

DevTest Solutions - 9.5

Administering

Date: 25-Jul-2016



This Documentation, which includes embedded help systems and electronically distributed materials, (hereinafter referred to as the "Documentation") is for your informational purposes only and is subject to change or withdrawal by CA at any time. This Documentation is proprietary information of CA and may not be copied, transferred, reproduced, disclosed, modified or duplicated, in whole or in part, without the prior written consent of CA.

If you are a licensed user of the software product(s) addressed in the Documentation, you may print or otherwise make available a reasonable number of copies of the Documentation for internal use by you and your employees in connection with that software, provided that all CA copyright notices and legends are affixed to each reproduced copy.

The right to print or otherwise make available copies of the Documentation is limited to the period during which the applicable license for such software remains in full force and effect. Should the license terminate for any reason, it is your responsibility to certify in writing to CA that all copies and partial copies of the Documentation have been returned to CA or destroyed.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CA PROVIDES THIS DOCUMENTATION "AS IS" WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. IN NO EVENT WILL CA BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, LOST INVESTMENT, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF CA IS EXPRESSLY ADVISED IN ADVANCE OF THE POSSIBILITY OF SUCH LOSS OR DAMAGE.

The use of any software product referenced in the Documentation is governed by the applicable license agreement and such license agreement is not modified in any way by the terms of this notice.

The manufacturer of this Documentation is CA.

Provided with "Restricted Rights." Use, duplication or disclosure by the United States Government is subject to the restrictions set forth in FAR Sections 12.212, 52.227-14, and 52.227-19(c)(1) - (2) and DFARS Section 252.227-7014(b)(3), as applicable, or their successors.

Copyright © 2016 CA. All rights reserved. All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

Table of Contents

General Administration	11
Default Port Numbers	11
DevTest Server Default Port Numbers	11
DevTest Workstation Default Port Numbers	13
Demo Server Default Port Numbers	13
Change Port Numbers for the DevTest Portal	13
Shared Installation Type	14
Directory Structure	15
DevTest Workstation Directories	15
DevTest Server Directories	17
Running Server Components as Services	19
Running DevTest Solutions with Ant and JUnit	20
Run DevTest Tests as JUnit Tests	20
DevTest Solutions Ant Tasks	21
junitlisa Ant Task	21
junitlisareport Ant Task	23
Ant and JUnit Usage Examples	24
Example Complete Ant Build File	24
Example junitlisa Task for Test Case	24
JUnit Usage Examples	24
JUnit 3 Test Cases	25
JUnit 3 Test Suites	25
JUnit 4 Test Cases	25
JUnit 4 Test Suites	26
Integration with CruiseControl	26
More Example Build Files	26
Memory Settings	27
Change Memory Allocation	27
Change the Java heap size on Windows and UNIX	27
Change the Java heap size on OS X	28
Adjust the memory for individual processes	28
Third-Party File Requirements	29
ActiveMQ File Requirements	30
JCAPS File Requirements	30
JMS Messaging File Requirements	30
Oracle OC4J File Requirements	30

RabbitMQ Requirements	30
SAP File Requirements	31
SAP IDoc Virtualization Requirements	31
SAP RFC Virtualization Requirements	31
SonicMQ File Requirements	32
TIBCO File Requirements	32
TIBCO Rendezvous Messaging	32
TIBCO EMS Messaging or TIBCO Direct JMS	33
TIBCO Hawk Metrics	33
WebLogic File Requirements	33
webMethods File Requirements	33
WebSphere MQ File Requirements	34
Enabling Additional Scripting Languages	35

Database Administration 36

External Registry Database Configuration	36
Configure DevTest to Use DB2	37
Configure DevTest to Use MySQL	38
Configure DevTest to Use Oracle	39
Configure DevTest to Use SQL Server	41
External Java Agent Database Configuration	42
Scenario 1: Same Schema for Registry and Agent	42
Scenario 2: Different Schema for Registry and Agent	43
DDL Statements for Creating and Deleting the Schema	43
External Enterprise Dashboard Database Configuration	43
Configure Enterprise Dashboard to Use DB2	44
Configure Enterprise Dashboard to Use MySQL	45
Configure Enterprise Dashboard to Use Oracle	46
Configure Enterprise Dashboard to Use SQL Server	47
Database Maintenance	48
Automatic Reporting Maintenance	48
Performance Summary Calculator	48
Report Cleaner	48
Automatic Deletion of Audit Log Entries	49
Automatic Deletion of Transactions	50
Automatic Deletion of Tickets	50
Internal Database Configuration	51

License Administration 52

Honor-Based Licensing	53
How Licensing, ACLs, and Audit Reports Work Together	54
Usage-Based License Agreement	55
License Activation	55
ACL Activation	55
Honor-based Compliance	56
Continuous Collection of Usage Data by User Type	56
Usage Audit Report	56
DevTest Solutions Usage Audit Report	56
Overview	57
User Type Chart	58
SV for Performance Chart	59
Historical Usage Tab	60
Component by User	60
Count Calculation Example	61
User Types	62
Example of How the User Type Is Determined for Auditing Purposes	63
User Type Inheritance Chart	64
Usage Audit Report FAQs	65
Licensing	66
Account Sharing	66
Reporting	67
Administration	69

Security 71

Using SSL to Secure Communication	71
SSL Certificates	72
Create Your Own Self-Signed Certificate	73
Use SSL with Multiple Certificates	74
Mutual (Two-Way) Authentication	75
SSL/TLS Protocol Configuration	76
Supported SSL/TLS Protocols and the Equivalent Values Expected by the JVM	76
Use the local.properties File to Configure All DevTest Components	76
Use .vmoptions Files to Configure the SSL/TLS Protocols for Each Component	77
Using HTTPS Communication with the invoke APIs	77
(Optional) Generate a New Key Pair and Certificate	77
Configure the Properties Files	79
Using Kerberos Authentication	80
DevTest support for principal + password authentication	81
Sample krb5.conf file	81

Sample krb5.conf file with Active Directory as KDC	81
Sample login.config file	82
Access Control (ACL)	82
Planning Deployment of ACLs	82
ACL Administration	82
Using the Lightweight Directory Access Protocol (LDAP) with DevTest Solutions	83
Authentication	83
Authorization	83
User Sessions	84
ACL and Backward Compatibility for CLIs and APIs That Ran Without Credentials	84
ACL Database	84
ACL and Command Line Tools or APIs	84
Permission Types	85
Boolean	85
Numeric Limit	85
Standard User Types and Standard Roles	86
Standard Permissions	87
User and Role Administration Permission	87
Resource Administration Permission	87
Test/Suite Administration Permission	87
Virtual Services Administration Permission	88
DevTest Server Administration Permission	89
CVS Administration Permission	90
Report Administration Permission	90
Metric/Event Administration	91
CAI Administration Permission	91
DevTest Workstation Permission	92
Standard Users	93
Change Passwords for Standard Users	94
View User Information from DevTest Workstation	95
Manage Users and Roles	95
Manage Users	96
Add and Update Roles	97
User Activity Report	100
Add Users With the adduser Command-Line Utility	101
Configure Authentication Providers for ACL	101
Using a Certificate with LDAP	105
Authorize Users Authenticated by LDAP	106
ACL Configuration Scenarios	107
Authenticate with LDAP/AD and manage user roles in DevTest	107
Authenticate with LDAP/AD and manage user roles with mapped LDAP groups and reject unmapped users	107

Authenticate with LDAP/AD and manage user roles with mapped LDAP groups and allow unmapped users to be assigned a default role	108
Resource Groups	108
Manage Resource Groups	108
Grant Roles to Resource Groups	109
Use Resource Groups to Control Access	110
Resources	110
Manage Resources	110

Logging 112

Main Log Files	112
Demo Server Log Files	113
Logging Properties File	113
Status Messages for Server Components	115
Automatic Thread Dumps	115
Test Step Logger	116
VSE Logging	117
VSE_Matches File	117
VSE.log File	117

Monitoring 118

Use DevTest Portal	118
View the Server Health Monitor	118
Use the ServiceManager	121
Service Manager Options	121
Service Manager Examples	123
Use the Registry Monitor	124
Registry Monitor - Test Tab	124
Registry Monitor - Simulators Tab	125
Registry Monitor - Coordinator Servers Tab	125
Registry Monitor - Virtual Environments Tab	125
Use the Enterprise Dashboard	125
Open the Enterprise Dashboard	126
Enterprise Dashboard Main Window	126
Peak User Counts	126
VSE(s) in Performance Mode	126
Running Registries	126
Enterprise Dashboard Registry Details Window	127
Monitor Usage and VSEs with Performance Mode	129

Peak User Counts	129
VSE(s) in Performance Mode - Peak Count	130
Peak Usage API	131
Reactivate a Registry or Enterprise Dashboard	132
Maintain Registries	133
Export Dashboard Data	134
Export Usage Audit Data	135
Purge Dashboard Data	135
Change Dashboard Language	136

Administering

This section provides administrators with procedures to help them perform regular administrative duties in DevTest Solutions.

- [General Administration \(see page 11\)](#)
- [Database Administration \(see page 36\)](#)
- [License Administration \(see page 52\)](#)
- [Security \(see page 71\)](#)
- [Logging \(see page 112\)](#)
- [Monitoring \(see page 118\)](#)

General Administration

This section contains the following topics that provide information on general administration of DevTest:

- [Default Port Numbers \(see page 11\)](#)
- [Shared Installation Type \(see page 14\)](#)
- [Directory Structure \(see page 15\)](#)
- [Running Server Components as Services \(see page 19\)](#)
- [Running DevTest Solutions with Ant and JUnit \(see page 20\)](#)
- [Memory Settings \(see page 27\)](#)
- [Third-Party File Requirements \(see page 29\)](#)
- [Enabling Additional Scripting Languages \(see page 35\)](#)

Default Port Numbers

This page defines each default port number that the following DevTest components use:

- DevTest Server
- DevTest Workstation
- Demo Server

This page also describes how to change the port numbers for the DevTest Portal.



Note: As with any socket communication, ephemeral ports are consumed as necessary when local endpoints are created.

DevTest Server Default Port Numbers

The following table lists the default port numbers that various components in DevTest Server use. The table also includes the property that sets each port number.

Port	DevTest Component	Property
1505	Embedded web server	lisa.webserver.port
1506	Enterprise Dashboard (embedded web server)	dradis.webserver.port
1507	DevTest Portal	devtest.port
1508	Broker (embedded web server)	broker.webserver.port

1528	Derby database	lisadb.internal.port, lisadb.pool.common.url
1529	Demo server	Not applicable
1530	Enterprise Dashboard Derby db port	dradisdb.internal.port
2003	Enterprise Dashboard network port	lisa.net.15.port
2004	VSE Manager network port	lisa.net.9.port
2005	Test Runner network port	lisa.net.5.port
2006	ServiceManager network port	lisa.net.11.port
2008	Workstation network port	lisa.net.0.port
2009	Broker network port	lisa.pathfinder.broker.port
2010	Registry network port	lisa.net.3.port
2011	Coordinator network port	lisa.net.2.port
2012	JUnit exec network port	lisa.net.4.port
2013	VSE network port	lisa.net.8.port
2014	Simulator network port	lisa.net.1.port
2999	JDBC simulation driver	Not applicable
3128	HTTP proxy	laf.httpproxy.port
3997	LPAR agent	lisa.mainframe.bridge.server.port
61617	LPAR agent	lisa.mainframe.bridge.port

If more than one simulator is created on the same computer, the port numbers for the new simulators are increased from 2014. For example, three more simulators would have the port numbers 2015, 2016, and 2017.

Typically, there is one coordinator for each lab. However, if you create more coordinators, then you define the port on the command line. For example:

```
CoordinatorServer - -name=tcp://hostname:2468/Coordinator2
```

The **lisa.properties** file defines most of the default port numbers.

If you want to change one or more port numbers after installation, add the property to the **local.properties** file and set the new value. Do not update the **lisa.properties** file.

The only time **site.properties** needs to be updated because of a port update is when the Enterprise Dashboard port changes. In itself, that is configured in **local.properties** like everything else. However, the registries must be configured to talk to the Enterprise Dashboard through **devtest.enterprisedashboard.host** and **devtest.enterprisedashboard.port**, which should be set in **site.properties**.

You can change the port number for the JDBC simulation driver by editing the **Virtual JDBC Listener** step in DevTest Workstation.

DevTest Workstation Default Port Numbers

The default port numbers in a DevTest Workstation installation are a subset of the default port numbers in DevTest Server.

Port	DevTest Component	Property
1505	Embedded web server	<code>lisa.webserver.port</code>
2004	VSE Manager	<code>lisa.net.9.port</code>
2005	Test Runner	<code>lisa.net.5.port</code>
2006	ServiceManager	<code>lisa.net.11.port</code>
2009	Broker	<code>lisa.pathfinder.broker.port</code>
2999	JDBC simulation driver	Not applicable
3128	HTTP proxy	<code>laf.httpproxy.port</code>

Demo Server Default Port Numbers

The following table lists the default port numbers that the demo server uses.

Port	DevTest Component
1098	JNDI
1099	JNDI
1528	Derby database
8080	HTTP

The following files define the default port numbers:

- `lisa-demo-server/jboss/server/default/conf/jboss-service.xml`
- `lisa-demo-server/jboss/server/default/deploy/jboss-web.deployer/server.xml`

If you want to change one or more port numbers after installation, you can update these files directly.

Change Port Numbers for the DevTest Portal

To change the port number of the DevTest Portal, you must configure both of the following property files:

- `local.properties`
- `phoenix.properties`

Follow these steps:

1. Go to the **LISA_HOME** directory.
2. If the **LISA_HOME** directory does not contain a **local.properties** file, copy **_local.properties** to **local.properties**.
3. Open the **local.properties** file.
4. Add the following property and set the value to the new port number.

```
# DevTest UI  
devtest.port=port-number
```
5. Save the **local.properties** file.
6. If the **LISA_HOME** directory does not contain a **phoenix.properties** file, copy **_phoenix.properties** to **phoenix.properties**.
7. Open the **phoenix.properties** file.
8. Uncomment the **phoenix.port** property and set the value to the new port number.
9. Save the **phoenix.properties** file.

Shared Installation Type

The shared installation type is designed for environments that have the following characteristics:

- Multiple users share an installation of DevTest from multiple computers.
- Each user has separate data.

In a shared installation, all data and temporary files are stored in user-specified directories. Each user has their own data, but they share a common DevTest installation. With a shared installation, users only need read access to the DevTest programs directory.



Note: If you want to install Component Services and you are an Administrator, use the local install.

If you select the shared installation type in the installer, the installer prompts you to specify the data directory and the temporary files directory. The default location of each directory is the **USER_HOME** directory.

In a shared installation, the **lisa.user.properties** file is added to the **LISA_HOME** directory and the **USER_HOME** directory for the user that is running the installation. This file contains the **lisa.data.dir** and **lisa.tmpdir** properties. You can specify the location of the file by setting the **lisa.user.properties** system property or the **LISA_USER_PROPERTIES** environment variable. If either of the properties have been defined as a system property, the system property definition takes precedence.



Note: Although **lisa.tmpdir** allows you to change the location of where you can store your temporary files, we do not recommend that you change this property to save the temporary files out to an external mount point or external share. If you encounter issues with product instability, and you are using an external share for temp file storage, Support may instruct you to go back to using a local disk for temp file storage for continued support of your environment.

New users on different computers must manually set up their own **lisa.user.properties** file in their **USER_HOME** directory before they attempt to access a shared installation. You can copy the file from the **LISA_HOME** directory, or you can create it manually if you want to use different directories.

If you attempt to start any DevTest component before setting up this property file, the components will not start. An error message appears on the screen and in the log files indicating an error reading the **lisa.user.properties** file.

Directory Structure

This page describes the directory structure of DevTest Workstation and DevTest Server.

- [DevTest Workstation Directories \(see page 15\)](#)
- [DevTest Server Directories \(see page 17\)](#)

This page assumes that the local installation type was selected during the installation of DevTest Solutions. If the [shared installation type \(see page 14\)](#) was selected, the following directories can be in a location other than **LISA_HOME**:

- cvsMonitors
- database
- hotDeploy
- locks
- tmp

DevTest Workstation Directories

The **LISA_HOME** directory for DevTest Workstation contains the following directories:



Note: The **locks** directory must have read/write permissions.

- **.install4j**
Contains the installer.
- **addons**
Contains the DevTest add-ons.
- **agent**
Contains the files for the DevTest Java Agent.
- **bin**
Contains the executable files for DevTest Workstation and other components.
- **defaults**
Contains the audit document and staging documents that are common across all projects. Project-specific staging documents are located in the **Projects** directory.
- **doc**
Contains the license agreement, JavaDocs for the DevTest Java Agent, and JavaDocs for the Software Development Kit (SDK).
- **examples**
A default project containing examples that use the demo server.
- **examples_src**
Examples source files and the Kiosk-related files.
- **hotDeploy**
A directory that DevTest monitors. This file contains Java classes and JAR files. Java classes and JAR files in this directory are on the DevTest classpath. Any new files or directories added to this directory are dynamically added to the DevTest classpath.



Putting a custom extension into hotDeploy still requires rebooting the DevTest component (for example, VSE or DevTest Workstation) before the extension is available for use. The HotDeployClassLoader will load the extension's class files, but the "lisaextension" file is not discovered automatically, so the extension is still unknown to DevTest. The "lisaextension" files are only discovered during start-up.

- **incontainer**
Contains the in-container testing instructions.
- **jre**
Contains the required JRE.
- **lib**
Contains the required JAR files.

- **licenses**
Contains the license files that are required for running DevTest Solutions.
- **locks**
Contains the lock files that are used for inter-process concurrency.
- **Projects**
Contains example projects, and any projects that you create.
- **reports**
Contains the XML-based test reports that DevTest creates.
- **snmp**
Contains the SNMP-related files.
- **tmp**
Contains the logging files that DevTest creates. If you communicate with Support on an issue, you could be asked to send one or more files from this directory. The tmp directory must be 1.5 times the size of the installer.
- **umetrics**
Contains the files that are related to collecting metrics.
- **webserver**
Contains the web server files.

DevTest Server Directories

The **LISA_HOME** directory for DevTest Server contains the following directories:



Note: The **locks** directory must have read/write permissions.

- **.install4j**
Contains the installer.
- **addons**
Contains the DevTest add-ons.
- **agent**
Contains the files for the DevTest Java Agent.
- **bin**
Contains the executable files for the registry, coordinator, simulator, VSE, DevTest Workstation, and other components. This directory also contains the following batch files:
 - startdefservers.bat - starts the default servers.
 - stopdefservers.bat - stops the default servers.

- **cvsMonitors**
Contains the deployed CVS monitors.
- **database**
Contains the DDL files and CAI upgrade scripts for various databases.
- **defaults**
Contains the audit document and staging documents that are common across all projects. Project-specific staging documents are located in the **Projects** directory.
- **doc**
Contains the license agreement, JavaDocs for the DevTest Java Agent, and JavaDocs for the Software Development Kit (SDK).
- **examples**
A default project containing examples that use the demo server.
- **examples_src**
Examples source files and the Kiosk-related files.
- **hotDeploy**
A directory that DevTest monitors. This file contains Java classes and JAR files. Java classes and JAR files in this directory are on the DevTest classpath. Any new files or directories added to this directory are dynamically added to the DevTest classpath.
- **incontainer**
Contains the in-container testing instructions.
- **jre**
Contains the required JRE.
- **lib**
Contains the required JAR files.
- **licenses**
Contains the license files that are required for running DevTest Solutions.
- **locks**
Contains the lock files that are used for inter-process concurrency.
- **Projects**
Contains example projects, and any projects that you create.
- **reports**
Contains the XML-based test reports that DevTest creates.
- **snmp**
Contains the SNMP-related files.
- **tmp**
Contains the logging files that DevTest creates. If you communicate with Support on an issue, you could be asked to send one or more files from this directory.

- **umetrics**
Contains the files that are related to collecting metrics.
- **webserver**
Contains the web server files.

Running Server Components as Services

If you plan to leave the server components running most of the time, you can use the service executables in the **LISA_HOME\bin** directory.

The default names of the server components are used when started from the command line without a specific name. The **lisa.properties** file is installed with the following properties and default values:

- **lisa.registryName=Registry**
- **lisa.coordName=Coordinator**
- **lisa.simulatorName=Simulator**
- **lisa.vseName=VSE**

If you want to override a default value for any of these properties, specify a new property value in your **local.properties** file.

We recommend that you start and stop the components in the order shown.

To start server components as services:

Enter the following commands:

```
EnterpriseDashboardService start
RegistryService start
PortalService start
BrokerService start
CoordinatorService start
SimulatorService start
VirtualServiceEnvironmentService start
```

To stop server components as services:

Enter the following commands:

```
SimulatorService stop
CoordinatorService stop
VirtualServiceEnvironmentService stop
BrokerService stop
PortalService stop
RegistryService stop
EnterpriseDashboardService stop
```

On UNIX, to configure the services to start automatically, consult your system administrator.

If DevTest Workstation started the report database (Derby), the report database shuts down when DevTest Workstation shuts down.

If the coordinator started the database (Derby), the database does not shut down when coordinator shuts down.

Running DevTest Solutions with Ant and JUnit

You can execute DevTest tests as JUnit tests within the execution path of an Ant build.

This feature provides real automated build and test integration opportunities. You can leverage the flexibility of Ant and the simplicity of JUnit with the power of DevTest Solutions to meet the following automation needs:

- Integration
- Load
- Stress
- Production monitoring

Ant is a Java-based, open source automated software building tool. Ant is extended with support for many third-party tools, including JUnit. For more information, see <http://ant.apache.org/>.

DevTest supports Ant 1.9.x and above with a 1.8 JDK.

JUnit is a Java-based, open source unit-testing framework. JUnit is widely supported by third-party testing tools, including DevTest Solutions. For more information, see <http://www.junit.org> (<http://junit.org/>).

- The [Execute JUnit Test Case/Suite step](https://docops.ca.com/display/DTS95/Execute+JUnit+Test+Case-Suite) (<https://docops.ca.com/display/DTS95/Execute+JUnit+Test+Case-Suite>) can execute a JUnit test. [DevTest Ant Tasks](#) (see page 21) can run any DevTest test or suite, making the DevTest tests look like they JUnit tests.
- The JUnit reports and log files are generated only when the **junitlisa** Ant task is used to run tests.
- The JUnit step enables existing JUnit test cases to be integrated into the testing workflow. You can use DevTest to wrap the JUnit tests and execute them through DevTest.
- The **junitlisa** task enables automating testing without going through the DevTest user interface. The **junitlisa** task generates an HTML report because there is no user interface that it can send results to.

Run DevTest Tests as JUnit Tests

The JUnit step supports JUnit3 and JUnit4 test cases and test suites. This procedure describes the steps that are required to execute DevTest tests as JUnit tests and have them report as native JUnit tests.

Follow these steps:

1. Make sure both Ant and JUnit are available on your PC and ANT_HOME\bin is set in your PATH.
2. Copy **junit.jar** from your JUnit installation into ANT_HOME\lib.
3. Define the system property **LISA_HOME** and set its value to the DevTest install directory.
4. Use the **junitlisa** task to execute DevTest tests as JUnit tests.
5. (Optional) Use the **junitlisareport** task to create HTML-based reports from the JUnit XML output.

Logging output is written to a **junitlisa_log.log** file in the **user.home\lisatmp** directory.

The logging level that is used is the same level that is set in **LISA_HOME\logging.properties**.

To change the logging level:

Edit the first line of this file, from:

```
log4j.rootCategory=INFO,A1
```

to

```
log4j.rootCategory=DEBUG,A1
```

The Standard JUnit output is available at **user.home\lisatmp\junit\index.html**.

DevTest Solutions Ant Tasks

This section describes DevTest Ant tasks.

- [junitlisa Ant Task \(see page 21\)](#)
- [junitlisareport Ant Task \(see page 23\)](#)

junitlisa Ant Task

The **junitlisa** task is a "drop in" replacement for the JUnit task available with Ant, but it executes DevTest tests instead of JUnit tests. Most continuous build systems recognize the XML output files and integrate the build dashboard with the test results.

This task is a direct subclass of the JUnit task. Therefore, the junitlisa task has the same attributes and nested elements as the JUnit task. However, be aware of the following differences in behavior:

- You cannot set the **fork** attribute to false.
- You cannot add nested **test** elements. Use the **test** attribute instead.
- You cannot add nested **batchtest** elements. Use the **suite** attribute instead.
- An implied classpath consisting of **LISA_HOME\bin*.jar**, **LISA_HOME\lib*.jar**, ***.zip**, and **LISA_HOME\lib\endorsed*.jar** is added.

- An implied **java.endorsed.dirs** system property pointing to **LISA_HOME\lib\endorsed** is added.
- If no formatter is specified, then a default formatter of type **xml** is added.
- The **printsummary** attribute is defaulted to true.
- The **maxmemory** attribute is defaulted to 1024m.
- The **showoutput** attribute is defaulted to true.

In addition to the attributes inherited from the JUnit task, the **junitlisa** task has the following attributes:

- **suite**
The file name of a suite document.
Example: suite="AllTestsSuite.ste"
- **test**
The file name of a test case.
Example: test="multi-tier-combo.tst"
- **stagingDoc**
The file name of a staging document.
Example: stagingDoc="Run1User1Cycle.stg"
- **config**
A named internal configuration set or a file name.
Example: config="project.config"
- **outfile**
The file name that is used to write reporting data. If the value does not comply with the standard naming scheme for junitlisareport, specify a fully configured junitreport task instead.
Example: outfile="report"
- **registry**
A pointer to the registry to use when you want to stage the test cases remotely.
Example: registry="<tcp://testbox:2010/Registry>"
- **preview**
Enables you to write out the name and description of each test case, without executing the test cases.
Example: preview="true"
- **user**
Specifies the user name for ACL.
Example: user="admin"
- **password**
Specifies the password for ACL.
Example: password="admin"

- **mar**
The file name of a MAR document.
Example: `mar="example.mar"`
- **mari**
The file name of a MAR info document.
Example: `mari="example.mari"`

The **junitlisa** task includes a nested element named **lisatest**. This element has the following attributes:

- **suite**
The file name of a suite document.
Example: `suite="AllTestsSuite.ste"`
- **test**
The file name of a test case.
Example: `test="multi-tier-combo.tst"`
- **stagingDoc**
The file name of a staging document.
Example: `stagingDoc="Run1User1Cycle.stg"`
- **mar**
The file name of a MAR document.
Example: `mar="example.mar"`
- **mari**
The file name of a MAR info document.
Example: `mari="example.mari"`

The attribute values can use curly braces, which are resolved in the usual way.

You are required to specify at least one test or suite. You can specify a test or suite in a **lisatest** nested element or in the **test** or **suite** attribute. You can specify multiple tests and suites by adding more **lisatest** elements. The tests and suites are executed in the order in which they appear in the XML.

When you run a single test with the **test** attribute, the test has the following default behavior:

- Staged with a single vuser.
- Run once.
- 100 percent think time.

To change this default behavior, wrap the test in a suite and specify an alternative staging document.

junitlisareport Ant Task

You can produce HTML-based reports with the **junitlisareport** task or the regular **junitreport** task.

The **junitlisareport** task is a subclass of the regular **junitreport** element, except that sensible defaults are specified. It is equivalent to the following code:

```
<junitreport todir="${testReportDir}">
  <fileset dir="<todir specified in the junitreport tag>">
    <include name="TEST-*.xml"/>
  </fileset>

  <report format="frames" todir="<todir specified in the junitreport tag>">/>
</junitreport>
```

You can specify your own file set and report. Because the task is a direct subclass of **junitreport**, all the attributes and nested elements that **junitreport** has are supported.

We recommend specifying the inherited **toDir** attribute in most cases, although it defaults to the current working directory.

Ant and JUnit Usage Examples

This section provides examples of Ant and JUnit use with DevTest.

Example Complete Ant Build File

The **build.xml** file in the **LISA_HOME\examples** directory is a complete Ant build file.

The build file contains two targets:

- The **lisaTests** target runs a suite as JUnit tests. The suite is specified with the **lisatest** nested element.
- The **oneTest** target runs a test case as a JUnit test. The test case is specified with the **test** attribute.

The build file assumes that access control (ACL) is enabled. Therefore, the **user** and **password** attributes are included.

Example junitlisa Task for Test Case

The following **junitlisa** task is configured to run a single test case on a remote registry.

```
<junitlisa test="MyTest.tst"
  config="dev"
  registry="tcp://testbox:2010/Registry"
  toDir="${testReportDir}"
  haltOnError="no"
  errorProperty="test.failure">
  <jvmarg value="-DmySystemProp=someValue"/>
</junitlisa>
```

JUnit Usage Examples

This page includes examples of how to use JUnit tests with DevTest.

- [JUnit 3 Test Cases](#) (see page 25)
- [JUnit 3 Test Suites](#) (see page 25)
- [JUnit 4 Test Cases](#) (see page 25)
- [JUnit 4 Test Suites](#) (see page 26)

JUnit 3 Test Cases

For JUnit 3 test cases, follow the JUnit conventions for writing your test case. In particular:

- Your test class should extend **junit.framework.TestCase**.
- Your method names start with **test**.

For example:

```
import junit.framework.TestCase;

public class JUnit3TestCase extends TestCase {

    public void testOneIsOne() {
        assertEquals (1, 1);
    }

    public void testTwoIsThree() {
        assertEquals (2, 3);
    }

}
```

JUnit 3 Test Suites

For JUnit 3 test suites, your suite is not required to extend **junit.framework.TestSuite**. However, it must implement the **suite()** method, with test cases wrapped by **JUnit4TestAdapter**.

For example:

```
import junit.framework.JUnit4TestAdapter;
import junit.framework.TestSuite;

public class JUnit3VanillaTestSuite {

    public static TestSuite suite() {
        TestSuite suite = new TestSuite();
        suite.addTest ( new JUnit4TestAdapter ( MyJUnit3TestCase.class ) );
        return suite;
    }

}
```

JUnit 4 Test Cases

For JUnit 4 test cases, the test methods must have the **@org.junit.Test** annotation on the methods, as required by JUnit 4.

For example:

```
import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class JUnit4TestCase {
```

```

@Test
public void oneIsOne() { assertEquals (1, 1); }

@Test
public void twoIsThree() { assertEquals (2, 3); }

}

```

JUnit 4 Test Suites

To implement a JUnit 4 test suite, add the **@RunWith** and **@Suite.SuiteClasses** annotations to flag the class as a test suite.

For example:

```

import org.junit.runner.RunWith;
import org.junit.runners.Suite;

@RunWith(Suite.class)
@Suite.SuiteClasses ( { JUnit4TestCase.class } )
public class JUnit4VanillaTestSuite { // empty }

```



Note: If loading your JUnit test returns an **IllegalArgumentException** on the JUnit step, confirm that you added **.class** to the end of the class name. You can manually verify the spelling of the classname or use the classpath browser to locate the class.

Integration with CruiseControl

If you are using CruiseControl to provide some continuous integration, you can configure it to include test failures in its Control Panel.

In the JUnit Ant task that is shown in [Ant and JUnit Usage Examples \(see page 24\)](#), **\${testReportDir}** is defined as **\${LISA_HOME}\bin\lisa-core.jar**.

The following sample shows a typical **CruiseControl config.xml**:

```

<project name="myproj " buildafterfailed="false">
  <log>
    <merge dir="/home/cruise/build"/>
    <merge dir="/path/to/lisajunit/output"/>
  </log>
  .
  .
  .
</project>

```

More Example Build Files

In addition to the **build.xml** file described in [Ant and JUnit Usage Examples \(see page 24\)](#), the **LISA_HOME\examples** directory contains the following example files for automating DevTest projects:

- **automated-build.xml**: The master build file that you place at the top of the project hierarchy.
- **lisa-project-build.xml**: The build file that you place in each project. The file name must be changed to **build.xml**.
- **common.xml**: The file that you place in each subdirectory between the root and the project.
- **common-macros.xml**: This file contains macros for performing various tasks with DevTest Server components. For example, you can start the registry, run a suite, or deploy a virtual service. This file is not a standalone build file and is meant to be incorporated into existing Ant-based frameworks.

For more information, see the comments in each file.

Memory Settings

On Windows operating systems, the default memory limit for DevTest Workstation is 512 MB (-Xmx512m).

For a Windows system, it is recommended that DevTest Server is on Windows 64-bit to leverage more memory when needed.

Change Memory Allocation

The **.vmoptions** files are used to pass more parameters to a Java process to modify the default settings that are used for the JVM. These files let you customize the memory allocation settings for each of the DevTest processes used in the server. When you install DevTest Server, the **LISA_HOME\bin** folder is populated with a **.vmoptions** file with the same name as each executable file.

A full list of JVM parameters is available from the Oracle website under Configuring the Default JVM and Java Arguments.

In CAI, the registry service may require more heap memory to process millions of business transactions. The recommended maximum heap memory setting for the registry is 1 GB per five million transactions.

Change the Java heap size on Windows and UNIX

1. Open the target file with a **.vmoptions** extension.
For example, open **RegistryService.vmoptions**, with the following content:

```
# Enter one VM parameter per line
# For example, to adjust the maximum memory usage to 512 MB, uncomment the following line:
# -Xmx512m
# To include another file, uncomment the following line:
# -include-options [path to other .vmoption file]
```

2. To change the maximum memory to be allocated (Xmx), uncomment the following line:

```
# -Xmx512m
```

3. To change the minimum memory to be allocated (Xms), add the VM argument on another line.

```
-Xms128M
```

The file would then specify the memory allocation range as in the following example.

```
-Xms128M  
-Xmx512M
```

4. Save the text file in the **LISA_HOME\bin** folder.

Change the Java heap size on OS X

Browse to the **LISAWorkstation.app** and edit the **Info.plist** file.

The path is **LisaHome/bin/LISAWorkstation.app/Contents/Info.plist**. You can modify the **Info.plist** file, but your changes only apply to DevTest Workstation.

Adjust the memory for individual processes

You can adjust memory for the following programs:

- Broker
- BrokerService
- CoordinatorServer
- CoordinatorService
- CVSManager
- EnterpriseDashboard
- EnterpriseDashboardService
- MakeMar
- PFCmdLineTool
- Portal
- PortalService
- Registry
- RegistryService
- ServiceImageManager
- ServiceManager

- Simulator
- SimulatorService
- TestRunner
- VirtualServiceEnvironment
- VirtualServiceEnvironmentService
- VSEManager
- Workstation

Some of these programs are available in installations of DevTest Server, but not DevTest Workstation.

To tweak the memory for each process, you can edit the individual script files. For example, you can allow the registry to have 128M but the simulator to have 1024M.

You can also copy these files and edit the copies. For example, you can copy **Registry** to **MyRegistry** and tweak **MyRegistry**.

Third-Party File Requirements

To use DevTest Solutions with various third-party applications, you must make JAR files from the third-party application available to DevTest.

Unless otherwise specified in the following sections, you can use any of these approaches:

- Place the files in the **LISA_HOME\hotDeploy** directory
- Place the files in the **LISA_HOME\lib** directory
- Define the **LISA_POST_CLASSPATH** variable

The following example shows the **LISA_POST_CLASSPATH** variable on a Windows computer. The files in this example are WebSphere MQ files.

```
LISA_POST_CLASSPATH="C:\Program Files\IBM\MQSeries\Java\lib\com.ibm.mq.commonservices.jar;C:\Program Files\IBM\MQSeries\Java\lib\com.ibm.mq.headers.jar;C:\Program Files\IBM\MQSeries\Java\lib\com.ibm.mq.jar;C:\Program Files\IBM\MQSeries\Java\lib\com.ibm.mq.jmqi.jar;C:\Program Files\IBM\MQSeries\Java\lib\com.ibm.mq.pcf.jar;C:\Program Files\IBM\MQSeries\Java\lib\com.ibm.mqjms.jar;C:\Program Files\IBM\MQSeries\Java\lib\connector.jar;C:\Program Files\IBM\MQSeries\Java\lib\dhbcore.jar"
```

CA does *not* provide the third-party files listed in this page.

The JAR files must support Java 8.

This page assumes that the local installation type was selected during the installation of DevTest Solutions. If the shared installation type was selected, the **hotDeploy** directory can be in a location other than **LISA_HOME**.

ActiveMQ File Requirements

The **activemq-all.jar** file is required.

JCAPS File Requirements

If you are using the JCAPS Messaging (Native) step, see the JCAPS documentation for information about the JAR files that you might need.

If you are using the JCAPS Messaging (JNDI) step, the **com.stc.jms.stcjms.jar** file is required. See the JCAPS documentation for information about other JAR files that you might need.

The JAR files are available in the **lib** directory of the JCAPS installation.

JMS Messaging File Requirements

If you are using the JMS Messaging (JNDI) step, see the JMS provider's documentation for information about the JAR files that you might need.

Oracle OC4J File Requirements

The following JAR files are required:

- dms.jar
- oc4j.jar
- oc4jclient.jar

See the OC4J documentation for more information about JAR files that you might need. The JAR files are available in the **lib** directory of the OC4J installation.

RabbitMQ Requirements

Copy the JAR files to the **LISA_HOME\lib** directory or define the **LISA_POST_CLASSPATH** variable. Do not copy the files to the **LISA_HOME\hotDeploy** directory.

The following JAR files are required:

- commons-io-1.2.jar
- rabbitmq-client.jar

SAP File Requirements

The following JAR files are required:

- sapidoc3.0.8/sapidoc3.jar - required for the SAP IDoc steps and virtualization.
- sapjco3.0.9/sapjco3.jar
- sapjco3.0.9/*platform*/ a native library file of the platform, where *platform* is your computer system (osx_64, windows_x86, and others) - required for the SAP RFC step and the SAP IDoc steps and virtualization.

SAP IDoc Virtualization Requirements

1. Create an RFC Destination of type T on both the client and server SAP systems. DevTest uses this RFC Destination to receive IDocs from both systems.
2. Update the outbound partner profile on the client and server SAP system. Update the outbound parameter in the partner profile for the respective IDoc type to send to the port number associated with the RFC Destinations created in Step 1. This allows DevTest to receive IDocs from the client and server SAP systems.
3. Create and import the connection property files in your project under the Data directory.
 - a. Create the RFC Connection (with properties from .jcoServer) for the client RFC Destination created in Step 1.
 - b. Create the System Connection property file (with properties from .jcoDestination) for the client SAP system.
 - c. Create the RFC Connection (with properties from .jcoServer) for the server RFC Destination created in Step 1.
 - d. Create the System Connection property file (with properties from .jcoDestination) for the server SAP system.

Before you begin recording, you must also create and import the connection property files in your project under the **Data** directory. You need these files to start JCo servers to receive IDocs from and forward IDocs to the client and server SAP systems. You need the RFC Connection (with properties from .jcoServer) and System Connection (with properties from .jcoDestination) property files. You need these files to start JCo servers to receive IDocs from and forward IDocs to the client and server SAP systems. Consult the SAP administrator about populating these property files.

SAP RFC Virtualization Requirements

1. Create an RFC Destination of type T on the client SAP system. DevTest uses this RFC Destination to intercept remote function calls that the client system makes.
2. Update the ABAP code that makes the remote function call to use the new destination.

3. Create and import the connection property files in your project under the Data directory. Consult the SAP administrator about populating these property files.
 - a. Create the RFC Connection (with properties from .jcoServer) for the client RFC destination that was created in Step 1. This file MUST NOT specify jco.server.repository_destination.
 - b. Create the Repository Connection property file (with properties from the .jcoDestination) for the system that acts as the RFC repository. This is typically the same as the system on which the RFC was run.
 - c. Create the System Connection property file (with properties from the .jcoDestination) for the system where the RFC runs. If this is the same as the Repository connection, only one properties file is needed.

SonicMQ File Requirements

The following JAR files are required:

- mfcontext.jar
- sonic_Client.jar
- sonic_XA.jar

The JAR files are available in the **lib** directory of the SonicMQ installation.

TIBCO File Requirements

The requirements for TIBCO vary depending on the application.

Copy the JAR files to the **LISA_HOME\lib** directory or define the **LISA_POST_CLASSPATH** variable. Do not copy the files to the **LISA_HOME\hotDeploy** directory.

TIBCO Rendezvous Messaging

The following JAR files are required:

- tibrvj.jar
- tibrvjms.jar
- tibrvjsd.jar
- tibrvjweb.jar
- tibrvnative.jar
- tibrvnativesd.jar

TIBCO Rendezvous .dll files are also required. Copy all .dll files from the TIBCO Rendezvous **bin** directory to the **LISA_HOME\bin** directory. Reference the **LISA_HOME\bin** location in your path environment.

In addition, add the TIBCO Rendezvous **bin** directory to your **PATH** environment variable.

TIBCO EMS Messaging or TIBCO Direct JMS

The following JAR files are required:

- tibcrypt.jar
- tibjms.jar
- tibjmsadmin.jar
- tibjmsapps.jar
- tibrvjms.jar

TIBCO Hawk Metrics

The following JAR files are required:

- console.jar
- talon.jar
- util.jar

TIBCO Rendezvous and/or TIBCO EMS JAR files, depending on which transport TIBCO Hawk is using, also need to be copied to the **LISA_HOME\lib** directory.

WebLogic File Requirements

The **weblogic.jar** file is required. If you are using security or JMX, other JAR files might be required.

See the WebLogic documentation for more information about JAR files that you might need. The JAR files are available in the **lib** directory of the WebLogic installation.

webMethods File Requirements

The following JAR files are required:

- (webMethods Integration Server 8.2) Copy the wm-isclient.jar from your webMethods installation to your DevTest installation_directory\lib\shared directory.
- (webMethods Integration Server 7.1 and later) installation_directory\lib\shared\wm-isclient.jar
- (webMethods Integration Server 7.0 and earlier) installation_directory\lib\client.jar

- wm-enttoolkit.jar
- wmbrokerclient.jar
- wmjmsadmin.jar
- wmjmsclient.jar
- wmjmsnaming.jar

The JAR files are available in the **lib** directory of the webMethods installation.

WebSphere MQ File Requirements

Copy the JAR files to the **LISA_HOME\lib** directory or define the **LISA_POST_CLASSPATH** variable. Do not copy the files to the **LISA_HOME\hotDeploy** directory.



Note: If the operating system is a Japanese version, use the files that are listed for WebSphere MQ 7.

The following JAR files are required for WebSphere MQ 5.2:

- com.ibm.mqjms.jar
- com.ibm.mqbind.jar
- com.ibm.mq.pcf.jar
- com.ibm.mq.jar
- connector.jar

The following JAR files are required for WebSphere MQ 6:

- com.ibm.mq.jar
- com.ibm.mq.pcf.jar
- com.ibm.mqjms.jar
- connector.jar
- dhibcore.jar

The following JAR files are required for WebSphere MQ 7:

- com.ibm.mq.commonservices.jar

- com.ibm.mq.headers.jar
- com.ibm.mq.jar
- com.ibm.mq.jmqi.jar
- com.ibm.mq.pcf.jar
- com.ibm.mqjms.jar
- connector.jar
- dhbcore.jar

The JAR files are available in the **MQ_HOME\java\lib** directory.

Enabling Additional Scripting Languages

To let users use more JSR-223 languages in the Execute Script test step, the Scriptable assertion, the match script editor, and the Scriptable data protocol, you can enable additional languages.

To enable an additional scripting language, put the language's jar file into the **hotdeploy** directory of the remote coordinator, server, or CA Service Virtualization. At next startup, the additional language will be in the **Language** drop-down in DevTest Workstation.

Database Administration

This section contains the following pages describing database administration:

- [External Registry Database Configuration \(see page 36\)](#)
- [External Java Agent Database Configuration \(see page 42\)](#)
- [External Enterprise Dashboard Database Configuration \(see page 43\)](#)
- [Database Maintenance \(see page 48\)](#)
- [Internal Database Configuration \(see page 51\)](#)

External Registry Database Configuration

You can configure the registry to use an external database by editing the **site.properties** file in the **LISA_HOME** directory.



Note: For information about database system requirements, see [System Requirements \(https://docops.ca.com/display/DTS95/System+Requirements\)](https://docops.ca.com/display/DTS95/System+Requirements).

The **site.properties** file includes properties for each of the supported databases:

- IBM DB2
- Derby
- MySQL
- Oracle
- Microsoft SQL Server

If you are running DevTest Server, you do not need to reconfigure each DevTest Workstation installation. The configuration in **site.properties** is propagated to each workstation, VSE, coordinator, simulator server, and any other DevTest component that connects to the registry.

The **LISA_HOME\database\dropSchema** directory contains the following files for the registry database:

- **db2_drop.ddl**
- **derby_drop.ddl**
- **mysql_drop.ddl**
- **oracle_drop.ddl**

- **sqlserver_drop.ddl**

These files can be useful if you need to revert the schema.

Configure DevTest to Use DB2

This page describes how to configure DevTest to use an IBM DB2 database.

The following properties in the **site.properties** file are relevant to database configuration:

```
lisadb.acl.poolName=common
lisadb.broker.poolName=common
lisadb.reporting.poolName=common

lisadb.pool.common.driverClass=com.ibm.db2.jcc.DB2Driver
lisadb.pool.common.url=jdbc:db2://[HOSTNAME]:[PORT]/[DATABASENAME]:
progressiveStreaming=2;
lisadb.pool.common.user=[USER]
lisadb.pool.common.password=[PASSWORD]

lisadb.pool.common.minPoolSize=0
lisadb.pool.common.initialPoolSize=0
lisadb.pool.common.maxPoolSize=10
lisadb.pool.common.acquireIncrement=1
lisadb.pool.common.maxIdleTime=45
lisadb.pool.common.idleConnectionTestPeriod=5
```

To minimize the number of connections to the database, connection pooling is used. The underlying implementation of the pooling functionality is c3p0. For detailed information about the settings, see http://www.mchange.com/projects/c3p0/index.html#configuration_properties. By default, all the components use the **common** connection pool. However, you can define a separate pool for each component or mix and match.

For the reporting database, we recommend at least 10 GB for optimal performance. A database for VSE is required only if you are working with legacy images created in a release earlier than 6.0.

The value of any property that ends with **password** is automatically encrypted at startup.



Note: The remote installations of DevTest Workstation, coordinator, simulator server, VSE, or any other remote DevTest component do not require more configuration. They all receive **site.properties** from the registry when they connect and configure their database access accordingly.

Follow these steps:

1. Ensure that the code page of the DB2 database is 1208.
2. Ensure that the page size of the DB2 database is at least 8 KB.
3. In the **LISA_HOME** directory, locate the **_site.properties** file.
4. Change the file name to **site.properties**.

5. Open the **site.properties** file.
6. Configure the database configuration properties.
You typically update the following properties:
 - `lisadb.pool.common.url`
 - `lisadb.pool.common.user`
 - `lisadb.pool.common.password`
7. The schema is automatically created in the database when the registry starts for the first time. However, if you do not want the DevTest user to have DBA privileges, you can manually create the schema beforehand. The **db2.ddl** file in the **LISA_HOME\database** directory contains SQL statements that can serve as the basis for creating the reporting tables and indexes.
8. Ensure that the **lisadb.internal.enabled** property is commented out or set to false.
9. Save the **site.properties** file.
10. Start the registry.

Configure DevTest to Use MySQL

This page describes how to configure DevTest to use a MySQL database.

The following properties in the **site.properties** file are relevant to database configuration:

```
lisadb.acl.poolName=common
lisadb.broker.poolName=common
lisadb.reporting.poolName=common

lisadb.pool.common.driverClass=com.mysql.jdbc.Driver
lisadb.pool.common.url=jdbc:mysql://DBHOST:DBPORT/DBNAME
lisadb.pool.common.user=database_username
lisadb.pool.common.password=database_password

lisadb.pool.common.minPoolSize=0
lisadb.pool.common.initialPoolSize=0
lisadb.pool.common.maxPoolSize=10
lisadb.pool.common.acquireIncrement=1
lisadb.pool.common.maxIdleTime=45
lisadb.pool.common.idleConnectionTestPeriod=5
```

To minimize the number of connections to the database, connection pooling is used. The underlying implementation of the pooling functionality is c3p0. For detailed information about the settings, see http://www.mchange.com/projects/c3p0/index.html#configuration_properties. By default, all the components use the **common** connection pool. However, you can define a separate pool for each component or mix and match.

The MySQL database must provide collation and characters set supporting UTF-8; double-byte characters are stored in the ACL and reporting tables. The default code page for the database has to be UTF-8; it is not enough only to define your database as UTF-8. If the MySQL database is not UTF-8, then runtime errors occur.

For the reporting database, we recommend at least 10 GB for optimal performance. A database for VSE is required only if you are working with legacy images created in a release earlier than 6.0.

The value of any property that ends with **password** is automatically encrypted at startup.



Note: The remote installations of DevTest Workstation, coordinator, simulator server, VSE, or any other remote DevTest component do not require more configuration. They all receive **site.properties** from the registry when they connect and configure their database access accordingly.

Follow these steps:

1. In the **LISA_HOME** directory, locate the **_site.properties** file.
2. Change the file name to **site.properties**.
3. Open the **site.properties** file.
4. Configure the database configuration properties.
You typically update the following properties:
 - **lisadb.pool.common.url**
 - **lisadb.pool.common.user**
 - **lisadb.pool.common.password**
5. The schema is automatically created in the database when the registry starts for the first time. However, if you do not want the DevTest user to have DBA privileges, you can manually create the schema beforehand. The **mysql.ddl** file in the **LISA_HOME\database** directory contains SQL statements that can serve as the basis for creating the reporting tables and indexes.
6. Ensure that the **lisadb.internal.enabled** property is commented out or set to false.
7. Save the **site.properties** file.
8. Add the MySQL JDBC driver to the **LISA_HOME\lib\shared** directory. The minimum version of the MySQL JDBC driver is 5.1.25.
9. Start the registry.

Configure DevTest to Use Oracle

This page describes how to configure DevTest to use an Oracle database.

The following properties in the **site.properties** file are relevant to database configuration:

```

lisadb.acl.poolName=common
lisadb.broker.poolName=common
lisadb.reporting.poolName=common

lisadb.pool.common.driverClass=oracle.jdbc.driver.OracleDriver
## Select one of the two connection URLs depending on usage of SID or SERVICE
lisadb.pool.common.url=jdbc:oracle:thin:@[HOST]:1521:[SID]
lisadb.pool.common.url=jdbc:oracle:thin:@//[HOST][:PORT]/SERVICE
lisadb.pool.common.user=[USER]
lisadb.pool.common.password=[PASSWORD]

lisadb.pool.common.minPoolSize=0
lisadb.pool.common.initialPoolSize=0
lisadb.pool.common.maxPoolSize=10
lisadb.pool.common.acquireIncrement=1
lisadb.pool.common.maxIdleTime=45
lisadb.pool.common.idleConnectionTestPeriod=5

```

To minimize the number of connections to the database, connection pooling is used. The underlying implementation of the pooling functionality is c3p0. For detailed information about the settings, see http://www.mchange.com/projects/c3p0/index.html#configuration_properties. By default, all the components use the **common** connection pool. However, you can define a separate pool for each component or mix and match.

For the reporting database, we recommend at least 10 GB for optimal performance. A database for VSE is required only if you are working with legacy images created in a release earlier than 6.0.

The value of any property that ends with **password** is automatically encrypted at startup.



Note: The remote installations of DevTest Workstation, coordinator, simulator server, VSE, or any other remote DevTest component do not require more configuration. They all receive **site.properties** from the registry when they connect and configure their database access accordingly.

Follow these steps:

1. Ensure that the character set of the Oracle database supports Unicode. In addition, for the initial creation of the database, the Oracle user must have the **CREATE VIEW** system privilege. Without this privilege, the following error might be generated during installation: **Schema creation failed: ORA-01031: insufficient privileges**.
2. In the **LISA_HOME** directory, locate the **_site.properties** file.
3. Change the file name to **site.properties**.
4. Open the **site.properties** file.
5. Configure the database configuration properties.
You typically update the following properties:
 - lisadb.pool.common.url
 - lisadb.pool.common.user

- `lisadb.pool.common.password`

For the `lisadb.pool.common.url` property, you can use the service name.

6. The schema is automatically created in the database when the registry starts for the first time. However, if you do not want the DevTest user to have DBA privileges, you can manually create the schema beforehand. The **oracle.ddl** file in the **LISA_HOME\database** directory contains SQL statements that can serve as the basis for creating the reporting tables and indexes.
7. Ensure that the **lisadb.internal.enabled** property is commented out or set to false.
8. Save the **site.properties** file.
9. Ensure that **LISA_HOME\lib\shared** contains the JDBC driver. In most cases, **ojdbc7.jar** is the correct driver for Java 1.7. If you are using Java 1.6, then use **ojdbc6.jar**. If you are using Java 1.5, then use **ojdbc5.jar**.
10. Start the registry.

Configure DevTest to Use SQL Server

This page describes how to configure DevTest to use a Microsoft SQL Server database.

The following properties in the **site.properties** file are relevant to database configuration:

```
lisadb.acl.poolName=common
lisadb.broker.poolName=common
lisadb.reporting.poolName=common

lisadb.pool.common.driverClass=com.microsoft.sqlserver.jdbc.SQLServerDriver
lisadb.pool.common.url=jdbc:sqlserver://SERVER:PORT;databaseName=DATABASENAME
lisadb.pool.common.user=database_username
lisadb.pool.common.password=database_password

lisadb.pool.common.minPoolSize=0
lisadb.pool.common.initialPoolSize=0
lisadb.pool.common.maxPoolSize=10
lisadb.pool.common.acquireIncrement=1
lisadb.pool.common.maxIdleTime=45
lisadb.pool.common.idleConnectionTestPeriod=5
```

To minimize the number of connections to the database, connection pooling is used. The underlying implementation of the pooling functionality is c3p0. For detailed information about the settings, see http://www.mchange.com/projects/c3p0/index.html#configuration_properties. By default, all the components use the **common** connection pool. However, you can define a separate pool for each component or mix and match.

For the reporting database, we recommend at least 10 GB for optimal performance. A database for VSE is required only if you are working with legacy images created in a release earlier than 6.0.

The value of any property that ends with **password** is automatically encrypted at startup.



Note: The remote installations of DevTest Workstation, coordinator, simulator server, VSE, or any other remote DevTest component do not require additional configuration. They all receive **site.properties** from the registry when they connect and configure their database access accordingly.

Follow these steps:

1. In the **LISA_HOME** directory, locate the **_site.properties** file.
2. Change the file name to **site.properties**.
3. Open the **site.properties** file.
4. Configure the database configuration properties.
You typically update the following properties:
 - **lisadb.pool.common.url**
 - **lisadb.pool.common.user**
 - **lisadb.pool.common.password**
5. The schema is automatically created in the database when the registry starts for the first time. However, if you do not want the DevTest user to have DBA privileges, you can manually create the schema beforehand. The **sqlserver.ddl** file in the **LISA_HOME\database** directory contains SQL statements that can serve as the basis for creating the reporting tables and indexes.
6. Ensure that the **lisadb.internal.enabled** property is commented out or set to false.
7. Save the **site.properties** file.
8. Start the registry.

External Java Agent Database Configuration

You can configure the DevTest Java Agent to use an external database by editing the **site.properties** file in the **LISA_HOME** directory.

- [Scenario 1: Same Schema for Registry and Agent \(see page 42\)](#)
- [Scenario 2: Different Schema for Registry and Agent \(see page 43\)](#)
- [DDL Statements for Creating and Deleting the Schema \(see page 43\)](#)

Scenario 1: Same Schema for Registry and Agent

If you want the registry and the agent to use the same database schema, follow the instructions in [External Registry Database Configuration \(see page 36\)](#) and the applicable subtopic.

Scenario 2: Different Schema for Registry and Agent

If you want the registry and the agent to use different database schemas, follow the instructions in [External Registry Database Configuration \(see page 36\)](#) and the applicable subtopic with the following modifications.

In the **site.properties** file, change the value of the **lisadb.broker.poolName** property from **common** to a new value. For example:

```
lisadb.broker.poolName=cai
```

When you configure the database configuration properties for your database, use the new pool name. For example:

```
lisadb.pool.cai.driverClass=com.microsoft.sqlserver.jdbc.SQLServerDriver
lisadb.pool.cai.url=jdbc:sqlserver://localhost:1433;databaseName=cai
lisadb.pool.cai.user=sa
lisadb.pool.cai.password=xyz
```

DDL Statements for Creating and Deleting the Schema

If you want to manually create the agent schema, obtain the DDL statements from one of the following files in the **LISA_HOME\database** directory:

- **db2_cai.ddl**
- **derby_cai.ddl**
- **mysql_cai.ddl**
- **oracle_cai.ddl**
- **sqlserver_cai.ddl**

The **LISA_HOME\database\dropSchema** directory contains the **drop_cai.ddl** file. This file can be useful if you need to revert the schema.

External Enterprise Dashboard Database Configuration

You can configure Enterprise Dashboard to use an external database by editing the **dradis.properties** file in the **LISA_HOME** directory.



Note: For information about database system requirements, see [System Requirements \(https://docops.ca.com/display/DTS95/System+Requirements\)](https://docops.ca.com/display/DTS95/System+Requirements).

The **_dradis.properties** file includes properties for each of the supported databases:

- IBM DB2
- Derby
- MySQL
- Oracle
- Microsoft SQL Server

The **LISA_HOME\database\dropSchema** directory contains the following files for the Enterprise Dashboard database:

- **db2_drop_enterprisedashboard.ddl**
- **derby_drop_enterprisedashboard.ddl**
- **mysql_drop_enterprisedashboard.ddl**
- **oracle_drop_enterprisedashboard.ddl**
- **sqlserver_drop_enterprisedashboard.ddl**

These files can be useful if you need to revert the schema.

Configure Enterprise Dashboard to Use DB2

This page describes how to configure Enterprise Dashboard to use an IBM DB2 database.

In the following procedure, you configure the DB2 version of the **dradis.properties** file. The following properties in this file are relevant to the database configuration:

```
dradis.db.driverClass=com.ibm.db2.jcc.DB2Driver
dradis.db.url=jdbc:db2://[HOSTNAME]:[PORT]/[DATABASENAME]
dradis.db.user=[USER]
dradis.db.password=[PASSWORD]
```

The value of any property that ends with *password* is automatically encrypted at startup.

Follow these steps:

1. Ensure that the code page of the DB2 database is 1208.
2. Ensure that the page size of the DB2 database is at least 16 KB.
3. In the **LISA_HOME** directory, locate the **_dradis.properties** file.
4. Change the file name to **dradis.properties**.
5. Open the **dradis.properties** file.

6. Configure the database configuration properties.
You typically update the following properties:

- dradis.db.url
- dradis.db.user
- dradis.db.password

The schema is automatically created in the database when the registry starts for the first time. However, if you do not want the Enterprise Dashboard user to have DBA privileges, you can manually create the schema beforehand. The **db2_enterprisedashboard.ddl** file in the **LISA_HOME\database** directory contains SQL statements that can serve as the basis for creating the Enterprise Dashboard tables and indexes.

7. Set the **dradis.db.internal.enabled** property to false.
8. Start Enterprise Dashboard.

Configure Enterprise Dashboard to Use MySQL

This page describes how to configure Enterprise Dashboard to use a MySQL database.

In the following procedure, you configure the MySQL version of the **dradis.properties** file. The following properties in this file are relevant to the database configuration:

```
dradis.db.driverClass=com.mysql.jdbc.Driver
dradis.db.url=jdbc:mysql://[DBHOST]:[DBPORT]/[DBNAME]
dradis.db.user=[USER]
dradis.db.password=[PASSWORD]
```

The MySQL database must provide collation and characters set supporting UTF-8. The default code page for the database has to be UTF-8; it is not enough only to define your database as UTF-8. If the MySQL database is not UTF-8, then runtime errors occur.

The value of any property that ends with *password* is automatically encrypted at startup.

Follow these steps:

1. In the LISA_HOME directory, locate the _dradis.properties file.
2. Change the file name to **dradis.properties**.
3. Open the **dradis.properties** file.
4. Configure the database configuration properties.
You typically update the following properties:
 - dradis.db.url
 - dradis.db.user
 - dradis.db.password

The schema is automatically created in the database when the registry starts for the first time. However, if you do not want the Enterprise Dashboard user to have DBA privileges, you can manually create the schema beforehand. The **mysql_enterprisedashboard.ddl** file in the **LISA_HOME\database** directory contains SQL statements that can serve as the basis for creating the Enterprise Dashboard tables and indexes.

5. Set the **dradis.db.internal.enabled** property to *false*.
6. Add the MySQL JDBC driver to the **LISA_HOME\lib\dradis** directory. The minimum version of the MySQL JDBC driver is 5.1.25.
7. Start Enterprise Dashboard.

Configure Enterprise Dashboard to Use Oracle

This page describes how to configure Enterprise Dashboard to use an Oracle database.

In the following procedure, you configure the Oracle version of the **dradis.properties** file. The following properties in this file are relevant to the database configuration:

```
dradis.db.driverClass=oracle.jdbc.driver.OracleDriver
dradis.db.url=jdbc:oracle:thin:@[HOST]:1521:[SID]
dradis.db.user=[USER]
dradis.db.password=[PASSWORD]
```

The value of any property that ends with *password* is automatically encrypted at startup.

Follow these steps:

1. Ensure that the character set of the Oracle database supports Unicode. In addition, for the initial creation of the database, the Oracle user must have the **CREATE VIEW** system privilege. Without this privilege, the following error might be generated during installation: **Schema creation failed: ORA-01031: insufficient privileges.**
2. In the **LISA_HOME** directory, locate the **_dradis.properties** file.
3. Change the file name to **dradis.properties**.
4. Open the **dradis.properties** file.
5. Configure the database configuration properties.
You typically update the following properties:
 - dradis.db.url
 - dradis.db.user
 - dradis.db.password

For the **dradis.db.url** property, you can use the service name.

The schema is automatically created in the database when Enterprise Dashboard starts for the first time. However, if you do not want the Enterprise Dashboard user to have DBA privileges,

you can manually create the schema beforehand. The **oracle_enterprisedashboard.ddl** file in the **LISA_HOME\database** directory contains SQL statements that can serve as the basis for creating the Enterprise Dashboard tables and indexes.

6. Set the **dradis.db.internal.enabled** property to *false*.
7. Ensure that **LISA_HOME\lib\dradis** contains the JDBC driver. In most cases, **ojdbc7.jar** is the correct driver for Java 1.7. If you are using Java 1.6, then use **ojdbc6.jar**. If you are using Java 1.5, then use **ojdbc5.jar**.
8. Start Enterprise Dashboard.

Configure Enterprise Dashboard to Use SQL Server

This page describes how to configure Enterprise Dashboard to use a Microsoft SQL Server database.

In the following procedure, you configure the SQL Server version of the **dradis.properties** file. The following properties in this file are relevant to the database configuration:

```
dradis.db.driverClass=com.microsoft.sqlserver.jdbc.SQLServerDriver
dradis.db.url=jdbc:sqlserver://[SERVER]:[PORT];databaseName=[DATABASENAME]
dradis.db.user=[USER]
dradis.db.password=[PASSWORD]
```

The value of any property that ends with *password* is automatically encrypted at startup.

Follow these steps:

1. In the **LISA_HOME** directory, locate the **_dradis.properties** file.
2. Change the file name to **dradis.properties**.
3. Open the **dradis.properties** file.
4. Configure the database configuration properties.
You typically update the following properties:

- dradis.db.url
- dradis.db.user
- dradis.db.password

The schema is automatically created in the database when the registry starts for the first time. However, if you do not want the Enterprise Dashboard user to have DBA privileges, you can manually create the schema beforehand. The **sqlserver_enterprisedashboard.ddl** file in the **LISA_HOME\database** directory contains SQL statements that can serve as the basis for creating the Enterprise Dashboard tables and indexes.

5. Set the **dradis.db.internal.enabled** property to *false*.
6. Start Enterprise Dashboard.

Database Maintenance

This section details processes available for database maintenance.

- [Automatic Reporting Maintenance \(see page 48\)](#)
- [Automatic Deletion of Audit Log Entries \(see page 49\)](#)
- [Automatic Deletion of Transactions \(see page 50\)](#)
- [Automatic Deletion of Tickets \(see page 50\)](#)

Automatic Reporting Maintenance

Two processes that run in the background affect reporting.

- Performance Summary Calculator
- Report Cleaner

Performance Summary Calculator

This process calculates the statistics for a test or suite run. Many of the graphs in the reporting portal depend on this process, so it cannot be disabled. You can vary the scheduling of the performance summary calculator by changing the following properties:

- **rpt.summary.initDelayMin=7**
Determines how many minutes to wait after starting the registry before starting this process.
- **rpt.summary.pulseMin=1**
Determines how long to wait between runs of the process.

Report Cleaner

This process is responsible for deleting "expired" report runs from the database.

- **perfmgr.rvwiz.whatrpt.autoExpire=true**
You can enable or disable the report cleaner by setting this property. If disabled, the report cleaner process does not run, and the remaining properties have no effect.
- **perfmgr.rvwiz.whatrpt.expireTimer=30d**
Sets the expiration value of a suite or test. Once a suite or test is older than this value, it is deleted. The property value is a number followed by a suffix. The following suffixes are valid, with **t** being the default:
 - t=milliseconds
 - s=second
 - m=minutes
 - h=hours

- d=days
- w=weeks

Keep in mind that tests and suites are deleted at approximately the same time of day that they were created. Adding several hours can minimize delete processes running during the day. However, there is no absolute way to ensure the times that a delete process runs.

- **perfmgr.rvwiz.whatrpt.forceCompleteTimer=24h**

To force the completion of a test (in the reporting database). Some suites or tests that fail to finish cleanly are never marked as "finished" in the database. Tests that are not marked "finished" are not deleted by the report cleaner and do not have performance summary calculations performed. Any test that is older than this value is marked as completed in the database. The completed tests are only removed from the database after the expireTimer has also been accounted for. If you have tests that run longer than 24 hours by default, increase the value for this process.

You can vary the scheduling of the report cleaner by changing the following properties:

- **rpt.cleaner.initDelayMin=10**

Determines how many minutes to wait after starting the registry before starting this process.

- **rpt.cleaner.pulseMin=60**

Determines how long to wait between runs of the process.

Automatic Deletion of Audit Log Entries

The access control (ACL) feature stores the audit log in the DevTest database. On a periodic basis, the registry runs a process to delete old audit log entries from the database.

The following properties control this behavior:

- **lisa.acl.audit.logs.delete.frequency**

Specifies how often to run the automatic deletion process. The default value is **1d**, which means that the process runs once a day. The valid units of time are d, h, m, and s (for days, hours, minutes, and seconds).

- **lisa.acl.audit.logs.delete.age**

Specifies the minimum age of audit log entries that are deleted by the automatic deletion process. The default value is 30d, which means that entries are considered to be old after 30 days. The valid units of time are d, h, m, and s (for days, hours, minutes, and seconds).

The default value of each property is located in the **lisa.properties** file. If you want to change the default value, add the property to the **local.properties** file.

Automatic Deletion of Transactions

You can configure CA Continuous Application Insight to delete old transactions automatically from the DevTest database.

By default, the automatic deletion of transactions is enabled.

The following broker properties are available:

- **Enable cleaner**
Controls whether the automatic deletion of transactions is enabled.
- **Cleanup frequency**
Specifies how often the cleaner process runs. The value is in number of minutes.
- **Maximum age**
Specifies the age of transactions that the cleaner process deletes. The value is in number of minutes.

You can configure these properties from the [Agents window \(https://docops.ca.com/display/DTS95/Agent+Configuration\)](https://docops.ca.com/display/DTS95/Agent+Configuration) of the DevTest Portal. The properties appear in the **Settings** tab.

Automatic Deletion of Tickets

You can configure CA Continuous Application Insight to delete old [tickets \(https://docops.ca.com/display/DTS95/Create+and+Manage+Tickets\)](https://docops.ca.com/display/DTS95/Create+and+Manage+Tickets) automatically from the DevTest database.

By default, the automatic deletion of tickets is enabled.

The following properties are available:

- **Enable cleaner**
Controls whether the automatic deletion of tickets is enabled.
- **Cleanup frequency**
Specifies how often the cleaner process runs. The value is in number of minutes.
- **Maximum age**
Specifies the age of tickets that the cleaner process deletes. The value is in number of minutes.

You can configure these properties from the [Agents window \(https://docops.ca.com/display/DTS95/Agent+Configuration\)](https://docops.ca.com/display/DTS95/Agent+Configuration) of the DevTest Portal. The properties appear in the **Settings** tab.

Internal Database Configuration

Apache Derby is provided as an out of the box database so that you have a functioning system as you prepare to move to the enterprise database solution for your organization. Derby is not supported as an enterprise database solution. This out of the box database is located in the **LISA_HOME\database\lisa.db** directory. To avoid manual data migration issues, we recommend that you configure enterprise databases as a post-installation task.

The following components interact with a database:

- Reporting
- Access control (ACL)
- Java Agent broker
- VSE
- Enterprise Dashboard

Use your site-specific procedures to back up the enterprise databases you use for DevTest Solutions.



Note: For information about database system requirements, see [Installing \(https://docops.ca.com/display/DTS95/Installing\)](https://docops.ca.com/display/DTS95/Installing).

License Administration

The DevTest 8.0 and later license agreement is based on the maximum allowed concurrent user sessions by user type. Contact your account team if you have any questions about your specific licensing agreement.

The license is file-based, where there is one file for the enterprise. This file activates DevTest Solutions at the initial startup following installation.

The Local License Server (LLS) and Internet Based License Server are no longer supported for DevTest Solutions 8.0 and later.

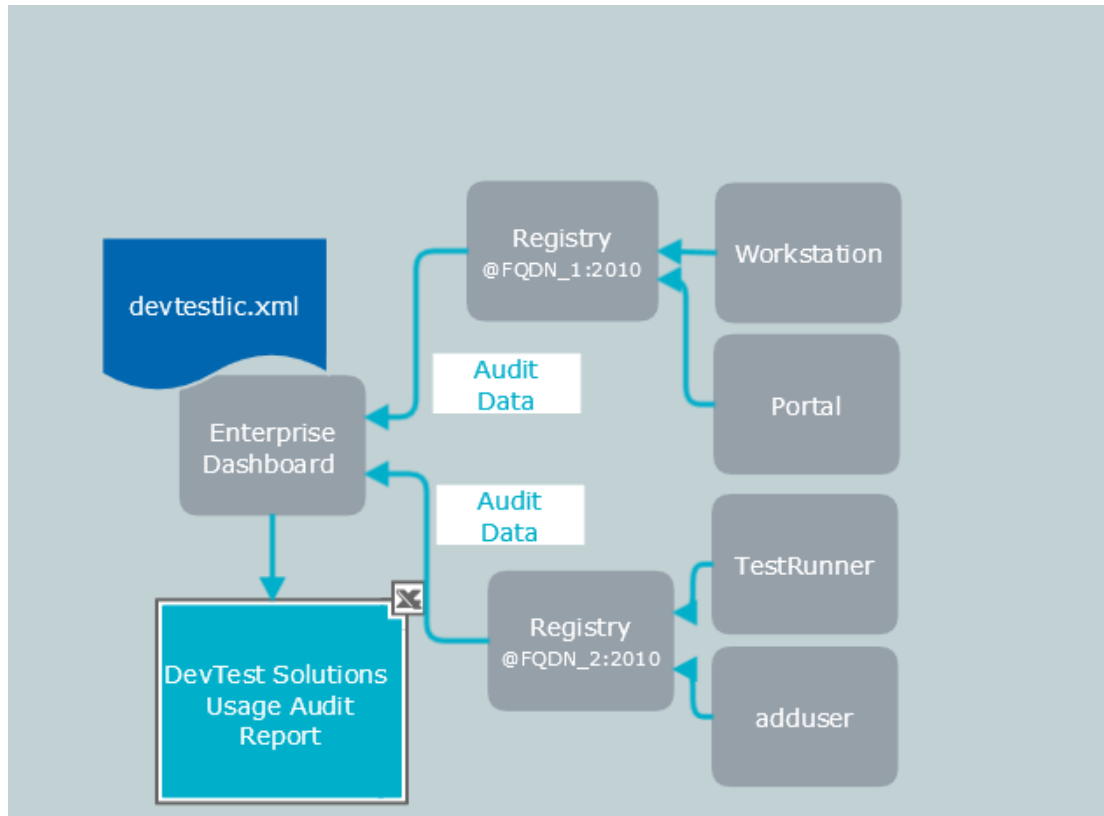
Concurrent usage data by user type is automatically collected.

The Enterprise Dashboard produces licensing reporting.

A Usage Audit Report, which reports maximum concurrent usage by user type, is a tool that helps you assess compliance with the license agreement.

For more information, see [How Licensing, ACLs, and Audit Reports Work Together \(see page 54\)](#).

The following diagram shows the activation of DevTest Solutions by a license file that is stored with the Enterprise Dashboard. The registries collect audit data from the UIs and CLIs that users log in to, including the Workstation, the Portal, and command-line utilities such as TestRunner and adduser. The registries forward the audit data to the Enterprise Dashboard. Administrators can generate a DevTest Solutions Usage Audit Report to verify compliance with the license agreement.



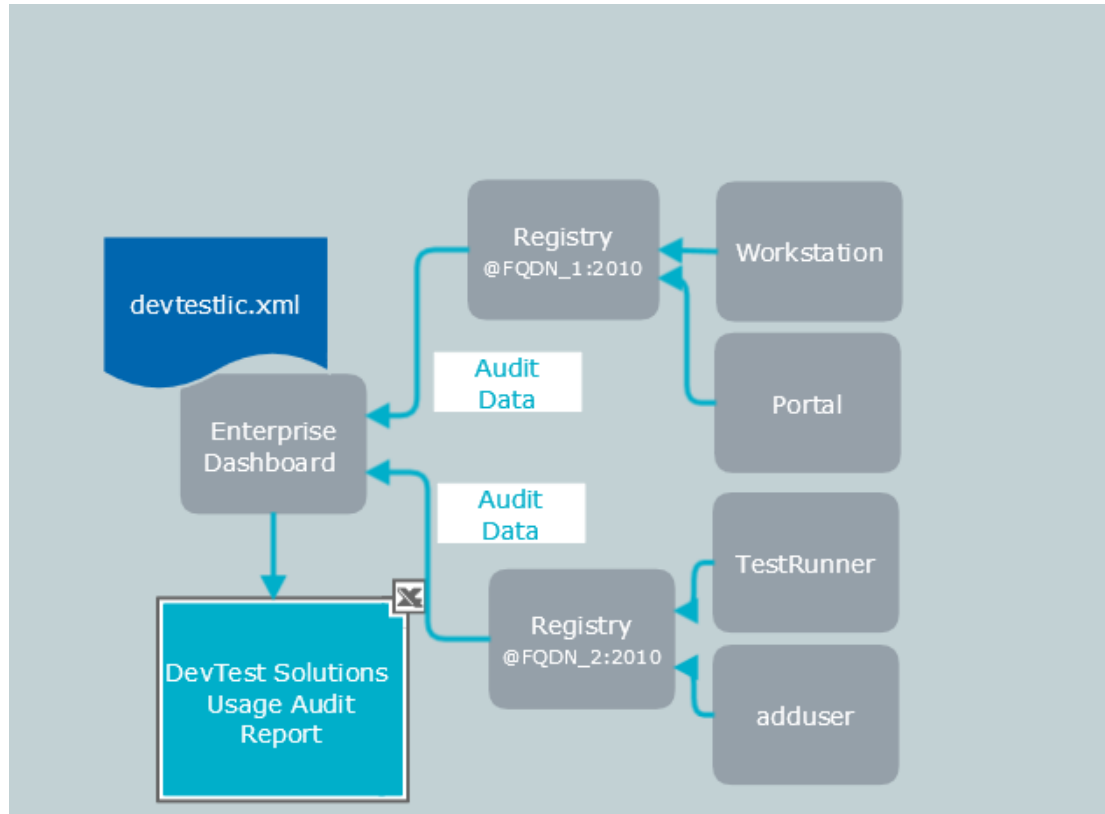
Honor-Based Licensing

Honor-based licensing overview:

- The DevTest 8.0 and later license agreement is based on the maximum allowed concurrent user sessions by user type. Contact your account team if you have any questions about your specific licensing agreement.
- The license is file-based, where there is one file for the enterprise. This file activates DevTest Solutions at the initial startup following installation.
- The Local License Server (LLS) and Internet Based License Server are no longer supported for DevTest Solutions 8.0.
- Concurrent usage data by user type is automatically collected.
- The Enterprise Dashboard produces licensing reporting.
- A Usage Audit Report, which reports maximum concurrent usage by user type, is a tool that helps you assess compliance with the license agreement. Users with administrative privileges can access this report.

For more information, see [How Licensing, ACLs, and Audit Reports Work Together](#) (see page 54).

The following diagram shows the activation of DevTest Solutions by a license file that is stored with the Enterprise Dashboard. The registries collect audit data from the UIs and CLIs that users log in to, including the Workstation, the Portal, and command-line utilities such as TestRunner and adduser. The registries forward the audit data to the Enterprise Dashboard. Administrators can generate a DevTest Solutions Usage Audit Report to verify compliance with the license agreement.



How Licensing, ACLs, and Audit Reports Work Together

The page includes details on how licensing, ACLs, and audit reports work together.

- [Usage-Based License Agreement \(see page 55\)](#)
- [License Activation \(see page 55\)](#)
- [ACL Activation \(see page 55\)](#)
- [Honor-based Compliance \(see page 56\)](#)
- [Continuous Collection of Usage Data by User Type \(see page 56\)](#)
- [Usage Audit Report \(see page 56\)](#)

Usage-Based License Agreement

Your license agreement with CA Technologies is based on the maximum number of concurrent user sessions by user type. User types are:

- Runtime User
- Test Power User
- SV Power User
- CAI Power User

Each activity that a user can perform with DevTest Solutions has a corresponding permission. Each permission is associated with a role. Each role is associated with one of the user types. Administrators assign permissions to users.

License Activation

License activation for DevTest Solutions is file-based. CA Technologies provides you with a license file (devtestlic.xml). During installation or upgrade of DevTest Solutions, the Setup wizard puts the file in the LISA_HOME directory on the host where the Enterprise Dashboard is installed. Subsequent installations of the Server component (DevTest Server) reference the URL of the Enterprise Dashboard. License activation occurs the first time that you start the Enterprise Dashboard and the registries. No other component requires license activation.

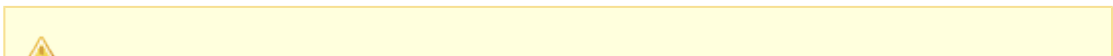
If you are upgrading, request a new devtestlic.xml license from CA Technologies. The current license key generation process is leveraged to generate a product key that is delivered in the devtestlic.xml file. This key is used to activate or unlock the Enterprise Dashboard.

ACL Activation

Access Control (ACL) is enabled by default. Before users can access DevTest Solutions, they must be authenticated with valid credentials and then authorized to perform the tasks that are associated with their role. You define for each user a user name, a password, and a role that is composed of a set of permissions. Roles are tied to user types.

Users must log in with valid credentials to access any UI (User Interface) or CLI (Command Line Interface). When a user opens the Workstation or browses to the Portal, a logon dialog is presented. The user logs in with the credentials defined in DevTest or in LDAP, if you have configured DevTest to use LDAP credentials. If the entered credentials match that of an authorized user, the UI opens. The features that are presented to the user are based on the permissions you grant to that user.

For backward compatibility, one exception exists. Although ACL is enabled, authorized users can run tests and can start virtual services without specifying a valid login user name and password.



Note: We recommend that you [change the passwords for the standard users \(see page 93\)](#) to ensure that only authenticated users gain access to DevTest Solutions.

Honor-based Compliance

The customer is responsible for license compliance with the terms of contract. The license that is issued does not enforce the maximum concurrency by user type. As outlined in the Usage Audit Report section, DevTest Solutions Usage Audit Reports are sent to CA Technologies upon request. The CA account manager may contact you to assess your interest in expanding your purchase agreement if the Audit Reports indicate concurrent usage of a user type that exceeds the agreement.

To evaluate functionality that has not been purchased, contact your Account Team to discuss a Proof of Concept trial, or to address questions about the new functionality. Any Support cases that are raised for product functionality that has not been purchased are treated as "Out of Scope".

Continuous Collection of Usage Data by User Type

When users log in to a DevTest Solutions UI or CLI, each registry captures session data by user type and forwards it to the Enterprise Dashboard at the top of each hour. If connectivity between the registries and the Enterprise Dashboard is lost, the usage audit data accumulates on the registries until connectivity is re-established. The Enterprise Dashboard compiles usage data from across the enterprise to count concurrent sessions by user type.

Usage Audit Report

Periodically, an administrator logs on to the Enterprise Dashboard and generates the Usage Audit Report for a specified time period. Data from the specific UI or CLI users log in to is reported. The concurrent sessions with the maximum number of users of a given user type are used in the report calculations. The report details concurrent usage counts by user type across registries. For each user type with data, the report generator creates a tab with drill-down details. The access that is detailed in the Audit Reports can be compared to the license agreement to determine the level of compliance.

The Usage Audit Report also reports instances of VSEs with the VSE Performance mode activated.

You can evaluate this report in light of your license agreement to verify compliance or to see if you want to upgrade your contract.

DevTest Solutions Usage Audit Report

A DevTest Solutions Usage Audit Report is generated as an Excel workbook with the following tabs:

- Overview
- User Type Chart

- SV for Performance Chart
- Historical Usage
- Component by User

Overview

The Overview tab provides details on the extent an enterprise is using the number of concurrent users of each user type that their license permits. A given report provides data from all registries for the specified date range. The Count data is shown by user type.

	A	B	C
1	DevTest Solutions	Usage Audit Report	
2	Company:	AppDelBU	
3	License Key ID:	1f8e856	
4	License Key Expiration:	December 08, 2016	
5	Enterprise Dashboard Product Version	9.5.0	
6	Reporting Period Start:	February 14, 2016	
7	Reporting Period End:	March 15, 2016	
8	Generated On:	March 22, 2016	
9	User Type	Maximum Concurrent Use Counts	Instances of Max Use Count
10	Continuous Application Insight Power User	2	1
11	Service Virtualization Power User	2	2
12	Application Test Power User	3	1
13	DevTest Runtime User	2	1
14	Maximum Concurrent Instances	Maximum Concurrent Use Counts	Instances of Max Use Count
15	Service Virtualization for Performance	2	1
16			
17	Usage Audit Documentation		
18			
19			

Usage Audit Report Overview Tab

Report Information

The report information section lists licensing information and the date range for this audit report.

- **Company:** Name of the company that generated this report
- **License Key ID:** The license key identifier
- **License Key Expiration:** The date the license expires
- **Enterprise Dashboard Product Version:** The version of DevTest
- **Reporting Period Start:** The starting date for this report
- **Reporting Period End:** The ending date for this report
- **Generated On:** The date on which the report was generated

User Type

Reported user types include Continuous Application Insight Power User, Service Virtualization Power User, Application Test Power User, and DevTest Runtime User.

If the user is granted permissions that are associated with more than one user type, the higher user type is used. The user type hierarchy, from highest to lowest, follows:

- 1.

- a. Continuous Application Insight Power User or Service Virtualization Power User (equivalent user types)
- b. Application Test Power User
- c. DevTest Runtime User
See [User Types](#) (see page 62).

▪ **Maximum Concurrent Use Counts**

All registries continuously collect usage data from the services that support the UIs and CLIs. Raw data is kept for concurrent usage by the same user type across the enterprise, where the concurrent login sessions are recorded by registry. Concurrent usage occurs when two or more users of the same user type access a DevTest UI or DevTest CLI where their sessions overlap in time. In this sample report, the highest concurrent usage was at least once where 3 users with a user type of Application Test Power User were logged on at the same time.

▪ **Instances of Max Use Count**

The instance calculations for the report are based on group counts where the maximum concurrency count is reached. In this sample report, there were two points in time between February 14 and March 15, 2016, where 2 Service Virtualization Power Users were logged in at the same time. See the [Count Calculation Example](#) (see page 61).

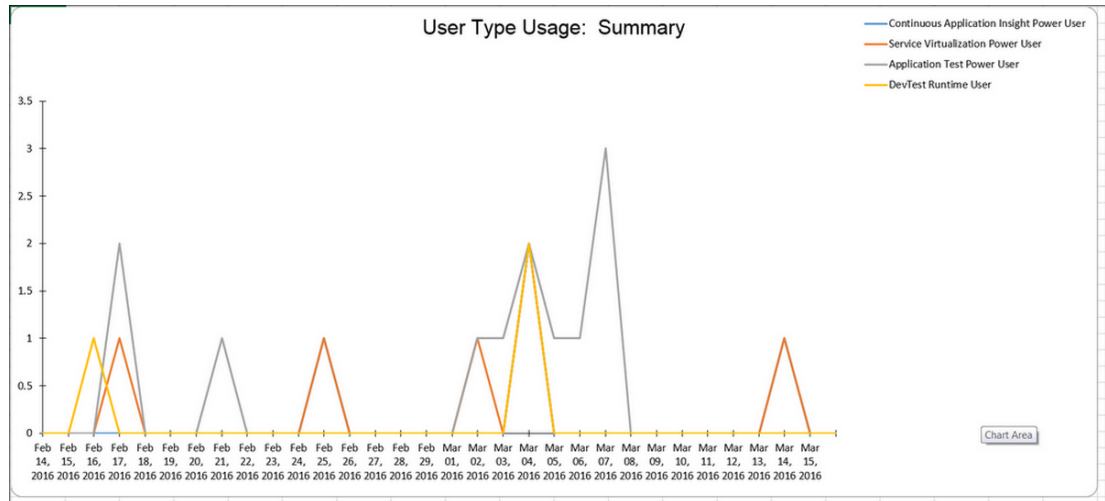
▪ **Maximum Concurrent Instances**

Counts concurrent VSEs that have the [VSE Performance](https://docops.ca.com/display/DTS95/Using+CA+Service+Virtualization) (<https://docops.ca.com/display/DTS95/Using+CA+Service+Virtualization>) option enabled. If the VSE Performance option is not enabled, the count is 0. In this sample report, between February 14 and March 15, 2016, there was one time when there were two VSEs with the Performance option enabled that were running at the same time.

The **Usage Audit Documentation** link at the bottom of the report links to this page of the product documentation.

User Type Chart

The User Type chart shows, by date, the concurrent usage by user type of the product during the time period for which the report was requested. The chart is derived from the data on the Historical Usage tab.



Usage Audit Report User Type Chart

To customize the chart, click on it.

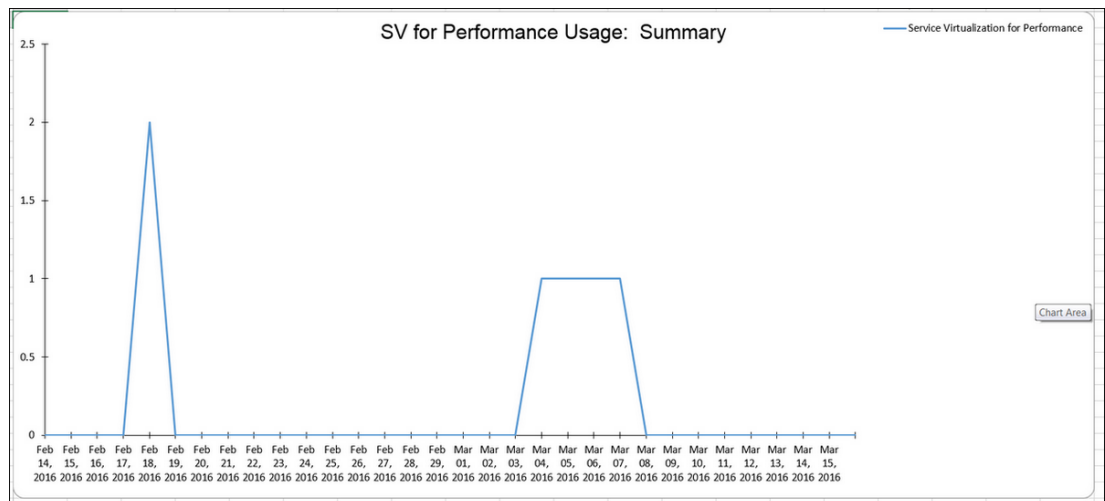
To customize the chart style and color, click the paintbrush icon.

To drill into a specific usage type, click the funnel icon and select a component.

For more information about customizing charts, see Microsoft Excel Help.

SV for Performance Chart

The SV for Performance Usage: Summary chart shows, by date, the maximum concurrent usage of SV for Performance during the time period for which the report was requested.



Usage Audit Report SV for Performance Chart

To customize the chart, click on it.

To customize the chart style and color, click the paintbrush icon.

To drill into a specific usage type, click the funnel icon and select a component.

For more information about customizing charts, see Microsoft Excel Help.

Historical Usage Tab

The Historical Usage report shows, by date, the highest concurrent usage by user type of the product for each day during the time period for which the report was requested. This tab contains the raw data that is shown on the User Type Chart.

Date	SV for Performance	Application Insight Power User	Service Virtualization Power User	Application Test Power User	DevTest Runtime User
Feb 14, 2016	0	0	0	0	0
Feb 15, 2016	0	0	0	0	0
Feb 16, 2016	0	0	0	0	1
Feb 17, 2016	0	0	1	2	0
Feb 18, 2016	2	0	0	0	0
Feb 19, 2016	0	0	0	0	0
Feb 20, 2016	0	0	0	0	0
Feb 21, 2016	0	0	0	1	0
Feb 22, 2016	0	0	0	0	0
Feb 23, 2016	0	0	0	0	0
Feb 24, 2016	0	0	0	0	0
Feb 25, 2016	0	1	1	0	0
Feb 26, 2016	0	0	0	0	0
Feb 27, 2016	0	0	0	0	0
Feb 28, 2016	0	0	0	0	0
Feb 29, 2016	0	0	0	0	0
Mar 01, 2016	0	0	0	0	0
Mar 02, 2016	0	0	1	1	0
Mar 03, 2016	0	0	0	1	0
Mar 04, 2016	1	2	2	2	2
Mar 05, 2016	1	0	0	1	0
Mar 06, 2016	1	0	0	1	0
Mar 07, 2016	1	0	0	3	0
Mar 08, 2016	0	0	0	0	0
Mar 09, 2016	0	0	0	0	0
Mar 10, 2016	0	0	0	0	0
Mar 11, 2016	0	0	0	0	0
Mar 12, 2016	0	0	0	0	0
Mar 13, 2016	0	0	0	0	0

User Audit Report Historical Usage Tab

Component by User

The Component by User tab reports user session data from each registry that is connected to the Enterprise Dashboard, sorted by registry, with a column for each UI or CLI with access during this time period. Entries on this tab include all access, not only concurrent access.

Registry	User Name	User Type	Total Sessions	Console	invoke2-api	Test Runner	Workstation
Registry@lenba0113068-2.ca.com:2010	admin	PF_POWER_USER	1	0	0	0	1
Registry@lenba0113068-2.ca.com:2010	admin	SV_POWER_USER	1	0	0	0	1
Registry@lenba0113068-2.ca.com:2010	tpower	TEST_POWER_USER	2	0	0	0	2
Registry@lenba0113933-2.ca.com:2010	admin	PF_POWER_USER	1	0	0	0	1
Registry@lenba0113933-2.ca.com:2010	admin	SV_POWER_USER	1	0	0	0	1
Registry@LENBA01.ca.com:2010	tpower	TEST_POWER_USER	1	0	0	0	1
Registry@lenba0113638-2.ca.com:2010	admin	ADMIN_USER	3	3	0	0	0
Registry@lenba0113638-2.ca.com:2010	barb	ADMIN_USER	4	4	0	0	0
Registry@lenba0113638-2.ca.com:2010	barb	PF_POWER_USER	6	0	2	0	4
Registry@lenba0113638-2.ca.com:2010	barb	SV_POWER_USER	6	0	2	0	4
Registry@lenba0113638-2.ca.com:2010	guest	SVT_RUNTIME_USER	3	0	1	0	2
Registry@lenba0113638-2.ca.com:2010	svpower	ADMIN_USER	5	5	0	0	0
Registry@lenba0113638-2.ca.com:2010	svpower	SV_POWER_USER	6	5	0	0	1
Registry@lenba0113638-2.ca.com:2010	tpower	ADMIN_USER	3	3	0	0	0
Registry@lenba0113638-2.ca.com:2010	tpower	TEST_POWER_USER	14	3	3	2	6

Usage Audit Report Component by User Tab

- **Registry**

Registry names are in the format:

registry@FQDN:2010

- **User Name**

Names of users who logged in to a user interface or command-line interface for one or more sessions.

- **User Type**

The user type that is assigned to the associated user.

- **Total Sessions**

The sum of counts from the reported columns for the associated user, where the columns represent either a UI session or a session using a CLI. User interfaces include DevTest Workstation. Examples of CLIs include the DevTest Portal, the REST Invoke API and Test Runner. Columns may vary by report, depending on where users log in. Some examples follow:

- **invoke2-api:** The number of times this user initiated the DevTest [Invoke API \(https://docops.ca.com/display/DTS95/REST+Invoke+API\)](https://docops.ca.com/display/DTS95/REST+Invoke+API).
- **Workstation:** The number of times this user opened a DevTest Workstation, connected to the associated registry, and logged in.



Note: For details on generating this report, see [Export Usage Audit Data \(see page 135\)](#).

Count Calculation Example

Calculating use counts for the [DevTest Solutions Usage Audit Report \(see page 56\)](#) is a two-step process.

1. Each time a concurrency is detected across the enterprise, a total is recorded for all involved registries. Consider the following example for the Application Test Power User user group.

In this example, there were totals recorded at five different times for three different registries. The largest number of concurrent logins by Application Test Power users was 3 and is highlighted below.

For reporting purposes, the totals with maximum concurrency are used. In this sample example, row 1 is used, because its total concurrent users is the peak count for the selected time interval.

Application Test Power User					
Row	Date	Registry 1	Registry 2	Registry 3	Total
1	10/07/2016	1	0	2	3
2	10/08/2016	0	0	2	2
3	10/09/2016	0	2	0	2
4	10/10/2016	0	0	1	1
5	10/11/2016	1	1	0	2

- This data is summarized on the Overview page of the report as User Type, Maximum Concurrent Use Counts, and Instances of Max Use Count. The count represents the largest total for that user type. The instances of max use count represent the number of times over the time period that this maximum concurrency occurred. For the example above, the instance count would be 1 (the number of times that the Application Test Power Users reached its maximum concurrent use count of 3).

User Type	Maximum Concurrent Use Count	Instances of Max Use Count
Application Test Power User	3	1

User Types

ACL has the following user types:

- SV Power User
- CAI Power User
- Test Power User
- Runtime User

A user type can be associated with one or more roles. Each role is associated with only one user type. For example, the Runtime User user type has three roles (System Administrator, Runtime, and Guest), but the Runtime role is associated with only the Runtime User user type.

A user can be granted one or more roles. When granted multiple roles, the role that is associated with the highest user type is used for usage auditing purposes. The user type hierarchy is made up of the following user types:

- CAI Power User and SV Power User are equal in the hierarchy and both are the highest user type.
- Test Power User is lower than CAI Power User and SV Power User but higher than Runtime User.

- Runtime User is the lowest user type.

Example of How the User Type Is Determined for Auditing Purposes

The permissions that are associated with a role assigned to a user determine the tasks that a user is authorized to perform. An administrator can assign one or more roles to a user. Typically, administrators assign a single role to each user because the permissions for the built-in roles are assigned with the user type hierarchy in mind. For example, the PF Power role is also assigned permissions that are associated with Test Power and Runtime.

To grant users permissions that are not part of the higher role, you can assign them multiple roles.

Consider an example where a user is assigned the following roles:

- Test Administrator
- Runtime

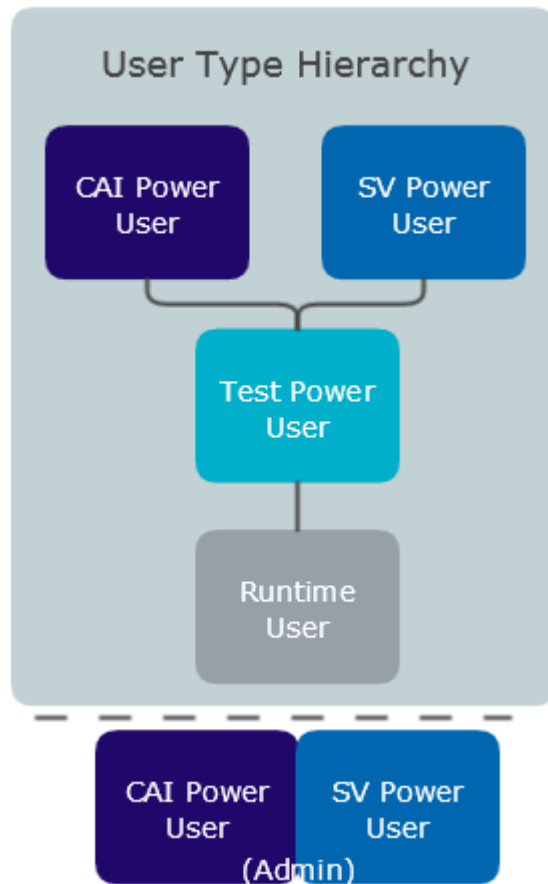
Different user types are associated with the assigned roles:

- The Test Administrator role is associated with the SV Power User user type.
- The Runtime role is associated with the Runtime User user type.



Note: See [Standard User Types and Standard Roles \(see page 86\)](#) for details.

User types are hierarchical.

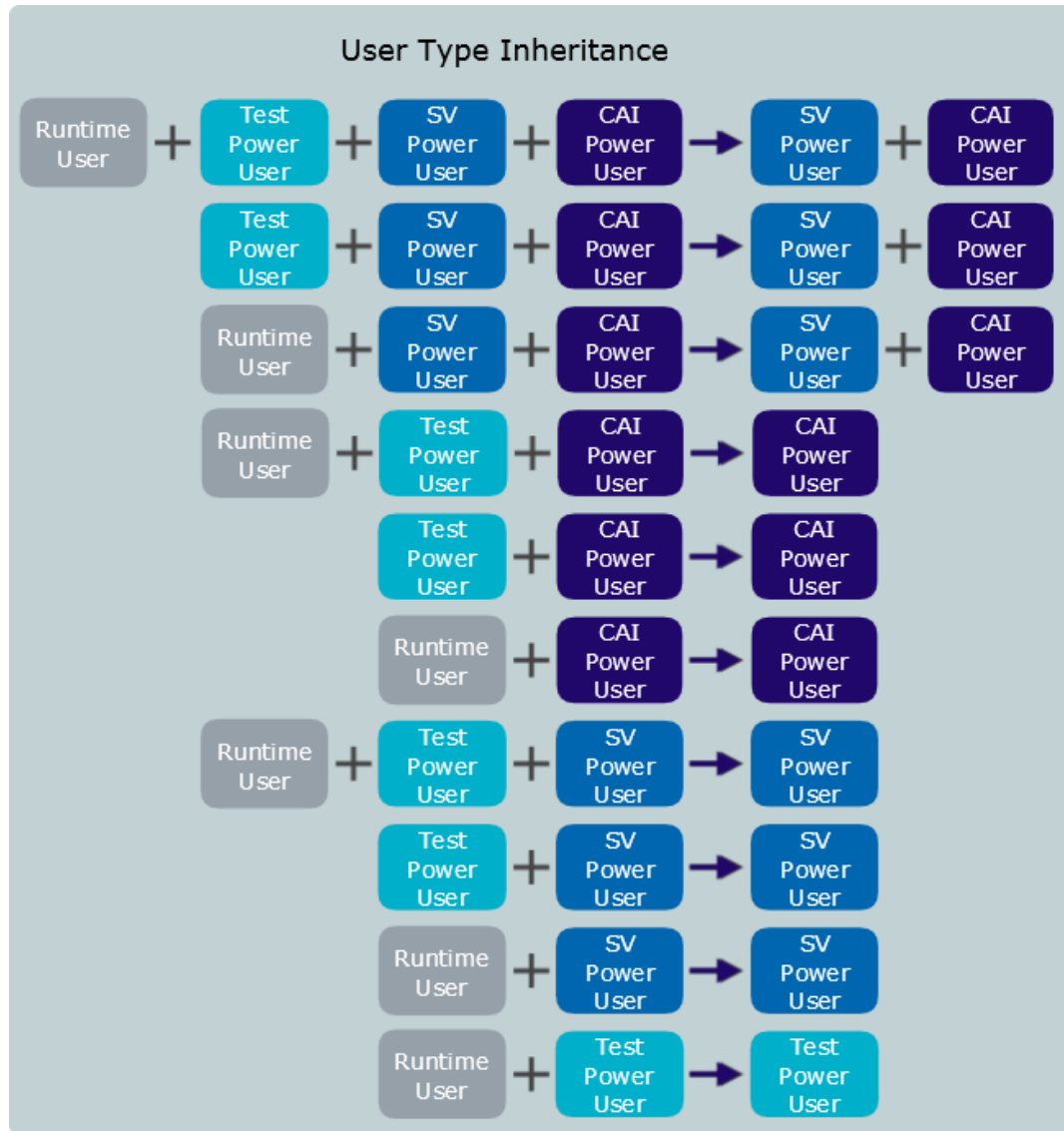


For purposes of the DevTest Solutions Usage Audit Report, the user type under which a user is counted is the highest user type that is associated with an assigned role. For example, when the *fakeName* user logs in to a DevTest UI or CLI, the user session is audited as belonging to the SV Power User user type, even if *fakeName* performs only tasks that are associated with the Runtime role, such as report administration.



User Type Inheritance Chart

The following diagram shows all cases of how the user type of a logged in user is determined if the user is assigned multiple roles from different user types. The "highest" user type takes precedence over all lower user types.



Usage Audit Report FAQs

This section addresses some of the frequently asked questions about the Usage Audit Report and license administration.

- [Licensing \(see page 66\)](#)
- [Account Sharing \(see page 66\)](#)
- [Reporting \(see page 67\)](#)
- [Administration \(see page 69\)](#)

Licensing

- **What is concurrent licensing?**

"Concurrent User" means a software license that is based on the number of simultaneous (concurrent) users accessing the program. DevTest does not prohibit more users from access, but does log license usage for auditing purposes. You are entitled to have the specified number of Concurrent Users access the software simultaneously. For licensing purposes, each Concurrent User is entitled to have one active session that is connected to the server either directly or indirectly. Each active session that is initiated by an individual, device, or process is counted as a Concurrent User. An active session means that either a workstation or Web browser is connected to the registry server, DevTest Portal, or both, or a custom application session accessing a virtualized service.

- **What types of licenses and how many licenses am I licensed for?**

Contact your account team for questions about your licensing agreement.

- **How many licenses are consumed when I use DevTest Workstation and DevTest Portal on the same machine?**

If you are using both DevTest Workstation and DevTest Portal on the same machine, one concurrent usage count is logged. The same is true for any combination of UIs and CLIs, on the same or on different machines.

- **Does my license limit the number of [Service Virtualization for Performance](https://docops.ca.com/display/DTS95/Using+CA+Service+Virtualization) instances that I can run? Is that also honor-based?**

You enable the Performance Option for VSE with the [VSE Manager](https://docops.ca.com/display/DTS95/VSE+Manager+Command++Manage+Virtual+Service+Environments) command or by setting a [property](https://docops.ca.com/display/DTS95/SV+Installation+and+Configuration). This number is also [honor-based](#) (see page 53).

Account Sharing

- **How are concurrent licenses calculated when we share a user: for example, *admin*?**

If more than one user uses the same user ID simultaneously on different machines, each is counted toward the concurrent use. If a user uses the same user ID to access different DevTest applications on the same machine, it is counted as a single use.

For example, if you have six users who are logged on to DevTest Workstation with the *admin* user ID, the Usage Report shows a concurrent use count of 6 for the user type that is associated with the *admin* user ID. Note: We do not recommend sharing of user IDs.

To prevent users from sharing user IDs, we recommend connecting DevTest to your AD/LDAP datasource. DevTest can enable AD/LDAP [authentication](#) (see page 106). DevTest continues to provide authorization. This technique does not bind a user ID or login to a DevTest Workstation. This technique does, however, provide some control because users are less likely to share their AD/LDAP credentials.

Reporting

- **What type of licenses and how many concurrent licenses am I consuming?**

To get this information, run a DevTest [Solutions Usage Audit Report \(see page 56\)](#) by [exporting usage audit data \(see page 134\)](#) from the Enterprise Dashboard to Excel. See the Overview tab to see the maximum concurrent sessions that are used by each license type for the time period you choose.

- **How do we calculate *runtime* users? Are there limitations, for example *runtime* users that do not log into DevTest?**

The license counts are all based on login and logout records.

Every DevTest user login creates a user session and increments the license count by one. Every DevTest user logout (including implicit logouts through timeouts) closes a user session and decrements the license count by one.

Note: Anything that connects to the virtual services deployed on a VSE server consumes a license, but these connections cannot be tracked in the usage audit report because they do not log in to DevTest.

- **How are concurrent licenses calculated when I'm logged into one DevTest application (for example, DevTest Workstation) on one machine and logged into another DevTest application (for example, DevTest Portal) on a different machine?**

In this example, because you are using two separate machines, you would consume two licenses. The license type that is consumed is based on the actions that are performed during the user session.

- **How frequently is usage polled?**

The information is collected continuously based on login and logout times. When an audit report is requested, the data is examined to determine when the concurrent access occurs.

- **If a user is logged in, but is not performing any activity, is that user counted in the usage count?**

Yes.

- **What about command-line interfaces like VSE Manager? Are those users counted in the usage count?**

Yes. Any utility that requires credentials is included in the usage count.

- **I am licensed for 10 SV Power Users, but I see as many as 15 concurrent users at one time. Is there a problem with my license?**

No, licensing for DevTest Solutions is [honor-based \(see page 53\)](#). Use the Usage Audit Report to monitor your compliance with the terms of your license.

- **My Component by User report shows a user type for ADMIN_USER. What is that user type and do I need to be concerned about licensing for it?**

While the ADMIN_USER type is included in the report, you do not need to consider it when evaluating your compliance to your licensing agreement. The ADMIN_USER user type, which is displayed on the Roles page as a combination CAI Power User and SV Power User types, is included in the DevTest Solutions Usage Audit Report for your information, and is counted as an SV_POWER user for licensing purposes.

- **The Overview tab indicates that I have 12 Concurrent users. When I click on the Component by User tab, I only see four different user IDs. Can you explain why we need to get licenses for more Power Users when we see only four different user IDs using the product?**

On the Overview tab, you see that CAI Power User and Service Virtualization Power User have each had 12 concurrent users during the reporting period, once for CAI Power User and twice for SV Power User.

DevTest Solutions		Usage Audit Report	
Company:	Forward, Inc.		
License Key ID:	1f8e856		
License Key Expiration:	December 08, 2016		
Enterprise Dashboard Product Version	9.5.0		
Reporting Period Start:	February 14, 2016		
Reporting Period End:	March 1, 2016		
Generated On:	March 15, 2016		
User Type	Maximum Concurrent Use Counts	Instances of Max Use Count	
Continuous Application Insight Power User	12	1	
Service Virtualization Power User	12	2	
Application Test Power User	0	0	
DevTest Runtime User	0	0	
Maximum Concurrent Instances	Maximum Concurrent Use Counts	Instances of Max Use Count	
Service Virtualization for Performance	2	1	
Usage Audit Documentation			

DevTest Usage Audit Report

On the Historical Usage tab, you can see when those high concurrent usage days occurred: for CAI Power User, on March 7, and for SV Power User, on March 5 and March 13.

Date	SV for Performance	Continuous Application Insight Power User	Service Virtualization Power User	Application Test Power User	DevTest Runtime User
Mar 01, 2016	0	0	0	0	0
Mar 02, 2016	0	0	1	0	0
Mar 03, 2016	0	0	0	0	0
Mar 04, 2016	1	2	2	0	0
Mar 05, 2016	1	0	12	0	0
Mar 06, 2016	1	0	0	0	0
Mar 07, 2016	1	12	3	0	0
Mar 08, 2016	0	0	0	0	0
Mar 09, 2016	0	0	0	0	0
Mar 10, 2016	0	0	0	0	0
Mar 11, 2016	0	0	0	0	0
Mar 12, 2016	0	0	0	0	0
Mar 13, 2016	0	0	12	0	0
Mar 14, 2016	0	1	1	0	0
Mar 15, 2016	0	0	0	0	0

Historical Usage Tab

The Component by User tab does not reflect any information about concurrent usage. It does indicate that in this example, there are only four different users who have used the product during this time period: *admin*, *barb*, *rob*, and *svpower*. They have used various interfaces to connect to three different registries.

Registry	User Name	User Type	Total Sessions	Console	invoke2-api	Test Runner	Workstation
Registry@forward13068-2.fw.com:2010	admin	CAI_POWER_USER	1	0	0	0	1
Registry@forward13068-2.fw.com:2010	admin	SV_POWER_USER	56	16	5	34	1
Registry@forward13933-2.fw.com:2010	rob	CAI_POWER_USER	10	9	0	0	1
Registry@forward13933-2.fw.com:2010	admin	SV_POWER_USER	1	0	0	0	1
Registry@forward13638-2.fw.com:2010	barb	CAI_POWER_USER	36	7	16	9	4
Registry@forward13638-2.fw.com:2010	barb	SV_POWER_USER	38	32	2	0	4
Registry@forward13638-2.fw.com:2010	svpower	SV_POWER_USER	14	5	8	0	1

Component by User Tab

In this example, we can tell that users at this company are sharing user IDs, thus accounting for the concurrent usage that is higher than the number of user IDs. To track product usage accurately, we recommend that your users do not share user IDs.

Administration

- **Where can I specify timeout values for DevTest Workstation and DevTest Portal, so that users who are not active will time out and not be counted in the Usage Audit Report?**

The **registry.max.user.lifetime.seconds** property is the main session lifetime property that determines how long a user session is valid after the last activity done by a user in a session.

This property specifies how many seconds the SSO security token is kept in memory for DevTest Workstation to bypass authentication for the DevTest Portal and DevTest Workstation.

- **Can we implement ACL so that a user can log in from only one DevTest Workstation instance and thus can occupy only one license per user ID?**

No, once a user ID is given access, DevTest does not perform actions that "bind" a given user ID to a DevTest Workstation IP address. User IDs are shown in the Usage Audit report.

- **When I perform an action (for example, log in to DevTest Workstation) how can I tell which license type is consumed?**

Each permission (listed in the Role UI, right-hand side) has a license type that is associated with it, identified within parentheses.

- **If a user has multiple user types, how can I determine the effective user type for a new or existing role?**

You can view or change the permissions and essentially update the effective user type of a role, within [access control \(see page 82\)](#).

- **How does the Usage Audit Report differ from the User Activity Report?**

The [User Activity Report \(see page 100\)](#) is useful to show an administrator what users are performing which activities in real time.

Security

This section contains the following pages that describe security options in DevTest :

- [Using SSL to Secure Communication \(see page 71\)](#)
- [SSL/TLS Protocol Configuration \(see page 76\)](#)
- [Using HTTPS Communication with the invoke APIs \(see page 77\)](#)
- [Using Kerberos Authentication \(see page 80\)](#)
- [Access Control \(ACL\) \(see page 82\)](#)

Using SSL to Secure Communication

By default, communication between components uses an unencrypted protocol. If necessary, the Secure Sockets Layer (SSL) can encrypt the network traffic. For example, if you run a lab in a public cloud and you want to ensure the traffic transmitted from your workstation is encrypted.

The easiest way to enable SSL is to set a DevTest property.

```
lisa.net.default.protocol=ssl
```

You cannot specify this property in **site.properties**; that is too late in the bootstrap phase. This property must be specified in **local.properties** (or on the command line).

If you then start a registry with no extra parameters - for example, it is listening on port 2010 (the usual port) but it expects clients to use the SSL protocol - the service name for the registry is *ssl://hostname:2010/Registry*.

If you want to connect to that registry from DevTest Workstation, use *ssl://hostname:2010/Registry* instead of the usual *tcp://hostname:2010/Registry*. If you start a simulator on the same computer, it is available on *ssl://hostname:2014/Simulator*, and it automatically connects to the registry at *ssl://hostname:2010/Registry* with no property changes.

You can also mix and match SSL and normal TCP protocols. If you leave the **lisa.net.default.protocol** property at its default setting (tcp), you can enable specific services for SSL by specifying the name of the individual service with the "ssl:" protocol prefix, instead of the default "tcp:" prefix. For example, to start a registry in SSL mode:

```
Registry --name=ssl://reghost.company.com:2010/Registry
```

To enable the SSL, use "ssl" in the service names instead of "tcp". For example:

```
Registry --name=ssl://reghost.company.com:2010/Registry
```

starts the registry with SSL enabled.

To connect a simulator to this registry, start the simulator with the fully qualified registry address:

```
Simulator --name=ssl://simhost.company.com:2014/Simulator --registry=ssl://reghost.
company.com:2010/Registry
```

This command tells the simulator to use SSL to talk to the registry while also securing the simulator. If you want the simulator itself to be unsecured, do this:

```
Simulator --registry=ssl://reghost.company.com:2010/Registry
```

Mixing secured and unsecured servers is not common. However, you may want to have unsecured servers inside your firewall and secured servers in a public cloud. There is some overhead using SSL encryption, which varies considerably depending on the hardware.

The **lisa.net.default.protocol** property defines the default protocol for ActiveMQ connections. The property does not influence the protocol that is used when DevTest components start.

If you have DevTest Enterprise Dashboard configured to use SSL, the **lisa.enterprisedashboard.server.url** property needs to be configured to have the value of **ssl**. Set this in the **local.properties** file.

All communication with the Enterprise Dashboard is done through the webserver port (1506 by default), and uses either HTTP or HTTPS.

To configure the Enterprise Dashboard to use HTTPS, create the **dradis.properties** file (by copying the **_dradis.properties** file), and uncomment the **dradis.webserver.https.enabled** property.

```
dradis.webserver.https.enabled=true
```

To configure the registry to know that the Enterprise Dashboard is using HTTPS, uncomment the **devtest.enterprisedashboard.https.enabled** property in the **site.properties** file.

```
devtest.enterprisedashboard.https.enabled=true
```

To configure the registry and all other components to use SSL by default, uncomment the **lisa.net.default.protocol** value in the **local.properties** file.

```
lisa.net.default.protocol=ssl
```

SSL Certificates

By default, a self-signed certificate encrypts and decrypts the messages between components. VSE also uses this certificate when recording https:// style web traffic. The certificate is in the **LISA_HOME\webrekeys.ks** file.

When you set the default protocol to SSL and you do not change anything else, you use an "internal DevTest" certificate. All DevTest users (not only your organization) share this internal certificate. Using this certificate encrypts the network traffic, but it does not prevent an unauthorized user from connecting to your simulator on a public cloud. To prevent this type of unauthorized access, use your own certificate. You can also continue to use the "well-known" DevTest certificate, but enable access control.

If you want to use your own certificate, you can override the certificate keystore by specifying the following properties:

```
lisa.net.keyStore=/path/to/keystore.ks
lisa.net.keyStore.password=plaintextPassword
```

The first time DevTest reads the plain text password, it converts the password to an encrypted property:


```
lisa.net.keyStore.password_enc=33aa310aa4e18c114daf86a33cee898
```

Create Your Own Self-Signed Certificate

This example uses the keytool utility, which is in the Java Runtime Environment (JRE).

To create your own self-signed certificate:

1. Enter the appropriate responses to the prompts.

```
prompt>keytool -genkey -alias serverA -keyalg RSA -validity 365 -
keystore keystore.ks
Enter keystore password: MyNewSecretPassword <the actual plaintext won't be shown>
Re-enter new password: MyNewSecretPassword
What is your first and last name?
[Unknown]: serverA
What is the name of your organizational unit?
[Unknown]: dev
What is the name of your organization?
[Unknown]: ITKO
What is the name of your City or Locality?
[Unknown]: Dallas
What is the name of your State or Province?
[Unknown]: TX
What is the two-letter country code for this unit?
[Unknown]: US
Is CN=serverA, OU=dev, O=ITKO, L=Dallas, ST=TX, C=US correct?
[no]: yes
Enter key password for <serverA>
(RETURN if same as keystore password) <just hit return>
```

The utility creates a file containing a certificate that is valid for 365 days.

2. Copy the file to LISA_HOME and update **local.properties**:

```
lisa.net.keyStore={{LISA_HOME}}keystore.ks
lisa.net.keyStore.password=MyNewSecretPassword
```

3. The first time DevTest reads the plain text password, it converts the password to an encrypted property:

```
lisa.net.keyStore.password_enc=33aa310aa4e18c114daf86a33cee898
```

The server side of the connection configuration is complete.

4. Configure the client.

Because this certificate is self-signed, you explicitly tell the clients to trust the certificate. Typically, when you connect to an SSL service (for example, using a browser to <https://www.MyBank.com>) a trusted Certification Authority certifies the certificate. Because a trusted third party does not certify self-signed certificates, you must add the certificate to a Trust Store:

```
lisa.net.trustStore={{LISA_HOME}}trustStore.ts
lisa.net.trustStore.password=MyNewSecretPassword
```

The same keytool utility manipulates trust stores. In general, a keystore contains one certificate and a trust store contains one or more certificates.

5. Export the certificate from the server keystore:

```
keytool -exportcert -rfc -alias serverA -keystore keyStore.ks -file serverA.cer
```

The **-rfc** means to export the certificate as ASCII text instead of binary, to make it easier to copy and paste. In our example, the resulting **serverA.cer** file looks like the following example:

```
-----BEGIN CERTIFICATE-----
MIICEzCCAXygAwIBAgIETnYzANBgkqhkiG9w0BAQUFADBOMQswCQYDVQQGEwJDQjELMAkGA1UE
CBM4Z20IxIzAJBgNVBACtAKNCMQswCQYDVQQKEwJDQjELMAkGA1UECmCQ0IxIzAJBgNVBAMTAKNC
MB4XDTEyMDcwODAyMTE0N1oXDTEyMDcwNzAyMTE0N1owTjELMAkGA1UEBhMCQ0IxIzAJBgNVBAGT
AKNCMQswCQYDVQQHEwJDQjELMAkGA1UECDMCQ0IxIzAJBgNVBAsTAkNCMQswCQYDVQQDEwJDQjCB
nzANBgkqhkiG9w0BAQEFAAOBjQAwYkCgYEAhYfaN+dCrKQwYZ+KaaaPUI8DeXNiQ/mS+KGnXnh
Pz08vdX/7HDLW4pzFhntjmKxx0i9dMwL02thTD1c0xI571PotenMENo4nyiUAEnMK9MTiWEYr2cQ
b6/TUueBCjRJ9I0GPCI0WPS+0Na2Q/wq8gPCHmDRpw1Xgo4uZ1v6C/ECaWEAATANBgkqhkiG9w0B
AQUFAA0BgQByCsX9EoBFiGhcSwoRwEvapIrv8wTaqQP0KKYeiEvSmbnERRu6+oi+cJftbdEfw6GG
CBddJH+dGZ9VeqLU8zBGasbU+JPzG5El0g0XcUGeQQEam1YMv6XWrIwNSljQk/MPZSt3R0tJ0lae
JPKJXSQ610xof9+yLHH0ebUGHuJdlQ==
-----END CERTIFICATE-----
```

6. Add this certificate to the client trust store.

Because you are creating a trust store file, you enter the password twice. If you add further certificates to this client trust store, you enter the password once.

```
prompt> keytool -importcert -file serverA.cer -keystore trustStore.ts
Enter keystore password:
Re-enter new password:
Owner: CN=serverA, OU=dev, O=itko, L=Dallas, ST=Texas, C=US
Issuer: CN=serverA, OU=dev, O=itko, L=Dallas, ST=Texas, C=US
Serial number: 4e155338
Valid from: Thu Jul 07 16:33:28 EST 2011 until: Wed Oct 05 17:33:28 EST 2011
Certificate fingerprints:
    MD5: 5B:10:F6:C8:02:3E:36:F5:AA:6D:FC:10:EF:F5:7F:54
    SHA1: 09:DA:8E:71:7C:D5:BB:44:89:14:13:07:F4:A1:C7:06:35:CD:BE:B1
    Signature algorithm name: SHA1withRSA
    Version: 3
Trust this certificate? [no]: yes
Certificate was added to keystore
```

Now you have a cryptographically strong way of talking to your DevTest servers in the public cloud. You must have the certificate on both sides for two DevTest components to talk to each other.

7. If your client talks to more than one remote SSL server, run the same keytool command to import the certificate to the trust store.



Note: In addition to the transport level security (the SSL), you can still enable fine-grain Access Control Lists (ACL). Access Control Lists let you require users to authenticate by user name and password. This type of security is similar to a banking website that uses HTTPS but still requires you to identify yourself.

Use SSL with Multiple Certificates

To have your local copy of DevTest configured to talk securely with multiple server certificates, add each server certificate to your local trustStore file.

In this example, we have serverA, serverB, and workstation.

The administrator of serverA wants to export the certificate using keytool:

```
serverA> keytool -exportcert -alias lisa -file serverA.cer -keystore serverA.ks
```

Similarly, the administrator for serverB wants to export the serverB certificate:

```
serverB> keytool -exportcert -alias lisa -file serverB.cer -keystore serverB.ks
```

Acquire a copy of **serverA.cer** and **serverB.cer**, and then import them into your client trust store:

```
workstation>keytool -importcert -alias serverA -file serverA.cer -keystore trustStore.ts
workstation>keytool -importcert -alias serverB -file serverB.cer -keystore trustStore.ts
```

Enter the password to your trustStore to modify it.

Ensure your workstation is using your trustStore, which now contains certificates for both serverA and serverB.

Copy this file to **LISA_HOME** and update **local.properties** as follows:

```
lisa.net.trustStore={{LISA_HOME}}trustStore.ts
lisa.net.trustStore.password_enc=33aa310aa4e18c114daf86a33cee898
```

When you run DevTest Workstation, you can select registries.

ssl://serverA:2010/Registry

and

ssl://serverB:2010/Registry

If you try to connect to **ssl://serverC:2010/Registry**, DevTest refuses the connection because you do not have the required certificate.

Mutual (Two-Way) Authentication

You can configure DevTest so that the server and client both need to authenticate each other. This type of authentication requires you to set a property on the server side:

```
lisa.net.clientAuth=true
```

In addition to each client needing a server certificate in the client trustStore, the server component needs a client certificate for each client in the server trustStore:

```
serverA>keytool -importcert -alias clientX -file clientX.cer -keystore trustStore.ts
serverA>keytool -importcert -alias clientY -file clientY.cer -keystore trustStore.ts
```

If clientZ attempts to connect to serverA, the connection fails because serverA does not have the clientZ certificate in the serverA trustStore. This failure occurs even though clientZ has the serverA certificate in the clientZ trustStore.

SSL/TLS Protocol Configuration

During HTTPS recording, playback, and live invocation, DevTest enables the following SSL/TLS protocols by default: TLS 1.0, SSL 3.0, and SSL 2.0.



If your installation of DevTest is using an IBM JVM, the default SSL/TLS protocols are only TLS 1.0 and SSL 3.0, as the IBM JVM does not support SSL 2.0.

To override these defaults, DevTest looks for the **https.protocols** system property to define which SSL/TLS protocols to enable. This property represents a comma-separated list of SSL/TLS protocols that the JVM supports. The values in this list must not have any quotes or white space before or after the commas.

Supported SSL/TLS Protocols and the Equivalent Values Expected by the JVM

SSL/TLS Protocol	Value Expected by JVM
TLS 1.2	TLSv1.2
TLS 1.1	TLSv1.1
TLS 1.0	TLSv1
SSL 3.0	SSLv3
SSL 2.0	SSLv2Hello

For example, setting the property to **TLSv1,SSLv3,SSLv2Hello** enables the TLS 1.0, SSL 3.0, and SSL 2.0 protocols.

There are two options for configuring the **https.protocols** system property.

Use the local.properties File to Configure All DevTest Components

Set the **https.protocols** system property in the **local.properties** file of your DevTest installation. Each component that is part of your DevTest installation (DevTest Workstation, VSE, Service Image Manager, for example) use this property when handling a SSL/TLS connection during HTTPS recording, playback, and live invocation.

For example, to make TLS 1.2, 1.1, and 1.0 the enabled SSL/TLS protocols for all components, add the following property to the **local.properties** file:

```
https.protocols=TLSv1.2,TLSv1.1,TLSv1
```

Use .vmoptions Files to Configure the SSL/TLS Protocols for Each Component

Configure the **https.protocols** system property on a per-component level by using the **.vmoptions** files in the **bin** directory of your DevTest installation.

For example, to make TLS 1.1 and 1.2 the enabled SSL/TLS protocols for the Virtual Service Environment, add the following line to the **bin/VirtualServiceEnvironment.vmoptions** file:

```
-Dhttps.protocols=TLSv1.1,TLSv1.2
```

Using HTTPS Communication with the invoke APIs

Complete the following tasks to enable HTTPS communication with the invoke APIs.

1. (Optional) Generate a New Key Pair and Certificate
2. Configure the Properties Files

(Optional) Generate a New Key Pair and Certificate

The simplest way to generate keys and certificates is to use the **keytool** application that comes with the JDK. This application generates keys and certificates directly into the keystore.

For more information, see <http://www.eclipse.org/jetty/documentation/current/configuring-ssl.html>.

Follow these steps:

1. Open a command prompt window.
2. Navigate to the **JAVA_HOME\bin** directory.
3. Type the following command:

```
keytool -keystore keystore -alias jetty -genkey -keyalg RSA
```



Note: You must use **jetty** as the alias.

This command prompts you for information about the certificate and for passwords to protect both the keystore and the keys within it.

4. Complete the following prompts.
 - **Enter the keystore password:**
The password is case-sensitive. The text of your password does not display.

- **Re-enter new password:**

The password is case-sensitive. The text of your password does not display.

- **What is your first and last name?**

[Unknown]:

Enter the same machine name that is used in the registry name. Normally, this is the unqualified host name of the server. For example, for a machine named jetty.eclipse.org, you would enter **jetty.eclipse.org**.

However, it is possible to start the registry with the -m command line parameter, using an IP address or a fully qualified host name. In these cases, the host name in the SSL certificate must match to prevent certificate errors in the web browser.



Note: This is the only mandatory prompt.

- **What is the name of your organizational unit?**

[Unknown]:

- **What is the name of your organization?**

[Unknown]:

- **What is the name of your City or Locality?**

[Unknown]:

- **What is the name of your State or Province?**

[Unknown]:

- **What is the two-letter country code for this unit?**

[Unknown]:

A confirmation of your entries displays.

5. Type **yes** to confirm.

The following prompt displays.

- **Enter key password for <jetty>**

<RETURN if same as keystore password>:

6. Press Enter.

The utility creates a new file named **keystore** in the current directory.

7. Copy the new keystore file to your LISA_HOME directory.

8. Rename the keystore file to **webserver.ks**.



Note: **webserver.ks** is the default file specified in the properties files. If you want to use a different file name, be sure to update the keystore location properties in the next section.

Configure the Properties Files

In this procedure, you configure the following files:

- **local.properties**
- **phoenix.properties**

Both files are in the **LISA_HOME** directory.



Note: The first time the system reads a password, it converts the password to an encrypted value.

Follow these steps:

1. Make a copy of the **_local.properties** file and save it as **local.properties**.
2. Uncomment and configure the following properties:
 - **dradis.webserver.https.enabled**
The default value is **true**. Do not change the value.
 - **dradis.webserver.ssl.keystore.location**
The default value is **{{LISA_HOME}}webserver.ks**. Change the value if you want to use a keystore file with a different name or in a different directory.
 - **dradis.webserver.ssl.keystore.password**
Set the value to the password that you defined when generating the keystore file.
 - **dradis.webserver.ssl.keymanager.password**
Set the value to the key manager password that you defined when generating the keystore file. Unless you specified a different password, this password is the same as the keystore password.
3. Uncomment and configure the following properties:
 - **lisa.webserver.https.enabled**
The default value is **true**. Do not change the value.
 - **lisa.webserver.ssl.keystore.location**
The default value is **{{LISA_HOME}}webserver.ks**. Change the value if you want to use a keystore file with a different name or in a different directory.
 - **lisa.webserver.ssl.keystore.password**
Set the value to the password that you defined when generating the keystore file.

- **lisa.webserver.ssl.keymanager.password**
Set the value to the key manager password that you defined when generating the keystore file. Unless you specified a different password, this password is the same as the keystore password.
 - **lisa.portal.url.prefix**
Set the value to **https://**.
4. Save the **local.properties** file.
 5. Make a copy of the **_phoenix.properties** file and save it as **phoenix.properties**.
 6. Uncomment and configure the following properties:
 - **registry.https.enabled**
The default value is **true**. Do not change the value.
 - **phoenix.https.enabled**
The default value is **true**. Do not change the value.
 - **phoenix.ssl.keystore**
The default value is **{{LISA_HOME}}webserver.ks**. Change the value if you want to use a keystore file with a different name or in a different directory.
 - **phoenix.ssl.keystore.password**
Set the value to the password that you defined when generating the keystore file.
 - **phoenix.ssl.keymanager.password**
Set the value to the key manager password that you defined when generating the keystore file. Unless you specified a different password, this password is the same as the keystore password.
 7. Save the **phoenix.properties** file.
 8. Restart the registry.

Using Kerberos Authentication

Kerberos support is similar to Basic Authentication and NTLM support in DevTest. DevTest uses Kerberos support when an application or resource that DevTest accesses through some of the steps is protected with Kerberos authentication. For example, HTTP/HTTPS, Web Service XML - the same steps as NTLM and Basic Authentication. Kerberos support uses the following properties in the [local.properties](https://docops.ca.com/display/DTS95/Local+Properties+File) (<https://docops.ca.com/display/DTS95/Local+Properties+File>) file:

- **lisa.java.security.auth.login.config**
The location of the login configuration file.
- **lisa.java.security.krb5.conf**
The location of the Kerberos configuration file that is used to override any preset locations.

- **`lisa.http.kerberos.principal`**

The name of the principal that is used for logging in when using DevTest support for principal + password authentication. When DevTest Workstation starts, it encrypts this principal.

- **`lisa.http.kerberos.pass`**

The password that is used for logging in when using DevTest support for principal + password authentication. When DevTest Workstation starts, it encrypts this principal.

You can authenticate with only the **`lisa.java.security.auth.login.config`** and the **`lisa.java.security.krb5.conf`** settings. These files and their settings vary depending on the operating system where DevTest runs. Consult the appropriate documentation about how to configure those two files for authentication that does not use DevTest support for principal + password authentication.

DevTest support for principal + password authentication

To support logging a user in by giving DevTest the credentials, the user must configure their login configuration file to use the DevTest login configuration file. The following example illustrates the contents of the file:

```
com.sun.security.jgss.initiate {
    com.itko.lisa.http.LisaKrb5LoginModule required doNotPrompt=false;
};
```

The custom **`LisaKrb5LoginModule`** is an extension of the standard **`com.sun.security.auth.module.Krb5LoginModule`** with one change. This extension submits the credentials in **`lisa.http.kerberos.principal`** and **`lisa.http.kerberos.pass`** instead of prompting the user for credentials.

Sample `krb5.conf` file

```
[libdefaults]
    default_realm = EXAMPLE.COM
    allow_weak_crypto = true

[realms]
    EXAMPLE.COM = {
        kdc = kdc.fakedomain.com:60088
    }

[domain_realm]
    .example.com = EXAMPLE.COM
    example.com = EXAMPLE.COM

[login]
    krb4_convert = true
```

Sample `krb5.conf` file with Active Directory as KDC

```
[libdefaults]
    default_realm = FAKEDOMAIN.COM
    allow_weak_crypto = false
    default_tkt_enctypes = arcfour-hmac-md5
    default_tgs_enctypes = arcfour-hmac-md5
    permitted_enctypes = RC4-HMAC arcfour-hmac-md5

[realms]
    FAKEDOMAIN.COM = {
        kdc = kdc.fakedomain.com
        master_kdc = kdc.fakedomain.com
        admin_server = kdc.fakedomain.com
```

```

        default_domain = FAKEDOMAIN.COM
    }
[domain_realm]
    fakedomain.com = FAKEDOMAIN.COM
[login]
    krb4_convert = true

```

Sample login.config file

```

com.sun.security.jgss.initiate {
    com.itko.lisa.http.LisaKrb5LoginModule required doNotPrompt=false;
};

```

Access Control (ACL)

Access control (ACL) is the mechanism that DevTest uses to authenticate users and enforce roles. ACL grants users access to only the application functionality they require to perform their role-based activities. ACL is required for DevTest so that you can monitor and maintain compliance to the license agreement. License agreements are based on the maximum number of concurrent user sessions.

You administer ACL using DevTest Portal.

Planning Deployment of ACLs

Starting with DevTest 8.0, controlling access through the Access Control Lists (ACL) system is required. Access to DevTest Workstation or DevTest Portal is not possible without authenticating against the ACL system.

Before you configure ACL, carefully consider how each user will use DevTest and the corresponding access that is required for each type of user.

By default, only the Super User and the System Administrator have administrative access to the ACL system.

ACL Administration

The ACL Administrator is responsible for the following activities:

- Creating users in the ACL system.
- Assigning roles to users, based on the responsibilities and required access of each user.
- Restricting access to various parts of DevTest by assigning components to Resource Groups.
- Assigning user access to defined resource groups.

The Administrator must have a thorough understanding of the various ACL roles and their associated privileges to assign the appropriate roles to each user.

Important! Do not use the default Super User or the System Administrator users to manage the ACL system. Use the default Super User to create new users with identical roles to the default Super User and System Administrator. Use the new users for managing the ACL system, and change the passwords for the default Super User and System Administrator users to prevent unauthorized access.

Using the Lightweight Directory Access Protocol (LDAP) with DevTest Solutions

You can also choose to manage the passwords for DevTest through LDAP, especially if an LDAP or Active Directory system is already available. When you use LDAP, user password changes are made by the LDAP administrator. The ACL administrator is no longer able to perform password changes.

Important! The implementation of the ACL system, and any integration with an LDAP service, remain the responsibility of the customer. If you need assistance with these implementation activities, contact CA Services. User administration through ACL is the responsibility of the ACL or LDAP administrator. CA Support is unable to progress cases where view access to the roles table is not available.

For example, a customer that reports a problem staging a test due to permission issues must ensure that the ACL or LDAP administrator is available when engaging CA support. ACL administrators must take these factors into account when assigning roles to their users and groups.

Authentication

You can determine how DevTest authenticates users. You can [manually add users \(see page 95\)](#) to the ACL database and specify credentials. The credentials are the user ID and password with which the user can log in to the DevTest user interface or command-line interface. Or, if your users are already defined with credentials in an LDAP database, you can use the LDAP server for authentication. In this case, you perform the steps in [Configure Authentication Providers for ACL \(see page 101\)](#).

Authorization

DevTest limits what features individual users can access based on their business role. DevTest is installed with over a dozen standard roles. You can experience how users with different roles experience DevTest by logging on as a standard user. [Standard users \(see page 93\)](#) are assigned unique [standard roles \(see page 86\)](#).

Important! To ensure security, the ACL administrator should change the default password for (admin, guest) as soon as possible to a password they will not forget.

When you manually add users to the ACL database, you assign a role to each user. The role grants a set of permissions. It is possible to assign multiple roles, but is rarely necessary as roles with more responsibility include permissions from lower related roles. When you use LDAP, the ACL is automatically populated with a row for each user. In this case, you assign only roles as described in [authorize users who LDAP authenticates](#).

The following activities are examples of the activities that you can control with permissions:

- Create a test case
- Create a staging document
- Stage a test case

User Sessions

When an authorized user logs in to a DevTest UI or CLI, a user session is created. User sessions are audited and form the basis of Usage Audit Reports. These reports include metrics and statistics on maximum concurrent user sessions by user type, where [user types \(see page 62\)](#) are categories that include multiple roles.

ACL and Backward Compatibility for CLIs and APIs That Ran Without Credentials

To access any DevTest user interface or command-line interface, users must log in with a valid user name and password. The requirement for credentials also applies to running tests and starting virtual services. If test cases that you automated in a previous release execute without credentials, you can temporarily override ACL so that they can continue to execute as scheduled. See [ACL and Command-Line Tools or APIs \(see page 84\)](#).

ACL Database

The ACL data is stored in the default internal Derby database after the installation. As outlined in *Installing*, the Derby database should be replaced with an enterprise database. For more information, see [Database Administration \(see page 36\)](#).

ACL and Command Line Tools or APIs

LISA releases 7.5.2 and before did not enforce ACL. DevTest Solutions enforces ACL by default. CA recognizes the need for transition time to update existing command-line tools and APIs that have been running without a specified user name and password. While you are updating your automated tests, virtual services, and programs like TestRunner with login credentials, you can set a parameter that tells your program to run with an internally derived Runtime user name. In this case, the identity of the user that is logged in to the OS is stored in the session object.

To override ACL temporarily, follow these steps:

1. Log on to the DevTest Server containing the in-use registry.
2. Navigate to the installation directory.
3. Open the local.properties file for edit.
4. Add the lisa.acl.use.runtime parameter and set it to true, that is:
`lisa.acl.use.runtime=true`
5. Save local.properties.
Your command-line program runs as ‘_runtime_<username>’ where ‘<username>’ comes from the ‘user.name’ system property.

To enforce ACL without exceptions, follow these steps:

1. Log on to the DevTest Server containing the in-use registry.
2. Navigate to the installation directory.
3. Open the local.properties file for edit.
4. Comment out the following parameter or set it to false.
`lisa.acl.use.runtime=`
5. Save local.properties
All UIs and CLIs run with valid login credentials.

Permission Types

Permissions can be divided into the following types:

- [Boolean \(see page 85\)](#)
- [Numeric Limit \(see page 85\)](#)

Boolean

Most of the permissions are Boolean. These permissions indicate whether an activity is allowed or not allowed.

For example, the Stop Registry permission indicates whether a user can stop the registry.

Numeric Limit

A permission can represent a numeric limit.

For example, the Stage Test permission allows a user to stage a test case with the specified maximum number of virtual users.

The value of a numeric limit permission must be -1, 0, or a positive integer.

If the value is -1, the permission is allowed and there is no numeric limit.

If the value is 0, the permission is denied.

Standard User Types and Standard Roles

DevTest Solutions creates standard user types with associated roles. Each role is associated with a unique set of [permissions \(see page 87\)](#). You can [change \(see page 95\)](#) the permissions that DevTest assigns to a standard role. In addition, you can [delete \(see page 95\)](#) a standard role.

- **CAI Power User and SV Power User**

The CAI Power User / SV Power User combination user types includes the following roles:

- Super User
- DevTest Administrator

- **CAI Power User**

The CAI Power User user type includes the following role:

- CAI Power

- **SV Power User**

The SV Power User user type includes the following roles:

- Test Administrator
- SV Power
- Test Runner

- **Test Power User**

The Test Power User user type includes the following roles:

- Test Power
- Test Observer
- Load Tester
- User

- **Runtime User**

The Runtime User user type includes the following roles:

- System Administrator
- Runtime
- Guest

Standard Permissions

A permission controls whether a user can perform a specific activity.

If a parent permission is allowed, then all its child permissions are allowed.

The following top-level permissions are automatically created.

- [User and Role Administration Permission \(see page 87\)](#)
- [Resource Administration Permission \(see page 87\)](#)
- [Test/Suite Administration Permission \(see page 87\)](#)
- [Virtual Services Administration Permission \(see page 88\)](#)
- [DevTest Server Administration Permission \(see page 89\)](#)
- [CVS Administration Permission \(see page 90\)](#)
- [Report Administration Permission \(see page 90\)](#)
- [Metric/Event Administration \(see page 91\)](#)
- [CAI Administration Permission \(see page 91\)](#)
- [DevTest Workstation Permission \(see page 92\)](#)

User and Role Administration Permission

The User and Role Administration permission is a Boolean permission that allows a user to manage access control (ACL).

Resource Administration Permission

The Resource Administration permission is a Boolean permission that allows a user to manage [resource groups \(see page 108\)](#).

Test/Suite Administration Permission

The Test/Suite Administration permission has the following child permissions.

Child Permission	Type	Description
Stage Test	Numeric Limit	Allows a user to stage a test case with the specified maximum number of virtual users.
Stage Suite	Numeric Limit	Allows a user to stage a suite with the specified maximum number of virtual users.
Stop Test	Boolean	Allows a user to stop a test.
Kill Test	Boolean	Allows a user to kill a test.
Optimize Test	Boolean	Allows a user to optimize a test.
View Test	Boolean	Allows a user to view a test.
Quick Stage Test	Boolean	Allows a user to stage a quick test.
Stage Local Suite	Boolean	Allows a user to run a suite locally.

Virtual Services Administration Permission

The Virtual Services Administration permission has the following child permissions.

Child Permission Level 1	Child Permission Level 2	Child Permission Level 3	Type	Description
View VSE Dashboard			Boolean	Allows a user to monitor VSEs.
VSE Server Administration			Boolean	
	Configure VSE Tracking Data Cleanup		Boolean	Allows a user to configure the process that deletes tracking data older than a specified amount of time.
	Stop VSE Server		Boolean	Allows a user to shut down a Virtual Service Environment.
	Reset VSE Server		Boolean	Allows a user to reset a Virtual Service Environment.
	Monitor VSE Server		Boolean	Allows a user to monitor a Virtual Service Environment.
VSE Service Administration			Boolean	
	VSE Service Deployment		Boolean	Allows a user to deploy a virtual service.
	VSE Service Archive Retrieval		Boolean	Allows a user to download the Model Archive (MAR) associated with a virtual service.
	VSE Service Execution		Boolean	
		Start VSE Service	Boolean	Allows a user to start a virtual service.
		Stop VSE Service	Boolean	Allows a user to stop a virtual service.
	Update Deployed VSE Service		Boolean	

	Update Virtual Service Capacity	Boolean	Allows a user to update the concurrent capacity for a deployed virtual service.
	Update Virtual Service Think Scale	Boolean	Allows a user to update the think time scale for a deployed virtual service.
	Update Virtual Service Auto-Restart	Boolean	Allows a user to update the auto-restart option for a deployed virtual service.
	Update Virtual Service Group Tag	Boolean	Allows a user to update the group tag on a deployed virtual service.
	Set Virtual Service Execution Mode	Boolean	Allows a user to select a new execution mode for a deployed virtual service.
	Reset Virtual Service Transaction Counts	Boolean	Allows a user to reset the transaction and error counts for a deployed virtual service.
	Heal Virtual Service Image	Boolean	Allows a user to perform model healing.

DevTest Server Administration Permission

The DevTest Server Administration permission has the following child permissions.

Child Permission	Type	Description
Debug DevTest Server	Boolean	Allows a user to create heap dumps, create thread dumps, and perform garbage collection for a server component.
Monitor Registry	Boolean	Allows a user to access the Registry Monitor.
Reset Registry	Boolean	Allows a user to reset the registry.
Stop Registry	Boolean	Allows a user to stop the registry.
Monitor Coordinator	Boolean	Allows a user to view a coordinator status message.
Reset Coordinator	Boolean	Allows a user to reset a coordinator.
Stop Coordinator	Boolean	Allows a user to stop a coordinator.
		Allows a user to view a simulator status message.

Monitor Simulator	Boolean	
Reset Simulator	Boolean	Allows a user to reset a simulator.
Stop Simulator	Boolean	Allows a user to stop a simulator.

CVS Administration Permission

The CVS Administration permission has the following child permissions.

Child Permission	Type	Description
View CVS Dashboard	Boolean	Allows a user to access the CVS tab in DevTest Portal.
Deploy or re deploy monitor to CVS	Boolean	Allows a user to deploy and redeploy CVS monitors.
Delete monitor from CVS	Boolean	Allows a user to delete CVS monitors.
Run monitor immediately on CVS	Boolean	Allows a user to run CVS monitors immediately, regardless of when they are scheduled to run.
Activate/Deactivate monitor on CVS	Boolean	Allows a user to activate and deactivate CVS monitors.

Report Administration Permission

The Report Administration permission has the following child permissions.

Child Permission	Type	Description
Access reporting console	Boolean	Allows a user to access the Reporting navigation menu.
View report data	Boolean	Allows a user to view his or her own report data.
View report data of other users	Boolean	Allows a user to view report data of other users.
View PDF report data	Boolean	Allows a user to view his or her own report data as a PDF file.
View PDF report data of other users	Boolean	Allows a user to view report data of other users as a PDF file.
Export Excel report data	Boolean	Allows a user to export his or her own report data as an Excel file.
Export Excel report data of other users	Boolean	Allows a user to export report data of other users as an Excel file.

Metric/Event Administration

The Metric/Event Administration permission has the following child permissions.

Child Permission	Type	Description
Add metric at runtime	Boolean	Allows a user to add a metric at runtime.
Remove metric at runtime	Boolean	Allows a user to remove a metric at runtime.
Pause metrics collection	Boolean	Allows a user to pause metric collection.
Save interval data	Boolean	Allows a user to save interval data.
Save metric data	Boolean	Allows a user to save metric data.
Save event data	Boolean	Allows a user to save event data.

CAI Administration Permission

The CAI Administration permission has the following child permissions.

Child Permission	Type	Description
View Paths	Boolean	Allows a user to view paths in DevTest Portal.
Create Baselines	Boolean	Allows a user to create baselines in DevTest Portal.
Create Virtual Service Models	Boolean	Allows a user to create virtual service models and raw traffic files in DevTest Portal.
Extract Data	Boolean	Allows a user to create data sets in DevTest Portal.
View Cases	Boolean	Allows a user to view tickets in DevTest Portal.
Edit Cases	Boolean	Allows a user to edit tickets in DevTest Portal.
View Agents	Boolean	Allows a user to view agents in DevTest Portal.
Agent Administration	Boolean	Allows a user to stop and start dispatching for an agent in DevTest Portal.
Import Paths	Boolean	Allows a user to import paths into DevTest Portal.
Export Paths	Boolean	Allows a user to export paths from DevTest Portal.
Delete Paths	Boolean	Allows a user to delete paths from DevTest Portal.
Create Document	Boolean	Allows a user to create a document from transactions in DevTest Portal.
View Documented Transactions	Boolean	Allows a user to view documented transactions in DevTest Portal.

Create Documented Transactions	Boolean	Allows a user to document transactions for testing in DevTest Portal.
Edit Point of Interest	Boolean	Allows a user to edit Point of Interest transactions in DevTest Portal.
View Point of Interest	Boolean	Allows a user to view Point of Interest transactions in DevTest Portal.
View Defects	Boolean	Allows a user to view transactions with defects in DevTest Portal.

DevTest Workstation Permission

The DevTest Workstation permission has the following child permissions.

Child Permission	Type	Description
Start DevTest Workstation	Boolean	Allows a user to start DevTest Workstation.
View a Configuration	Boolean	Allows a user to view configurations.
Edit a Configuration	Boolean	Allows a user to edit configurations.
Create a New Configuration	Boolean	Allows a user to create configurations.
View a Staging Document	Boolean	Allows a user to view staging documents.
Edit a Staging Document	Boolean	Allows a user to edit staging documents.
Create a New Staging Document	Boolean	Allows a user to create staging documents.
View a Suite	Boolean	Allows a user to view suites.
Edit a Suite	Boolean	Allows a user to edit suites.
Create a New Suite	Boolean	Allows a user to create suites.
View a Test Case	Boolean	Allows a user to view test cases.
Edit a Test Case	Boolean	Allows a user to edit test cases.
Create a New Test Case	Boolean	Allows a user to create test cases.
View an Audit Document	Boolean	Allows a user to view audit documents.
Edit an Audit Document	Boolean	Allows a user to edit audit documents.

Create a New Audit Document	Boolean Allows a user to create audit documents.
View Virtual Service Model	Boolean Allows a user to view virtual service models.
Edit Virtual Service Model	Boolean Allows a user to edit virtual service models.
Create a New Virtual Service Model	Boolean Allows a user to create virtual service models.
View Virtual Service Image	Boolean Allows a user to view service images.
Edit Virtual Service Image	Boolean Allows a user to edit service images.
Create a New Virtual Service Image	Boolean Allows a user to create service images.
Execute ITR	Boolean Allows a user to start the Interactive Test Run utility.
Execute Instant Replay	Boolean Allows a user to replay a test case to a specific point in the Interactive Test Run utility.
View ITR Properties	Boolean Allows a user to view the Properties tab in the Interactive Test Run utility.
View ITR Test Events	Boolean Allows a user to view the Test Events tab in the Interactive Test Run utility.

Standard Users

When you start the registry for the first time, standard users are created. Each standard user is assigned a role within a user type, and default password. See [Standard User Types and Standard Roles \(see page 86\)](#) for permissions that are associated with each role.



Important! To help prevent unauthorized access, we recommend that you change the default passwords for these users as soon as possible.

- **admin**
The admin user has the Super User role, a CAI Power User and SV Power User combination user type. The default password is **admin**.
- **caipower**
The caipower user has the CAI Power role, a CAI Power User user type. The default password is **caipower**.

- **svpower**
The svpower user has the SV Power role, an SV Power User user type. The default password is **svpower**.
- **tpower**
The tpower user has the Test Power role, a Test Power User user type. The default password is **tpower**.
- **devtest**
The devtest user has the Runtime role, a Runtime User user type. The default password is **devtest**.
- **sysadmin**
The sysadmin user has the System Administrator role, a Runtime user type. The default password is **sysadmin**.
- **guest**
The guest user has the Guest role. The default password is **guest**.

Change Passwords for Standard Users

When you start the registry for the first time, seven standard users are created. Each standard user is assigned a role, user type, and default password. To help prevent unauthorized access, we recommend that you change the default passwords for these users as soon as possible.

Follow these steps:

1. Ensure that DevTest Solutions is running.
2. Browse to the DevTest Portal.
`http://localhost:1507/devtest/#/main/dashboard`
3. Log in to the DevTest Portal. If you have no credentials, log in as *admin*.
4. Click **Settings**, then **Access Control**.
5. Click **Users**.
The standard users are displayed.
6. For each standard user, perform the following steps:
 - a. In the **Actions** column, click the **Options** icon.
 - b. Click **Set Password**.
 - c. Type a new password in the **Password** field.
 - d. Type the new password in the **Confirm Password** field.
 - e. Click **Save**.

View User Information from DevTest Workstation

When you are logged in to DevTest Workstation, you can open a dialog that lets you view the following user information:

- The unique ID of your user name
- The roles that are assigned to you
- The permissions that you have

Follow these steps:

1. Select **System, View Security Permissions** from the main menu.
The **User Security Permissions** dialog opens.
2. When you finish viewing the information, close the dialog.

Manage Users and Roles

Administrators can manage users and roles from the Access Control option in the **DevTest Portal**. Consider the following approach:

1. Review roles and user types.
 - a. Select **Roles**.
 - b. Notice that the available roles are listed with the associated user types. For example, the Test Administrator role is associated with the SV Power User user type.
 - c. Your license agreement specifies the maximum number of concurrent users that is allowed for each user type. Keep this figure in mind as you assign roles to users. Notice that many roles map to the same user type.
 - d. Examine permissions that are associated with each role and, optionally, customize permissions.
 - e. Optionally, add a new role that is based on a standard role. The new role inherits the user type of the role on which it is based.
2. Add all DevTest users.
 - a. Select **Users**.
 - b. DevTest uses the user ID and password you specify to authenticate the named user.
 - c. DevTest uses the role that you specify to authorize the user to perform various activities that are based on the granted permissions.
3. Verify that you have satisfactorily configured all your users with the appropriate roles. If you customized existing roles or added new ones, verify the configuration.

4. **Back up your database** strategically according to your in-house maintenance policies. This is a precaution that enables you to recover your configuration of users and roles quickly if there is database corruption. The resolution to a corrupted ACL database is to perform a Restore that should be managed by your database administrator.



More Information:

- [Manage Users \(see page 96\)](#)
- [Add and Update Roles \(see page 97\)](#)
- [User Activity Report \(see page 100\)](#)

Manage Users

You use **DevTest Portal** to add users, change the details for a user, copy a user, and delete users. The details that you can change include the password.

User IDs and passwords differ regarding case-sensitivity.

- User IDs are not case-sensitive. For example, the user IDs **AAAA1** and **aaaa1** are treated as the same value.
- Passwords are case-sensitive.

You cannot delete the user that you are currently logged in as.

You can show or hide any of the columns on the **Users** tab. Click the drop-down arrow in a column. Select **Hide Column** to hide the column. To show a hidden column, select the **Filter** icon and click the name of the hidden column.

To add a user:

1. In **DevTest Portal**, click **Settings, Access Control, Users**.
2. Click **Add User**. The **Add New User** dialog appears.
3. In the **User ID** field, enter a unique ID for the user.
You can enter any combination of alphanumeric, hyphen (-), underscore (_), period (.), and ampersand (@) characters. The maximum number of characters is 100.
4. In the **Name** field, enter the user name.
You can enter any combination of alphanumeric, hyphen (-), underscore (_), and space characters. The maximum number of characters is 100.
5. In the **Description** field, enter a description for the user.

6. In the **Password** field, enter a password for the user.
You can enter any combination of alphanumeric, hyphen (-), underscore (_), and ampersand (@) characters. Passwords must have at least five characters.
7. In the **Confirm Password** field, enter the password again.
8. Click the arrow between the columns to move a role from the **Available** column to the **Assigned** column.
To filter the lists of roles, type in the **Type your filter** field at the top of the column.
9. Click **Save**.

To update a user:

1. In **DevTestPortal**, click **Settings, Access Control, Users**.
2. Select a user from the list on the left.
3. On the **User Profile** tab, click the arrow between the columns to move a role from the **Available** column to the **Assigned** column.
To filter the lists of roles, type in the **Type your filter** field at the top of the column.
4. (Optional) To change a user password, in the **Action** column, select the **Options** icon, then **Set Password**.
5. Click **Save**.

To copy a user:

1. In **DevTestPortal**, click **Settings, Access Control, Users**.
2. Select a user from the list on the left.
3. In the **Action** column, select the **Copy** icon.
4. The **Add New User** dialog appears on the right.
5. Complete the dialog as previously described in **How to add a user**.

To delete a user:

1. In the **Users** tab, select the user name and select the check box on the left for as many users as you are deleting.
2. Click **Delete User(s)**.
3. Click **Yes**.

Add and Update Roles

You use **DevTest Portal** to add roles, change the details for a role, and delete roles.

DevTest Portal displays the roles and permissions in the **Roles** tab.

- Each role is shown with its effective user type. This identifies the user type that is reported in the [Usage Audit Report \(see page 56\)](#). Selecting a role displays the **Permissions** tab, with available permissions shown in the left column and permissions that are assigned to the role in the right column.
- When a user has more than one role, all the permissions from each role are aggregated into a single list of permissions from which actions are interrogated.

When you select a role, the **Permissions** tab on the right is populated. On this tab, if you select a parent permission, the child permissions are automatically selected. If you clear a parent permission, the child permissions are automatically cleared. If you clear a child permission, the parent check becomes a dash to indicate partial child selection. If you unselect the last child permission, the parent becomes unselected.

You can show or hide any of the columns on the **Roles** tab. Click the drop-down arrow in a column. Select **Hide Column** to hide the column. To show a hidden column, select the **Filter** icon and click the name of the hidden column.

To add a role:

1. In DevTest **Portal**, click **Settings, Access Control, Roles**.
2. At the upper left, click **Add Role**.
The **Add New Role** tab appears.
3. In the **Name** field, enter the role name.
You can enter any combination of alphanumeric, hyphen (-), underscore (_), and space characters. The maximum number of characters is 50.
4. In the **Description** field, enter the role description.
The maximum number of characters is 200.



The **All Effective User Types** field identifies all the user types that this role consumes. For example, there is at least one permission from each of the identified user types assigned to the role.

5. From the **Permissions** tab, select a permission or permissions by highlighting the permission's row.
6. Click the arrow between the columns to move a permission from the **Available** column to the **Assigned** column.
To filter the lists of permissions, type in the **Type your filter** field at the top of the column.
7. Click **Save**.
8. Use the same process on the **Users** and **Resource Groups** tabs to add users and resource groups to a role.

Roles tab

Role Information - Test Power

*Name: Test Power

Description: Test Power

All Effective User Types: Runtime User (R) , Test Power User (TP) , Administration User (A)

Permissions	Users in Role	Resource Groups in Role
Available		Assigned
Type your filter		Type your filter
<input type="checkbox"/> DevTest Console Administration		<input type="checkbox"/> DevTest Console Administration
<input type="checkbox"/> User and Role Administration (A)		<input type="checkbox"/> Test/Suite Administration
<input type="checkbox"/> Resource Administration (A)		<input type="checkbox"/> Virtual Services Administration
<input type="checkbox"/> Virtual Services Administration		<input type="checkbox"/> DevTest Server Administration
<input type="checkbox"/> DevTest Server Administration		<input type="checkbox"/> CVS Administration
<input type="checkbox"/> CAI Administration		<input type="checkbox"/> Report Administration
<input type="checkbox"/> DevTest Workstation		<input type="checkbox"/> Metric/Event Administration
		<input type="checkbox"/> DevTest Workstation
		<input type="checkbox"/> Cloud Lab Integration

Roles tab

To update a role:

1. In DevTest **Portal**, click **Settings, Access Control, Roles**.
2. Select a role from the list on the left.
3. On the **Permissions** tab, select a permission and select the **Expand** icon to see its child permissions.
4. To update a [Boolean \(see page 85\)](#) permission, select or clear the check box.
5. To update a [numeric limit \(see page 85\)](#) permission:
 - a. Click **Numeric Limits**.
 - b. Enter a value in the numeric limit.
 - c. Click **Save**.
6. To assign a user type to a role quickly, click the **Quick Selection** button and select a user type from the drop-down.
7. Click **Save**.

Role Information - Test Administrator Save

*Name:

Description:

All Effective User Types: Runtime User (R) , Administration User (A) , Test Power User (TP) , SV Power User (SVP)

Permissions **Users in Role** Resource Groups in Role

Available

Type your filter

- ☐ CAI Administration
- ☐ DevTest Console Administration
- ☐ CVS Administration
- ☐ User and Role Administration (A)
- ☐ DevTest Server Administration
- ☐ Resource Administration (A)
- ☐ Report Administration
- ☐ Metric/Event Administration

Assigned Quick Selection

Type your filter

- ☐ DevTest Console Administration
- ☐ Virtual Services Administration
- ☐ DevTest Workstation
- ☐ DevTest Server Administration
- ☐ Test/Suite Administration
 - ☐ Stage Test (R)
 - ☐ Stage Suite (R) Unlimited
 - ☐ Stop Test (R)
 - ☒ Kill Test (R)
 - ☒ Optimize Test (R)
 - ☐ View Test (R)
 - ☐ Quick Stage Test (R)
 - ☐ Stage Local Suite (R)
- ☐ Cloud Lab Integration

Save Cancel

Example of updating permissions within a role

To delete a role:

1. In the **Roles** tab, select the role name and select the check box on the left for as many roles as you are deleting.
2. Click **Delete Role(s)**.
3. Click **Yes**.

User Activity Report

The User Activity report contains a series of entries for user activities that ACL controls. You access the report from the Administration window. Open the window by selecting **Reporting, Administration** from the left navigation menu. System administrators can search and filter the report for a specific event and user.

For more information about viewing and manipulating reports see [View and Navigate Reports \(https://docops.ca.com/display/DTS95/View+and+Navigate+Reports\)](https://docops.ca.com/display/DTS95/View+and+Navigate+Reports).



More Information:

- [Testing Reports \(https://docops.ca.com/display/DTS95/Testing+Reports\)](https://docops.ca.com/display/DTS95/Testing+Reports)

- [Virtual Service Metrics Reports \(https://docops.ca.com/display/DTS95/Virtual+Service+Metrics+Reports\)](https://docops.ca.com/display/DTS95/Virtual+Service+Metrics+Reports)
- [CAI Top N and Metrics Reports \(https://docops.ca.com/display/DTS95/CAI+Top+N+and+Metrics+Reports\)](https://docops.ca.com/display/DTS95/CAI+Top+N+and+Metrics+Reports)

Add Users With the adduser Command-Line Utility

Use the **adduser** command-line utility to add a user. The utility is located in the **LISA_HOME\bin** directory of a DevTest Server installation. To use the utility, you must have the User and Role Administration permission.

This utility has the following format:

```
adduser -d userid -w password [-r role] -u userid -p password [-m registry_url]
```

The following options let you assign basic information to the new user:

- Use the **-d** or **--adduser** option to assign a user ID to the new user.
- Use the **-w** or **--addpassword** option to assign a password to the new user.
- (Optional) Use the **-r** or **--rolename** option to assign a role to the new user. If you do not assign a role, the user has no permissions until a role is assigned.

The following options let you specify your credentials:

- Use the **-u** or **--username** option to specify your user ID.
- Use the **-p** or **--password** option to specify your password.

If the registry is on a remote computer, use the **-m** or **--registry** option to specify the registry URL.

Example

This example adds a user who is named **user1**. Because the role name has more than one word, quotation marks are used.

```
adduser -d user1 -w password1 -r "Load Tester" -u admin -p myadminpassword
```

Configure Authentication Providers for ACL

You can configure access control (ACL) so that user authentication is based on the information in an LDAP server, multiple LDAP servers, the database, or LDAP servers and the database. The ACL administrator should consult with your LDAP administrator for configuration and implementation that is based on the following properties.

If LDAP successfully authenticates the user and the user does not exist in the database, the user can be automatically added to the database.



Note: To use this LDAP integration, you cannot have DevTest Workstations older than 8.4. For example, if you upgrade the registry to 8.4, and you then leverage LDAP integration, all the DevTest Workstations under that registry must also be upgraded to 8.4.

Follow these steps:

1. Edit the **authentication-providers.xml** file in the home directory,



Note: The **authentication-providers.xml** file must exist in the home directory.

2. Authentication providers that are listed in this file are tried in the order they appear in the file. Enter as many authentication providers as you need, in the order they should be used.

3. Complete the following fields for each authentication provider:

- **name**

The label for this authentication provider.

Required

- **type**

The type of authentication provider.

Required

Values: LDAP, ActiveDirectory, Embedded, Custom, or Legacy. These values are case-sensitive and **Legacy** is deprecated and will be removed in a future release.

The <authentication-provider> element has two required attributes of **name** and **type**. The name attribute provides a user-defined name that is associated with the respective authentication provider that is used in the UI and in error reporting. The **type** attribute determines the category of the authentication provider and how it is constructed

ActiveDirectory or LDAP

When the type is defined with **ActiveDirectory** or **LDAP** then an internal AuthenticationProviderFactory is used to create an authentication provider that is used for looking up users in the Microsoft Active Directory server or a normal LDAP server. All attributes are valid except for **factory-class**.

Example:

```
<authentication-provider
  name="Corp. Active Directory Server"
  autoAddUsers="false"
  authenticateOnly="false"
  enabled="true"
  type="ActiveDirectory"
  defaultRole="SV Power"
  rejectUnmappedUsers="true">
  <url>ldaps://server.example.com:3269</url>
  <user-dn>cn=readOnlyUser,ou=users,dc=example,dc=com</user-dn>
  <user-password>drowssap</user-password>
```

```

<user-dn-pattern>cn={0},ou=users,dc=example,dc=com</user-dn-pattern>
<user-search-base>dc=example,dc=com</user-search-base>
<user-search-filter>(objectClass=user)</user-search-filter>
<group-search-base>ou=groups</group-search-base>
<group-search-filter>(member={0})</group-search-filter>
</authentication-provider>

```

Embedded

The **Embedded** type creates an authentication provider for the internal DevTest database. Only **name**, **type**, and **enabled** are valid attributes.

Example:

```

<authentication-provider
  name="DevTest ACL Database"
  type="Embedded"
  enabled="true"/>

```

Legacy

A type attribute value of **Legacy** creates an authentication provider that is based off the older and now *deprecated* `com.itko.lisa.acl.IAuthenticationModule` interface. Users of the `IAuthenticationModule` who are using it to access the internal DevTest database or an external LDAP/Active Directory server can switch to using the direct type replacements of **Embedded** and **LDAP/ActiveDirectory**. Any other implementations should be re-implemented as **Custom** authentication providers.

Only **name**, **type**, **enabled**, and **defaultRole** are valid attributes.

```

<authentication-provider
  name="ITKO Authentication Module"
  type="Legacy"
  enabled="true"
  defaultRole="Guest"/>

```

Custom

A value of **Custom** for the type attribute allows you to create your own `DevTestAuthenticationProvider` instance. When the value of the type attribute is **Custom**, then you must also define the **factory-class** attribute and the value of the **factory-class** attribute must be the fully qualified name of a Java class that implements the `com.ca.dts.security.authentication.AuthenticationProviderFactory` interface.

Only **name**, **type**, **enabled**, **defaultRole**, and **factory-class** are valid attributes. Any subelements like `<property1>value1</property1>` are passed to the instantiated factory-class as properties with the element name (`property1`) being the key and the text of the element (`value1`) as the value.

Example:

```

<authentication-provider
  name="My Authentication Provider"
  type="Custom"
  enabled="true"
  defaultRole="Guest"
  factory-class="com.example.FooAuthenticationProviderFactory">
  <property1>value1</property1>
  <property2>value2</property2>
</authentication-provider>

```

- **enabled**
Controls whether this authentication provider is available for authenticating users.
Values: true, false
Default: true
- **autoAddUsers**
Controls whether successfully authenticated users are automatically added to the database.
When this field has the value of *false*, you must explicitly create accounts in for your LDAP users.
If you enable this field, you must manually remove users from the DevTest database as users are removed from the system.
Values: true, false
Default: true
- **defaultRole**
The role to assign to new users upon successful login if they have no other roles assigned.
Default: Guest
- **rejectUnmappedUsers**
Prevent users with no LDAP groups mapped to roles from logging in. If this value is set to *false*, and a user has no entry in the **ldap-mapping.xml** file, the user is given the role that is associated with the **defaultRole** parameter.
Values: true, false
Default: true
- **authenticateOnly**
Controls whether LDAP/AD is only used to authenticate a user. When this field has the value of *true*, you must explicitly create accounts in the ACL database or you must configure **autoAddUsers** to *true*.
Values: true, false
Default: false
If the value of the **authenticateOnly** attribute is **true**, then the **rejectUnmappedUsers** attribute must be **false**
- **url**
The URL of the server.
- **user-dn**
The distinguished name of the LDAP user that is used to connect to the server.
- **user-password**
The password of the LDAP user that is used to connect to the server. Unencrypted passwords will automatically be encrypted shortly after you save the file when the registry is running.
- **user-dn-pattern**

A pattern that is used to generate a distinguished name for the LDAP/AD user who is used to bind to the server. This element may be specified 0 or more times and the patterns are tried in the order in which they occur in the <authentication-provider> element. The pattern argument {0} will contain the username. An example is: "cn={0},ou=users,dc=example,dc=com".
- **user-search-base**
The relative name to use in searching for users.

- **user-search-filter**

The LDAP filter that is used to find user entries.

The search filter can include the placeholder '{0}' that contains the username of the user trying login. As an example, if the filter '(&(objectClass=user)(sAMAccountName={0}))' used and a user with username of demoUser, then the filter will return only entries that have the objectClass of user and a sAMAccountName attribute value of 'demoUser'. The user-search-filter should only return a single entry.

- **group-search-base**

The relative name to start searching for groups.

The search filter can include the placeholder '{0}' that contains the username of the user trying login. As an example, if the filter '(member={0})' used and a user with username of demoUser, then the filter will return only entries that have an attribute entry with 'member' as the attribute name and demoUser as the value. The group-search-filter can return 0 or more entries.

- **group-search-filter**

The LDAP filter that is used to find group entries.

- **factory-class**

The fully qualified name of a Java class that implements the **com.ca.dts.security.authentication.AuthenticationProviderFactory** interface.



Note: If you use Microsoft Active Directory as your LDAP server, use the Global Catalog port number in the LDAP configuration. The Global Catalog port number is 3268 by default, or 3269 by default if using SSL. You may need to contact your system administrator to determine the correct port number to use.

Using a Certificate with LDAP

Follow these steps:

1. Import the trusted public certificate for the LDAP server into the Java keystore for the JRE that uses.
2. For a normal install, the jre keystore is the keystore at **LISA_HOME\jre\lib\security\cacerts**.
3. Execute the following command in the LISA_HOME directory. <aliasname> can be anything that you choose, and <public_cert_file> is the path and filename for the public certificate from the LDAP server.

```
jre\bin\keytool -import -alias <aliasname> -keystore jre\lib\security\cacerts -trustcacerts -file "<public_cert_file>" -storepass changeit -noprompt
```

More Information:

For information about adding LDAP users to the database, see [Authorize Users Authenticated by LDAP \(see page 106\)](#).

Authorize Users Authenticated by LDAP

Use the **ldap-mapping.xml** file in the DevTest home directory to assign roles to groups of LDAP users.

Follow these steps:

1. Edit the **ldap-mapping.xml** file.
2. For each role, enter the LDAP distinguished name for each LDAP group to associate with that role.
3. A sample **ldap-mapping.xml** file appears below.

```
<?xml version="1.0" encoding="UTF-8" ?>

<mappings>
  <mapping role="Super User">
    <groupDN>cn=administrators,ou=groups,dc=example,dc=com</groupDN>
    <groupDN>CN=Team - Forward Cars - QA2 - Team Only,OU=Groups,OU=North
America,DC=ca,DC=com</groupDN>
  </mapping>
  <mapping role="DevTest Administrator">
  </mapping>
  <mapping role="Test Administrator">
  </mapping>
  <mapping role="System Administration">
    <groupDN>cn=administrators,ou=groups,dc=example,dc=com</groupDN>
  </mapping>
  <mapping role="CAI Power">
  </mapping>
  <mapping role="SV Power">
  </mapping>
  <mapping role="Test Power">
  </mapping>
  <mapping role="Runtime">
  </mapping>
  <mapping role="Test Runner">
  </mapping>
  <mapping role="Test Observer">
  </mapping>
  <mapping role="Load Tester">
  </mapping>
  <mapping role="User">
    <groupDN>cn=users,ou=groups,dc=example,dc=com</groupDN>
    <groupDN>CN=Team - Forward Cars - All,OU=Groups,OU=North America,DC=ca,
DC=com</groupDN>
  </mapping>
  <mapping role="Guest">
  </mapping>
</mappings>
```

In the example, members of the **Team - Forward Cars - QA2 - Team Only** distinguished name group are given the role of Super User, and members of **Team - Forward Cars - All** are given the role of User. If a user was a member of both groups, both roles and the permissions for each role are assigned.

To update the DevTest Users table by making individual role assignments

1. Ask all DevTest users to log in to DevTest Solutions and then log out.

2. Browse to the DevTest Portal and log in.
3. Click **Settings, Access Control, Users**.
The DevTest Users table opens.
4. For each user, clear the default role and select the appropriate role.
5. (Optional) To view permissions for a role in the right pane of the User Profile dialog, click the icon to the right of the role name.
6. Click **Save**.

ACL Configuration Scenarios

This section provides examples of various ACL configuration scenarios.

- [Authenticate with LDAP/AD and manage user roles in DevTest \(see page 107\)](#)
- [Authenticate with LDAP/AD and manage user roles with mapped LDAP groups and reject unmapped users \(see page 107\)](#)
- [Authenticate with LDAP/AD and manage user roles with mapped LDAP groups and allow unmapped users to be assigned a default role \(see page 108\)](#)

Authenticate with LDAP/AD and manage user roles in DevTest

```
<authentication-provider
  name="Corp. Active Directory Server"
  autoAddUsers="false"
  authenticateOnly="true"
  enabled="true"
  type="ActiveDirectory"
  defaultRole="SV Power"
  rejectUnmappedUsers="false">
  <url>ldaps://server.example.com:9999</url>
  <user-dn>cn=readOnlyUser,ou=users,dc=example,dc=com</user-dn>
  <user-password>drowssap</user-password>
  <user-dn-pattern>cn={0},ou=users,dc=example,dc=com</user-dn-pattern>
  <user-search-base>dc=example,dc=com</user-search-base>
  <user-search-filter>(objectClass=user)</user-search-filter>
  <group-search-base>ou=groups</group-search-base>
  <group-search-filter>(member={0})</group-search-filter>
</authentication-provider>
```

Authenticate with LDAP/AD and manage user roles with mapped LDAP groups and reject unmapped users

```
<authentication-provider
  name="Corp. Active Directory Server"
  autoAddUsers="false"
  authenticateOnly="false"
  enabled="true"
  type="ActiveDirectory"
  defaultRole="SV Power"
  rejectUnmappedUsers="true">
  <url>ldaps://server.example.com:3269</url>
  <user-dn>cn=readOnlyUser,ou=users,dc=example,dc=com</user-dn>
  <user-password>drowssap</user-password>
  <user-dn-pattern>cn={0},ou=users,dc=example,dc=com</user-dn-pattern>
  <user-search-base>dc=example,dc=com</user-search-base>
  <user-search-filter>(objectClass=user)</user-search-filter>
```

```
<group-search-base>ou=groups</group-search-base>
<group-search-filter>(member={0})</group-search-filter>
</authentication-provider>
```

Authenticate with LDAP/AD and manage user roles with mapped LDAP groups and allow unmapped users to be assigned a default role

```
<authentication-provider
  name="Corp. Active Directory Server"
  autoAddUsers="false"
  authenticateOnly="false"
  enabled="true"
  type="ActiveDirectory"
  defaultRole="SV Power"
  rejectUnmappedUsers="false">
  <url>ldaps://server.example.com:3269</url>
  <user-dn>cn=readOnlyUser,ou=users,dc=example,dc=com</user-dn>
  <user-password>drowssap</user-password>
  <user-dn-pattern>cn={0},ou=users,dc=example,dc=com</user-dn-pattern>
  <user-search-base>dc=example,dc=com</user-search-base>
  <user-search-filter>(objectClass=user)</user-search-filter>
  <group-search-base>ou=groups</group-search-base>
  <group-search-filter>(member={0})</group-search-filter>
</authentication-provider>
```

Resource Groups

Resource groups are one or more DevTest Servers or VSEs. Define resource groups to determine the resources that a user or a project can access.

When using ACL, associate roles with resource groups to determine which roles can act on which resources.



Tip: If you have a small environment, such as a site that has a local registry and a VSE, you do need to set up resource groups to control access such as restricting VSE specific access. However, you may want to set up Resource Groups to limit the access between your Production and your Development group.

Manage Resource Groups

Use the **DevTest Portal** to add resource groups, change the details for a resource group, delete resource groups, and remove a resource from a resource group.

You can show or hide any of the columns on the **Resource Groups** tab. Click the drop-down arrow in a column. Select **Hide Column** to hide the column. To show a hidden column, select the **Filter** icon and click the name of the hidden column.

To add a resource group:

1. In the **DevTest Portal**, click **Settings, Access Control, Resource Groups**.
2. Click **Add Resource Group**. The **Add New Resource Group** dialog appears.

3. In the **Name** field, enter a unique name for the resource group. The name must contain only alphanumeric characters, spaces, hyphens, or underscores.
4. In the **Description** field, enter a description of the resource group.
5. From the **Resources** tab, select a resource or resources by highlighting the group's row.
6. Click the arrow between the columns to move a resource from the **Available** column to the **Assigned** column.
To filter the lists of resources, type in the **Type your filter** field at the top of the column.
7. Click **Save**.
The Resource Groups tab shows the resource with the associated resource group displayed in the **Resource Groups** column.

To display resource groups:

1. To refresh the display, click **Refresh** at the upper right corner of the window.
2. Resources that are inactive, but still associated with a resource group, display with a Disconnected status.

To delete a resource group or groups:

1. In the **Action** column for the resource group, click the **Delete** icon.
2. From the drop-down list, select **Delete Resource Group**.
3. Click **Delete** in the **Please Confirm** dialog.
4. To delete multiple resource groups, select the check box to the left of the groups you are deleting and click the **Delete Resource Group(s)** button.

To copy a resource group:

1. In the **Action** column for the resource group, click the **Copy** icon.
2. A copy of the resource group appears in the **Add New Resource Group** dialog.
3. Continue as if you were adding a resource group.

To remove a resource from a resource group:

1. From the **Resources** tab, highlight the resource in the **Assigned** column and click the left arrow at the top of the column.
2. Click **Save**.

Grant Roles to Resource Groups

To grant roles to resource groups, follow these steps.

1. In the **DevTest Portal**, click **Settings, Access Control, Resource Groups**.

2. Select a resource group.
3. Select the **Roles in Resource Group** tab.
4. In the **Available** column, select a role, then click the right arrow between the columns to move it into the **Assigned** column.
5. Click **Save**.

Use Resource Groups to Control Access

You can use resource groups to control access to resources in addition to using Access Control (ACL).

Follow these steps:

1. Add the resource groups that are appropriate for your organization. For instructions, see *Manage Resource Groups*.
2. To open a configuration file in DevTest Workstation, double-click it from the **Project Panel**.
3. Click **Add** at the bottom of the panel.
4. Click the **Key** column in the **Properties Editor**.
5. From the list of properties, select **RESOURCE_GROUP**.
6. Click the **Value** column.
7. Select the resource group that you want to associate with this configuration.
You can only select one resource group from the drop-down list, but you can edit the **Value** column to enter a list of resource groups, which are separated by commas.
8. To save the configuration, click **Save**.

Any test case, test suite, or VSM that you stage using this configuration is restricted to using the resources that are in the resource group.

Note: The resource group association is only in effect when the configuration is the active configuration.

Resources

Resources are coordinators, servers, and VSEs.

Manage Resources

Use the **DevTest Portal** to delete disconnected resources and modify the assignment of resource groups for a selected resource.

You can show or hide any of the columns on the **Resources** tab. Click the drop-down arrow in a column. Select **Hide Column** to hide the column. To show a hidden column, select the **Filter** icon and click the name of the hidden column.

To delete a resource or resources:

You can only delete resources that are disconnected.

1. In the **DevTest Portal**, click **Settings, Access Control, Resources**.
2. In the **Action** column for the resource, click the **Delete** icon.
3. From the drop-down list, select **Delete Resource**.
4. Click **Delete** in the **Please Confirm** dialog.
5. To delete multiple resources, select the check box to the left of the resources you are deleting and click the **Delete Resource(s)** button.

To change the assignment of resource groups for a resource:

1. In the **DevTest Portal**, click **Settings, Access Control, Resources**.
2. Select a resource by highlighting the resource's row.
The **Resource Information** tab displays.
3. Click the arrow between the columns to move a resource group from the **Available** column to the **Assigned** column.
To filter the lists of resource groups, type in the **Type your filter** field at the top of the column.
4. Click **Save**.

Logging

This section contains the following pages that provide details about logging in DevTest:

- [Logging Properties File \(see page 113\)](#)
- [Status Messages for Server Components \(see page 115\)](#)
- [Automatic Thread Dumps \(see page 115\)](#)
- [Test Step Logger \(see page 116\)](#)
- [VSE Logging \(see page 117\)](#)

Main Log Files

The main log files include:

- **coordinator.log**: the logging output for the coordinator.
- **cvsmgr.log**: the logging output for the Continuous Validation Service (CVS).
- **devtest_broker_pid.log**: the logging output for the broker component of the DevTest Java Agent.
- **devtest_console_pid.log**: the logging output for the console component of the DevTest Java Agent.
- **marmaker.log**: the logging output for the Make Mar command-line utility.
- **pfbroker.log**: the cumulative logging output for the broker component of the DevTest Java Agent.
- **portal.log**: the logging output for DevTest Portal.
- **registry.log**: the logging output for the registry.
- **simulator.log**: the logging output for the simulator.
- **svcimgmgr.log**: the logging output for the Service Image Manager command-line utility.
- **svcmgr.log**: the logging output for the Service Manager command-line utility.
- **trunner.log**: the logging output for the Test Runner command-line utility.
- **vse.log**: the logging output for the VSE server.
- **vsemgr.log**: the logging output for the VSE Manager command-line utility.
- **vse_xxx.log**: the logging output for VSE conversations and service image navigation, where xxx is the service image name.
- **workstation.log**: the logging output for DevTest Workstation.

Generally, looking at the main log for the component in question or suspected to have an issue should be enough to get an idea of the problem. The main component logs are: Registry, Coordinator, Simulator, and VSE.

The **`lisa.tmpdir`** property controls the location of these log files. To see the value of the **`lisa.tmpdir`** property from DevTest Workstation:

1. Click **Help, DevTest Runtime Info** from the main menu.
2. In the **System Properties** tab, locate **`lisa.tmpdir`**.

If the registry, coordinator, simulator, or VSE is [running as a Windows service \(see page 19\)](#), the log files are located in the **`LISA_HOME\lisatmp`** directory.

Note: Although **`lisa.tmpdir`** allows you to change the location of where you can store your temporary files, we do not recommend that you change this property to save the temporary files out to an external mount point or external share. If you encounter issues with product instability, and you are using an external share for temp file storage, Support might instruct you to go back to using a local disk for temp file storage for continued support of your environment.

Demo Server Log Files

The demo server has its own log files, which are located in the **`lisa-demo-server/jboss/server/default/log`** directory.

The **`lisa.tmpdir`** property does not control this location.

The demo server log files are:

- `boot.log`
- `server.log`

Logging Properties File

DevTest uses the Apache log4j logging framework. The **`logging.properties`** (<https://docops.ca.com/display/DTS95/logging.properties>) file in the **`LISA_HOME`** directory lets you configure the logging behavior. This file is your main source for manipulating what is being logged across the multiple components of DevTest.

To get more logging information from DevTest Workstation, you can change the logging level in **`log4j.rootCategory`**:

```
log4j.rootCategory=INFO,A1
```

The **logging.properties** file contains a set of loggers for third-party components that are included in DevTest. The default log levels for these loggers are intended to prevent the third-party components from flooding the log files with too many messages. You typically do not need to change the log levels.

```
log4j.logger.com.teamdev=WARN
log4j.logger.EventLogger=WARN
...
```

The default appender is **com.itko.util.log4j.TimedRollingFileAppender**. Log statements for a component are appended to a file that is backed up when it reaches a certain size. The default maximum file size is 10 MB. The default number of backup files is 5.

```
log4j.appender.A1=com.itko.util.log4j.TimedRollingFileAppender
log4j.appender.A1.File=${lisa.tmpdir}/${LISA_LOG}
log4j.appender.A1.MaxFileSize=10MB
log4j.appender.A1.MaxBackupIndex=5
log4j.appender.A1.layout=org.apache.log4j.EnhancedPatternLayout
log4j.appender.A1.layout.ConversionPattern=%d{ISO8601}{UTC}Z (%d{HH:mm}) [%t] %-5p %
-30c - %m%n
```

To perform log rotation by date or time, use the **DailyRollingFileAppender**. For more information, see the [log4j documentation \(https://logging.apache.org/log4j/1.2/apidocs/org/apache/log4j/DailyRollingFileAppender.html\)](https://logging.apache.org/log4j/1.2/apidocs/org/apache/log4j/DailyRollingFileAppender.html).

```
log4j.appender.A1=
org.apache.log4j.DailyRollingFileAppender
log4j.

appender.A1.File

=${lisa.tmpdir}/${LISA_LOG}

log4j.appender.A1.Append=true

log4j.appender.A1.DatePattern='
'
yyyy-MM-dd
```

The backup files are placed in the same directory as the log file. For example, if the log file for the registry has been backed up three times, the directory contains the following files:

- registry.log
- registry.log.1
- registry.log.2
- registry.log.3

Layouts control the format of log statements. The default layout is **org.apache.log4j.EnhancedPatternLayout**. The default conversion pattern is **%d{ISO8601}{UTC}Z (%d{HH:mm}) [%t] %-5p %-30c - %m%n**. This conversion pattern specifies that a log statement includes the date, thread, priority, category, and message. For example:

```
2014-11-20 14:09:08,152Z (07:09) [main] INFO com.itko.lisa.net.
ActiveMQFactory - Starting amq broker
```

The date uses Coordinated Universal Time (UTC). This convention makes it easier to follow log events when the registry is running in a different time zone than DevTest Workstation.

For information about the thread dump properties, see [Automatic Thread Dumps \(see page 115\)](#).

Status Messages for Server Components

The following server components write status messages to their log files at a specified interval:

- Registry
- Coordinator
- Simulator
- VSE

The following example was written to a log file by the registry.

```
2015-01-15 17:40:12,228Z (09:40) [Event Sink Thread Pool Thread 5] INFO com.itko.
lisa.coordinator.TestRegistryImpl -
Coordinator Servers: 0 Simulator Servers: 0 VSEs: 0 Running vusers: 0 Labs: 1
Memory used 356mb, allocated 538mb, max 569mb (62%) labSims: 0 labVSEs: 0 labCoords: 0
Our cpu usage 0%, system cpu used 6% GC time 0% db ping no connection
```

The default interval of the status messages is 30 seconds. You can change the interval by editing the following properties in the **lisa.properties** file:

- lisa.defaultRegistry.pulseInterval
- lisa.coordinator.pulseInterval
- lisa.simulator.pulseInterval
- lisa.vse.pulseInterval

Automatic Thread Dumps

You can use the **logging.properties** file to enable automatic thread dumps, which are helpful in debugging performance issues.

Locate the following property and change **WARN** to **INFO**.

```
log4j.logger.threadDumpLogger=WARN, THREAD_DUMPS
```

To disable the thread dumps, change **INFO** back to **WARN**.

The default interval of the thread dumps is 30 seconds. You can change the interval by editing the **lisa.threadDump.interval** property in the **lisa.properties** file.



You can change the setting from WARN to 1 for more granular logging without generating a dump

Test Step Logger

As described in [Elements of a Test Step \(https://docops.ca.com/display/DTS95/Elements+of+a+Test+Step\)](https://docops.ca.com/display/DTS95/Elements+of+a+Test+Step), each test step includes a log message element.

To send the log message to a specific file at runtime, add the following properties to the **logging.properties** file in the **LISA_HOME** directory.

```
log4j.logger.com.itko.lisa.test.StepLogger=DEBUG, A2
log4j.additivity.com.itko.lisa.test.StepLogger=false
log4j.appender.A2=org.apache.log4j.RollingFileAppender
log4j.appender.A2.File=${lisa.tmpdir}/log.log
log4j.appender.A2.MaxFileSize=10MB
log4j.appender.A2.MaxBackupIndex=5
log4j.appender.A2.layout=org.apache.log4j.PatternLayout
log4j.appender.A2.layout.ConversionPattern=%d [%t] %-5p %-30c - %m%n
```

The values of the **File**, **MaxFileSize**, and **MaxBackupIndex** properties can be different from the values that are shown in the preceding example.



Note: After you add the properties, restart DevTest Workstation (if currently running).

The resulting message in the log file is similar to the following example:

```
2012-07-11 17:32:59,390 [basic-test/basic-test [QuickStageRun]/0] DEBUG com.itko.lisa.
test.StepLogger -
LOG basic-test,basic-test [QuickStageRun],local,0,3,my log message
```

The portion of the message after **LOG** consists of six components:

- The name of the test case.
- The name of the staging document.
- The name of the simulator.
- The instance/vuser number. In a ten vuser test, the number varies from 1 to 10.
- The cycle number. This number applies to situations in which the staging document is configured to run the test over and over until some condition occurs. The value is the number of times this particular vuser executed the test.
- The log message that is set in the test step.

VSE Logging

VSE_Matches File

DevTest generates a log file with the services matches into a file named VSE_matches. You can use this log file to debug errors at the request-response level. It is located in the **lisatmp** directory. It contains the matching characteristics for the request being processed.

VSE.log File

The VSE.log file contains the errors found when matching requests and responses.

Monitoring

You can monitor DevTest Solutions by using the DevTest Portal, the Service Manager command-line utility, the Registry Monitor, or the Enterprise Dashboard.

This section contains the following pages :

- [Use DevTest Portal \(see page 118\)](#)
- [Use the ServiceManager \(see page 121\)](#)
- [Use the Registry Monitor \(see page 124\)](#)
- [Use the Enterprise Dashboard \(see page 125\)](#)

Use DevTest Portal

You can use DevTest Portal to monitor DevTest Solutions servers.

View the Server Health Monitor

The **Server Health Monitor** in **DevTest Portal** provides an indicator of the overall health of running components.

To access the Server Health Monitor, in DevTest **Portal**, click **Monitor, Server Health**. The Server Health Monitor displays.

If the user does not have permission to view the health of a component, the component displays with the message "No permission to view status of any components of this type." To remove those components from the display, clear the **No permission** check box. A DevTest administrator specifies permissions for types of components in ACL using [roles \(see page 86\)](#).

If the user does not have permission to view the health of a specific component; for example, a specific VSE, the component displays with the message, "Not authorized to monitor status of this resource." To remove those components from the display, clear the **Not authorized** check box. A DevTest administrator specifies permissions for specific components in ACL using [resources \(see page 110\)](#) and [resource groups \(see page 108\)](#).

Home x Server Health x

Summary of Server Status

Last Refreshed: 05/05/2016 7:34:17 AM ☐ Auto Refresh 60 Sec

1

1

1

VSEs Coordinators Simulators

☒ No permission ☒ Not authorized

Type	Name	Version	Host	Port	OS	Total Up Time	Current CPU Usage (%)			JVM GC	JVM Heap Memory Consumption			
							JVM	System	% of CPU Time	Max (MB)	Allocated (MB)	Current (MB)	Current (%)	
Registry	Registry	9.5.0 (9.5...	ad-buil...	2010	Windo...	7:27:04	12	35	0	892	420	380	42	Cur
Coordi...	Coordi...	9.5.0 (9.5...	ad-buil...	2011	Windo...	7:27:03	0	49	0	683	78	48	7	Use Ava Ma
Simula...	Simula...	9.5.0 (9.5...	ad-buil...	2014	Windo...	7:27:03	0	5	0	569	57	47	8	Use Ava Ma
VSE	VSE	9.5.0 (9.5...	ad-buil...	2013	Windo...	7:27:04	0	48	0	910	509	108	11	Per

1 10 Items per page

1 - 4 of 4 items

Server Health Monitor window

For each server, the following fields are displayed.

- **Type**
The server type that is displayed is either Registry, Simulator, Coordinator, or VSE.
- **Name**
Displays the name of the server.
- **Version**
Displays the version of DevTest Server
- **Host**
Displays the hostname for the server.
- **Port**
Displays the port used by the server.
- **OS**
Displays the operating system of the server.
- **Total Up Time**
Displays the time, in an hours:minutes:seconds format, since the server was started.
- **Current CPU Usage (%) - JVM**
Displays the percentage of total CPU usage for Java virtual machine (DevTest Server).
- **Current CPU Usage (%) - System**
Displays the percentage of total CPU usage for the system.
- **% of CPU Time - JVM GC**
Displays the percentage of CPU that the JVM (DevTest Server) garbage collection is using.
- **JVM Heap Memory Consumption - Max (MB)**
Displays the maximum consumption of heap memory that the JVM (DevTest Server) attempts to use, in megabytes.

- **JVM Heap Memory Consumption - Allocated (MB)**

Displays the total amount of heap memory in the JVM (DevTest Server), in megabytes.

- **JVM Heap Memory Consumption - Current (MB)**

Displays the current heap memory consumption in the JVM (DevTest Server), in megabytes.

- **JVM Heap Memory Consumption - Current (%)**

Displays the current heap memory consumption in the JVM (DevTest Server), as a percentage of the total.

- **Server Info**

Displays information about servers, depending on the server type. Information that can be displayed includes:

- **Current User Count**

(for registries) A snapshot of how many users are logged in to the registry at this moment.

- **Used Capacity**

(for coordinators and simulators) Displays roughly the number of running test cases and suites.

- **Available Capacity**

(for coordinators and simulators) Displays roughly the number of how many more test cases and suites can be run on that component.


- **Max Capacity**

(for coordinators and simulators) Displays the maximum number of test cases and suites that can be run on that component.

Default: 128 for coordinators and 256 for simulators. This can be adjusted with [properties \(https://docops.ca.com/display/DTS95/Local+Properties+File\)](https://docops.ca.com/display/DTS95/Local+Properties+File).

- **Mobile Assets**

(for simulators) To display a table of mobile assets that are associated with the server, click the blue arrow to the right of the **Mobile Assets** label. For each asset, the name, version, and type are shown.

0	569	196	161	28	Used Capacity : 0 Available Capacity : 256 Max Capacity : 256 Mobile Assets 
Name	Version	Type			
2_7_QVGA_API_23	6.0	Simulator			
Nexus_7_2012_API_23	6.0	Simulator			
Nexus_5_API_23_x86	6.0	Simulator			
Galaxy_Nexus_API_23	6.0	Simulator			

1 - 5 of 6 items

1 - 4 of 4 items

- **Performance Mode**

(for VSEs) Displays either **Enabled** or **Disabled**, to indicate whether the VSE is in [performance mode](https://docops.ca.com/display/DTS95/SV+Installation+and+Configuration) (<https://docops.ca.com/display/DTS95/SV+Installation+and+Configuration>).

Use the ServiceManager

The ServiceManager command-line utility lets you perform various actions on a registry, coordinator, simulator, or VSE server.

- [Service Manager Options \(see page 121\)](#)
- [Service Manager Examples \(see page 123\)](#)

The ServiceManager command-line utility has the following format:

```
ServiceManager [--command]=service-name
```

The **service-name** is the name of the service to affect.

The command/name pair can be repeated.

To look up the name, wrap a lisa.properties key with double braces. For example: **{{lisa.registryName}}**

Example service names:

- tcp://localhost:2010/Registry
- Simulator (resolves to tcp://localhost:2014/Simulator)

Service Manager Options

- **-h, --help**
Displays help text.
- **-s service-name, --status=service-name**
Displays a status message about the service. Entering *all* for the service-name returns status messages about all registered services.
- **-r service-name, --reset=service-name**
Keeps the service in memory but refreshes its state.
- **-o service-name, --stop=service-name**
Instructs the service to end.
- **-i valid-remote-init-service-name, --initialize=valid-remote-init-service-name**
Remotely initializes a service.
- **-t service-name, --threaddump=service-name**
Instructs the service to generate a thread dump (stack trace) for diagnostics.

- **-b simulator-name, --attached=simulator-name**
Instructs simulator-name to return a list of attached mobile devices.
- **-e service-name, --heapdump=service-name**
Instructs the service to create an .hprof file for memory diagnostics.
- **-g service-name, --gc=service-name**
Instructs the service to force a Java garbage collection.
- **-d service-name, --diagnostic=service-name**
Creates a zip file that contains diagnostic files for the service. If the service is a registry, then the zip file also contains diagnostic files for all connected coordinators, simulators, and VSE servers. Typically, you use this option when requested to do so by Support.
loglevel
This option also includes an optional, secondary parameter that is named **loglevel**, followed by one of the following loglevel keywords: error, warn, info, debug, trace.
For example, **ServiceManager -d tcp://10.1.1.23:2010/Registry loglevel debug** sets the log level of all components, including agents, to the debug level. Service Manager then writes the following message:
Log levels set to debug. Run your repro steps, then press return to restore log levels and capture diagnostic.
The generated zip file contains debug log level logs for all connected components and thread dumps and license information and properties. This zip file also contains the logs for any agents that are connected to the registry broker.



Note: If the -d command is sent to a VSE, coordinator, or simulator server, only the logs and diagnostics for that component are included in the zip. Agents do not have a trace log level, but the dev log level is similar and treated as such.

- **-u username, --username=username**
Use this command to specify your user name.
- **-p password, --password=password**
Use this command to specify your password.
- **--version**
Print the version number.
- **-m registry-name, --registry-name=registry-name**
Used with *initialize* to specify a registry to which to attach.
- **-n component-name, --component-name=registry-name**
Used with *initialize* to specify a component name.
- **-l lab-name, --lab-name=lab-name**
Used with *initialize* to specify a lab name to create.
- **-a app-name, --app=app-name**
Used with *initialize* to specify a server appID.

Example:

If you have a simulator that you want to name *MySim* and you want to attach it to *MyRegistry* and start a new lab that is named *MyDevLab*, enter:

```
./SimulatorService -n MySim -m tcp://1.2.3.4:2010/MyRegistry -l MyDevLab
```

To add a second simulator to that lab:

```
./SimulatorService -- component-name=MySecondSim -- registry-name=tcp://1.2.3.4:2010/MyRegistry -- lab-name=MyDevLab
```

To add a VSE to that registry but in a different lab:

```
./VirtualServiceEnvironment -n CoreServices -m tcp://1.2.3.4:2010/MyRegistry -l QA
```

Service Manager Examples

The following example checks the status of the registry.

```
ServiceManager -s Registry
Coordinator Servers: 1 Simulator Servers: 2 VSEs: 1 Running vusers: 0
Labs: 1 Memory used 76mb, allocated 155mb, max 253mb (30%)
labSims: 2 labVSEs: 1 labCoords: 1
```

The following example checks the status of all registered services.

```
Coordinator Server: tcp://bdert-mbp.local:2011/Coordinator
OK: 1 Coordinators running. Memory used 223mb, allocated 461mb, max 910mb (24%) Our cpu usage 0%, system cpu used 8%
Simulator Server: tcp://bdert-mbp.local:2014/Simulator
OK: 1 Simulators running. Memory used 301mb, allocated 437mb, max 910mb (33%) Our cpu usage 0%, system cpu used 8%
```

The following example stops the registry.

```
ServiceManager -o Registry
Sending stop request to Registry.
```

The following example generates a thread dump for the VSE server.

```
ServiceManager --threaddump=tcp://remote.host.com:2013/VSE
< a bunch of stack traces >
```

The following example forces a Java garbage collection for the VSE server.

```
ServiceManager --gc=VSE
After GC: Memory used 55mb, allocated 225mb, max 246mb (22%)
```

The following example generates a zip file that contains trace level logs for all components that are connected to the registry.

```
ServiceManager --diagnostic=Registry loglevel TRACE
```

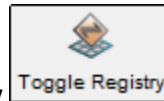
The following example generates a zip file that contains debug level logs for the simulator.

```
ServiceManager -d Simulator loglevel debug
```

Use the Registry Monitor

The Registry Monitor lets you monitor the test cases, simulators, coordinators, and virtual environments for a test suite.

- [Registry Monitor - Test Tab \(see page 124\)](#)
- [Registry Monitor - Simulators Tab \(see page 125\)](#)
- [Registry Monitor - Coordinator Servers Tab \(see page 125\)](#)
- [Registry Monitor - Virtual Environments Tab \(see page 125\)](#)



To open the **Registry Monitor**, click **Toggle Registry** on the main toolbar of DevTest Workstation.

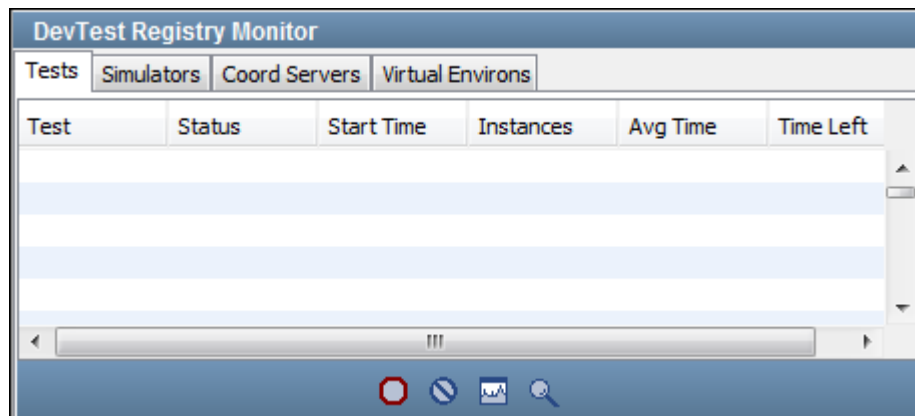


Image of the Registry Monitor

The **Registry Monitor** has the following tabs:





- [Tests Tab \(see page 124\)](#)
- [Simulators Tab \(see page 125\)](#)
- [Coordinator Servers Tab \(see page 125\)](#)
- [Virtual Environments Tab \(see page 125\)](#)

Registry Monitor - Test Tab

The **Tests** tab lists information about all test cases that are currently running in this test suite.

For each test case, you can view the test name, status, start time, instances, average time, and time left.

You can perform the following actions on a selected test:

- To stop a test, click **Stop** .
- To kill a test (stop immediately), click **Kill** .
- To optimize a test, click **Optimize Test** .
For more information, see [Using the Load Test Optimizer \(https://docops.ca.com/display/DTS95/Use+the+Load+Test+Optimizer\)](https://docops.ca.com/display/DTS95/Use+the+Load+Test+Optimizer).
- To view a test, click **View Test** .
The test information is displayed in the test monitor.

Registry Monitor - Simulators Tab




The **Simulators** tab lets you add virtual users in real time to the selected simulator server.

This tab lists the simulator server and its available instances.

Registry Monitor - Coordinator Servers Tab

The **Coordinator Servers** tab lists information about coordinator servers that are running.

You can perform the following actions on a selected coordinator server:

- To shut down this service, click **Stop** .
- To reset this service (stop and clear current activities), click **Reset** .
- To view a status message, click **View Status Message** .

Registry Monitor - Virtual Environments Tab

The **Virtual Environments** tab lists information about the virtual environments (if any) that are running.

Use the Enterprise Dashboard

This section contains the following pages describing the functionality of the Enterprise Dashboard:

- [Monitor Usage and VSEs with Performance Mode \(see page 129\)](#)
- [Reactivate a Registry or Enterprise Dashboard \(see page 132\)](#)
- [Maintain Registries \(see page 133\)](#)
- [Export Dashboard Data \(see page 134\)](#)
- [Purge Dashboard Data \(see page 135\)](#)
- [Change Dashboard Language \(see page 136\)](#)

Open the Enterprise Dashboard

To open the Enterprise Dashboard from a web browser:

1. Navigate to the directory where you installed the Enterprise Dashboard.
2. The Enterprise Dashboard server must be started if it isn't already running. Navigate to the **bin** directory, and run **EnterpriseDashboard.exe**, or select **Enterprise Dashboard Server** from the **Start, Programs** menu.
3. Enter **http://localhost:1506/** in a web browser or select **DevTest Enterprise Dashboard UI** from the **Start, Programs** menu.
If the **Enterprise Dashboard** is running on a remote computer, replace **localhost** with the name or IP address of that computer.
The **Enterprise Dashboard** opens.

Enterprise Dashboard Main Window

The main window in the **Enterprise Dashboard** lets you view performance statistics about peak user counts and running registries.

Peak User Counts

The top of the main window displays peak usage counts for four [user types \(see page 62\)](#) (Continuous Application Insight Power User, Service Virtualization Power User, Application Test Power User, and DevTest Runtime User). This count identifies the largest number of simultaneous users for each user type during the specified time period. For more information, click **More User Info**.

VSE(s) in Performance Mode

The top of the main window displays the maximum number of VSE or VSEs being with [performance mode \(https://docops.ca.com/display/DTS95/SV+Installation+and+Configuration\)](#) activated. For more information, click **More VSE Info**.

Running Registries

- **Display Name**
The Display Name is a name that you assign when you configure the registry.

- **Name**
The URL of the registry.
- **Last Update**
The last date and time that Enterprise Dashboard received data from this registry.
- **Version**
The DevTest version for that registry.
- **Coordinators, Simulators, VSEs, Workstations, Agents, and Labs**
The number of each resource that is associated with the registry.



Note: To view agents, you must [start broker.exe \(https://docops.ca.com/display/DTS95/Start+the+Broker\)](https://docops.ca.com/display/DTS95/Start+the+Broker).

- **DevTest Console URL**
The address of the landing page for DevTest Console, which provides links to DevTest Portal.

To display information about the database that the **Enterprise Dashboard** uses, hover over the database icon



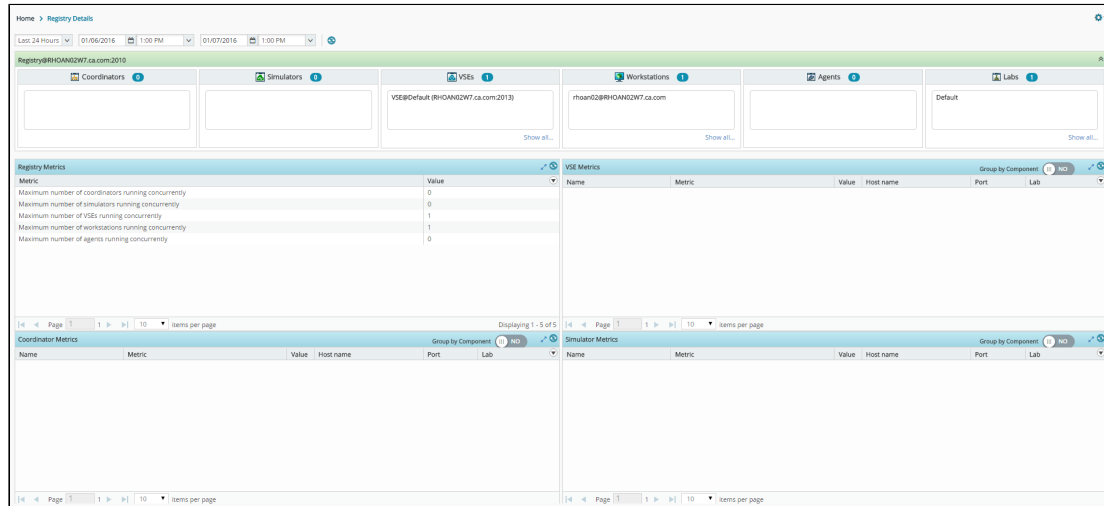
in the upper right corner of the panel. The popup window displays the database URL, database type, database version, driver name, driver version, and user.

Enterprise Dashboard Registry Details Window

The Registry Details window in the Enterprise Dashboard lets you view details about registries. Click on a registry display name on the main page to bring up the registry details window




Note: This window only displays data for registries running version 7.5 or later.



The top part of the window lists all of this registry's coordinators, simulators, VSE servers, labs, Workstations, and agents that were running during the timeframe of the supplied filter.

To designate the timeframe to use in displaying metrics, use the drop-down fields at the upper left of the panel. The first drop-down field provides a list of standard timeframes. To be more specific, specify a start date and start time and end date and end time in the other drop-down fields. Click **Refresh** to refresh the display.

If the **Enterprise Dashboard** has been idle for some time, click **Refresh**  to refresh the data for each panel.

The bottom part of the window shows metrics for registries, VSE servers, coordinators, and simulators.

The **Registry Metrics** area of the panel shows the maximum number of workstations, simulators, coordinators, agents, and VSEs concurrently running in the designated time period.

The **VSE Metrics** area of the panel shows the number of transactions and the number of labs that were deployed for each VSE active during the designated time period.

The **Coordinator Metrics** area of the panel shows the number of tests that were started for each coordinator during the designated time period.

If you run a test suite and the **Enterprise Dashboard** "tests started" count is not what you expect, see the suite results section of DevTest Workstation. Check the results for each failed test. If any tests failed to stage, the tests started count does not include them.

The **Simulator Metrics** area of the panel shows the maximum number of virtual users active during the designated time period.

To return to the main window, click **Home** in the breadcrumbs in the top left corner of the page.



Enterprise Dashboard may sporadically display inadvertent stop and restart of components as network connectivity is lost between the registry and its underlying components. When a registry loses connectivity to an underlying component for a period of time, it is assumed that the component has shut down, so an “End Event” is generated and submitted to Enterprise Dashboard for that component.

Monitor Usage and VSEs with Performance Mode

From the main window of the Enterprise Dashboard, you can drill down into more detail on [peak user counts](#) (see page 61) by [user type](#) (see page 62) and peak counts of VSEs being run in [performance mode](#) (<https://docops.ca.com/display/DTS95/SV+Installation+and+Configuration>).

You can use this information to have a real-time picture of your peak concurrent user and VSE count for a designated time period, to help you monitor your compliance with the terms of your [license agreement](#) (see page 52).

Use the filter at the upper right corner of the **Peak Counts** pane to designate the time period to display the peak user and VSE counts.

The **Registry** pane displays all registries that this Enterprise Dashboard controls, with information about each. Use the filter drop-down at the upper right of the **Registry** pane to control the time frame to display related registries.

Peak User Counts

Instances of peak user counts are shown for each of four user types: Continuous Application Insight Power User, Service Virtualization Power User, Application Test Power User, and DevTest Runtime User.

Use the date and time range fields at the top of the page to specify a time period for the report.

For each user type, the Start and Stop dates and times are shown, with a peak user count. Click the number of the user count for more details of all users who are included during the time period of that peak count occurrence.

Example

The following graphic shows the CAI Power User peak user counts from the Top Instances window of the Enterprise Dashboard.

Home > Top Instances of User Counts

Select a time 04/25/2016 7:00 AM 05/09/2016 7:00 AM

Continuous Application Insight Power User - Peak User Count

Start Time	End Time	# of Users
4/25/2016 3:58:24 PM	4/25/2016 4:22:45 PM	2
4/26/2016 7:06:27 AM	4/26/2016 7:57:23 AM	2
4/26/2016 8:04:23 AM	4/26/2016 9:21:23 AM	2
4/26/2016 9:22:59 AM	4/26/2016 10:38:23 AM	2
4/26/2016 10:57:09 AM	4/26/2016 11:17:16 AM	2
4/26/2016 2:06:35 PM	4/26/2016 2:38:26 PM	2
4/27/2016 10:06:23 AM	4/27/2016 10:06:51 AM	2
4/28/2016 8:56:04 AM	4/28/2016 9:36:33 AM	2
4/28/2016 10:14:22 AM	4/28/2016 10:40:36 AM	2

Page 1 of 2 10 items per page Displaying 1 - 10 of 11

CAI Power User Top Instances

During the time period that is specified, from 7:00 AM on April 25 to 7:00 AM on May 9, there was a maximum of two concurrent users. There are 11 rows, indicating that there were 11 instances where two CAI users were being consumed. If you click on the number 2, you see the User Count Usage Details window. This window shows information about each of the two user sessions: the user, the hostname of the client application, their registry, login and logout times, and the application they were logged in to.

The timeframe that is shown on this window is the timeframe that reflects the concurrency; that is, in this example, when two users were logged in simultaneously. Essentially, this timeframe is the time period from the time that the concurrency started to when the concurrency ended. In the User Count Usage Details pane that is shown below, the concurrency happened on April 26. The first moment that two users were both logged in was 7:06:27, when the second user logged in. Both users were logged in until the first user logged out at 7:57:23.

The User Count Usage Details window displays all the usage activity that makes up the concurrent count from the Top Instances of User Counts from the previous page. The only entries that display are those that comprise the peak user count. Other logins and user activity during this period are not detailed on this page.

User Count Usage Details

Continuous Application Insight Power User [4/26/2016 7:06:27 AM - 4/26/2016 7:57:23 AM]

User ID	Hostname	Registry	Login Time	Logout Time	DevTest Portal	test-invoke
admin	xiopi01e.ca.com	Registry@ISHED01.c...	4/26/2016 6:58:29 AM	4/26/2016 7:57:23 AM	⊙	⊙
admin	ISHED01.ca.com	Registry@ISHED01.c...	4/26/2016 7:06:27 AM	4/26/2016 8:11:23 AM	⊙	

User Count Usage Details screenshot

VSE(s) in Performance Mode - Peak Count

Peak VSE(s) in Performance Mode show the number of VSEs running in [Performance Mode](https://docops.ca.com/display/DTS95/SV+Installation+and+Configuration) (<https://docops.ca.com/display/DTS95/SV+Installation+and+Configuration>) that were running simultaneously during the specified time period. Information that is shown includes start time, stop time, and the peak number of VSEs.

When you click on the number of VSEs, you see details for each VSE that was reported on the Peak Count window.

Similar to the User Count Usage Details window, the timeframe that is shown on this window is the timeframe that reflects the concurrency. This timeframe is the time period from the time that the concurrency started to when the concurrency ended.

Peak Usage API

You can use the Peak Usage API to get the same information about user and VSE usage that is available through the Enterprise Dashboard UI.

Follow these steps:

1. From your browser, navigate to **http://localhost:1506/api**.
2. Click **List Operations**.
3. Click **Get concurrent usage information**.
The Registry Usage information appears.

registry: Registry Controller Show/Hide List Operations Expand Operations Raw

GET `/registry/index` [Get a list of registries](#)

GET `/registry/usage` [Get concurrent usage information](#)

Implementation Notes
Times are in ISO 8601 format. Example: 2016-03-23T20:56:21.111Z or 2016-03-23T15:56:21.111-05:00

Parameters

Parameter	Value	Description	Parameter Type	Data Type
beginTime	<input type="text"/>	Starting time range.	query	string
endTime	<input type="text"/>	Ending time range	query	string
maxCount	true (default) ▼	Only show entries where maximum concurrent count occurred	query	string
include	<input type="text"/>	Includes the session information for each concurrent group	query	string
q	<div> <div>usage_type=SV_POWER_USER</div> <div>usage_type=CAT_POWER_USER</div> <div>usage_type=SVT_RUNTIME_USER</div> <div>usage_type=TEST_POWER_USER</div> </div>	Filter on usage type	query	string

[Try it out!](#)

Peak Usage API screenshot

4. Click **Try it out!**
Results appear in the Response Body, Response Code, and Response Headers areas.

Reactivate a Registry or Enterprise Dashboard

The DevTest Installation Setup Wizard configures new registries. When you start the Enterprise Dashboard and the registries, the registries are automatically activated. However, if there are subsequent changes to the Enterprise Dashboard URL, reactivation is needed. This procedure is valid for DevTest 9.1 and later. For reactivating registries for earlier versions, see [Maintain Registries \(see page 133\)](#).

Follow these steps:

1. If the host name or port of the Enterprise Dashboard changes:
 - a. Log on to the computer with the registry.
 - b. Navigate to LISA_HOME and open **site.properties**.
 - c. Update the following lines if the host name or port of the Enterprise Dashboard changes.


```
devtest.enterprisedashboard.host=localhost
devtest.enterprisedashboard.port=1506
```
 - d. Update the following line if the URL changed from HTTP to HTTPS or from HTTPS to HTTP.


```
devtest.enterprisedashboard.https.enabled=false|true
```
 - e. Save the file.
 - f. Restart the registry.
2. Verify that the Enterprise Dashboard is running.
3. If the host name of the server where the registry is installed changes, restart the registry to reactivate it.
4. If the registry name, or registry port of any registry changes:
 - a. Log on to the computer where there has been a change to the registry.
 - b. Open a command prompt or terminal window and navigate to LISA_HOME.
 - c. Start the registry with a new name or port. For example:


```
./bin/Registry.exe -n "tcp://localhost:2093/MyRegistry"
```
5. [Verify registry reconfiguration \(https://docops.ca.com/display/DTS95/Verify+Registry+Activation\)](https://docops.ca.com/display/DTS95/Verify+Registry+Activation).

Maintain Registries

The registry gets a license from the Enterprise Dashboard and stores and forwards its use counts to the dashboard.

Maintaining registries involves the following procedure:

- Add the registry of a DevTest Server and then validate the registry.

To add a registry (DevTest 7.5.x through 9.0.x):

1. Navigate to the LISA_HOME directory of a newly installed DevTest Server.
2. Copy **_site.properties** and rename the copy to **site.properties**.
3. Open **site.properties** for edit. Locate Section 1 - Enterprise Dashboard.
4. Uncomment the following line and substitute *localhost* or the valid hostname for *somehost*. Use the hostname for the CIC Enterprise Dashboard bridge.

For 8.x and later:

```
lisa.enterprisedashboard.service.url=tcp://somehost:2003/EnterpriseDashboard
```

For 7.5.x:

```
lisa.enterprisedashboard.url=tcp://somehost:2003/EnterpriseDashboard
```

5. Save **site.properties** and exit.
6. Restart the registry.

To activate CIC for registries older than Release 9.1:

DevTest uses CIC to support pre-9.1 registries with Enterprise Dashboard 9.1 or later. If your registries are Release 9.1 or later, you do not need to activate CIC.

1. Start Enterprise Dashboard
2. Start CIC by executing **LISA_HOME/bin/EnterpriseDashboardCIC.exe**.
3. Modify the **lisa.enterprisedashboard.service.url** property for every legacy registry to point to the host where CIC is running: `lisa.enterprisedashboard.service.url=tcp://<host>:2003/EnterpriseDashboard`
4. Restart the legacy registry.
5. Start the DevTest Enterprise Dashboard UI if it is not running.
6. Confirm that the legacy registry appears on the home page display.

Export Dashboard Data

The Enterprise Dashboard lets you export current and historical dashboard data in an Excel format. You can export data from either the main window or the Registry Details window.

The following metrics are exported:

- **Registry Metrics**

- Number of Workstations Running (Minimum/Maximum)
- Number of Simulators Running (Minimum/Maximum)
- Number of Agents Running (Minimum/Maximum)
- Number of Coordinators Running (Minimum/Maximum)
- Number of Virtual Service Environments (VSEs) Running (Minimum/Maximum)

- **VSE Metrics**

- Maximum Number of Models Deployed

- **Coordinator Metrics**

- Number of Tests Started

- **Simulator Metrics**

- Maximum Number of VUs Allocated

Follow these steps:

1. Click **Options** in the upper right corner of the window and select **Export To Excel**.
If you are exporting from the main window, the Export Registry Data window opens.
2. Complete the following fields:
 - **Export live data only**
Selecting this check box limits your export to live data.
Clearing this check box exports historical data.
 - **Ping registries**
Selecting this check box pings each registry to determine the status of the registry to display in the exported file.
Clearing this check box results in a status of Unknown for each registry in the exported file.



Note: This option only applies to historical data. For a live data export, the status of each registry is automatically determined.

3. You are prompted to either save or open the exported Excel file.
4. Click one of the following:
 - Click **Save** to save your exported file to a specified directory.
 - Click **Open** to view the exported file.
5. Click **OK**.

Export Usage Audit Data

The DevTest Solutions Usage Audit Report provides details on compliance with your license agreement, which is based on maximum concurrent usage by the following user types:

- Continuous Application Insight Power User
- Service Virtualization Power User
- Application Test Power User
- DevTest Runtime User

You can generate the Usage Audit Report on demand.

Follow these steps:

1. Browse to the Enterprise Dashboard.
`http://hostname:1506`
2. Click the gear icon and select **Export Usage Audit Data**.
The Export Usage Audit Data dialog opens.
3. To select a start date and end date for the date range on which to report, click the calendar icons, then click **OK**.
The generated report from the specified date range is downloaded to the bottom left of the window.
4. Click the DevTestSolutionsUsageAuditReport.xlsx to open the Excel book containing your report.
See [DevTest Solutions Usage Audit Report \(see page 56\)](#) for details on each tab.

Purge Dashboard Data

The Enterprise Dashboard lets you purge historical event logs and metrics that are older than a specified date and time.

Follow these steps:

1. Click **Options** in the upper right corner of the window and select **Purge Data**.
The Purge Data dialog opens.

2. Select a date and time for the data that you want to purge.
The purge feature permanently deletes all event logs and metrics from the database that are older than the specified date and time.
The default values delete event logs and metrics that are 100 days older than the current date and time.
3. Click **OK**.
A confirmation dialog opens.
4. Click **Yes**.
The selected data is purged from the Enterprise Dashboard.

Change Dashboard Language

The Enterprise Dashboard lets you change the language that is used on the Dashboard user interface.

Follow these steps:

1. Click **Options** in the upper right corner of the window and select **Language**.
The available languages are listed.
2. Select a language.