If you are a licensed user, then the *Member* role is the default role assigned and is restricted to specific actions.

You can then see the team space under **My Team Spaces** on the **Team Space Dashboard** page.

You might want to follow up with the *Team Space Owner* if your request is pending for some time.

### Results

You have become a member of another team space.

### What to do next

You can view the actions that you can perform based on your assigned role. See Adding members to a team space on page 485.

## Managing members and their roles in a team space

After you assign roles to the members of a team space, as a *Team Space Owner*, you might want to remove a member from the team space or reassign the role to the existing members of your team space.

### Before you begin

You must have been assigned the role of a *Team Space Owner*.

### About this task

As a licensed user, you are assigned a role of a *Member* in any team space after the approval of the *Team Space Owner*.

As an owner of a team space, you can remove or reassign any member of your team space with the roles of a *Team Space Owner*, *Project Creator*, or *Architect*. You can also remove any member from the list of members of the team space.

1. Log in to HCL OneTest™ Server.

   **Result**

   The **Projects** page of the initial team space is displayed.
2. Click **Dashboard**.

   **Result**

   The **Team Space Dashboard** page is displayed.
3. Identify the team space for which you want to reassign the roles of the members or remove a member from the list of members.
4. Click the **Settings** icon ⚙ of the identified team space.

   **Result**

The **Details** tab of the **Team Space Configuration** page is displayed.

5. Identify the member from the list of **Members**.
6. Click the **Menu** icon ⋮ next to the selected member.
7. Perform any one of the following operations to the identified member of the team space:

   **Choose from:**

   - Click **Remove Member** to remove the identified member from the team space.
   - Select the role that you want to assign or remove the assigned role for the identified member.

     You can choose multiple roles for the members based on the associated actions:

| Roles | Actions |
|---|---|
| *Team Space Owner* | ▪ Create a team space.<br>▪ Update and delete a team space.<br>▪ Add members to a team space.<br>▪ Remove members from the team space.<br>▪ Assign role to the members.<br>▪ Modify roles of the members.<br>▪ Create and modify a project.<br>▪ Add or update the system model.<br>▪ Add a repository to store a system model.<br>▪ Update and delete a repository in a team space.<br>▪ Add, update, or delete Docker.<br>▪ Configure or update agent and intercepts.<br>▪ Register, update, delete, and list Resource Monitoring sources in a team space. |
| *Project Creator* | ▪ Create, update, and view a project.<br>▪ Add, update, or delete Docker.<br>▪ Configure or update agent and intercepts. |
| *Architect* | ▪ Create, update, delete, and view a repository in a team space.<br>▪ Create, update, and delete components in the system model.<br>▪ Add, update, or delete Docker.<br>▪ Configure or update agent and intercepts. |
| *Member* | ▪ View the list of team spaces in the **Team Space Dashboard.**<br>▪ View the configuration of the team space.<br>▪ Add, update, or delete Docker.<br>▪ Configure or update agent and intercepts. |

| Roles | Actions |
|---|---|
| | ▪ View the registered Dockers, agents, or intercepts.<br>▪ View the Resource Monitoring agents in a team space.<br>▪ View the registered Resource Monitoring agents.<br>▪ View components in the system model. |

**Note:** The *Team Space Owner*, *Project Creator*, *Architect*, or *Member* of a team space can be assigned with any of the following project roles:

- *Project Owner*
- *Tester*
- *Viewer*

Similarly, you can remove the assigned role of any member of your team space.

### Results

You have completed any of the following tasks in your team space:

- Reassigned the role of the members.
- Removed the assigned role of the members.
- Removed the members.

Related information

Adding members to a team space on page 485

## Deleting a team space

You might want to delete a team space when it is no longer needed.

### Before you begin

You must have been assigned the role of a *Team Space Owner* of HCL OneTest™ Server.

### About this task

As a *Team Space Owner*, you can delete a team space that you do not need. When you delete a team space, it is a permanent deletion. You cannot undo the delete action on a team space.

When you delete the initial team space, you can only view the **Team Space Dashboard** page without any navigation pane.

After you delete the team space, you can use the name of the team space again.

1. Log in to HCL OneTest™ Server.

   **Result**

   The **Projects** page of the initial team space is displayed.
2. Click **Dashboard**.

   **Result**

   The **Team Space Dashboard** page is displayed.
3. Click **Manage > Configuration**.

   **Result**

   The **Team Space Configuration** page is displayed.
4. Click the **Details** tab.
5. Click **Delete Team Space**.
6. Confirm that you want to delete the team space when prompted.

   If you proceed, the team space is deleted including all of its projects and members.

**Results**

You have deleted the selected team space.

**What to do next**

You can view the available team spaces on the **Team Space Dashboard** page. See Viewing team spaces on page 484.

After you delete a team space, the license that is used for that team space is released and is available to configure the other team spaces, if required. See Configuring named user licenses on page 101.

# Test assets and a server project

HCL OneTest™ Server projects manage access to your test assets, which are stored in a Git repository. Projects are either public by default or private. Private projects are not discoverable by other users. You can either add your own project or you can request to be a member of another public project.

When you add your own project, you can configure it now or later. Configuring the project includes adding details about the project, adding one or more Git repositories, optionally adding secrets, classifying encrypted datasets, and adding a change management system. To run test assets that are associated with an encrypted dataset, you must categorize the encrypted datasets by creating a classification. You can also add users to your server project so they can access your test assets. To use Jira to create defects for the tests available in your project, you must configure the Jira server for your project on HCL OneTest™ Server.

Related information

Becoming a project member on page 503

Managing access to server projects on page 504

Working with Git repositories

Protecting API test assets by using secrets on page 512

Managing an encrypted dataset on page 518

Default user administration on page 68

Configuration of Jira as a change management system on page 397

## Working with projects

After you log in to HCL OneTest™ Server, you must have a project to perform the test activities. To work with projects, you must create and configure projects.

### Before you begin

You must have logged into HCL OneTest™ Server.

1. Go to the **Projects** page from the Home page of the initial Team Space. See Viewing projects on page 502.
2. Add a project. See Adding a project on page 499.
3. Add a repository. See Adding a repository to a team space on page 184.
4. Add secrets in secrets collections. See Secrets configuration on page 501.
5. Encrypt the data set resources. See Data security.
6. Configure a change management system. See Change management system.
7. Add members to a project and configure their roles. See Adding users to a server project on page 501.
8. View the test assets or resources maintained in the repositories. See Test resource access by using the global branch
9. Perform the following tasks based on the type of test assets or resources that you want to run:
    1. Add agents to your project, if you are using remote agents as a location to run certain tests. See Adding an agent to a project on page 220.
    2. Add remote Dockers, if you are using a remote Docker host as a location to run tests. See Adding a remote Docker host to the project for running tests on page 234.
    3. Read the considerations that you must take into account before you configure a run or start a virtual service. See Prerequisites for running virtual services on page 321.
    4. Read the considerations that you must take into account before you configure a test run. See Prerequisites to running tests on page 207.
10. Configure a run of the test asset or resource. See Test run configurations on page 240.
11. View the progress of a test run. See Viewing the progress of running test assets on page 312.
12. Manage a running test. See Management of running tests on page 312.
13. View the results of a test run after the run is completed. See Test results on page 359.
14. Review the test results and create defects for test runs from the Results page. See Creating Jira defects on page 401.

### Results

You have successfully performed all the tasks with the projects.

## Repository considerations for a server project

To collaborate with other project stakeholders, you can open test assets from a local clone of the Git repository, pull project test assets from a Git repository, and push changes made to your local test assets to a Git repository. Before you add a project and add a repository to that project, you must consider some information about repositories.

Consider the following sections about using Git repositories with HCL OneTest™ Server. For more information about installing, setting up, and using Git, see the Git documentation.

### Git

You must install Git or upgrade the version if you already have installed Git.

### Repositories and user identities

After you install Git, you must set up your Git repository and set up access for members. You must ensure that the repository contains your test assets.

Optionally, you can use a command line utility or Git tool to access the repository, upload your test assets, fetch or pull from the repository, push to the repository, clone the repository, and other operations you want to perform in Git.

### Local and shared repositories

After you create a remote or shared repository in Git, you can create a local version of the repository by cloning the remote repository. You must ensure that your test assets are available in the remote repository and are also cloned to the local repository.

Alternatively, if your test assets are on your local system, you can set up a Git repository in the bare mode, add the project files to the local repository, and then commit and push from the local repository to the remote repository in Git by using your preferred method.

**Note:** While copying the test assets from your local system to the repository, you must ensure that you copy the entire project that contains the test assets.

### User authentication for the Git repository

The administrator can set up different types of authentication for accessing the Git repository. HCL OneTest™ Server supports the following authentication types:

- HTTP with user name and password
- HTTP without user name and password
- HTTPS with user name and password
- HTTPS without user name and password
- SSH with SSH key and passphrase
- SSH with SSH key and without a passphrase

Based on the authentication type that is set for a repository, you must provide the same authentication values in HCL OneTest™ Server when you add a repository.

### Test assets

You must complete the following tasks in the desktop client where you are authoring your test before you check in and commit the test assets to the Git repository.

| Test type | Task | More information |
|---|---|---|
| API suite in HCL OneTest™ API | Change the local stub to a remote stub. | See Test run considerations for API Suites on page 208. |
| | Add the library files. | |

## Adding a project

The first step is to add a project and provide some details about it.

### Before you begin

To add one or more projects to manage access to your test assets, you must log in to HCL OneTest™ Server by providing the application URL in a browser.

If you are a new user, and LDAP and Active Directory are not configured, you must first sign up by completing a form that specifies user information such as an email, user name, and password, then you can log in by using that information.

### About this task

You add a project and give it a name and description.

1. Log in to HCL OneTest™ Server.
2. From the Home page, add a project. Give it a name and if you want, add a description.
   The Details page is displayed.
3. Decide if you want a public project, which is the default, or a private project.

### What to do next

You can add a repository to your project.

Related information

Default user administration on page 68

## Adding repositories to a server project

You can add repositories to a server project to access the test assets available in the respective repository.

### Before you begin

You must have completed the following tasks:

- Added a project on HCL OneTest™ Server.
- Been granted permission to access the repository.

### About this task

As a project owner or a tester, you can add one or more repositories to your project. When you add a repository, the Git repository is cloned to your project. While adding a repository, you must provide the necessary authentication credentials that are set for the Git repository that you want to add to your project. For example, if the authentication type is SSH, then you must provide the Git URL, a deploy key, and a passphrase.

After you log in, from the Home page, you can add one or more repositories to your project by following these steps.

1. Open your project.
2. Click the **Repositories** tab.
3. Click **Add repository**.

   The Add repository page is displayed.
4. Enter the URL of the Git repository that you want to add to your project.
5. Click the **Expand** icon to enter the required credentials based on any of the following authentication methods configured in the repository.

   To gain access to the repository, you must use any one of the authentication methods:

   | Authentication method | Credentials required |
   |---|---|
   | SSH | ◦ Deploy key<br>◦ Passphrase |
   | HTTPS | ◦ User name<br>◦ Password |
   | HTTP | ◦ User name<br>◦ Password |

   **Notes:**

   - You must have defined the authentication type and set the authentication credentials in the Git repository.
   - If you use SSH to connect to your remote repository and HCL OneTest™ Server displays an `Auth Fail` exception while using the deliver changes option, you can resolve this exception error by regenerating your SSH keys by using the -m PEM option.
6. Click **Add**.

The Git repository is cloned on HCL OneTest™ Server.

**Note:** Depending on the size of the repository you are cloning, it can take a few to several minutes to clone the repository.

7. **Optional:** Repeat the steps to add another repository.

**What to do next**

You can perform the following actions on the repository that you added:

- Update the authentication credentials if they are changed in the Git repository configuration.
- Delete a repository if it is no longer required.
- Refresh a repository to fetch and synchronize changes from the remote repository.
- Configure a webhook to notify the server there is a push event in the remote repository.
- Add a system model to the repository.

Related information

Working with Git repositories

## Secrets configuration

If you are working with an API suite and the project test asset contains environment variables that are required for the test runs, you must configure the environment variables as secrets in a secrets collection by using the **Secrets** tab. Configuring secrets enable the API suite to run correctly on HCL OneTest™ Server.

Related information

Protecting API test assets by using secrets on page 512

## Adding users to a server project

You can add users to your server project so they can access your test assets.

**Before you begin**

You own a project.

**About this task**

As a project owner, you can add other users to your public or private project. For example, if you have a large number of test assets that must be run, you might want your test team to help run them, and see the results.

1. Add one or more users to be members of your project. Click the **Settings** icon ⚙ on the Project card in the list of **My Projects** or click **Configure** in the navigation when the project is already open.
2. Add users by entering the user name of the user you want to add. You can add a partial name and then press **Enter** to see the user that you want to add.
3. Select that user and then assign a role. Click **Viewer**, **Tester**, or **Owner**. When you add a user, the default role is Viewer.

   The added user after logging in can see the owner's project in their list of **My Projects**.

4. Repeat the applicable steps to add another user.

#### What to do next

You can add or remove roles to one or more projects that you own. All users can request to be a member of another public project.

Related information

Becoming a project member on page 503

Managing access to server projects on page 504

## Viewing projects

After you log in to HCL OneTest™ Server, you can view the existing projects in the initial Team Space.

#### About this task

As a project owner, you can view your projects under the **My Projects** panel, and the projects that are created by other users as public projects are displayed under the **Other Projects** panel.

Log in to HCL OneTest™ Server.

The **Projects** page of the initial Team Space is displayed.

You can view the following projects:

- The projects that you created or were added to as a member with either the *Tester* or *Viewer* role are displayed under the **My Projects** panel.
- The projects that are created by others as public projects are displayed under the **Other Projects** panel.

#### Results

You have viewed the projects that are created by you and other users in the Team Space.

#### What to do next

You can perform any of the following tasks:

- Create a project. See Adding details to a server project on page 499.
- Request to become a member of a project that is displayed under the **Other Projects** panel. See Becoming a project member on page 503.
- View the other pages in the Team Space and perform tasks on those pages. See Tasks or operations in a Team Space on page 92.

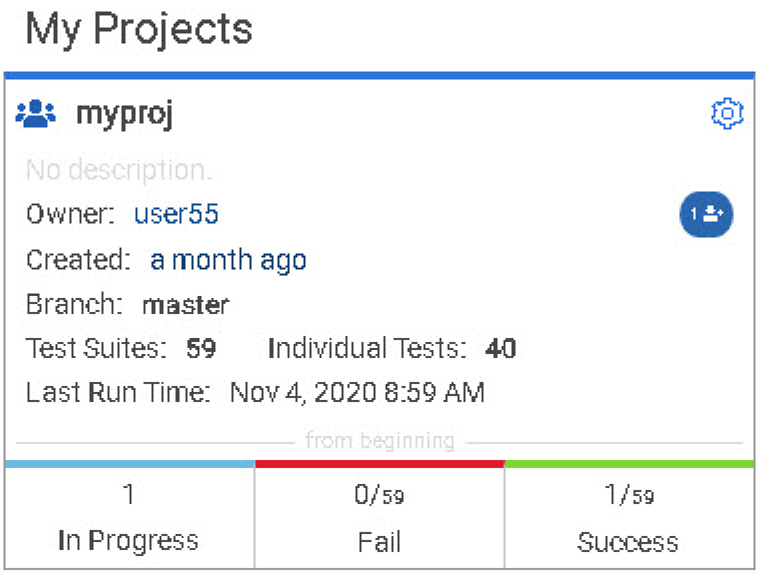
## Becoming a project member

You might want to request to be a member of another project. New users without any projects might also want to be a member of an existing project.

### About this task

As a project owner, you can add users to your public or private projects. All users can request to become a member of a public project.

1. Request to be a member of another project. Search for the project that you want to join in the list of other projects from the project Home page. Click that Project card or the **Key** icon 🔑. Only public projects are visible in the list of projects.
2. Confirm your request when prompted. If you agree, the project owner is notified. The project owner sees a pending request **Notification** icon on their Project card.



My Projects

myproj

No description.

Owner: user55

Created: a month ago

Branch: master

Test Suites: 59 Individual Tests: 40

Last Run Time: Nov 4, 2020 8:59 AM

from beginning

| 1<br>In Progress | 0/59<br>Fail | 1/59<br>Success |
|---|---|---|

After opening the notification, the owner can see the user that is requesting access and can accept or decline it.

If the project owner accepts your request, you are added as a member of the project with a Viewer role. A Viewer role is the default role assigned and is restricted to specific actions.

You then see the project under My Projects on the Home page.

If the project owner declines your request, follow up with the project owner.

## Managing access to server projects

You might want to add or remove a project member role from your project.

**About this task**

As a project owner, you can assign access by specifying a role when you add a user to your project. You can also assign more access or remove access for a user. Roles enable users to perform tasks on project resources such as running tests and viewing test results.

1. Choose the role in a project based on its associated actions.

| Option | Description |
|---|---|
| **Owner of a project** | ◦ All Tester's actions<br>◦ Update, archive, list, and view a project<br>◦ Assign, update, delete, and list user roles for a project<br>◦ Accept new member requests<br>◦ Create, update, and delete a dataset classification in a project<br>◦ Add, update, and remove an encrypted dataset from a classification<br>◦ Update current row for a listed dataset in a project<br>◦ Delete, list and view a report<br>◦ Update, delete, and list a repository in a project<br>◦ Get the test assets from a repository in a project<br>◦ Create, update, delete and list secrets in a project<br>◦ Create, update, delete, and list resource monitoring sources in a project<br>◦ Get the content of a resource monitoring source in a project<br>◦ Register, update, delete, and list resource monitoring agents<br>◦ Create, configure, and delete the Jira change management system |

| Option | Description |
|---|---|
| | ◦ Create and update defects in the Jira change management system<br>◦ Create and update components in the system model |
| **Tester of a project** | ◦ All Viewer's actions<br>◦ List repositories in a project<br>◦ Run a test and create a report<br>◦ Stop a test while running<br>◦ Get the test assets from a repository in a project<br>◦ Get a list of datasets in a project<br>◦ Get the content of a dataset<br>◦ Update the current row for a listed dataset in a project<br>◦ Add, update, and remove an encrypted dataset from a classification<br>◦ Delete a report if you created the report<br>◦ Create and update defects in the Jira change management system<br>◦ Create and update components in the system model |
| **Viewer of a project** | ◦ List your own roles on a project<br>◦ List running tests<br>◦ List and view a report<br>◦ List resource monitoring sources in a project<br>◦ Get the content of a resource monitoring source in a project<br>◦ List the resource monitoring agents<br>◦ View components in the system model<br>◦ View defects in the Jira change management system |
| **Non-member of a project** | ◦ Request to be a member of a project |

2. Open the project that you own.
   From the project details, you can see a list of project members for your project and then add or remove a members role.
3. Add or remove the member access by using the **Menu** icon ⋮ next to each user in the member list.

## Archiving or unarchiving server projects

You can archive a project when that project is no longer needed but you want to retain it for future reference. If you do want to use an archived project, you can unarchive it.

**About this task**

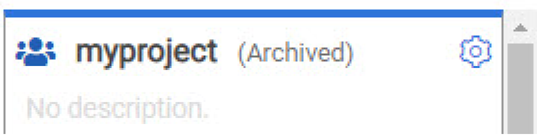
As a project owner,you can archive projects that are inactive. Archived projects are not visible to users or project members, but owners can show or hide their archived projects. Archived projects can be unarchived.

**Note:**

After you migrate the server data from a previous version to a newer version, the **Project Owner** must log in to HCL OneTest™ Server so that other members of the project can access the test assets in that project.

- Archive a project by following these steps:
  1. Open the project that you want to archive.
  2. From the project details, archive the project.
  3. Confirm that you want to archive the project when prompted. If you proceed, the project is archived and is hidden in your project list. To show or hide the archive projects, select the appropriate option by using the **Menu** icon ⋮ from the menu bar.

     When visible, the archived project shows as archived in the list of **My Projects**.

     myproject (Archived)

     No description.

     **Note:** The option to configure a project continues to display after you archive a project.
- Unarchive a project by following these steps:
  1. Open an archived project and unarchive it.
  2. Confirm that you want to unarchive the project when prompted. If you proceed, the project is unarchived and is shown in your project list.

## Deleting server projects

You might want to delete a project when the project is no longer needed, and you want to free up disk space.

**About this task**

As a project owner, you can delete a project that you no longer want. When you delete a project it is permanent. You cannot undo the delete project action.

After you delete the project, you can use the project name again.

1. Open the project that you want to delete.
2. From the project details, delete the project.
3. Confirm that you want to delete the project when prompted. If you proceed, the project is deleted including all of its test assets, results, and members.

**Note:** The repository that you used for the project is not deleted.

## An overview of test assets, modifications, and scheduled runs

After you configure a project with one or more Git repositories and run some test assets, you can view statistics about those test assets.

From the **Overview** page, several cards display charts, sliders, and tables of the statistics of the test assets in your server project. You can click some of the charts, sliders, or tables to see more information, such as the progress of a scheduled run or the results of a run. You can also review some statistics by date and see more details such as the proportion of test assets by type.

You can view a server project, which might contain multiple repositories and multiple branches, by selecting the name of the branch from the **Branch** drop-down list. For more information about selecting branches, see the related links.

**Test suites**

This card displays a slider of the percentage of test suites by type. A test suite is a collection of tests, including schedules, compound tests, API Suites, and AFT Suites. You can hover over the slider and see the proportion of test suites by test suite type. The number of suites shown are relative to the selected branch.

**Individual tests**

This card displays a slider of the percentage of individual tests by type that are used in the test suites. You might have performances tests, API tests, or UI tests according to the desktop clients that you used to create the tests. You can hover over the slider and see the proportion of individual tests by type. The number of tests shown are relative to the selected branch.

**Execution results**

This card displays a chart of the proportion of test suites that were run by status and by date. If no date is selected, the card displays all the run results associated with the project since it was created. You can see the run results by clicking a test suite status. You can also hover over the chart and see the test suite status and the proportion of test assets. You can view statistics by test suites run.

**Notes:**

- The results shown are generated from test suites run.
- The statistics exclude canceled test suites.

- The number of test suites and individual tests that are associated with your project are shown on the **Home** page along with a verdict summary. You can see the run results by clicking a verdict. You can also see the last date and time when you ran one or more test assets.
- The number of suites shown are relative to the selected branch.

**Last run**

This card displays a table of the last three runs by date and time. You can see the run results by clicking each last run.

**New or modified**

This card displays a chart of new and modified test assets that were made in your Git repository for a week or for the last seven days based on a selected date range. You can hover your mouse over each bar in the chart and see the proportion of test assets by type. The data shown is relative to the selected branch.

**Scheduled runs**

This card displays a table of the next three scheduled runs by date and time. You can see the progress of a scheduled run by clicking each scheduled run.

Related information

Test assets and a server project on page 496

Test resource access by using the global branch

# Managing repositories

After you create a project in HCL OneTest™ Server, you can add repositories that contain test assets and resources to the project. You can also change the repository details, update the user credentials of a repository, delete a repository, create a webhook for the configured repository.

### Tasks in a repository

To add a repository in your project and work on the repository, you can perform the tasks listed in the following table:

| Task | More information |
|---|---|
| Adding a repository | Adding repositories to a server project on page 499 |
| Deleting a repository | Deleting a repository on page 510 |

## Adding repositories to a server project

You can add repositories to a server project to access the test assets available in the respective repository.

**Before you begin**

You must have completed the following tasks:

- Added a project on HCL OneTest™ Server.
- Been granted permission to access the repository.

**About this task**

As a project owner or a tester, you can add one or more repositories to your project. When you add a repository, the Git repository is cloned to your project. While adding a repository, you must provide the necessary authentication credentials that are set for the Git repository that you want to add to your project. For example, if the authentication type is SSH, then you must provide the Git URL, a deploy key, and a passphrase.

After you log in, from the Home page, you can add one or more repositories to your project by following these steps.

1. Open your project.
2. Click the **Repositories** tab.
3. Click **Add repository**.

   The Add repository page is displayed.
4. Enter the URL of the Git repository that you want to add to your project.
5. Click the **Expand** icon to enter the required credentials based on any of the following authentication methods configured in the repository.

   To gain access to the repository, you must use any one of the authentication methods:

   | Authentication method | Credentials required |
   |---|---|
   | SSH | ◦ Deploy key<br>◦ Passphrase |
   | HTTPS | ◦ User name<br>◦ Password |
   | HTTP | ◦ User name<br>◦ Password |

   **Notes:**

   - You must have defined the authentication type and set the authentication credentials in the Git repository.
   - If you use SSH to connect to your remote repository and HCL OneTest™ Server displays an `Auth Fail` exception while using the deliver changes option, you can resolve this exception error by regenerating your SSH keys by using the -m PEM option.
6. Click **Add**.

The Git repository is cloned on HCL OneTest™ Server.

**Note:** Depending on the size of the repository you are cloning, it can take a few to several minutes to clone the repository.

7. **Optional:** Repeat the steps to add another repository.

**What to do next**

You can perform the following actions on the repository that you added:

- Update the authentication credentials if they are changed in the Git repository configuration.
- Delete a repository if it is no longer required.
- Refresh a repository to fetch and synchronize changes from the remote repository.
- Configure a webhook to notify the server there is a push event in the remote repository.
- Add a system model to the repository.

Related information

Working with Git repositories

## Deleting a repository

You can delete repositories that you no longer require in your test environment.

**Before you begin**

You must have completed the following tasks:

- Configured the repository that contains the test assets in your project.
- Been assigned a project owner or a tester role with access to delete repositories.

**About this task**

You can delete a repository that you have configured for your project.

1. Open your project and go to the Repositories page.

   You can see a list of repositories that you added to the project.
2. Delete a repository that you want to remove from the project.

   The repository is deleted and removed from the list.

## Selecting the global branch in a project

After you add one or more repositories to your project, you can start accessing all branches in all repositories of that project. You can select only one branch as a global branch from the list of branches in the **Execution**, **Datasets**, and **Overview** pages.

### Before you begin

You must have completed the following tasks:

- Been assigned a member or Project Creator role in a team space
- Added at least one repository in your project.

### About this task

You can access all the branches contained in all the repositories, which are added to the project, by using the **Branch** field on the **Execution**, **Datasets**, and **Overview** pages.

After you select a branch from any of the repositories as the global branch, this selection is retained for you as the global branch unless you change the branch on any of the following pages: **Overview**, **Execution**, or **Datasets** pages.

**Note:**

- When you add the first repository to a project, the global branch is set to the default branch of the repository.
- Branches are arranged in an increasing order of their names in the list.
- When you change the branch in one of the pages, then the same branch is displayed as selected in the other pages and the contents of that page displays the assets from the selected branch.

1. Log in to HCL OneTest™ Server and open the team space that contains your project.
2. Open the project.

   The project **Overview** page is displayed.
3. Go to the **Overview** page.
4. Select a branch from the **Branch** field.

   The branch that you selected in the **Overview** page is reflected in the **Execution** and **Datasets** pages and the test assets are filtered based on the global branch selection on each of these pages.

### Results

You have selected a global branch in a project.

# Protecting API test assets by using secrets

Secrets are key-value pairs that are created for your project in HCL OneTest™ Server under a secrets collection. You can create secrets collections for your project that enable you or members in your project to use secrets at test runtime either in HCL OneTest™ Server or in desktop clients.

The secrets collections in a project in HCL OneTest™ Server has a separate access control list managed by the members with access to the secrets collections. Controlling access to secrets means controlling access to applications and systems under test. The introduction of secrets (under secrets collections) for a project has simplified managing access to separate environments. If a member of a project does not have access to a secret, for example, *a server credential* then the member cannot accidentally or maliciously run tests against that server. For example, tests that must access the database server by using the server credentials to retrieve stored data can only be run by a member if the access to the secrets is granted.

**Note:** Secrets and secrets collections are applicable to test assets authored in HCL OneTest™ API that enable running tests in defined environments. Secrets are not applicable to tests authored in HCL OneTest™ UI or HCL OneTest™ Performance.

As a project member with the *Owner* or *Tester* role, you can create secrets collections in the project. You can grant or restrict access to the secrets collection that you create in the project.

Members with access to a secrets collection can access, edit, or delete the secrets collection in HCL OneTest™ Server and can view secrets, edit secrets, or delete secrets.

Members with access to secrets collections can grant access to or remove the following:

- Other members added specifically
- All members with a specific role

Members in the project with the *Owner* or *Tester* role and with access the secrets collection can use the secrets in the secrets collection, in tests at runtime.

If you are configuring a project to run an API Suite with tests that refer to secret values, you must configure the secrets under a secrets collection by using the **SECRETS** tab. You must complete the following tasks:

1. Create a secrets collection. See Step 1 on page 513 in Managing secrets collections on page 512.
2. Add secrets in the secrets collection created. See Step 1 on page 515 in Creating a secret in a secrets collection on page 514.
3. Grant access to project members or member roles, who can access the secrets collection. See Step 1 on page 516 in Granting access to members or member roles on page 515.

## Managing secrets collections

You can create secrets in a secrets collection for your project. Secrets are credentials required in certain tests during test runs. Secrets stored in the collection can be used by members to run tests on different environments and eliminates the need to store secrets in multiple locations. You can opt to edit or delete a secrets collection that you configured for your project any time after you create a secrets collection.

#### Before you begin

- You must have created a project on HCL OneTest™ Server. See Test assets and a server project on page 496.
- You must have completed the following tasks before you edit or delete a secrets collection:
  - Configured a secrets collection in your project.
  - Created secrets in the selected secrets collection. See Creating a secret in a secrets collection on page 514.
- You must be a member with the *Owner* or *Tester* role to create a secrets collection.
- You must be a member with access to the secrets collection to edit or delete the secrets collection. See Granting access to members or member roles on page 515.

#### About this task

You must configure secrets collections in your project so that the members of the project can use secrets contained in a collection during test runs. You can configure secrets so that you can use them in different test environments.

As a member with access to the secrets collection, you can opt to edit or delete a secrets collection configured in a project. For example, you might want to edit the secrets collection name or delete the secrets collection if the testing environment has changed and if secrets that are configured earlier are not required.

- To create a secrets collection, go to Step 1 on page 513.
- To edit or delete a secrets collection, go to Step 4 on page 514.

To create a secrets collection:

1. To create a secrets collection while configuring a new project in the HCL OneTest™ Server UI, open the **SECRETS** tab in the **Project Configuration** and create a secrets collection. Use **Add Collection**.
2. Alternatively, to create a secrets collection in an existing project, complete the following steps:
   a. Log in to HCL OneTest™ Server and from the User Interface (UI) open the project listed under **My Projects** for which you want to create a secrets collection.
   b. Open the Project Configuration page, and then open the **SECRETS** tab to create a secrets collection.
3. Enter a name for the secrets collection as its *Identifier*.

   > **Tip:** You can create a secrets collection that contains secrets for a particular test environment in your project. For example, the secrets collection *test_env* can contain secrets that application testers can use in tests that they run while the secrets collection *dev_env* can contain secrets that application developers can use in tests they run.

   A message is displayed for the successful creation of the secrets collection.

The secrets collection created is displayed.

You can add secrets to the secrets collection you created.

To edit or delete a secrets collection:

4. Log in to HCL OneTest™ Server and from the UI open the project listed under **My Projects**.
5. Open the secrets collection from the **SECRETS** tab in the Project Configuration page.
   If there are multiple secrets collections in the project, select the secrets collection that you want from the list.
   - To edit a secrets collection, go to Step 6 on page 514.
   - To delete a secrets collection, go to Step 7 on page 514.
6. To edit a secrets collection, complete the following steps:
   a. Click the **Edit** icon ✎ to edit the selected secrets collection.

      **Note:** The **Edit** icon ✎ is displayed only for the project owner.

   b. Edit the name of the secrets collection, and update the secrets collection.

      The secrets collection is updated with the updated name.
7. To delete a secrets collection, click the **Delete** icon 🗑 to delete the selected secrets collection.

   The selected secrets collection is removed from the list of secrets collections configured for the project.

**Results**

You have completed the following tasks:

- Created a secrets collection for your project.
- Edited the name of a secrets collection in your project.
- Removed a secrets collection from your project.

**What to do next**

- If you have created a new secrets collection, you must add secrets to your secrets collection.
- You must provide access to project members or member roles to the secrets collection by selecting members or member roles.

## Creating a secret in a secrets collection

You must create secrets in the secrets collections configured in your project so that the secrets contained in a collection can be used in certain tests by members of the project with access to the secrets collections during an API suite run.

### Before you begin

You must have created a project on HCL OneTest™ Server and configured a secrets collection in your project.

You must be a member with access to the secrets collection.

### About this task

You can also configure secrets such that the secrets can be used across different test environments by members with access to the secrets collection. Secrets correspond to the environment variables or tags that you create in a HCL OneTest™ API project specific to an environment.

1. To create a secret under a secrets collection while configuring a new project in the HCL OneTest™ Server UI, select the secrets collection listed in the **SECRETS** tab in the Project Configuration page and create a secret under the secrets collection.
2. Alternatively, to create a secret under a secrets collection in an existing project, complete the following tasks:
    a. Log in to HCL OneTest™ Server and from the UI open the project listed under **My Projects**.
    b. Open the secrets collection from the **SECRETS** tab in the Project Configuration page.
3. Enter a name for the secret as its *Identifier* and enter the `password` as the *Value* for the secret.
   For example, under the secrets collection (named as `test_env`), enter the name of the secret to access a database as `dbcred` and enter the `password` required to access the database as its value.

   A message is displayed for successful creation of the secret.

### Results

You have created secrets in the selected secrets collection for your project.

### What to do next

- You can view, edit, or delete the secrets created under a secrets collection any time you want.
- You can use the secrets in the tests that require these secrets during test runs.

## Granting access to members or member roles

You can grant or revoke access to the secrets collection in your project to individual members with different roles or the all members with a specific role. Without access to the secrets collection, members cannot view, create, edit, delete, or use the secrets in the secrets collection.

### Before you begin

You must have created a project on HCL OneTest™ Server and configured a secrets collection in your project.

You must be a member with access to the secrets collection.

1. To grant access to a secrets collection while configuring a new project in HCL OneTest™ Server UI, select the secrets collection listed in the **SECRETS** tab in the Project Configuration page.
2. Alternatively, to grant access to a secrets collection in an existing project, complete the following tasks:

   a. Log in to HCL OneTest™ Server and from the UI open the project listed under **My Projects**.

   b. Open the secrets collection from the **SECRETS** tab in the Project Configuration page.
   If there are multiple secrets collections in the project, select the secrets collection that you want from the list.

3. To grant access to a secrets collection in a new project or an existing project, select from the following methods:
   - To add all members with a specific role, click the role listed under **Grant access to role**. For example, if you select **Testers**, then all members in the project with a tester role are granted access to the secrets collection. You can select any role or all the roles listed.
   - To select specific members to grant access to the selected secrets collection, enter the name or the email ID of the member in the field box and add them from the list that is displayed.

**Note:** Members added specifically are listed under **Members with access to this collection** but all the members granted access solely due to their roles are not listed.

**Important:** Irrespective of the role that the member (*Owner*, *Tester* or *Viewer*) was assigned in the project, the access to the secrets collections has to be specifically granted to the members from the **SECRETS** tab.

#### Removing access to a secrets collection

4. To remove access granted to all members with a specific role or a specific member, select from the following methods:
   - To remove all members with a specific role, click the role listed under **Grant access to role** to clear the selection. For example, if **Testers** is selected and you clear it, then all members in the project with a tester role are removed from the access list to the secrets collection.
   - To remove specific members with access to the secrets collection, select the member and click the **Delete** icon ⊖.

**Notes:**

- Any member with access to the secrets collection can remove access of other members specifically added or of all members with a specific role.
- Members with access to the secrets collection can remove themselves from the access list. Members can do this when there is at least one member remaining in the list. After removing themselves, members cannot add themselves back to the access list and must be added by any of the other remaining members in the list.

**Results**

You have added members from your project or members with specific role to the access list of people who can access secrets in the selected secrets collection, or you have removed specific members or members with specific role from the access list.

**What to do next**

You can create secrets under secrets collections for your project.

## Managing secrets

You can view, edit, or delete the secrets configured under a secrets collection any time after you have created secrets or after you were granted access to the secrets collection. You can change the value of the secret by editing the secret. You can delete secrets that you no longer require in your test environment.

**Before you begin**

You must have created a project on HCL OneTest™ Server and configured a secrets collection in your project.

You must have created secrets in the selected secrets collection or the secrets collection must contain secrets.

You must be a member with access to the secrets collection.

1. Log in to HCL OneTest™ Server and from the UI open the project listed under **My Projects**.
2. Complete the following steps:
   a. Open the secrets collection from the **SECRETS** tab in the Project Configuration page.
   b. Optionally, select the secrets collection that you want from the list if there are multiple secrets collections in the project.

   The secrets configured in the selected secrets collection are displayed.
3. Complete the steps for the task you want to perform as listed in the following table:

| **Task** | **Steps** |
|---|---|
| Viewing a secret value | Click the **Show** icon 👁 for the secret you want to view its value, which most likely is a password for the secret. |

| Task | Steps |
|---|---|
| | The value configured for the secret is displayed. |
| Editing a secret value | Click the **Edit** icon ✎ for the secret you want to edit, and enter a `new value` for the secret as its *Value*. The value can be a password for the secret.<br>**Note:** You can only change the value of the secret.<br>The value of the selected secret is changed. |
| Deleting a secret | Click the **Delete** icon 🗑 in the row of the secret you want to delete.<br>After deleting it, the secrets list in the collection is removed from the list. |

**Results**

- You viewed the password configured of the secret under a secrets collection that you created or were granted access.
- You changed the secret value of the secret under a secrets collection in your project.
- You deleted and removed the secret from the selected secrets collection in your project.

**What to do next**

You can use secrets in the tests that require these secrets during test runs.

Related information

Test run configurations on page 240

Becoming a project member on page 503

# Managing an encrypted dataset

You can use encrypted datasets to limit access to confidential information such as account number or passwords. You can arrange data by an appropriate category so that project members can use datasets more effectively in certain tests and protect them.

**About this task**

A dataset can contain classified information that other members can access with permission. As a project owner, you can group encrypted datasets into different classifications and enable project members to view and edit datasets and run tests associated with the encrypted datasets. After you have created a classification, you can change the classification for a dataset. You can also delete a classification if you do not require it in your test environment.

**Notes:**

- You must grant access and provide an encryption key of an encrypted dataset to other members of the project to work with the encrypted dataset.
- A project member who has been added as a **Tester** role can work with the encrypted dataset.

## Creating a classification

As a project **Owner**, you can organize encrypted datasets by creating a classification so that project members can use and protect datasets more efficiently.

### Before you begin

You must have completed the following tasks:

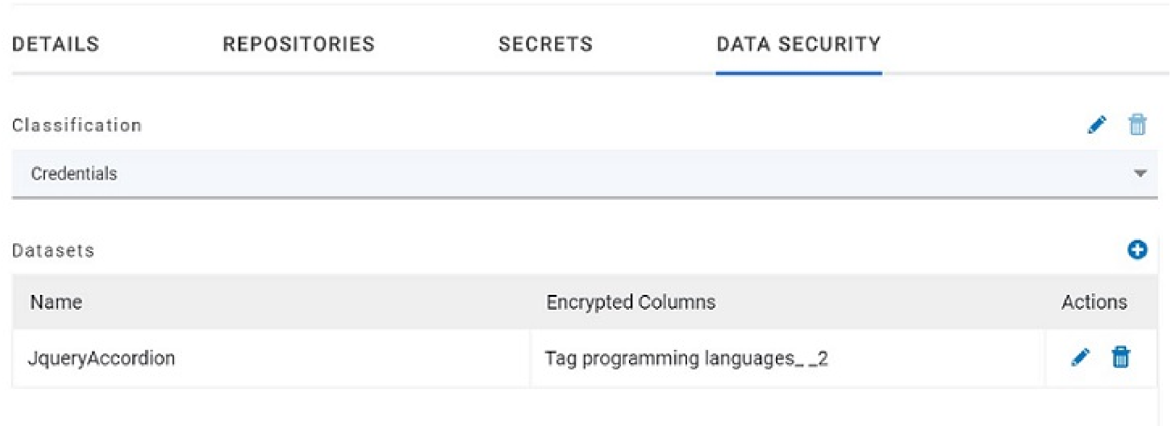
- Created a project in . See Test assets and a server project on page 496.
- Configured the repository that contains the test assets in your project. See Adding repositories to a server project on page 499.
- Created at least one dataset and encrypted the dataset with an encryption key. See Dataset encryption on page 113.

### About this task

After creating a classification, you can grant access and provide the encryption key to other members of the project to work with the encrypted dataset.

1. Go to the  URL.
2. Enter your user name and password, and then click **Login**.
3. Open your project from the  UI.
4. Click **Manage**, and then the **DATA SECURITY** tab.
5. Click **New classification** and enter a name for the classification.
6. Click **Create**.
7. Click the **Add** icon ⊕ to select the encrypted dataset to become part of the new classification.
8. Select a dataset from the list and enter the encryption key in the **Password** field for the dataset, and then click the **Add** icon.

The encrypted dataset that is added to the classification is displayed.



Classification

Credentials

Datasets

| Name | Encrypted Columns | Actions |
|---|---|---|
| JqueryAccordion | Tag programming languages_ _2 | |

### What to do next

You can grant access to other project members to use the encrypted dataset.

## Editing or deleting a classification

After you have created a classification, you can edit the name of the classification. You can also delete a classification when it is not required in your test environment.

### Before you begin

You must have completed the following tasks:

-
- Created at least two classifications. See Creating a classification on page 519.

### About this task

You can opt to edit or delete a classification for your project any time after you create a classification. For example, you might want to edit the name of the classification or delete the classification if the classification that is created earlier are not required.

**Note:** You must be a project **Owner** to create, edit, or delete a classification.

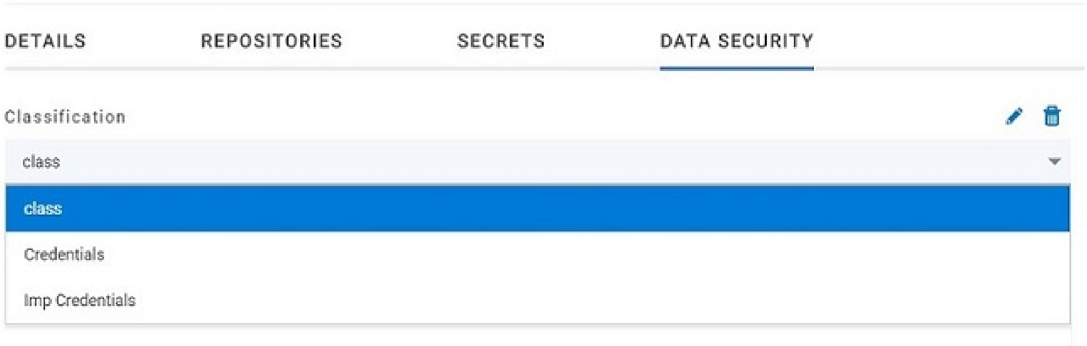
- To edit a classification, go to Step 1 on page 520.
- To delete a classification, go to Step 2 on page 521.

1. Perform the following steps to edit a classification:

    a.

    b.

c. Select a classification that you want to edit from the list.



d. Click the **Edit** icon ✎ to edit the selected classification.

e. Edit the name of the classification, and then click **Save**.

2. Perform the following steps to delete a classification:

   a.

   b. Select a classification that you want to delete from the list.

   c. Click the **Delete** icon 🗑.

   d. Click **Delete** in the **Delete classification** window to confirm the deletion of the classification.

   > **Note:** Before deleting a classification, you must move the added encrypted datasets to another classification else the delete icon is disabled.

**Results**

- You have edited the name for a classification in your project.
- You have deleted a classification.

## Moving an encrypted dataset to another classification

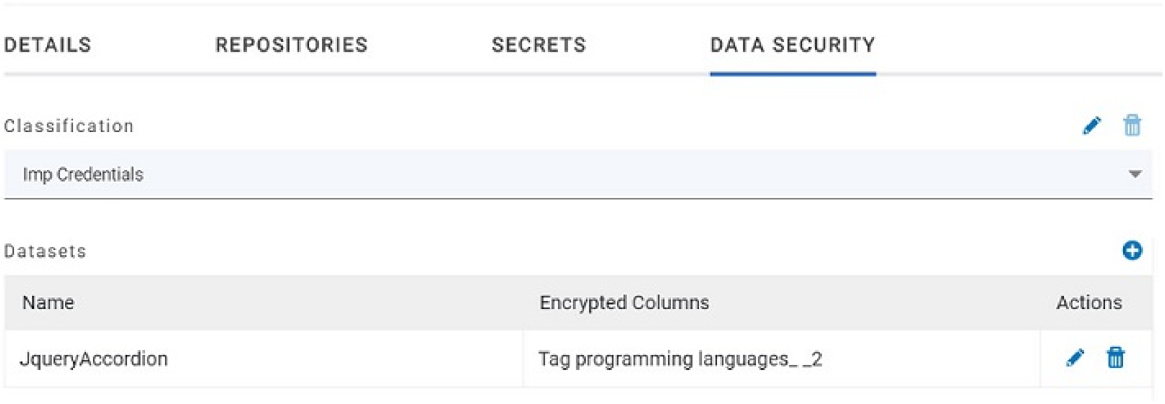
When you add many encrypted datasets to the same classification, you can move some of them to another classification.

**Before you begin**

1.
2.
3. Select a classification from the list that has the encrypted dataset.
4. Click the **Edit** icon ✎ from the **Actions** column of a dataset.

5. Select a classification from the list and enter the encryption key of the dataset in the **Password** field.
6. Click **Save**.

   A classification for a dataset is updated successfully.



Classification: Imp Credentials

Datasets

| Name | Encrypted Columns | Actions |
|---|---|---|
| JqueryAccordion | Tag programming languages_ _2 | |

### Results

You have moved the encrypted dataset from one classification to another.

## Removing a dataset from the classification

You can remove a datasets added to a classification when they are no longer required.

### Before you begin

You must have completed the following tasks:

-
- Created a classification and added the encrypted dataset to it. See Creating a classification on page 519.

1.
2.
3. Select a classification from the list that has an encrypted dataset.
4. Click the **Delete** icon 🗑 from the **Actions** column of a dataset.
5. Click **Remove** in the **Change the classification for the Dataset** window.

   **Note:** Removing dataset from the classification also removes the password stored in  for encrypted data. You must enter the password again to gain access to the encrypted columns.

### Results

You have removed the datasets from a classification.

## Granting classification access to members or members roles

You can grant or revoke access to the classification in your project to individual members with different roles or the all members with a specific role. Without access to the classification, members cannot view, create, edit, delete, or use the classification.

**Before you begin**

You must be a project **Owner** and have completed the following tasks:

-
- Added one or more users to your project. See Adding users to a server project on page 501.
-
- Created at least one classification. See Creating a classification on page 519.

1.
2.
3. Select the classification from the drop-down list.
4. Choose any of the following methods to grant access to a member:
   **Choose from:**
   - To add all members with a specific role, click the role listed under **Grant access to role**. For example, if you select **Testers** then all members in the project with a tester role are granted access to the selected classification. You can select any role or all the roles listed.
   - To select specific members to grant access to the selected classification, enter the name or the email ID of the member in the **Grant access to member** field and add them from the list that displays.

   **Note:** Members added specifically are listed under **Members with access to this classification** but the members added for a role are not displayed.

   **Important:** Irrespective of the role that the member (**Owner**, **Tester** or **Viewer**) was assigned in the project, the access to the classification has to be specifically granted to the members from the **DATA SECURITY** tab.
5. Choose any of the following methods to revoke access from a member:
   **Choose from:**
   - To remove all members with a specific role, click the role listed under **Grant access to role** to clear the selection. For example, if **Testers** is selected and you clear it, then all members in the project with a tester role are removed from the access list to the classification.
   - To remove specific member with access to the classification, select the member and click the **Delete** icon ⊖.

**Notes:**

- ◦ Any member with access to the classification can remove access of other members specifically added or of all members with a specific role.
- ◦ Members with access to the classification can remove themselves from the access list provided that there is at least one member in the list. After removing themselves, members cannot add themselves back to the access list and must be added by any of the other members in the list.

**Results**

You have granted or revoked the classification access to project members.

Related information

Adding a project on page 499

# Managing notifications

HCL OneTest™ Server provides a feature to display notifications for different events within the user interface (UI) of HCL OneTest™ Server. You can configure an SMTP server on HCL OneTest™ Server when you want HCL OneTest™ Server to send out notifications about the different events as emails to the subscribed users.

HCL OneTest™ Server provides you the following features to receive notifications about server events:

| Channels | Description |
|---|---|
| In-app | Notifications are displayed within the UI of HCL OneTest™ Server. |
| SMTP | Notifications are sent by HCL OneTest™ Server in emails to subscribed users. |

When you register as a user in HCL OneTest™ Server, the in-app and SMTP channels are automatically subscribed for you.

## Managing notifications in the server UI

Viewing the in-app notifications or notifications in the server UI is a feature that is enabled automatically when you register as a user in HCL OneTest™ Server.

You are subscribed to the following channels for all events as the default option when you sign up on the server.

- In-app
- SMTP

If you want to receive notifications about specific events and not all events, you can change the default subscriptions and modify your subscription for the server events. You can also view the in-app notifications that are displayed for you. See Managing in-app notifications on page 525.

### Managing SMTP notifications

You, as a *Server Administrator* or *Team Space Owner* can configure an existing Simple Mail Transfer Protocol (SMTP) server so that HCL OneTest™ Server can send out notifications about the server events in emails to the registered users. See Configuring an SMTP server to manage email notifications on page 535.

You are subscribed to the following channels for all events as the default option when you sign up on the server.

- In-app
- SMTP

If you, as a registered user of HCL OneTest™ Server want to receive the email notifications about specific events and not all events, you can change the default subscriptions and modify the subscription for the server events. See Managing email notifications on page 531.

## Managing in-app notifications

You can manage the in-app notifications by changing the subscriptions that are automatically enabled for you for all server events. You can view the server events that are automatically subscribed and then change the subscription for any or all events. For example, as a team space owner, you can choose to be notified when members of the team space create new projects or delete their projects but not be notified when tests are run.

### Before you begin

-
-

### About this task

The in-app notifications that are automatically subscribed for you, are displayed in the server UI under the following categories for the server events that occur:

| Category | Notifications displayed |
|---|---|
| User | Include information about actions performed by team space members in the team space. |
| System | Include information about actions performed by registered users at the server level. |

1.
2. Select from the following actions that you want to perform:

  - Modify notification settings. Go to Step 3 on page 526.
  - View notifications. Go to Step 4 on page 527.
  - Unsubscribe from the in-app notifications. Go to Step 5 on page 528.

3. Perform the following steps to modify the notification settings:

   a. Click the **User** icon on the menu bar.

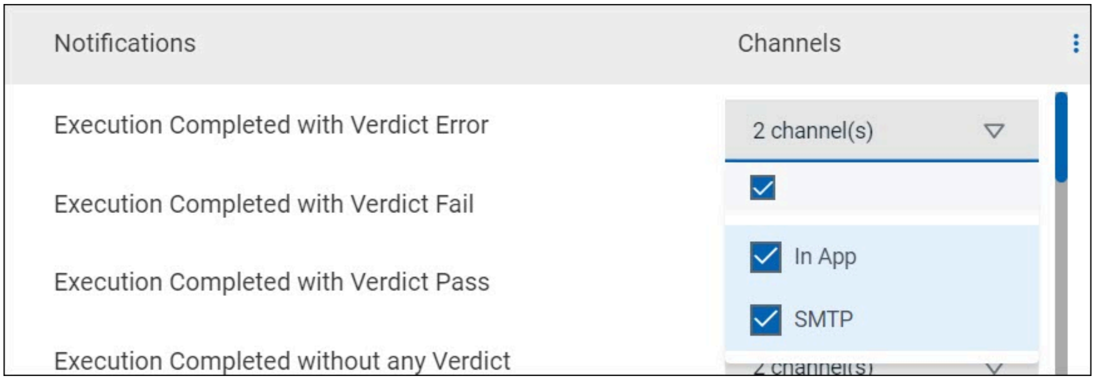
   b. Click **Notification Settings**.

      The **Notification Settings** dialog is displayed.

   c. Change the language in which you want to receive the notifications, if you do not want to use the default language.

      The default language is the locale language set on the computer that hosts .

      You can notice that for each of the server events, both the channels are displayed as selected, which is the default selection unless you have modified the subscriptions.



| Notifications | Channels |
|---|---|
| Execution Completed with Verdict Error | 2 channel(s) |
| Execution Completed with Verdict Fail | |
| Execution Completed with Verdict Pass | |
| Execution Completed without any Verdict | 2 channel(s) |

   d. Identify the server events for which you want to change the subscription.

   e. Perform the actions indicated in the following table to clear the subscriptions for server events that you do not want to be notified and to select the in-app subscription for the server events that you want to be notified.

| When | Action |
|---|---|
| You do not want to receive the in-app notifications for a specific server event. | Perform the following steps:<br>i. Expand the channel list in the row of the server event.<br>ii. Clear the checkbox in the header row to clear the subscriptions to both channels.<br>iii. Click **Apply**. |

| When | Action |
|---|---|
| | You have unsubscribed from both channels and the in-app notifications for the server event is not displayed for you when the event occurs. |
| You want to receive the in-app notifications for a specific server event. | Perform the following steps:<br>i. Expand the channel list in the row of the server event.<br>ii. Select any of the following actions:<br>▪ Clear the **SMTP** checkbox.<br>The **In App** selection is retained while the checkbox in the header row and for the **SMTP** options are cleared.<br>▪ Clear the checkbox in the header row to clear the subscriptions to both channels, and then select the **In App** checkbox.<br>iii. Click **Apply**.<br>You have unsubscribed from the SMPT channel and retained the subscription for the in-app notifications for the server event. |

f. Go to Results on page 530.

4. Perform the following steps to view the in-app notifications:

a. Click the **Notifications** icon 🔔 on the menu bar.

The **Notifications** dialog is displayed. The notifications are displayed in the following tabs:

- User
- System

The notifications in the **USER** tab is displayed as the default option.

b. View the notifications in the **USER** tab.

c. Click the **SYSTEM** tab to view the notifications at the server level.

You can find the information about the default notifications that you can view based on your role in the server as in-app notifications in the **USER** tab and **SYSTEM** tab.

| Category | Notification details | or | or | | |
|---|---|---|---|---|---|
| User notifications | Information about events at the server level or across team | ✓ | ✓ | ✗ | ✗ |

| Category | Notification details | or | or | | |
|---|---|---|---|---|---|
| | spaces performed by other team space owners. | | | | |
| | Information about events in the team space performed by other team space members. | ✓ | ✗ | ✗ | ✗ |
| | Information about events in the project performed by other project members. | ✗ | ✓ | ✓ | ✗ |
| | Information about events performed by you in a project. | ✗ | ✗ | ✓ | ✓ |
| System notifications | Information about events at the server level or across team spaces. | ✓ | ✓ | ✗ | ✗ |
| | Information about events in the team space performed by other team space members. | ✓ | ✓ | ✗ | ✗ |
| | Information about events in the projects at the team space level performed by other project members. | ✓ | ✓ | ✗ | ✗ |

d.

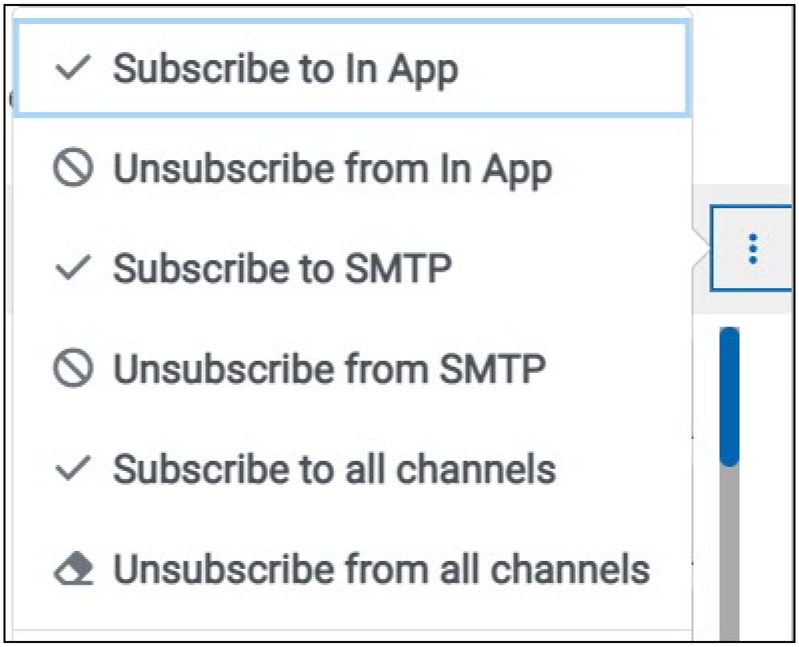
5. Perform the following steps to unsubscribe to the in-app notifications:

a.

b.

c. Click the **Open action menu** icon ⋮ in the row of **Notifications**.

The following panel is displayed:



✓ Subscribe to In App

⃠ Unsubscribe from In App

✓ Subscribe to SMTP

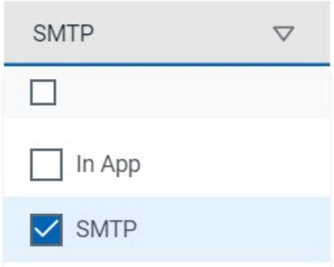
⃠ Unsubscribe from SMTP

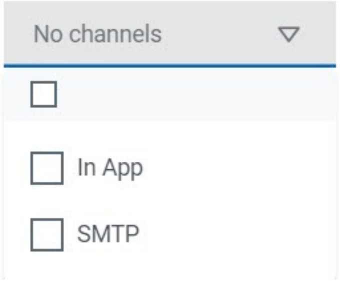
✓ Subscribe to all channels

Unsubscribe from all channels

You can notice that the subscriptions to in-app, SMTP, and all channels are displayed as selected.

d. Perform the actions indicated in the following table for the option you want:

| When | Action | Results in |
|---|---|---|
| You do not want to receive any in-app notifications. | Click the **Unsubscribe from In App** option, and then click **Apply**. | The in-app channel selection for all server events is cleared and you are unsubscribed from the in-app channel. The SMTP notifications is displayed as selected for subscription for all server events.<br><br>SMTP<br>□<br>□ In App<br>☑ SMTP |
| You do not want to receive any in-app or the SMTP email notifications. | Click the **Unsubscribe from all Channels** option, and then click **Apply**. | The in-app channel and the SMTP channel selections for all server events are cleared and you are unsubscribed from both |

| When | Action | Results in |
|---|---|---|
| | | the channels. No channel is displayed as selected for subscription for all server events.<br><br>No channels ▽<br>□<br>□ In App<br>□ SMTP |
| You want to subscribe to the in-app notifications for all server events. | Click the **Subscribe to In App** option, and then click **Apply**. | The in-app channel for all server events is selected and you are subscribed to the in-app channel. The in-app notifications is displayed as selected for subscription for all server events. |
| You want to receive notifications as in-app and as SMTP email notifications. | Click the **Subscribe to all Channels** option, and then click **Apply**. | The in-app channel and the SMTP channel are displayed as selected for all server events and you are subscribed to both the channels. |

e.

**Results**

You completed the following tasks:

- Modified the notification subscriptions.
- Viewed the notifications displayed for you in the **USER** tab and **SYSTEM** tab.
- Unsubscribed from or subscribed to either the in-app channel or both the channels.

Related information

Managing notifications on page 524

## Managing email notifications

You can manage the email notifications by changing the subscriptions that are automatically enabled for you for all server events. You can view the server events that are automatically subscribed and then change the subscription for any or all events. For example, as a project owner, you can choose to be notified when members of the project add a repository or run a test in a team space but not be notified when projects are created in the team space.

### Before you begin

- Ensured that the  or  has configured a Simple Mail Transfer Protocol (SMTP) server. See Configuring an SMTP server to manage email notifications on page 535.
-
-

### About this task

The email notifications in the SMTP channel are automatically subscribed for you for all server events. You can view the server events that are subscribed in the **Notifications Settings** dialog. You can either unsubscribe from the SMPT channel or change your subscriptions for the server events for which you do not want the email notifications.

1.
2. Select from the following actions that you want to perform:
   - Modify notification settings. Go to Step 3 on page 531.
   - Unsubscribe from the email notifications. Go to Step 4 on page 533.
3. Perform the following steps to modify the email notifications settings:

   a. Click the **User** icon 👤 on the menu bar.

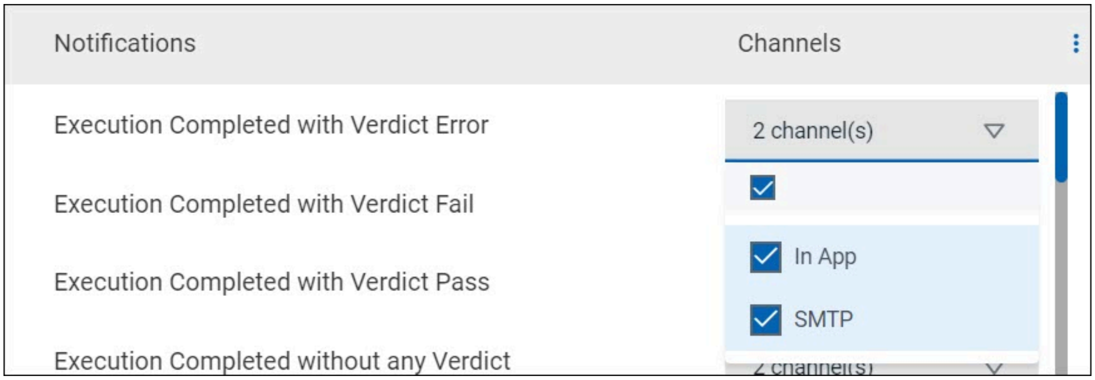
   b. Click **Notification Settings**.

   The **Notification Settings** dialog is displayed.

   c. Change the language in which you want to receive the email notifications, if you do not want to use the default language.

   The default language is the locale language set on the computer that hosts .

   You can notice that for each of the server events, both the channels are displayed as selected, which is the default selection unless you have modified the subscriptions.



Notifications | Channels

Execution Completed with Verdict Error | 2 channel(s)

☑

☑ In App

☑ SMTP

Execution Completed with Verdict Fail

Execution Completed with Verdict Pass

Execution Completed without any Verdict | 2 channel(s)

d. Identify the server events for which you want to change the subscription.

e. Perform the actions indicated in the following table to clear the subscriptions for server events that you do not want to be notified and to select the SMTP subscription for the server events that you want to be notified in emails.

| When | Action |
|---|---|
| You do not want to receive the email notifications for a specific server event. | Perform the following steps:<br>i. Expand the channel list in the row of the server event.<br>ii. Clear the checkbox in the header row to clear the subscriptions to both channels.<br>iii. Click **Apply**.<br>You have unsubscribed from both channels and the email notifications for the server event are not sent to you when the event occurs. |
| You want to receive the email notifications for a specific server event. | Perform the following steps:<br>i. Expand the channel list in the row of the server event.<br>ii. Select any of the following actions:<br>▪ Clear the **In App** checkbox.<br>The **SMTP** selection is retained while the checkbox in the header row and for the **In App** options are cleared.<br>▪ Clear the checkbox in the header row to clear the subscriptions to both channels, and then select the **SMTP** checkbox.<br>iii. Click **Apply**. |

| When | Action |
|---|---|
| | You have unsubscribed from the in-app channel and retained the subscription for the email notifications for the server event. |

f. Go to Results on page 535.

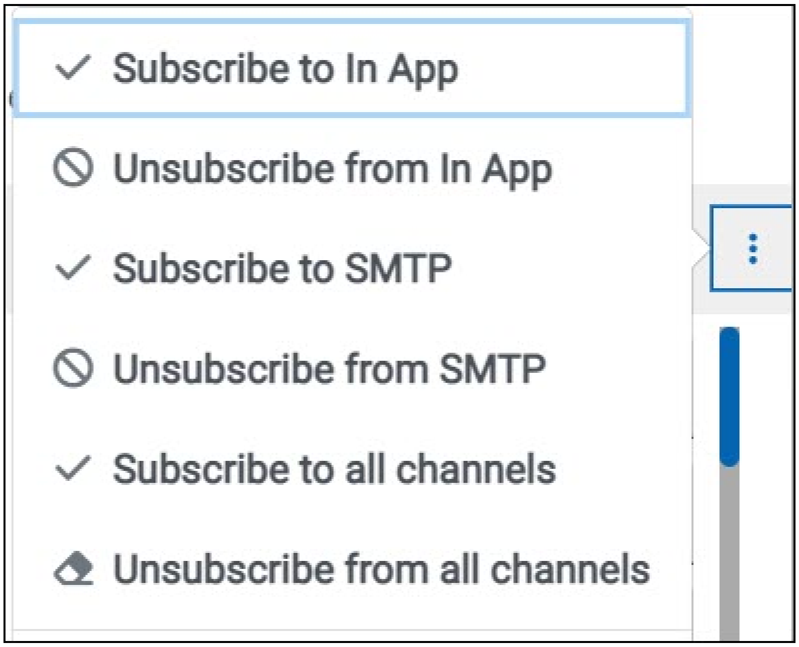
4. Perform the following steps to unsubscribe to the email notifications:

a.

b.

c. Click the **Open action menu** icon ⋮ in the row of **Notifications**.

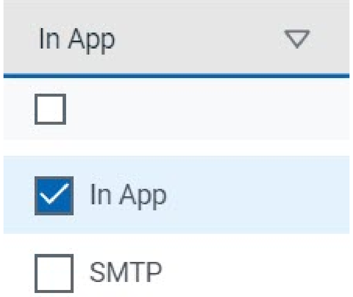
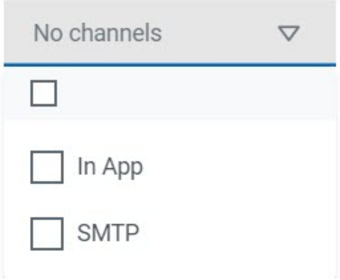
The following panel is displayed:



You can notice that the subscriptions to in-app, SMTP, and all channels are displayed as selected.

d. Perform the actions indicated in the following table for the option you want:

| When | Action | Results in |
|---|---|---|
| You do not want to receive any email notifications. | Click the **Unsubscribe from SMTP** option, and then click **Apply**. | The SMTP channel selection for all server events is cleared and you are unsubscribed from the SMTP channel. The **In App** option is displayed as selected |

| When | Action | Results in |
|---|---|---|
| | | for subscription for all server events.<br><br> |
| You do not want to receive any in-app or the SMTP email notifications. | Click the **Unsubscribe from all Channels** option, and then click **Apply**. | The in-app channel and the SMTP channel selections for all server events are cleared and you are unsubscribed from both the channels. No channel is displayed as selected for subscription for all server events.<br><br> |
| You want to subscribe to the email notifications for all server events. | Click the **Subscribe to SMTP** option, and then click **Apply**. | The SMTP channel for all server events is selected and you are subscribed to the in-app channel. The **In App** option is displayed as selected for subscription for all server events. |
| You want to receive notifications as in-app and as SMTP email notifications. | Click the **Subscribe to all Channels** option, and then click **Apply**. | The in-app channel and the SMTP channel are displayed as selected for all server events |

| When | Action | Results in |
|---|---|---|
| | | and you are subscribed to both the channels. |

e.

**Results**

You completed the following tasks:

- Modified the notification subscriptions.
- Unsubscribed from or subscribed to either the SMTP channel or both the channels.

Related information

Managing notifications on page 524

## Configuring an SMTP server to manage email notifications

Before you can receive email notifications about events that occur in HCL OneTest™ Server, you need to configure an existing Simple Mail Transfer Protocol (SMTP) server in HCL OneTest™ Server. When events occur in HCL OneTest™ Server, email notifications are sent to the registered users of HCL OneTest™ Server if they are subscribed to the SMTP notification channel.

**Before you begin**

You must have completed the following tasks:

- Ensured that you are assigned a role as a *Team Space Owner* in the team space. See Managing members and their roles in a team space on page 493.
- Ensured that an existing SMTP server is available to be connected to HCL OneTest™ Server.

**About this task**

The configuration works for emails that are supported by the SMTP server. To use this capability, you must enable HCL OneTest™ Server to connect to the SMTP server to send email notifications.

As a server administrator, you can configure email notifications so that other users can receive information about actions on the server projects.

1. Log in to HCL OneTest™ Server and open the team space that contains your project.
2. Click the **Settings** icon ⚙ from the menu bar.

   The **Notifications** page is displayed.
3. Click **Add Configuration**.

4. Provide the following details to configure the SMTP server:
    - A name to the configuration.
    - The host name and port number of the SMTP server.

      **Note:** The default configuration uses a local SMTP server that listens to the default port 25. This SMTP server does not require any authentication.

    - The email address of the sender. The email address is displayed in email notifications. The email notifications are sent to the users based on the notification settings.
    - The encryption type for secure communications is when the SMTP server is on an external network. You can enable either **STARTTLS** or **SSL/TLS**.

      **Note:** The SMTP server for the secure communication listens to the default port 465.

    - Login credentials are required if the SMTP server requires authentication.
5. Click **Add**.

   **Note:** The SMTP configuration is added after a valid connection is established with the SMTP server.

**Results**

You configured an SMTP server to connect with HCL OneTest™ Server and send out email notifications about the server events to users who have subscribed to the SMTP channel.

**What to do next**

You can modify the default subscriptions for the SMTP channel, see Managing email notifications on page 531.

Related information

Managing notifications on page 524

# Chapter 8. Troubleshooting Guide

This guide describes how to analyze and resolve some of the common problems that you might encounter while you work with HCL OneTest™ Server.

Known problems are documented. For more information, see the download document https://support.hcltechsw.com/csm?id=kb_article&sysparm_article=KB0081911. You can also search this information center for troubleshooting documentation.

You can contact HCL Support if you are unable to troubleshoot the problem. Gather all the required background information and provide the details to HCL Support for investigation. For more information, see HCL Customer Support.

## Troubleshooting issues

You can find information about the issues or problems that you might encounter while working with HCL OneTest™ Server. Details about issues, their causes and the resolutions that you can apply to fix the issues are described.

The troubleshooting issues are presented to you in the following tables based on where or when you might encounter these issues on HCL OneTest™ Server.

- Table 10: Troubleshooting issues: installation on page 537
- Table 11: Troubleshooting issues: server administration on page 540
- Table 12: Troubleshooting issues: resource monitoring on page 540
- Table 13: Troubleshooting issues: configuring test runs on page 540
- Table 14: Troubleshooting issues: test or stub runs on page 542
- Table 15: Troubleshooting issues: test results and reports on page 545

**Table 10. Troubleshooting issues: installation**

| Problem | Description | Solution |
|---|---|---|
| On Ubuntu, when you are installing the server software and you encounter errors in the scripts that are running. | At times, scripts might not appear to be running due to any of the following reasons:<br>• Slow connection speeds.<br>• Insufficient CPU, memory, or disk resources.<br>• A firewall that was configured incorrectly is already enabled. | You can complete any of the following tasks:<br>• To identify the issue, you can perform a diagnostic check by running the following command:<br>`journalctl -u k3s`<br>This command displays the log that you can use to check for the problem. |

**Table 10. Troubleshooting issues: installation (continued)**

| Problem | Description | Solution |
|---|---|---|
| | | • Run the following command to see which pods are running and which pods are not running:<br>`kubectl get pods -A`<br>Run the following command to get details about a specific pod:<br>`kubectl describe pod -n <namespace> <pod name>`<br>• Follow the on-screen instructions to resolve the errors.<br>• Some issues can be solved by re-running the following script:<br>`sudo ./ubuntu-init.sh` |
| On Ubuntu, DNS is not working as expected. | | The DNS configuration that is used by the cluster can be displayed by using the following command:<br>`kubectl get cm -n kube-system coredns -ojsonpath="{.data.Corefile}"`<br>The **forward** setting displays the *nameservers* that are used. For example, you might see the following in the `corefile`:<br>`.:53 {`<br>`  :`<br>`     forward . 8.8.8.8 9.9.9.9`<br>`  :`<br>` }` |

**Table 10. Troubleshooting issues: installation (continued)**

| Problem | Description | Solution |
|---|---|---|
| | | A script (`ubuntu-set-dns.sh`) is supplied for managing these values.<br>For example, to set the DNS values for the values shown in the previous example:<br>`sudo ./ubuntu-set-dns.sh --server 8.8.8.8 --server 9.9.9.9`<br>**Note:** If you do not use sudo in the command, the script runs but the configuration might be lost if the cluster is restarted.<br>To learn more about the behavior of the script, run the following command:<br>`sudo ./ubuntu-set-dns.sh --help` |
| When running helm install the created pods keep crashing, and the logs contain: ACCESS_REFUSED when trying to connect to RabbitMQ | In some instances, the RabbitMQ password is not automatically setup correctly. | Manually apply the necessary password:<br>`kubectl exec -n <namespace> <release-name>-rabbitmq-0 -- rabbitmqctl change_password user \`<br>`"$(kubectl get secret -n <namespace> <release-name>-rabbitmq -o jsonpath='{.data.rabbitmq-password}' \| base64 --decode)"` |

**Table 11. Troubleshooting issues: server administration**

| Problem | Description | Solution |
|---|---|---|
| When a user is assigned an additional role, the change in the permissions is not observed in the browser. | | You must log out of the session and log in again for the changed role to take effect. |
| You see the following message displayed on HCL OneTest™ Server:<br>`You can't request to join a project that has no owners` | You requested to join an project that no longer has an owner. Orphaned projects occur when the project owners are deleted. This can occur, for example, when the person leaves the organization. | Ask an administrator to take ownership of the project, and then add you as a member. |

**Table 12. Troubleshooting issues: resource monitoring**

| Problem | Description | Solution |
|---|---|---|
| You are not able to add a Prometheus server as a Resource Monitoring source. | The cause might be that you have not installed the Prometheus server at the time of server installation. | Verify that the Prometheus server was installed in Helm at the time of server installation. See Installing the server software on Ubuntu by using k3s on page 50. If not, consult your cluster administrator to get the Prometheus server installed and configured. |

**Table 13. Troubleshooting issues: configuring test runs**

| Problem | Description | Solution |
|---|---|---|
| When you configure a run of a schedule that matches the following conditions: | The cause might be because of the following reasons: | To resolve the problem, select from either of the following methods: |

**Table 13. Troubleshooting issues: configuring test runs (continued)**

| Problem | Description | Solution |
|---|---|---|
| - The schedule has two user groups configured to run on static agents when the schedule was created in HCL OneTest™ Performance V10.1.<br>- One of the user groups is disabled and the asset is committed to the remote repository.<br><br>Both the static agents are displayed as available for the test run in the **Location** tab of the **Execute test asset** dialog box when only one agent that is configured for the user group must be available. | - The schedule was created in HCL OneTest™ Performance V10.1.<br>- The user group that is disabled is not removed or deleted from the test resources.<br>- The agent configured on the disabled user group is already added as an agent to the server project and is available for selection. | - By using HCL OneTest™ Performance V10.1.1.<br><br>Perform the following steps:<br>1. Open the schedule in HCL OneTest™ Performance V10.1.1.<br>2. Save the schedule and the project.<br>3. Commit your test asset to the remote repository.<br>4. Proceed to configure a run for the schedule on HCL OneTest™ Server V10.1.1.<br>- By using HCL OneTest™ Performance V10.1.<br><br>Perform the following steps:<br>1. Select the disabled user group.<br>2. Click **Remove**.<br>3. Save the schedule and the project.<br>4. Commit your test asset to the remote repository.<br>5. Proceed to configure a run for the schedule on HCL OneTest™ Server V10.1.1. |
| You have added a remote repository to your project that contains the test assets or resources of the following types: | This problem occurs if the server extension is not enabled. Although the extension was enabled at the time of installation of HCL OneTest™ Server, | You must verify if the server extension is enabled and running by running the following command:kubectl get pod -n <test-system>, where <test-system> is the namespace that |

Table 13. Troubleshooting issues: configuring test runs (continued)

| Problem | Description | Solution |
|---|---|---|
| • Postman<br>• JMeter<br>• JUnit<br>The test assets or resources are not displayed on the **Execution** page for you to select the asset for a run. | it was disabled subsequently by the server administrator. | you created to install the server software. The server extensions that are running are displayed.<br>If the server extension that you want is not running implying that the server extension is not enabled. You must enable the server extension. Contact the server administrator to enable the server extension. |

Table 14. Troubleshooting issues: test or stub runs

<table>
<tr><th>Problem</th><th>Description</th><th>Solution</th></tr>
<tr>
<td>You encounter any of the following issues:<br>• When many tests are run simultaneously on the default cluster location and you observe the following issues:<br>◦ Out of memory</td>
<td>The issue is seen when any of the following events occur:<br>• Many tests are run in parallel.<br>• The memory that is used by the tests during the test run exceeds the allocated default memory of 1 GB.<br>• The default</td>
<td>To resolve the problem, you can increase the resource allocation for test runs.<br>You can enter arguments in the <b>Additional configuration options</b> field in the <b>Advanced</b> settings panel of the <b>Execute test asset</b> dialog box when configuring a test run.<br><b>Important:</b> The memory settings that you configure for a test run is persisted for the test when ever you run it. You must use this setting judiciously. Configuring all tests for an increased memory limit might affect subsequent test runs or cause other memory issues when tests run simultaneously.<br>You can increase the resource allocation for test runs by using any of the following arguments:
<table>
<tr><th>Requirement</th><th>Configuration option name</th><th>Default value, if no value is set</th><th>An example value</th><th>Result of using the example value</th></tr>
<tr><td>Specifying the memory</td><td><code>init.resource.memory.limit</code></td><td><i>1024Mi</i></td><td><i>2048Mi</i></td><td>Increases the memory limit of the</td></tr>
</table>
</td>
</tr>
</table>

**Table 14. Troubleshooting issues: test or stub runs (continued)**

| Problem | Description | Solution | | | | |
|---|---|---|---|---|---|---|
| | | **Requirement** | **Configuration option name** | **Default value, if no value is set** | **An example value** | **Result of using the example value** |
| errors.<br>◦ Observe that the test runs are slow with a high CPU usage.<br>◦ The Kubernetes pods are getting evicted.<br>• When you run an AFT suite that contains multiple Web UI tests and you observe the following issues: | memory of the container is not adequate for the test run.<br>• Pods are evicted due to low node memory. | limit of the *init container*. | | | | *init container* from the default value to *2048Mi*. |
| | | Configuring a larger memory request for the *init container* to avoid pod eviction. | `init.resource.memory.request` | *64Mi* | *1024Mi* | Increases the initial memory request for the *init container* from the default value to *1024Mi*. |
| | | Specifying the cpu request for the *init container*. | `init.resource.cpu.request` | *50m* | *60m* | Increases the cpu request for the *init container* from the default value *60m*. |
| | | Specifying the memory limit of the container used for the test run. | `resource.memory.limit` | The larger of *3Gi* or *maximum heap size + 1Gi* | *4Gi* | Changes the memory limit of the main container from the default value to *4Gi*. |
| | | Specifying the memory request for the container used by the test run. | `resource.memory.request` | *64Mi* | *1024Mi* | Increases the memory request for the main container from the |

**Table 14. Troubleshooting issues: test or stub runs (continued)**

<table>
<tr><th>Problem</th><th>Description</th><th>Solution</th></tr>
<tr>
<td>◦ Error stating that the browser might not be installed or the browser version is unsupported.<br>◦ Error stating multiple random timeouts or an in-</td>
<td></td>
<td>
<table>
<tr><th>Requirement</th><th>Configuration option name</th><th>Default value, if no value is set</th><th>An example value</th><th>Result of using the example value</th></tr>
<tr><td></td><td></td><td></td><td></td><td>default value to <i>1024Mi</i>.</td></tr>
<tr><td>Specifying the cpu request for the main container used by the test run.</td><td><code>resource.cpu.request</code></td><td><i>50m</i></td><td><i>70m</i></td><td>Increases the cpu request for the main container from the default value to <i>70m</i>.</td></tr>
</table>
In addition, in the <b>JVM Arguments</b> field under the <b>Advanced</b> settings you can set the maximum heap size for the test runtime. For example, adding the JVM argument <b>-Xmx3g</b> sets the maximum heap size to <i>3Gi</i>.
</td>
</tr>
</table>

**Table 14. Troubleshooting issues: test or stub runs (continued)**

| Problem | Description | Solution |
|---|---|---|
| ternal error. | | |
| You are not able to run the Istio stubs from the **Execution** page. | The cause might be that the fully qualified domain name is not specified in the **Host** field for the stub when it was created. | Verify and ensure to add the fully qualified domain name of the server in the **Host** field when the physical transport for the stub is configured in HCL OneTest™ API. |

**Note:** You can refer to the Kubernetes documentation for information about the different units that can be used for resources in the **Additional configuration option** fields.

**Table 15. Troubleshooting issues: test results and reports**

| Problem | Description | Solution |
|---|---|---|
| You are not able to view the Jaeger traces for the tests you ran. | The cause can be as follows:<br>• Jaeger was not pre-installed in OpenShift.<br>• The Jaeger trace is not supported for the particular test that you ran. | Check for any of the following solutions:<br>• Verify that Jaeger was installed in Helm at the time of server installation. See Installing the server software on Ubuntu by using k3s on page 50. If not, consult your administrator to get Jaeger installed and configured.<br>• Verify that the tests you ran are supported for Jaeger traces. See Test results and reports overview on page 360. |

**Table 15. Troubleshooting issues: test results and reports (continued)**

| Problem | Description | Solution |
|---|---|---|
| | | • Verify if you provided the program variables when you configured the test run. See Considerations for using Jaeger traces in reports on page 213. |

# Troubleshooting issues

You can find information about the issues or problems that you might encounter while working with HCL® OneTest™ Data. Details about issues, their causes and the resolutions that you can apply to fix the issues are described.

## Schema designing error or warning messages

The HCL® OneTest™ Data analyzer validates the logical and structural consistency of the data definition of a schema. In case of any inconsistency in the schema structure, the analyzer issues error or warning messages based on the type of the analysis.

- Logical analysis addresses the integrity of the relationships that you define in the schema.
- Structural analysis addresses the integrity of the underlying database.

Warnings indicate a successful analysis and are relatively insignificant. Warning messages provide information about inconsistencies that occurred, when you changed the schema, and are automatically resolved.

Error messages are important. Error messages provide information about errors in the type definitions that you must correct. An error might result in unpredictable results in the mapping of the data definitions.

## Return codes and error messages

You can find information about both error messages and warnings based on structural analysis or logical analysis.

## Schema analysis logic error messages

The following table lists the logic error messages that result from a logical analysis of a schema:

**Return Code**

**Message**

**L100**

COMPONENT neither inherited nor local: `*type name*' of TYPE: `*type name*'

Hint: Look at the super-type's component list. The component is a valid type, but the supertype has a component list that restricts you from using this type as a component. You may have added subtype components before adding supertype components. Either remove all supertype components or add the components in error to the component list of the supertype.

**L101**

This GROUP must have at least one component - TYPE: `*type name*'

Hint: If you want to map this group, add components. If you do not want to map it, make it a category.

**L102**

Circular reference found in COMPONENT # (`*type name*') - TYPE: `*type name*'

Hint: Look at the type of the component in error. It is probably missing an initiator or terminator.

**L103**

Circular reference found in Floating Component type - TYPE: `*type name*'

Hint: Look at the floating component type in error. It is probably missing an initiator or terminator.

**L104**

DELIMITER for TYPE - `*type name*' must have a value

Hint: All delimited groups need a delimiter. Edit delimited group properties to insert the missing delimiter.

**L105**

DELIMITER type neither inherited nor local - TYPE: `*type name*'

Hint: The delimiter name has been entered incorrectly. It should be the name of a local type, or the name of an inherited delimiter, or the name of a type in the sub-tree of the inherited delimiter.

**L106**

Default DELIMITER not specified - TYPE: `*type name*'

Hint: This Type was specified with a FIND option for its delimiter. Please add a default value to define what to use for building outputs.

**L107**

Default DELIMITER not in restriction list - TYPE: `*type name*'

Hint: This delimiter was specified as a syntax item. Add the default value to the restriction list for that syntax item.

**L108**

DELIMITER type is not a SYNTAX ITEM - TYPE: `*type name*'

Hint: Delimiters specified as an item must be specified to be interpreted as SYNTAX to set the value of the delimiter if it appears as a component in a data stream.

**L109**

DELIMITER type has no restriction list - TYPE: `*type name*'

Hint: All syntax items need a restriction list.

**L110**

RELEASE CHARACTER neither inherited nor local - TYPE: `*type name*'

Hint: The release character name has been entered incorrectly. It should be either the name of a local type, the name of an inherited release character, or the name of a type in the sub-tree of the inherited release character.

**L111**

Default RELEASE CHARACTER not specified - TYPE: `*type name*'

Hint: This Type was specified with a syntax item for its release character. Please add a default value to define a value for the release character that has not been encountered in the data.

**L112**

Default RELEASE CHARACTER not in restriction list - TYPE: `*type name*'

Hint: This Type was specified with a syntax item for its release character. Please add the default value to the restriction list of that syntax item.

**L113**

RELEASE CHARACTER type is not a SYNTAX ITEM - TYPE: `*type name*'

Hint: Release characters specified as an item must be specified to be interpreted as SYNTAX to set the value of the release character if it appears as a component in a data stream.

**L114**

RELEASE CHARACTER type has no restriction list - TYPE: `*type name*'

Hint: All syntax items need a restriction list.

**L115**

Floating Component TYPE neither inherited nor local - TYPE: `*type name*'

Hint: The floating component name has been entered incorrectly. It should be either the name of a local type, the name of an inherited floating component, or the name of a type in the sub-tree of the inherited floating component.

**L116**

INITIATOR type neither inherited nor local - TYPE: `type name'

Hint: The initiator name has been entered incorrectly. It should be either the name of a local type, the name of an inherited initiator, or the name of a type in the sub-tree of the inherited initiator.

**L117**

Default INITIATOR not specified - TYPE: `*type name*'

Hint: This Type was specified with a syntax item for its initiator. Add a default value to define a value for that initiator has not been encountered in the data.

**L118**

Default INITIATOR not in restriction list - TYPE: `*type name*'

Hint: This Type was specified with a syntax item for its initiator. Add the default value to the restriction list of that syntax item.

**L119**

INITIATOR type is not a SYNTAX ITEM - TYPE: `*type name*'

Hint: Initiators specified as an item must be specified to be interpreted as SYNTAX to set the value of the initiator if it appears as a component in a data stream.

**L120**

INITIATOR type has no restriction list - TYPE: `*type name*'

Hint: All syntax items need a restriction list.

**L121**

TERMINATOR type neither inherited nor local - TYPE: `*type name*'

Hint: The terminator name has been entered incorrectly. It should be either the name of a local type, the name of an inherited terminator, or the name of a type in the sub-tree of the inherited terminator.

**L122**

Default TERMINATOR not specified - TYPE: `*type name*'

Hint: This Type was specified with a syntax item for its terminator. Add a default value to define a value for that terminator has not been encountered in the data.

**L123**

Default TERMINATOR not in restriction list - TYPE: `*type name*'

Hint: This Type was specified with a syntax item for its terminator. Please add the default value to the restriction list of that syntax item.

**L124**

TERMINATOR type is not a SYNTAX ITEM - TYPE: `*type name*'

Hint: Terminators specified as an item must be specified to be interpreted as SYNTAX to set the value of the terminator if it appears as a component in a data stream.

**L125**

TERMINATOR type has no restriction list - TYPE: `*type name*'

Hint: All syntax items need a restriction list.

**L126**

COMPONENT range minimum (#) greater than range maximum (#) - COMPONENT `*type name*' - TYPE: `*type name*'

Hint: The minimum range must be less than or equal to the maximum range.

**L127**

COMPONENT range minimum (#) less than inherited range minimum(#) - COMPONENT `*type name*' - TYPE: `*type name*'

Hint: The component in error has been inherited. Look at the range of the component with the same name in the super-type's component list.

**L128**

COMPONENT range maximum (#) greater than inherited range maximum(#) - COMPONENT `*type name*' - TYPE: `*type name*'

Hint: The component in error has been inherited. Look at the range of the component with the same name in the super-type's component list.

**L129**

COMPONENT RULE references a COMPONENT later in the component list - `*type name*' - TYPE: `*type name*'

Hint: Move the component rule to the component later in the list.

**L130**

COMPONENT RULE references undefined type - COMPONENT # of TYPE: `*type name*'

Hint: Verify the spelling of the data object name. The rule should reference a data object name of the component or a data object name of a component earlier in the component list.

**L131**

COMPONENT RULE references components of a partitioned group - COMPONENT # of TYPE *type name*

**L132**

Invalid partitioning: TYPE has no SUBTYPES - TYPE: `*type name*'

Hint: Remove the partitioned option from the class window or add sub-types to the Type in error.

**L133**

Type of COMPONENT exists, but its relative name is not valid: `*type name*' in TYPE: `*type name*'

Hint: To get the correct relative name, drag the type you want to use as a component and drop it in the component list of the Type. (Remember to delete the invalid component!)

**L134**

Reference to `ANY' not allowed: COMPONENT number # of TYPE: `*type name*'

Hint: In this case, the Type in error is a group and it is not the root of a partitioned tree. ANY cannot be used if that component needs to be validated. So, if that group is partitioned, you cannot use ANY for a component up to and including the identifier (if there is one). If that group is not partitioned, you cannot use ANY at all.

**L135**

COMPONENT number # cannot reference a CATEGORY in TYPE: `*type name*' (because group is not partitioned)

Hint: In this case, the Type in error is a group and it's not the root of a partitioned tree. A category cannot be used if the component must be validated. So, if that group is partitioned, you cannot use a category for a component up to and including the identifier (if there is one). If that group is not partitioned, you cannot use a category as a component at all.

**L136**

COMPONENT `*type name*' occurs more than once in list - TYPE: `*type name*'

Hint: Each component in the same component list must have a unique type name. Try to make sub-types of the type name in error and replace each non-unique component with one of the new sub-types.

**L137**

COMPONENT `*type name*' and its super-type cannot be in same COMPONENT LIST (in TYPE: `*type name*')

Hint: Try making another sub-type of the super-type and replace the super-type reference with the new sub-type.

**L138**

COMPONENT `*type name*' is same type as delimiter - TYPE: `*type name*'

Hint: A component and a delimiter cannot have the same name. You may need to add sub-types to the type name used in error to resolve this one.

**L139**

COMPONENT `*type name*' is sub-type of delimiter - TYPE: `*type name*'

Hint: This occurs when a syntax item is used to specify a delimiter. You can add another sub-type to the syntax item and replace the delimiter name with the new sub-type name.

**L140**

COMPONENT `*type name*' is super-type of delimiter - TYPE: `*type name*'

Hint: This occurs when a syntax item is used to specify a delimiter. You can add another sub-type to the syntax item and replace the component name with the new sub-type name.

**L141**

COMPONENT `*type name*' is same type as initiator - TYPE: `*type name*'

Hint: A component and an initiator cannot have the same name. You can add sub-types to the type name used in error and replace both the component name and the initiator name.

**L142**

COMPONENT `*type name*' is sub-type of initiator - TYPE: `*type name*'

Hint: This occurs when a syntax item is used to specify an initiator. You can add another sub-type to the syntax item and replace the initiator name with the new sub-type name.

**L143**

COMPONENT `*type name*' is super-type of initiator - TYPE: `*type name*'

Hint: This occurs when a syntax item is used to specify an initiator. You can add another sub-type to the syntax item and replace the component name with the new sub-type name.

**L144**

COMPONENT `*type name*' is same type as terminator - TYPE: `*type name*'

Hint: A component and a terminator cannot have the same name. Try adding sub-types to the type name used in error and replace both the component name and terminator name with one of the new sub-types.

**L145**

COMPONENT `*type name*' is sub-type of terminator - TYPE: `*type name*'

Hint: This occurs when a syntax item is used to specify a terminator. You can add another sub-type to the syntax item and replace the terminator name with the new sub-type name.

**L146**

COMPONENT `*type name*' is super-type of terminator - TYPE: `*type name*'

Hint: This occurs when a syntax item is used to specify a terminator. You can add another sub-type to the syntax item and replace the component name with the new sub-type name.

**L147**

COMPONENT `*type name*' is same type as Floating Component - TYPE: `*type name*'

Hint: Make both the floating component name and the component name sub-types of the floating component.

**L148**

COMPONENT `*type name*' is sub-type of Floating Component - TYPE: `*type name*'

Hint: Make both the floating component name and the component name sub-types of the floating component.

**L149**

COMPONENT `*type name*' is super-type of Floating Component - TYPE: `*type name*'

Hint: Make both the floating component name and the component name sub-types of the floating component.

**L150**

COMPONENT `*type name*' is same type as release character - TYPE: `*type name*'

Hint: This occurs when a syntax item is used to specify a release character. You can add sub-types to the syntax item and replace both the component name and the release character name with the new sub-type names.

**L151**

COMPONENT `*type name*' is sub-type of release character - TYPE: `*type name*'

Hint: This occurs when a syntax item is used to specify a release character. You can add another sub-type to the syntax item and replace the release character name with the new sub-type name.

**L152**

COMPONENT `*type name*' is super-type of release character - TYPE: `*type name*'

Hint: This occurs when a syntax item is used to specify a release character. You can add sub-types to the syntax item and replace the component name with the new sub-type name.

**L153**

DELIMITER `*type name*' is same type as initiator - TYPE: `*type name*'

Hint: A delimiter and an initiator cannot have the same name. You may need to add sub-types to the type name used in error and replace both the delimiter and initiator names to refer to the new sub-types.

**L154**

DELIMITER `*type name*' is sub-type of initiator - TYPE: `*type name*'

Hint: This occurs when a syntax item is used to specify both an initiator and a delimiter. You can add another sub-type to the syntax item and replace the initiator name with the new sub-type name.

**L155**

DELIMITER `*type name*' is super-type of initiator - TYPE: `*type name*'

Hint: This occurs when a syntax item is used to specify both an initiator and a delimiter. You can add another sub-type to the syntax item and replace the delimiter name with the new sub-type name.

**L156**

DELIMITER `*type name*' is same type as terminator - TYPE: `*type name*'

Hint: A delimiter and a terminator cannot have the same name. You may need to add sub-types to the type name used in error and replace both the delimiter and terminator names to refer to the new sub-types.

**L157**

DELIMITER `*type name*' is sub-type of terminator - TYPE: `*type name*'

Hint: This occurs when a syntax item is used to specify both a delimiter and a terminator. You can add another sub-type to the syntax item and replace the terminator name with the new sub-type name.

**L158**

DELIMITER `*type name*' is super-type of terminator - TYPE: `*type name*'

Hint: This occurs when a syntax item is used to specify both a delimiter and a terminator. You can add another sub-type to the syntax item and replace the delimiter name with the new sub-type name.

**L159**

DELIMITER `*type name*' is same type as release character - TYPE: `*type name*'

Hint: A delimiter and a release character cannot have the same name. You may need to add sub-types to the type name used in error and replace both the delimiter and release character names to refer to the new sub-types.

**L160**

DELIMITER `*type name*' is sub-type of release character - TYPE: `*type name*'

Hint: This occurs when a syntax item is used to specify both a delimiter and a release character. You can add another sub-type to the syntax item and replace the release character name with the new sub-type name.

**L161**

DELIMITER `*type name*' is super-type of release character - TYPE: `*type name*'

Hint: This occurs when a syntax item is used to specify both a delimiter and a release character. You can add another sub-type to the syntax item and replace the delimiter name with the new sub-type name.

**L162**

DELIMITER `*type name*' is same type as Floating Component - TYPE: `*type name*'

Hint: A delimiter and a floating component cannot have the same name. Try adding sub-types to the type name used in error and replace both the delimiter and floating component names to refer to the new sub-types.

**L163**

DELIMITER `*type name*' is sub-type of Floating Component - TYPE: `*type name*'

Hint: This occurs when a syntax item is used to specify both a delimiter and a floating component. You can add another sub-type to the syntax item and replace the floating component name with the new sub-type name.

**L164**

DELIMITER `*type name*' is super-type of Floating Component - TYPE: `*type name*'

Hint: This occurs when a syntax item is used to specify both a delimiter and a floating component. You can add another sub-type to the syntax item and replace the delimiter name with the new sub-type name.

**L165**

INITIATOR `*type name*' is same type as terminator - TYPE: `*type name*'

Hint: An initiator and a terminator cannot have the same name. You may need to add sub-types to the type name used in error and replace both the initiator and terminator names to refer to the new sub-types.

**L166**

INITIATOR `*type name*' is sub-type of terminator - TYPE: `*type name*'

Hint: This occurs when a syntax item is used to specify both an initiator and a terminator. You can add another sub-type to the syntax item and replace the terminator name with the new sub-type name.

**L167**

INITIATOR `*type name*' is super-type of terminator - TYPE: `*type name*'

Hint: This occurs when a syntax item is used to specify both an initiator and a terminator. You can add another sub-type to the syntax item and replace the initiator name with the new sub-type name.

**L168**

INITIATOR `*type name*' is same type as release character - TYPE: `*type name*'

Hint: This occurs when a syntax item is used to specify both an initiator and a release character. You can add sub-types to the syntax item and replace both the initiator name and the release character name with a new sub-type name.

**L169**

INITIATOR `*type name*' is sub-type of release character - TYPE: `*type name*'

Hint: This occurs when a syntax item is used to specify both an initiator and a release character. You can add another sub-type to the syntax item and replace the release character name with the new sub-type name.

**L170**

INITIATOR `*type name*' is super-type of release character - TYPE: `*type name*'

Hint: This occurs when a syntax item is used to specify both an initiator and a release character. You can add another sub-type to the syntax item and replace the initiator name with the new sub-type name.

**L171**

INITIATOR `*type name*' is same type as Floating Component - TYPE: `*type name*'

Hint: An initiator and a floating component cannot have the same name. Try adding sub-types to the type name used in error and replace both the initiator and floating component names with the new sub-types.

**L172**

INITIATOR `*type name*' is sub-type of Floating Component - TYPE: `*type name*'

Hint: This occurs when a syntax item is used to specify both an initiator and a floating component. You can add another sub-type to the syntax item and replace the floating component name with the new sub-type name.

**L173**

INITIATOR `*type name*' is super-type of Floating Component - TYPE: `*type name*'

Hint: This occurs when a syntax item is used to specify both an initiator and a floating component. You can add another sub-type to the syntax item and replace the initiator name with the new sub-type name.

**L174**

TERMINATOR `*type name*' is same type as release character - TYPE: `*type name*'

Hint: A terminator and a release character cannot have the same name. You may need to add sub-types to the type name used in error and replace both the terminator and release character names with the new sub-types.

**L175**

TERMINATOR `*type name*' is sub-type of release character - TYPE: `*type name*'

Hint: This occurs when a syntax item is used to specify both a terminator and a release character. You can add another sub-type to the syntax item and replace the release character name with the new sub-type name.

**L176**

TERMINATOR `*type name*' is super-type of release character - TYPE: `*type name*'

Hint: This occurs when a syntax item is used to specify both a terminator and a release character. You can add another sub-type to the syntax item and replace the terminator name with the new sub-type name.

**L177**

TERMINATOR `*type name*' is same type as Floating Component - TYPE: `*type name*'

Hint: A terminator and a floating component cannot have the same name. You may need to add sub-types to the type name used in error and replace both the terminator and floating component names with the new sub-types.

**L178**

TERMINATOR `*type name*' is sub-type of Floating Component - TYPE: `*type name*'

Hint: This occurs when a syntax item is used to specify both a terminator and a floating component. You can add another sub-type to the syntax item and replace the floating component name with the new sub-type name.

**L179**

TERMINATOR `*type name*' is super-type of Floating Component - TYPE: `*type name*'

Hint: This occurs when a syntax item is used to specify both a terminator and a floating component. You can add another sub-type to the syntax item and replace the terminator name with the new sub-type name.

**L180**

RELEASE CHARACTER `*type name*' is same type as Floating Component - TYPE: `*type name*'

Hint: A release character and a floating component cannot have the same name. You may need to add sub-types to the type name used in error and replace both the release character and floating component names to refer to the new sub-types.

**L181**

RELEASE CHARACTER `*type name*' is sub-type of Floating Component - TYPE: `*type name*'

Hint: This occurs when a syntax item is used to specify both a release character and a floating component. You can add another sub-type to the syntax item and replace the floating component name with the new sub-type name.

**L182**

RELEASE CHARACTER `*type name*' is super-type of Floating Component - TYPE: `*type name*'

Hint: This occurs when a syntax item is used to specify both a release character and a floating component. You can add another sub-type to the syntax item and replace the release character name with the new sub-type name.

**L183**

COMPONENT NAME ambiguous: `*type name*' in TYPE: `*type name*'

Hint: This type has a component whose relative name can be associated with more than one type in the schema. Rename the conflicting types.

**L184**

RESTRICTION longer than max TYPE size - RESTRICTION # of TYPE: `*type name*'

Hint: The Type in error is an item. Either change the maximum size of the item or remove the restriction.

**L185**

RESTRICTION used in an earlier partition - RESTRICTION # of TYPE: `*type name*'

Hint: Item Partitions must have mutually exclusive restrictions. Remove the restriction from one of the partition restriction lists.

**L186**

Type of COMPONENT does not exist - `*type name*' in TYPE: `*type name*'

Hint: You probably entered an incorrect type name. Try the drag and drop approach to get the correct one.

**L187**

TYPE must be partitioned (since in a partitioned tree and has sub-types) - TYPE: `*type name*'

Hint: All types in a partitioned sub-type must have mutually exclusive data objects. Set the partitioned property for the type in error.

**L188**

TYPE is FIXED, COMPONENT # must have a maximum range value - TYPE *type name*

**L189**

TYPE is FIXED, but COMPONENT # is not fixed - TYPE: `*type name*'

Hint: If the component is not intended to be fixed in size, change the group format for the Type to implicit. If the group format is intended to be fixed, check the component: if that component is an item, make sure it has a Padded To length; if that component is a group, change its type to be of fixed syntax.

**L190**

BINARY text ITEM used as COMPONENT neither FIXED nor SIZED - COMPONENT # of TYPE: `*type name*'

Hint: The size of a binary text item must either have a Padded To length or it must be sized by the previous component.

**L191**

COMPONENT with SIZED attribute is not an UNSIGNED INTEGER ITEM TYPE - COMPONENT # of TYPE: `*type name*'

Hint: A component used to size the component that follows it must be defined as an unsigned integer item type.

**L192**

The last COMPONENT in the COMPONENT LIST may not have a SIZED attribute: TYPE: `*type name*'

Hint: Specify a component to follow the one with the sized attribute.

**L193**

Range of COMPONENT # must have a maximum value to indicate how many placeholders are needed for its series in TYPE: `type name'.

Hint: Change the range maximum to a specific value (not "s") if you may re-define the data this way.

**L194**

Cannot distinguish delimiter from terminator in TYPE: `type name'.

Hint: Make the range of the last component in the type fixed or make the delimiter of the type different from its terminator.

**L195**

Cannot distinguish delimiter contained in COMPONENT # from terminator of TYPE: `type name'.

Hint: Make that component bound or make that contained delimiter different from the type terminator.

**L196**

Cannot distinguish delimiter of COMPONENT # from delimiter of TYPE: *type name* because COMPONENT # has no placeholder.

Hint: Make that component bound or make that component's delimiter different from the type delimiter.

**L197**

Cannot distinguish delimiter of COMPONENT # from delimiter of TYPE: *type name* because COMPONENT # has no range maximum.

Hint: Make that component bound, or make that component's delimiter different from the type delimiter, or specify a range maximum that has a specific value (not "s") for the last component of COMPONENT #.

**L198**

Cannot distinguish delimiter contained in COMPONENT # from delimiter of TYPE: `type name'.

Hint: Make that contained component bound or make that contained component's delimiter different from the type delimiter.

**L200**

Cannot distinguish delimiter contained in COMPONENT # from delimiter of TYPE: `type name'.

Hint: Either make that contained component bound, make that contained component's delimiter different from the type delimiter, or specify a range maximum that has a specific value (not "s") for the last component of the contained component.

## Logic error and warning messages

The tables in this section list the logic warning messages that result from a logic analysis of a schema.

The following table lists the warnings than can result when a map is compiled.

Warnings should be resolved because they may produce unpredictable results at mapping time.

**Return Code**

**Message**

**L199**

COMPONENT # is not distinguishable from COMPONENT # that may follow in TYPE: `type name'.

Hint: Make the first COMPONENT bound, or look at the tables in "Distinguishable objects" to see how you can define the two component types as distinguishable.

**L201**

Different data objects of COMPONENT # are not distinguishable in TYPE: `type name'.

Hint: See "Distinguishable objects" for more information about distinguishable objects.

**L202**

RESTRICTION list deleted: TYPE is not an ITEM - TYPE: *`type name'*

Hint: Type class was changed from an item to a group or category, so the restriction list was deleted. If this was not your intent, change it back to the way it was.

**L203**

COMPONENT list deleted: TYPE is an ITEM - TYPE: *`type name'*

Hint: Type class was changed from a group to a item or category, so the program deleted its component list. If this was not your intent, change it back to the way it was.

**L204**

DELIMITER deleted: TYPE is not a DELIMITED GROUP - TYPE: *`type name'*

Hint: Group format was changed from delimited to something else, so the program deleted its delimiter. If this was not your intent, change it back to the way it was.

**L205**

COMPONENT RULE deleted: TYPE is a CATEGORY - TYPE: *`type name'* (warning)

Hint: Type class was changed from a group to a category, so its component rule was deleted. If this was not your intent, change it back to the way it was.

**L206**

DELIMITER cannot be found (because first component is not required) - TYPE: *`type name'* (warning)

Hint: If the delimiter is missing, a previously set initiator value or the default value is used.

**L251**

COMPONENT NAME could apply to more than one type:'type name' in TYPE: `type name' (warning).

## Schema analysis structure error messages

The following table lists the structure error messages that result from a structural analysis of a schema:

**Return Code**

**Message**

**S100**

Invalid TYPE Name: SubTYPE # of TYPE: `*type name*'

**S101**

Invalid TYPE chain: SubTYPE # of TYPE: `*type name*'

**S118**

Invalid TYPE NAME WhereUsed chain - TYPE NAME: `*type name*' (error).

**S133**

Referenced COMPONENT not `InUse' - COMPONENT # of TYPE: `*type name*' (error).

**S134**

COMPONENT previously referenced - COMPONENT # (COMP #) of TYPE: `*type name*' (error).

**S149**

Bad Parent COMPONENT Index - COMPONENT `*type name*' - TYPE: `*type name*' (error)

## Schema analysis structure warning messages

The following table lists the structure warning messages that result from a structural analysis of a schema:

**Return Code**

**Message**

**S102**

Unused DELIMITER deleted: `*type name*' (at index #)

**S103**

Invalid DELIMITER pointer deleted - TYPE: `*type name*'

**S104**

Invalid default DELIMITER pointer deleted - TYPE: `*type name*'

**S105**

Invalid RELEASE Char pointer deleted - TYPE: `*type name*'

**S106**

Invalid default RELEASE Char pointer deleted - TYPE: `*type name*'

**S107**

Invalid INITIATOR pointer deleted - TYPE: `*type name*'

**S108**

Invalid default INITIATOR pointer deleted - TYPE: `*type name*'

**S109**

Invalid TERMINATOR pointer deleted - TYPE: `*type name*'

**S110**

Invalid default TERMINATOR pointer deleted - TYPE: `*type name*'

**S111**

Resetting DELIMITER Use Count (was # now #) - DELIMITER: `*type name*'

**S112**

Unused DESCRIPTION deleted: `*type name*' (at index #)

**S113**

Invalid DESCRIPTION pointer deleted - TYPE: `*type name*'

**S114**

Resetting DESCRIPTION Use Count (was # now #) - DESCRIPTION: `*type name*'

**S115**

Invalid Floating Component TYPE pointer deleted - TYPE: `*type name*'

**S116**

Invalid TYPE UsedInComp chain repaired - TYPE: `*type name*'

**S117**

Unused TYPE NAME deleted - TYPE NAME: `*type name*' (at index #)

**S119**

Resetting TYPE NAME use count (was # now #) - TYPE NAME: `*type name*'

**S120**

Repaired empty TYPE NAME WhereUsed chain - TYPE NAME: `*type name*'

**S121**

Unused RESTRICTION NAME deleted: `*type name*' (at index #)

**S122**

Invalid RESTRICTION NAME deleted no DESCRIPTION was available - TYPE: `*type name*' .

**S123**

Invalid RESTRICTION NAME deleted DESCRIPTION was `*type name*' - TYPE: `*type name*'

**S124**

Resetting RESTRICTION NAME Use Count (was # now #) - RESTRICTIONS: `*type name*'

**S125**

Unused RESTRICTION DESCRIPTION deleted: `*type name*' (at index #)

**S126**

Invalid RESTRICTION DESCRIPTION deleted - TYPE: `*type name*'

**S127**

Resetting RESTRICTION DESCRIPTION Use Count (was # now #) - RESTRICTIONS: `*type name*'

**S128**

Unused RULE deleted: `*type name*' (at index #)

**S129**

Invalid RULE pointer deleted - COMPONENT # of TYPE: `*type name*'

**S130**

Resetting RULE Use Count (was # now #) - RULE: `*type name*'

**S131**

Invalid COMPONENT TYPE Description pointer - COMPONENT #

**S132**

COMPONENT marked `InUse' found in Free Chain- COMPONENT #

**S135**

COMPONENT in Free Chain referenced by a TYPE - COMPONENT #

**S136**

COMPONENT recovered and added to Free Chain - COMPONENT #

**S137**

TYPE in Free Chain referenced by another TYPE - TYPE #

**S138**

TYPE recovered and added to Free Chain - TYPE X'%04X'

**S139**

TYPE marked `InUse' but not referenced - TYPE #

**S140**

Referenced TYPE not marked `InUse' - TYPE #

**S141**

TYPE Free Chain not in order: sorting

**S142**

COMPONENT Free Chain not in order: sorting

**S143**

Overlap found in LIST Free Chain

**S144**

Free Chain extends into unallocated region

**S145**

Overlap found in COMPONENT LIST SPACE: list cleared COMPONENTS will be deleted

**S146**

Invalid COMPONENT LIST pointer: all COMPONENTS DELETED - TYPE: `*type name*'

**S147**

Resetting COUNT in COMPONENT LIST: some COMPONENTS may be lost

**S148**

RULE truncated (due to internal error): `*type name*' (at index #)

**S150**

CATEGORY `*type name*' was missing GROUP and/or ITEM attributes

**S151**

GROUP `*type name*' was missing GROUP attributes

**S152**

ITEM `*type name*' was missing ITEM attributes

## Compile-time error messages

When you generate the test data of a schema, you might encounter compile-time error messages. The schemas with compilation errors fail to generate the test data. You can view the error messages on the schema designer.

You can download the schema definition file (.mmc) to troubleshoot the compile-time errors from the following location of the HCL® OneTest™ Data pod:

`/opt/hcl/hip-rest/maps/<genMapPath>`

The following table lists the compile-time errors that can occur when you generate the test data:

**Note:** In the following messages, the characters `x` and `y` are used as variables:

- `x` = the map where the error occurred
- `y` = the output that displays the rule with error

| Compile-time Error Message | Description |
|---|---|
| `Map:x`<br>`Output:y`<br>`Output argument of rule does not match output item sub-class` | This error appears when you enter an invalid regular expression for an item type in **Item Properties** of the **Properties** dialog box.<br>For example, If you select **Number** as **Item subclass** in **Item Properties** of the **Properties** dialog box and enter text as the value of the **Regular Expression** field, then you encounter this error.<br>To resolve this error, enter the valid values for the regular expression. |
| `No such file or directory` | This error appears when you enter an incorrect filename for the **Values file** field while you set up the restrictions for an item type in the **Properties** dialog box.<br>To resolve this error, enter the correct filename that you want HCL® OneTest™ Data to use to import the values of the item type. |
| `Map:x`<br>`Output:y`<br>`Rule references unknown` | This error appears when you apply an invalid function for any item type.<br>To resolve this error, apply the correct function. |
| `Group <component_name> Root does not have any components (or subtypes if it is a partitioned group).` | This error appears when you define an invalid schema.<br>To resolve this error, ensure that you correctly define the type definitions of the schema and its properties. |
| `There is a missing component in group <component_name> Root` | This error appears when the reference of any type definitions exists in the **Structure** dialog box and the definition of that type is missing in the **Dictionary** of the schema.<br>For example, if in the **Structure** dialog box, you have a reference of an item type **Age**, and the definition of this item type is missing in the **Dictionary**, then you encounter this error.<br>To resolve this error, you must delete the reference of the type from the **Structure** dialog box that does not exist in the dictionary. |

# Configuring log files

When you run into issues to trace the log files or the log files end abruptly without the complete information about the action, you must configure settings for the log file in the `configMap` file.

### Before you begin

- You must have installed HCL OneTest™ Server.
- You must have access to the Secure Shell (SSH) console.

### About this task

To troubleshoot the issue, you can configure the following settings:

- The maximum number of log files.
- The maximum file size of each log file.

**Note:** The default value of the maximum number of log files is 5 and the maximum file size of each log file is 10 MB.

1. Open the HCL® OneTest™ Data `configMap` file to edit by using the following command: `kubectl edit configmap -n {namespace} {my-ots}-data-config -o yaml`.

   **Note:** You can use the following command to get the list of the `configMap` files:`kubectl get configmaps`

2. Modify the values of the following parameters that you want to change: **log_file_size** and **max_no_of_log_files**
3. Save the changes.
4. Restart the `onetestdata` pod by using the following command: `kubectl delete pod {my-ots}-data-app-0`.

### Results

You have configured the settings for the log files. You have modified the maximum number of log files and the maximum size of each log file.

# Audit log overview

When execution of a map to generate the test data fails, HCL® OneTest™ Data creates an audit log. The audit log records the detailed information about the execution of the failed job.

You can download the audit log of the failed job from the **Jobs** page by clicking the **Download Log File** button.

## Audit log contents

The audit log provides an execution summary of the following information:

| ExecutionSummary Fields | Description |
|---|---|
| `CommandLine` | Specifies the name and the location of the compiled map. |
| `Elapsedsec` | Specifies the total time spent on the map execution. The time spent is presented in seconds only (rounded). For example, `ElapsedSec="25"`. |
| `mapreturn` | Returns a code to specify the result of map execution. For example, `mapreturn="0"`. |
| `MapStatus` | Specifies the status of the map executed. |
| `Message` | Notifies the map execution result. |
| `ObjectsBuilt` | Specifies the number of objects generated based on the data. Each type in a schema represents an object. |
| `ObjectsFound` | Specifies the number of objects identified based on the data read. Each type in a schema represents an object. |
| `SourceReport` | Provides a detailed report about the source adapter. |

**Note:** Some of the execution information is optional, and are not always displayed in the `ExecutionSummary` of the audit log.

### Source data report

The source data report provides the following information:

- Value of the source adapter
- Byte count of the source data
- Return code of the source adapter with a message
- Adapter command line or file path
- Time stamp of the data source

### Sample of an execution summary in an audit log file

The following sample is the execution summary of the audit log file.

```
<ExecutionSummary MapStatus="Valid" mapreturn="0" ElapsedSec="0.0373" BurstRestartCount="0"> <Message>Map
completed successfully</Message> <CommandLine>'install_dir\examples\CallsSummary.mmc'</CommandLine>
<ObjectsFound>18</ObjectsFound> <ObjectsBuilt>12</ObjectsBuilt> <SourceReport card="1" adapter="File"
bytes="52" adapterreturn="0"> <Message>Data read successfully</Message> <Settings>install_dir\examples
\stores.txt</Settings> <TimeStamp>18:10:04 December 26, 2019</TimeStamp> </SourceReport> <SourceReport card="2"
adapter="File" bytes="69" adapterreturn="0"> <Message>Data read successfully</Message> <Settings>install_dir
```

```
\examples\CALLS.TXT</Settings> <TimeStamp>18:10:04 December 26, 2019</TimeStamp> </SourceReport> <TargetReport
card="1" adapter="File" bytes="119" adapterreturn="0"> <Message>Data written successfully</Message>
<Settings>install_dir\examples\summary.txt</Settings> <TimeStamp>10:14:34 Dec 27, 2019</TimeStamp>
</TargetReport> <WorkArea type="File"> <inputarea card="1" Path="install_dir\examples\CallsSummary.I01"
TimeStamp="10:14:34 Dec 27, 2019" bytes="65695"/> <inputarea card="2" Path="install_dir\examples
\CallsSummary.I02" TimeStamp="10:14:34 Dec 27, 2019" bytes="65695"/> <outputarea card="1" Path="install_dir
\examples\CallsSummary.O01" TimeStamp="10:14:34 Dec 27, 2019" bytes="65695"/> </WorkArea> </ExecutionSummary>
```

## Disabling audit logs

As a default configuration, HCL® OneTest™ Data generates audit logs for each failed test data generation job. However, you might want to disable the audit logs when you no longer require them. In such case, you must change the default configuration.

### Before you begin

- You must have cluster-admin permissions.
- You must have the IP address of the computer where HCL OneTest™ Server is installed.

### About this task

You must set the value of the **AUDIT_ENABLED** property in the `configmap` file to disable the audit log for all the failed test data generation jobs.

1. Log in to the SSH console of HCL OneTest™ Server.
2. Run the following command to edit the configuration file:

   ```
   kubectl edit configmap -n test-system {my-ots}-data-config -o yaml
   ```

3. Search for the **AUDIT_ENABLED** property in the configuration file, and then set the value as *false*.

   **Note:** The default value of **AUDIT_ENABLED** is *true*.

4. Save your changes, and then exit from the configuration file.
5. Run the following command to delete and restart the *<onetest data>* pod:

   ```
   kubectl delete pod {my-ots}-data-app-0
   ```

### Results

You have successfully disabled the audit logs for all the failed test data generation jobs.

## Commands used in HCL OneTest Data

When you want to view the list of HCL® OneTest™ Data pods and to manage each of these pods, you can run the kubectl or oc commands on an SSH console.

The following table provides you the list of commands that you can use to manage the pods on the Kubernetes environment (k3s):

| Commands | Purpose | Examples |
|---|---|---|
| `kubectl get pods` | Shows the list of all pods with the status of each pod. | - |
| `kubectl get configmaps` | Shows the list of all configuration files. | - |
| `kubectl exec -it <podname> bash` | Helps to access the pod. | `kubectl exec -it {my-ots}-data-app-0 bash` |
| `kubectl edit configmap -n <namespace> <configmapName> -o yaml` | Edits the configuration file. | `kubectl edit configmap -n test-system {my-ots}-data-config -o yaml` |
| `kubectl delete pod <podname>` | Deletes and restart the pod. | `kubectl delete pod {my-ots}-data-app-0` |
| `kubectl logs <podname>` | Shows the logs of any specific pod. | `kubectl logs {my-ots}-data-app-0` |

# 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 majority of communications are sent over TLS to port 443 (see Ports, protocols, and services on page dlxx). During the installation, an X.509 certificate is generated for the user provided DNS name, which is used to connect to the server. This certificate is self-signed and hence untrusted by other applications.

This self-signed certificate must be replaced by a certificate signed by a certificate authority trusted by your organization. For more information, see X.509 Certificate User Authentication in the Keycloak documentation.

For information about how the self-signed certificate was created, see the `ssl.sh` file in the `<install-directory>/prepare/` directory.

For information about importing a certificate authority trusted by your organization, see Importing Certificate Authority into a browser.

## Ports, protocols, and services

TCP port 443 is used by the majority of communications with the server.

The port 7085 is the default port for communications with agents registered with HCL OneTest™ Server.

The ports starting from 7085, are used in pairs such as 7085 and 7086, and are allotted for the Schedule that is executed first. The next Schedule is allotted the next pair (7087,7088), and so on for the Schedules that are running simultaneously.

You must open the required ports in pairs for each of the Schedules that you want to run simultaneously.

## Customizing your security settings

You can customize your security settings through user registration.

### User registration

By default, users can sign up themselves with the server. In some environments, this self sign-up might be undesirable. It can be changed by switching off user registration. For more information, see User Registration in the Keycloak documentation.

By default, user email addresses are not verified. This verification must be enabled in production environments. For more information, see Email settings on page 69.

## Setting up user roles and access

You can manage user roles and access through single sign on (SSO) and administration only accounts.

### Single sign-on

By default, Keycloak manages users and passwords locally. In production environments, it is normally appropriate to use single sign-on. For more information, see LDAP user administration on page 70.

### Administration only accounts

Users in the Administrator group can manage the team space where they can configure licenses and notifications and add a repository to store the System model. For more information, see Team Space overview on page 91.

Users in the Administrator group can also discover all projects stored on the server (including private ones) and assign themselves and others roles in those projects.

For this reason, users who use the server to perform both administration and non-administration tasks must have two different accounts, one for each purpose. For more information, see Default user administration on page 68.

# 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.

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

HCL® 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:

*HCL*
*330 Potrero Ave.*
*Sunnyvale, CA 94085*
*USA*
*Attention: Office of the General Counsel*

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

*HCL*
*330 Potrero Ave.*
*Sunnyvale, CA 94085*
*USA*
*Attention: Office of the General Counsel*

HCL TECHNOLOGIES LTD. 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. HCL® 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-HCL® 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 HCL® product and use of those websites is at your own risk.

HCL® 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:

*HCL*
*330 Potrero Ave.*
*Sunnyvale, CA 94085*
*USA*
*Attention: Office of the General Counsel*

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 HCL® under terms of the HCL® Customer Agreement, HCL® 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-HCL® products was obtained from the suppliers of those products, their published announcements or other publicly available sources. HCL® has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-HCL® products. Questions on the capabilities of non-HCL® products should be addressed to the suppliers of those products.

Statements regarding the future direction or intent of HCL® 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 HCL®, 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. HCL®, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. HCL® shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work must include a copyright notice as follows:
© (your company name) (year).
Portions of this code are derived from  HCL Ltd. Sample Programs.
© Copyright HCL Ltd. 2000, 2021.

# Trademarks

HCL®, the HCL® logo, and hcl.com® are trademarks or registered trademarks of HCL Technologies Ltd., registered in many jurisdictions worldwide. Other product and service names might be trademarks of HCL® or other companies.

# 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 HCL® 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 HCL®.

## 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 HCL®.

## 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.

HCL® 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 HCL®, 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.

HCL® 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.

# Index