DevTest Solutions

Administering

Version 8.0

Contact CA Technologies

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information that you need for your Home Office, Small Business, and Enterprise CA Technologies products. At http://ca.com/support, you can access the following resources:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Providing Feedback About Product Documentation

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at http://ca.com/docs.

Contents

Chapter 1: General Administration

This section contains the following topics:

Default Port Numbers

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

- DevTest Server (see page 9)
- DevTest Workstation (see page 11)
- Demo Server (see page 11)

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
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
8443	Embedded web server (if SSL is enabled)	lisa.webserver.ssl.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 **site.properties** file and set the new value. Do not update the **lisa.properties** file.

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	lisa.webserver.port
2004	VSE Manager	lisa.net.9.port
2005	Test Runner	lisa.net.5.port
2006	ServiceManager	lisa.net.11.port
2009	Broker	lisa.pathfinder.broker.port
2999	JDBC simulation driver	Not applicable
3128	HTTP proxy	laf.httpproxy.port
8443	Embedded web server (if SSL is enabled)	lisa.webserver.ssl.port

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.

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.

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, CA does 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 topic describes the directory structure of DevTest Workstation (see page 14) and DevTest Server (see page 16).

This topic assumes that the local installation type was selected during the installation of DevTest Solutions. If the shared installation type (see page 12) 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 Agent.

bin

Contains the executable files for DevTest Workstation and other components.

database

Contains the DDL files for various databases.

defaults

Contains the audit document and staging documents common across all projects. Project-specific staging documents are located in the Projects directory.

doc

Contains the license agreement.

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.

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.

When you create a project within DevTest Workstation, the **Projects** directory is added.

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 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 for various databases.

defaults

Contains the audit document and staging documents common across all projects. Project-specific staging documents are located in the Projects directory.

doc

Contains the license agreement.

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.

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.

When you create a project within DevTest Workstation, the **Projects** directory is added.

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/.

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.

- The Execute JUnit Test Case/Suite step can execute a JUnit test. DevTest Ant Tasks (see page 20) 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

The following Ant tasks are included with DevTest Solutions:

- junitlisa (see page 21)
- junitlisareport (see page 24)

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

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

JUnit3 Test Cases (see page 25)

JUnit3 Test Suites (see page 26)

JUnit4 Test Cases (see page 26)

JUnit4 Test Suites (see page 27)

JUnit3 Test Cases

For JUnit3 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);
   }

}
```

JUnit3 Test Suites

For JUnit3 test suites, your suite is not required to extend the junit.framework.TestSuite. However, it must implement the suite() method, with test cases wrapped by the 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 JUnit4 test cases, the test methods must have the "@org.junit.Test" annotation on the methods, as required by JUnit4.

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); }

}
```

JUnit4 Test Suites

To implement a JUnit4 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 are used to customize the memory allocation settings for each of the DevTest processes used in the server. When you install the DevTest Server, the LISA_HOME\bin folder is populated with a .vmoption file with the same name as each executable file. A full list of JVM parameters are detailed in the Oracle website under Configuring the Default JVM and Java Arguments.

To 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.

To change the Java heap size on Mac 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.

To adjust the memory for individual processes:

You can adjust memory for the following programs:

- Broker

- BrokerService
- CoordinatorServer
- CoordinatorService
- Registry
- RegistryService
- Simulator
- SimulatorService
- VirtualServiceEnvironment
- VirtualServiceEnvironmentService
- Workstation

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

If you reinstall DevTest Solutions, these edits are overwritten by the installer.

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 topic.

Note: This topic assumes that the local installation type was selected during the installation of DevTest Solutions. If the shared installation type (see page 12) was selected, the **hotDeploy** directory can be in a location other than **LISA_HOME**.

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.

SAP File Requirements

The following JAR files are required:

- sapjidoc3.0.8/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 TIBCO 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:

- 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 WebSphere MQ 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
- dhbcore.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.

Chapter 2: Database Administration

This section contains the following topics:

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

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

Note: For information about database system requirements, see *Installing.*

External Database Configuration

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

Note:

- For information about database system requirements, see *Installing.*

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 general steps are as follows:

1. Locate the **_site.properties** file in the **LISA_HOME** directory and change the name to **site.properties**.
2. Modify the new **site.properties** file, uncommenting the properties to match the target database.
3. When configuring to a DevTest supported external database, such as Oracle, DB2, MySQL, or SQL Server:
    a. Uncomment the following properties for that external database in the appropriate section.

    For example:

    lisadb.pool.common.driverClass=oracle.jdbc.driver.OracleDriver lisadb.pool.common.url

    lisadb.pool.common.url=jdbc:oracle:thin:@HOST:1521:SID

    b. Update the **lisadb.pool.common.url** property with the appropriate hostname, port, and SID values.

    c. Update the following user and password properties with the correct values for accessing the database:

    lisadb.pool.common.user

    lisadb.pool.common.password

d. Comment out the following two properties for the Derby database:

#lisadb.pool.common.driverClass=org.apache.derby.jdbc.ClientDriver

#lisadb.pool.common.url=jdbc:derby://localhost:1528/database/lisa.db;create=true

e. Set the following property to false:

lisadb.internal.enabled=false

4. Start the registry or DevTest Workstation.

Configure DevTest to Use DB2

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

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

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

lisadb.pool.common.driverClass=com.ibm.db2.jcc.DB2Driver
lisadb.pool.common.url=jdbc:db2://HOSTNAME:PORT/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 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. Set the **lisadb.internal.enabled** property to false.
9. Add the DB2 JDBC driver to both **LISA_HOME\lib\shared** directory and **LISA_HOME\webserver\phoenix\phoenix-1.0.0\WEB-INF\lib** directory.

   Important! The JDBC driver must be added to both directories.
10. Start the registry.

Configure DevTest to Use MySQL

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

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

```
lisadb.reporting.poolName=common
lisadb.vse.poolName=common
lisadb.acl.poolName=common
lisadb.broker.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. Set the **lisadb.internal.enabled** property to false.
7. Add the MySQL JDBC driver to both the **LISA_HOME\lib\shared** directory and the **LISA_HOME\webserver\phoenix\phoenix-1.0.0\WEB-INF\lib** directory. The minimum version of the MySQL JDBC driver is 5.1.25.

   Important! The JDBC driver must be added to both directories.
8. Start the registry.

Configure DevTest to Use Oracle

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

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

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

lisadb.pool.common.driverClass=oracle.jdbc.driver.OracleDriver
lisadb.pool.common.url=jdbc:oracle:thin:@HOST:1521:SID
lisadb.pool.common.user=oracle_username
lisadb.pool.common.password=oracle_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.
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. Set the **lisadb.internal.enabled** property to false.
8. Add the Oracle JDBC driver to both LISA_HOME\lib\shared directory and **LISA_HOME\webserver\phoenix\phoenix-x.0.0\WEB-INF\lib** directory.

   You can download the driver from http://www.oracle.com/technetwork/database/features/jdbc/index-091264.html. We recommend that you download the latest driver, even if you are using an older database server. In most cases, **ojdbc7.jar** is the correct driver for Java 1.7 (which ships with DevTest). If you are using Java 1.6, then use **ojdbc6.jar**. If you are using Java 1.5, then use **ojdbc5.jar**.

   Important! The JDBC driver must be added to both directories.
9. Start the registry.

Configure DevTest to Use SQL Server

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

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

```
lisadb.reporting.poolName=common
lisadb.vse.poolName=common
lisadb.acl.poolName=common
lisadb.broker.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. Set the **lisadb.internal.enabled** property to false.
7. Add the JDBC driver for SQL Server to both **LISA_HOME\lib\shared** directory and **LISA_HOME\webserver\phoenix\phoenix-x.0.0\WEB-INF\lib** directory.

   Important! The JDBC driver must be added to both directories.

8. Start the registry.

External Enterprise Dashboard Database Configuration

The Enterprise Dashboard uses an internal Derby database by default. We recommend that you use an external database instead.

The Enterprise Dashboard does not support IBM DB2.

Follow these steps:

1. Log on to the server where the Enterprise Dashboard is installed.
2. Navigate to LISA_HOME and open the **local.properties** file.
3. Locate the following lines in the Enterprise Dashboard Options section, which specifies whether to use the internal Derby database in the Enterprise Dashboard.
   ```
   # Should the internal Derby DB instance in the Enterprise Dashboard be started?
   dradisdb.internal.enabled=true
   ```
4. Change the value to false.
   ```
   dradisdb.internal.enabled=false
   ```
5. Perform one of the following actions:
   - To use an Oracle database, edit the following properties to set values to an external database similar to the following example:

     ```
     lisadb.pool.dradis.driverClass=oracle.jdbc.driver.OracleDriver

     lisadb.pool.dradis.url=Jdbc:oracle:thin:@[HOST]:[PORT]:[SID]

     lisadb.pool.dradis.user=oracle_username

     lisadb.pool.dradis.password=oracle_password
     ```
   - To use a SQL Server database, edit the following properties to set values to an external database similar to the following example:

     ```
     lisadb.pool.dradis.driverClass=com.microsoft.sqlserver.jdbc.SQLServerDriver

     lisadb.pool.dradis.url=jdbc:sqlserver://[SERVER]:[PORT];databaseName=[DATABASENAME]

     lisadb.pool.dradis.user=sqlserver_username

     lisadb.pool.dradis.password=sqlserver_password
     ```
   - To use a MySQL database, edit the following properties to set values to an external database similar to the following example:

     ```
     lisadb.pool.dradis.driverClass=com.mysql.jdbc.Driver

     lisadb.pool.dradis.url=jdbc:mysql://DBHOST:DBPORT/DBNAME

     lisadb.pool.dradis.user=mysql_username

     lisadb.pool.dradis.password=mysql_password
     ```

6. Save the local.properties file.

Database Maintenance

You can use various properties to manage the amount of data in the database.

- For information about the reporting properties, see Automatic Reporting Maintenance (see page 50).
- For information about the access control (ACL) properties, see Automatic Deletion of Audit Log Entries (see page 51).
- For information about the CA Continuous Application Insight properties, see Automatic Deletion of Transactions (see page 52) and Automatic Deletion of Cases (see page 52).

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 (see page 116) 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 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 of the DevTest Portal. The properties appear in the Settings tab.

Automatic Deletion of Cases

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

By default, the automatic deletion of cases is enabled.

The following properties are available:

Enable cleaner

Controls whether the automatic deletion of cases is enabled.

Cleanup frequency

Specifies how often the cleaner process runs. The value is in number of minutes.

Maximum age

Specifies the age of cases that the cleaner process deletes. The value is in number of minutes.

You can configure these properties from the Agents window of the DevTest Portal. The properties appear in the Settings tab.

Chapter 3: License Administration

This section contains the following topics:

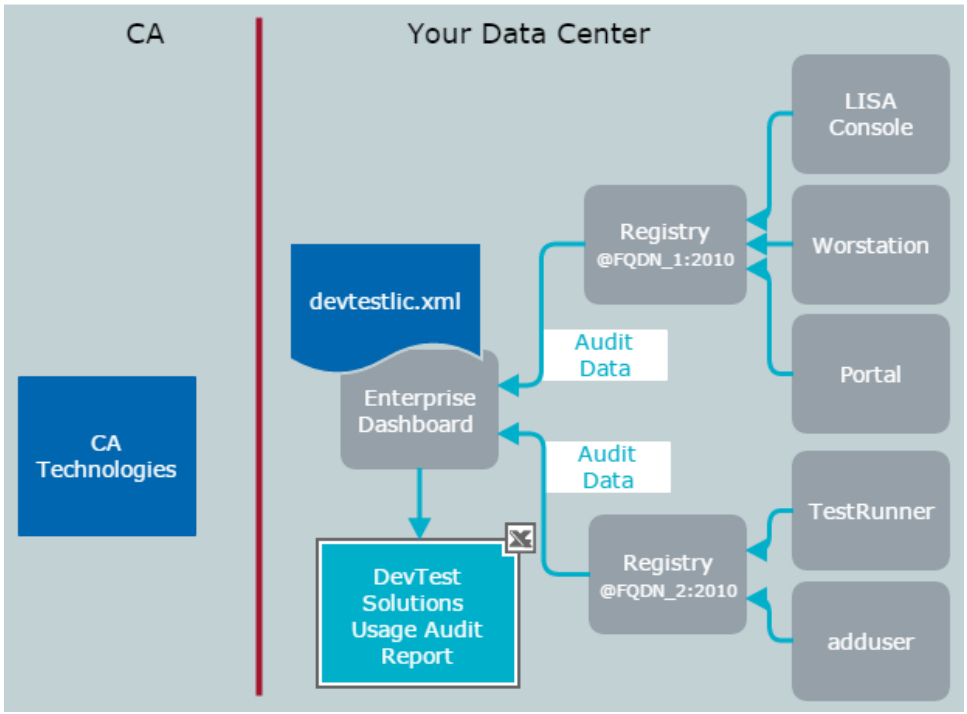
Honor-Based Licensing

Honor-based licensing overview:

- The 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 55).

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 DevTest Console, 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 topic includes:

- Usage-based license agreement
- File-based license activation
- ACL activation
- Honor-based compliance
- Continuous collection of usage data by user type
- Usage Audit Report

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 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 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 delivered via 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 associated with their role. Using the Administration tab on the Server Console, 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, browses to the Portal, or browses to the DevTest Console, 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 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 108) 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 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.

Customers who wish to evaluate functionality that has not been purchased should contact their Account Team to discuss a Proof of Concept trial, or to address questions about the new functionality. Any Support cases 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 onto the Enterprise Dashboard and generates the Usage Audit Report for a specified time period. Data from the specific UI or CLI users log into 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 detailed in the Audit Reports can be compared to the license agreement to determine the level of compliance. 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:

- First tab: Overview
- User Type tabs: Admin User, PF Power User, SV Power User, Test User, Runtime User

  There is a tab for each user type with any activity over the time period, either concurrent or single usage. While Admin User is included in the report, you do not need to consider it when evaluating your compliance to your licensing agreement.
- Last tab: Component by User, with a column for each UI or CLI with access during this time period. Entries on this tab do not necessarily represent concurrent access.

Overview

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

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.
- Key ID: The license key identifier
- Expiration: The date the license expires
- Reporting Period Start: The starting date for this report.
- Reporting Period End: The ending date for this report.

User Type and Count

Reported user types include PF Power User, SV Power User, Test Power User, and Runtime User. Admin User is also reported, although it is not counted as a user type for licensing.

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:

a. PF Power User or SV Power User (equivalent user types)

b. Test Power User

c. Runtime User

d. Admin User

See User Types (see page 61).

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.

The Count calculations for the report are based on group counts where the maximum concurrency count is reached. The Min, Max, and standard deviation convey the distribution across registries. See the Count Calculation Example (see page 60).

User Type tabs (Admin User, PF Power User, SV Power User, Test User, Runtime User)

Each User Type tab reports metrics and statistics on concurrency groups that reached the maximum concurrency count for this reporting period.

- Average:
  - The average is calculated for each registry that is part of the concurrent group.
  - Total is for the user type. This value also appears on the Overview page.
- STDEV: Standard deviation that is calculated for each registry sample set.
- Minimum: The lowest value for each registry sample set.
- Maximum: The highest value for each registry sample set.

Component By User

The Component by User tab reports user session data from each registry that is connected to the Enterprise Dashboard.

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. User types are displayed as:

- ADMIN_USER
- PF_POWER_USER
- SV_POWER_USER
- TEST_POWER_USER
- SVT_RUNTIME_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 Console, DevTest Workstation, and Portal. Examples of CLIs include AddUser and Test Runner. Columns may vary by report, depending on where users log in. Some examples follow:

- Add User: The number of times this user initiated the adduser command-line utility.
- Console: The number of times this user logged in to the DevTest Console for this registry.
- Workstation: The number of times this user opened a DevTest Workstation, connected to the associated registry, and logged in.
- Portal: The number of times this user browsed to http://*hostname*:1507/devtest for this registry and logged in.

Note: For details on generating this report, see Export Usage Audit Data (see page 153).

Count Calculation Example

The Count calculation for the DevTest Solutions Usage Audit Report (see page 57) is a three-step process.

1. Each time a concurrency is detected across the enterprise, a group count is recorded for all involved registries. Consider the following example for the Test Power User user group.
2. For reporting purposes, the group counts with maximum concurrency are used. In this sample example, the shaded rows (1, 3, and 4) are used.

Test Power User				
Timestamp	**Registry 1**	**Registry 2**	**Registry 3**	**Group Count**
1	1	0	3	4
2	0	0	2	2
3	1	2	1	4
4	0	0	4	4
5	1	1	1	3

3. After identifying the maximum group count data by registry, the average of the counts by registry is calculated. The minimum and maximum count values are stored. These three values (average, minimum, and maximum) appear on the report.

Test Power User						
Registry	**Count 1**	**Count 2**	**Count 3**	**Avg**	**Min**	**Max**
Registry 1	1	1	0	0.67	0	1
Registry 2	0	2	0	0.67	0	2
Registry 3	3	1	4	2.67	1	4
TOTAL				**4**	1	7

This data is summarized on the Overview page of the report as User Type and Count. The Count data that is displayed is combined for each user type. The data is calculated by taking the average of the total Min and Max across the registries in the sample for each user type. The count represents the maximum concurrency for that user type because the sample is composed of only group counts with maximum concurrency.

User Type	Count
Test Power User	**4**

User Types

ACL has the following user types:

- PF Power User
- SV Power User
- Test Power User
- Runtime User

Note: The Admin User user type, which is displayed on the Roles page as a combination PF Power User and SV Power User, is not used in licensing. The Admin User is, however, included in the DevTest Solutions Usage Audit Report for your information.

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:

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

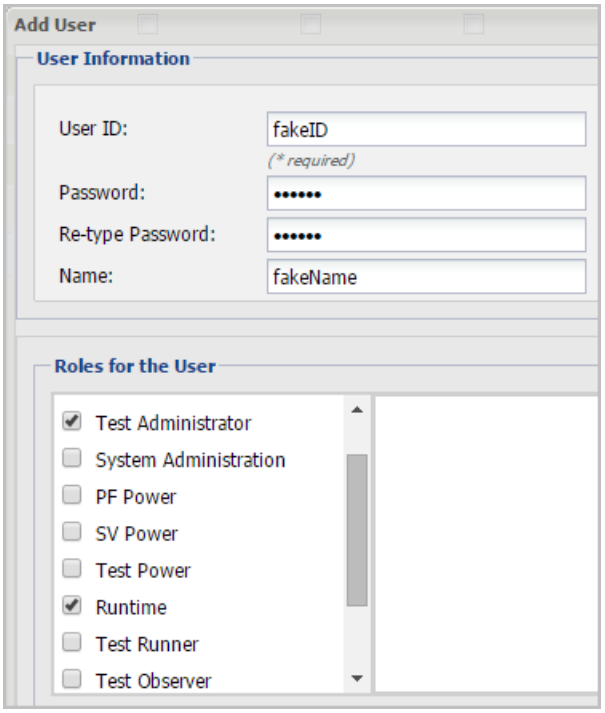
Example of How the User Type is Determined for Auditing Purposes

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

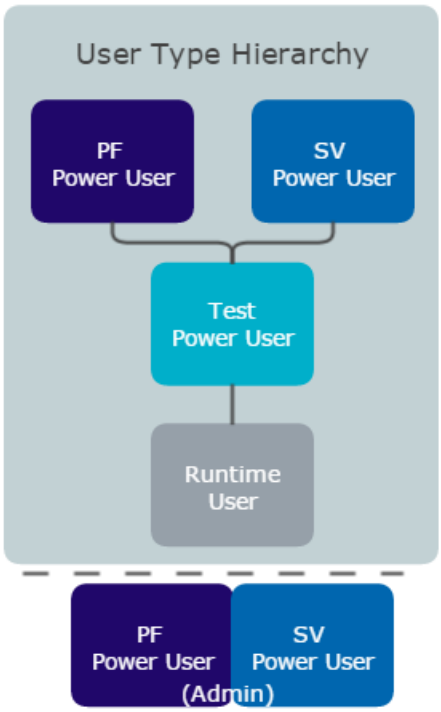


Different user types are associated with the assigned roles:

- 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 84) for details.

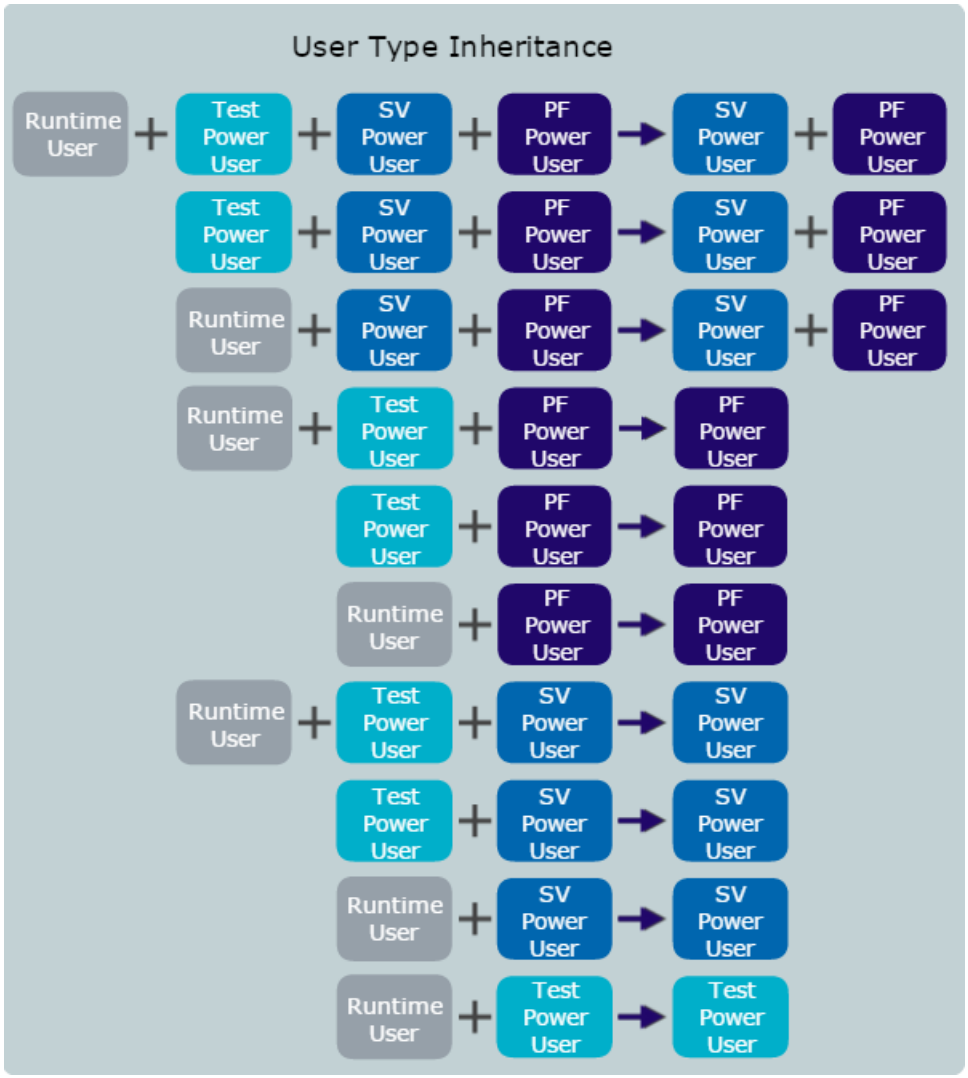
User types are hierarchical.



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. In this 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.



Chapter 4: Security

This section contains the following topics:

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.

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\webreckeys.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=33aa310aa4e18c114dacf86a33cee898
```

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=33aa310aa4e18c114dacf86a33cee898
```

   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-----
   ```

   ```
   MIICEzCCAXygAwIBAgIEThZnYzANBgkqhkiG9w0BAQUFADBOMQswCQYDVQQGEwJ
   DQjELMAkGA1UE
   ```

   ```
   CBM420IxCzAJBgNVBAcTAkNCMQswCQYDVQQKEwJDQjELMAkGA1UECxMCQ0IxCzA
   JBgNVBAMTAkNC
   ```

   ```
   MB4XDTExMDcwODAyMTE0N1oXDTEyMDcwNzAyMTE0N1owTJELMAkGA1UEBhMCQ0I
   xCzAJBgNVBAgT
   ```

   ```
   AkNCMQswCQYDVQQHEwJDQjELMAkGA1UECDMCQ0IxCzAJBgNVBAsTAkNCMQswCQY
   DVQQDEwJDQjCB
   ```

   ```
   nzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAhYfaN+dCrKQwYZ+KeaaPUI8DeXN
   iqQ/mS+KGnXnh
   ```

   ```
   Pz08vdX/7HDLW4pzFhntjmkxxOi9dMwlO2thTD1cOxI571PotenMENo4nyiUAEn
   MK9MTiWEYr2cQ
   ```

   ```
   b6/TUueBCjRJ9I0GPCI0WPS+0Na2Q/wq8gPCHmDRpw1Xgo4uZ1v6C/ECAwEAATA
   NBgkqhkiG9w0B
   ```

   ```
   AQUFAAOBgQByCsX9EoBFIGhcSwoRwEvapIrv8wTaqQPOKKyeIevSmbnERRu6+oi
   +cJftbdEfw6GG
   ```

   ```
   CBddJH+dGZ9VeqLU8zBGasbU+JPzG5ElOgOXcUGeQQEaM1YMv6XWrIwNSljQk/M
   PZSt3ROtJOlae
   ```

   ```
   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=33aa310aa4e18c114dacf86a33cee898
```

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.

Using HTTPS Communication with the DevTest Console

Complete the following tasks to enable HTTPS communication with the DevTest Console.

1. Generate a New Key Pair and Certificate (see page 73)
2. Copy the New Keystore to LISA_HOME (see page 74)
3. Update lisa.webserver Properties (see page 75)

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://wiki.eclipse.org/Jetty/Howto/Configure_SSL.

Follow these steps:

1. Open a command prompt window.
2. Type the following command:
   `cd JAVA_HOME\bin`
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.

Copy the New Keystore to LISA_HOME

Follow these steps:

1. Copy the new keystore file to your **LISA_HOME** directory.
2. Rename the keystore file to **webserver.ks**.

   Note: webserver.ks is the default file specified in the lisa.properties file. If you want to use a different file name, open lisa.properties and modify the **lisa.webserver.ssl.keystore.location** property to reflect the correct path and file name. For more information, see Update Webserver Properties (see page 75).

Update Webserver Properties

Follow these steps:

1. Open the local.properties file in your LISA_HOME directory.
2. Add the following properties to this file.

```
# enable https and setup the webserver ssl keystore
 lisa.webserver.https.enabled=true
 lisa.webserver.ssl.keystore.location={{LISA_HOME}}webserver.ks
 lisa.webserver.ssl.keystore.password=yourpassword
 lisa.webserver.ssl.keymanager.password=yourpassword
 lisa.webserver.port=8443
# should lisa workstation use https when launching the portals?
 lisa.portal.use_https=true
 lisa.portal.url.prefix=http://
```

3. Modify each property to specify the correct value.

   lisa.webserver.https.enabled

   Set this property to **true** to use HTTPS with the DevTest Console.

   lisa.webserver.ssl.keystore.location

   The default value for this property is **{{LISA_HOME}}webserver.ks**. Modify this value if you want to use a keystore file with a different name or in a different directory.

   lisa.webserver.ssl.keystore.password

   Set this property to the password you defined when generating your keystore file.

   lisa.webserver.ssl.keymanager.password

   Set this property to the key manager password you defined when generating your keystore file. Unless you specified a different password, this password is the same as your keystore password.

   lisa.webserver.port

   Setting this property is optional, but the default port for HTTPS is 8443.

   lisa.portal.url.prefix

   Change the value for this property from **http://** to **https://**.

   Note: The first time the system reads the passwords in this local.properties, it converts the password to an encrypted property.

4. Save your changes and close local.properties.
5. 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 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. Access control (ACL) is enabled by default.

This section contains the following topics:

ACL Overview

Access Control Lists (ACL) is a widely used security mechanism. ACL grants users access to only the application functionality they require to perform their role-based activities. ACL is required for DevTest Solutions so that you can monitor and maintain compliance to the license agreement. License agreements are based on the maximum number of concurrent user sessions.

This overview discusses the following ACL topics:

- Planning Deployment of ACLs
- Authentication
- Authorization
- User Sessions
- Automated Test Cases
- ACL Database

Planning Deployment of ACLs

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

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

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

ACL Adminstration

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 Solutions 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 Solutions 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 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 needs to ensure that the ACL/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 111) 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 Solutions 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 ACL to Use LDAP Authentication (see page 118).

Authorization

DevTest limits what DevTest features individual users can access based on their business role. DevTest Solutions 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 105) are assigned unique standard roles (see page 84).

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 authenticated by LDAP (see page 120).

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 61) are categories that include multiple roles. See ACL and User Sessions

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 82).

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 37).

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 will run with valid login credentials.

Permission Types

Permissions can be divided into the following types:

- Boolean
- Numeric limit
- List
- Custom

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.

List

A permission can be associated with a list of strings. The strings could be regular expressions.

Custom

Cloud-based provisioning of development and test environments use custom properties. DevTest creates custom properties for specifying the user name and password that is required to access the Virtual Lab Manager (VLM) provider.

For more information, see Configuring DCM Properties in *Using CA Application Test.*

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 97). You can change (see page 113) the permissions that DevTest assigns to a standard role. In addition, you can delete (see page 113) a standard role.

PF Power User and SV Power User (see page 85)

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

- Super User
- DevTest Administrator

PF Power User (see page 87)

The PF Power User user type includes the following role:

- PF Power

SV Power User (see page 88)

The SV Power User user type includes the following roles:

- Test Administrator
- SV Power
- Test Runner

Test Power User (see page 91)

The Test Power User user type includes the following roles:

- Test Power
- Test Observer
- Load Tester
- User

Runtime User (see page 95)

The Runtime User user type includes the following roles:

- System Administrator
- Runtime
- Guest

Combination PF Power User and SV Power User User Types

The PF Power User / SV Power User combination user type has the following roles:

- Super User
- DevTest Administrator

Super Users and System Administrators can reassign role permissions. See Add and Update Roles (see page 113).

Super User

The Super User role has all the standard permissions.

- LISA Console Administration (see page 97)
- User and Role Administration (see page 98)
- Resource Administration (see page 98)
- Test and Suite Administration (see page 98)
- Virtual Services Administration (see page 98)
- DevTest Server Administration (see page 100)
- CVS Administration (see page 100)
- Report Administration (see page 101)
- Metric and Event Administration (see page 102)
- Pathfinder Administration (see page 102)
- DevTest Workstation (see page 103)
- Cloud Lab Integration (see page 104)
- Can access any target

DevTest Administrator

The DevTest Administrator role has full standard permissions, except for the LISA Console Administration, User and Role Administration, and Resource Administration. That is:

- Partial from LISA Console Administration (see page 97)
  - View VSEasy Console
- Test and Suite Administration (see page 98)
- Virtual Services Administration (see page 98)
- DevTest Server Administration (see page 100)
- CVS Administration (see page 100)
- Report Administration (see page 101)
- Metric and Event Administration (see page 102)
- Pathfinder Administration (see page 102)
- DevTest Workstation (see page 103)
- Cloud Lab Integration (see page 104)
- Can access any target

PF Power User User Type

The PF Power User user type is comprised of the PF Power role.

Super Users and System Administrators can reassign role permissions. See Add and Update Roles (see page 113).

PF Power

The PF Power role has the following permissions:

- Partial from DevTest Console Administration (see page 97)
  - Access Pathfinder Console (PFP)
  - View VSEasy Console
- Test and Suite Administration (see page 98)
- Partial from Virtual Services Administration (see page 98)
  - View VSE Dashboard
  - VSE Service Deployment
  - VSE Service Archive Retrieval
  - Start VSE Service
  - Stop VSE Service
  - Update Virtual Service Capacity
  - Update Virtual Service Think Scale
  - Update Virtual Service Auto-Restart
  - Set Virtual Service Execution Mode
  - Reset Virtual Service Transaction Counts
- CVS Administration (see page 100)
- Report Administration (see page 101)
- Metric and Event Administration (see page 102)
- CAI Administration (see page 102)
- Partial from DevTest Workstation (see page 103)
  - Start DevTest Workstation
  - Create, Edit, and View a Configuration
  - Create, Edit, and View a Staging Document
  - Create, Edit, and View a Suite
  - Create, Edit, and View a Test Case
  - Create, Edit, and View an Audit Document

        - View Virtual Service Model
        - View Virtual Service Image
        - Execute ITR
        - Execute Instant Replay
        - View ITR Properties
        - View ITR Test Events
        - List Available Labs
        - Start and Stop Labs
        - Expand Labs
        - Kill Labs of other users
    - Can access any target

SV Power User User Type

The SV Power User user type has the following roles:

- Test Administrator
- SV Power
- Test Runner

Super Users and System Administrators can reassign role permissions. See Add and Update Roles (see page 113).

Test Administrator

The Test Administrator role has the following permissions:

- Partial from LISA Console Administration (see page 97)
    - View VSEasy Console
- Test and Suite Administration permission (see page 98)
- Virtual Services Administration permission (see page 98)
- DevTest Workstation permission (see page 103)
- Cloud Lab Integration permission (see page 104)
- Can access any target

SV Power

The SV Power role has the following permissions:

- Partial from LISA Console Administration (see page 97)
  - View VSEasy Console
- Test and Suite Administration permission (see page 98)
- Partial from Virtual Services Administration (see page 98)
  - View VSE Dashboard
  - VSE Service Deployment
  - VSE Service Archive Retrieval
  - Start VSE Service
  - Stop VSE Service
  - Update Virtual Service Capacity
  - Update Virtual Service Think Scale
  - Update Virtual Service Auto-Restart
  - Update Virtual Service Group Tag
  - Set Virtual Service Execution Mode
  - Reset Virtual Service Transaction Counts
  - Heal Virtual Service Image
- CVS Administration (see page 100)
- Report Administration (see page 101)
- Metric and Event Administration (see page 102)
- Partial from LISA Workstation (see page 103)
  - Start DevTest Workstation
  - Create, Edit, and View a Configuration
  - Create, Edit, and View a Staging Document
  - Create, Edit, and View a Suite
  - Create, Edit, and View a Test Case
  - Create, Edit, and View an Audit Document
  - Create, Edit, and View Virtual Service Model
  - Create, Edit, and View Virtual Service Image
  - Execute ITR
  - Execute Instant Replay

- View ITR Properties
- View ITR Test Events
- List Available Labs
- Start and Stop Labs
- Expand Labs
- Kill Labs of other users

- Can access any target

Test Runner

The Test Runner role has the following permissions:

- Partial from LISA Console Administration (see page 97)
  - View VSEasy Console
- Test and Suite Administration permission (see page 98)
- DevTest Workstation permission (see page 103)
- Cloud Lab Integration permission (see page 104)
- Can access any target

Test Power User User Type

The Test Power User user type includes the following roles:

- Test Power
- Test Observer
- Load Tester
- User

Super Users and System Administrators can reassign role permissions. See Add and Update Roles (see page 113).

Test Power

The Test Power role has the following permissions:

- Partial from LISA Console Administration (see page 97)
  - View VSEasy Console
- Test and Suite Administration (see page 98)
- Partial from Virtual Services Administration (see page 98)
  - View VSE Dashboard
  - VSE Service Deployment
  - VSE Service Archive Retrieval
  - Start VSE Service
  - Stop VSE Service
  - Update Virtual Service Capacity
  - Update Virtual Service Think Scale
  - Update Virtual Service Auto-Restart
  - Update Virtual Service Group Tag
  - Set Virtual Service Execution Mode
  - Reset Virtual Service Transaction Counts
- CVS Administration (see page 100)
- Report Administration (see page 101)
- Metric/Event Administration (see page 102)
- All from DevTest Workstation Permission (see page 103) except:
  - Create or Edit a Virtual Service Model
  - Create or Edit a Virtual Service Image
- Cloud Lab Integration (see page 104)

- Can access any target
- Can only access a given list of targets.

Test Observer

The Test Observer role has the following permissions:

- Partial from LISA Console Administration (see page 97)
  - View VSEasy Console
- Test/Suite Administration permission (see page 98)
- Partial from LISA Workstation (see page 103)
  - Start DevTest Workstation
  - View a Configuration
  - View a Staging Document
  - View a Suite
  - View a Test Case
  - View an Audit Document
  - View Virtual Service Model
  - View Virtual Service Image
  - View ITR Properties
  - View ITR Test Events
- Cloud Lab Integration permission (see page 104)
- Can only access a given list of targets

Load Tester

The Load Tester role has the following permissions:

- Partial from LISA Console Administration (see page 97)
  - View VSEasy Console
- Test/Suite Administration permission (see page 98)
- CVS Administration permission (see page 100)
- Partial from LISA Workstation (see page 103)
  - Start DevTest Workstation
  - View a Configuration
  - View a Staging Document
  - View a Suite
  - View a Test Case
  - View an Audit Document
  - View Virtual Service Model
  - View Virtual Service Image
  - View ITR Properties
  - View ITR Test Events
- Cloud Lab Integration permission (see page 104)
- Can only access a given list of targets

User

The User role has the following permissions:

- Partial from LISA Console Administration (see page 97)
  - View VSEasy Console
- Test/Suite Administration permission (see page 98)
- Partial from LISA Workstation (see page 103)
  - Start DevTest Workstation
  - View a Configuration
  - View a Staging Document
  - View a Suite
  - View a Test Case
  - View an Audit Document
  - View Virtual Service Model
  - View Virtual Service Image
  - View ITR Properties
  - View ITR Test Events
  - List Available Labs
  - Start/Stop Labs
  - Expand Labs
- Can access any target

Runtime User User Type

The Runtime User user type has the following roles:

- System Administrator
- Runtime
- Guest

Super Users and System Administrators can reassign role permissions. See Add and Update Roles (see page 113).

System Administrator

The System Administrator role has the following default permissions:

- Partial from DevTest Console Administration (see page 97)
  - Access DevTest Console
  - Access Server Console
- User and Role Administration (see page 98)
- Resource Administration (see page 98)
- Partial from Virtual Services Administration (see page 98)
  - Configure VSE Tracking Data Cleanup
  - Stop VSE Server
  - Reset VSE Server
  - Monitor VSE Server
- DevTest Server Administration (see page 100)

Runtime

The Runtime role has the following default permissions:

- Partial from DevTest Console Administration (see page 97)
  - View VSEasy Console
- Test and Suite Administration (see page 98)
- Partial from Virtual Services Administration (see page 98)
  - View VSE Dashboard
  - VSE Service Deployment (R)
  - VSE Service Archive Retrieval (R)
  - Start VSE Service (R)
  - Stop VSE Service (R)
  - Update Virtual Service Capacity
  - Update Virtual Service Think Scale
  - Update Virtual Service Auto-Restart
  - Update Virtual Service Group Tag
  - Set Virtual Service Execution Mode (R)
  - Reset Virtual Service Transaction Counts (R)
- CVS Administration (see page 100)
- Report Administration (see page 101)
- Partial from DevTest Workstation (see page 103)
  - Start DevTest Workstation
  - View a Configuration
  - View a Staging Document
  - View a Suite
  - View a Test Case
  - View an Audit Document
  - View Virtual Service Model
  - View Virtual Service Image
  - View ITR Properties
  - View ITR Test Events
- Partial from Cloud Lab Integration (see page 104)
  - List Available Labs

- Expand Labs
- Can only access a given list of targets

Guest

The Guest role has the following default permissions:

- Partial from DevTest Workstation (see page 103)
  - Start DevTest Workstation
  - View a Suite
  - View a Test Case

Standard Permissions

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

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

The following top-level permissions are automatically created.

Permissions:

LISA Console Administration Permission

The LISA Console Administration permission has the following child permissions.

Child Permission	Type	Description
Access LISA Console	Boolean	Allows a user to log in to the DevTest Console.
Access Server Console	Boolean	Allows a user to log in to the Server Console.

Access Pathfinder Console	Boolean	Allows a user to log in to the Pathfinder Console.
View VSEasy Console	Boolean	Allows a user to view the VSEasy Console.

User and Role Administration Permission

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

Resource Administration Permission

The Resource Administration permission is a Boolean permission that allows a user to manage resource groups (see page 121).

Test and 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 in the Registry Monitor.
Kill Test	Boolean	Allows a user to kill a test in the Registry Monitor.
Optimize Test	Boolean	Allows a user to optimize a test in the Registry Monitor.
View Test	Boolean	Allows a user to view a test in the Registry Monitor.
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 view the VSE Dashboard in the Server Console.
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 in the Server Console or the Registry Monitor.
Reset Coordinator	Boolean	Allows a user to reset a coordinator in the Server Console or the Registry Monitor.
Stop Coordinator	Boolean	Allows a user to stop a coordinator in the Server Console or the Registry Monitor.
Monitor Simulator	Boolean	Allows a user to view a simulator status message in the Server Console or the Registry Monitor.
Reset Simulator	Boolean	Allows a user to reset a simulator in the Server Console or the Registry Monitor.
Stop Simulator	Boolean	Allows a user to stop a simulator in the Server Console or the Registry Monitor.

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 Dashboard.
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 Console.
View report data	Boolean	Allows a user to view his or her own report data in the Reporting Console.
View report data of other users	Boolean	Allows a user to view other users' report data in the Reporting Console.
View PDF report data	Boolean	Allows a user to view his or her own report data as a PDF file in the Reporting Console.
View PDF report data of other users	Boolean	Allows a user to view other users' report data as a PDF file in the Reporting Console.
Import XML report data	Boolean	Allows a user to import XML data in the Reporting Console.
Export XML report data	Boolean	Allows a user to export his or her own report data as an XML file from the Reporting Console.
Export XML report data of other users	Boolean	Allows a user to export others users' report data as an XML file from the Reporting Console.
Export Excel report data	Boolean	Allows a user to export his or her own report data as an Excel file from the Reporting Console.
Export Excel report data of other users	Boolean	Allows a user to export other users' report data as an Excel file from the Reporting Console.
Delete report data	Boolean	Allows a user to delete his or her own report data from the Reporting Console.
Delete report data of other users	Boolean	Allows a user to delete other users' report data from the Reporting Console.
Create report filter	Boolean	Allows a user to create a filter in the Reporting Console.
Delete report filter	Boolean	Allows a user to delete his or her own filters from the Reporting Console.
Delete report filter of other users	Boolean	Allows a user to delete other users' filters from the Reporting Console.

View all filters	Boolean	Allows a user to view all users' filters in the Reporting Console.

Metric and 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.

CA Continuous Application Insight Administration

The CA Continuous Application Insight (CAI) Administration permission has the following child permissions.

Child Permission	Type	Description
View Paths	Boolean	Allows a user to view paths in the DevTest Portal.
Create Baselines	Boolean	Allows a user to create baseline test cases and suites in the DevTest Portal.
Create Virtual Service Models	Boolean	Allows a user to create virtual service models and raw traffic files in the DevTest Portal.
Extract Data	Boolean	Allows a user to extract test data from XML requests or responses in the DevTest Portal.
View Tickets	Boolean	Allows a user to view tickets in the DevTest Portal.
Edit Tickets	Boolean	Allows a user to edit tickets information in the DevTest Portal.
View Agents	Boolean	Allows a user to view agent information in the DevTest Portal.
Agent Administration	Boolean	Allows a user to stop and start dispatching for an agent in the <dt> Portal.
Import Paths	Boolean	Allows a user to import one or more paths in the DevTest Portal.
Export Paths	Boolean	Allows a user to export one or more paths from the DevTest Portal.
Create Document	Boolean	Allows a user to create a document from transactions in the DevTest Portal.
View Manual Cases	Boolean	Allows a user to view manual test cases in the DevTest Portal.
Create Manual Cases	Boolean	Allows a user to create manual test cases in the DevTest Portal.

Edit Point of Interest	Boolean	Allows a user to edit Point of Interest transactions in the DevTest Portal.
View Point of Interest	Boolean	Allows a user to view Point of Interest transactions in the DevTest Portal.
View Defects	Boolean	Allows a user to view transactions with defects in the 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.

Cloud Lab Integration

The Cloud Lab Integration permission has the following child permissions.

Child Permission	Type	Description
List Available Labs	Boolean	Allows a user to view the list of available labs.
Start / Stop Labs	Boolean	Allows a user to start and stop labs.
Expand Labs	Boolean	Allows a user to dynamically expand a test lab.
Kill Other User's Labs	Boolean	Allows a user to kill a lab that another user started.

Debug DevTest Server

This permission allows a user to create heap dumps and thread dumps and force garbage collection. The permission applies to right-clicking from the Server Console and issuing the commands through the ServiceManager (see page 133) command-line 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 84) for permissions that are associated with each role.

Important! To help prevent unauthorized access, we recommend that you change the default passwords (see page 108) for these users as soon as possible.

admin

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

pfpower

The pfpower user has the PF Power role, a PF Power User user type. The default password is **pfpower**.

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**.

Experience the Roles of Standard Users

DevTest Solutions is installed with standard users, each with unique credentials, and a different role. To prepare for assigning roles to new users, use the standard users to get hands-on familiarity with roles. You can log in with the credentials of each standard user to experience what users will experience as they use the permissions that are associated with the various roles.

Follow these steps:

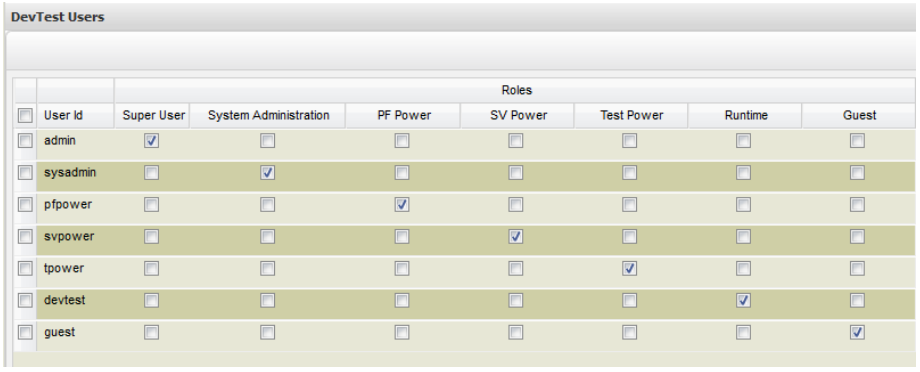
1. Browse to the Server Console and log in as a user with a Super User role.

   `http://hostname:1505`

2. Expand the Administration pane in the left navigation bar. In the Security area, the Super User provides credentials to authenticate users and grants permissions to access features through role assignments.

3. Click Users.

   Standard users are displayed.



   The default password is the same as the User Id for each standard user.

   - Super User: admin, admin
   - PF Power: pfpower, pfpower
   - SV Power: svpower, svpower
   - Test Power: tpower, tpower
   - Runtime: devtest, devtest
   - System Administration: sysadmin, sysadmin
   - Guest: guest, guest

4. Log in to each UI and CLI with the credentials of a Super User.
5. Examine the functionality that the users with this role can access.
6. Repeat the last two steps with other standard users.

This process prepares you to set up access control (ACL). You set up access control by adding users and assigning roles that grant only those permissions that are necessary for the tasks they perform.

Important! As part of access control, we recommend that you change the passwords for standard users (see page 108). Changing passwords is one way to prevent unauthorized users from accessing your system.

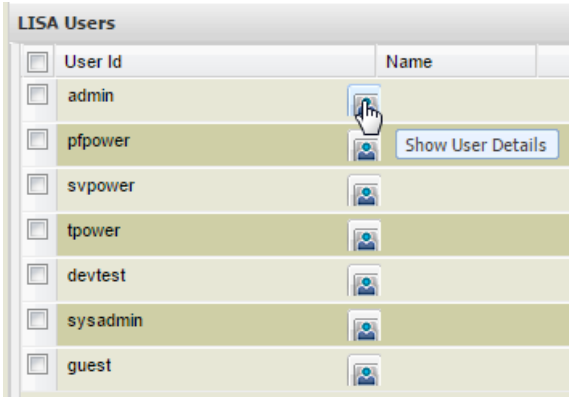
Change Passwords for Standard Users

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. See Start the Server Components.
2. Browse to the DevTest Console.
   `http://localhost:1505`
3. Log in to the DevTest Console. If you have no credentials, take one of the following approaches:
   - If you plan to use LDAP for authentication, log in as the standard Super User.
   - If you plan to define the credentials for users and you have not yet defined yourself as a user, create a user with the Super User role.
4. Click Server Console.
5. Click the Administration navigation tab.
6. Click Users.

   The standard users are displayed.
7. For each standard user, perform the following steps:
   a. Click Show User Details for one of the standard users.



   b. Type a new password in the Password field.
   c. Type the new password in the Re-type Password field.
   d. Click Save.
8. Click Logout.

View User Information from DevTest Workstation

When you are logged in to DevTest Workstation, you can open a dialog that provides the following 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 with Super User access manage users and roles from the Administration panel in the Server Console. Consider the following approach:

1. Review roles and user types.
    a. Select Roles.
    b. On the Permissions tab, 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 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 associated with each role and, optionally, customize permissions.
    e. Optionally, add a new role 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 you specify to authorize the user to perform various activities based on the granted permissions.
3. Verify that you have satisfactorily configured all of your users with the appropriate roles. If you customized existing roles or added new ones, verify the configuration.
4. **Back up your database** strategically per your in-house maintenance policies. This is a precaution that enables you to quickly recover your configuration of users and roles in case of database corruption. The resolution to a corrupted ACL database is to perform a Restore that should be managed by your database administrator.

More information:

Change the Priority Order of Roles (see page 115)
View the Audit Log (see page 116)
adduser Command-Line Utility (see page 117)
Add, Update, and Delete Roles (see page 113)
Add, Update, and Delete Users (see page 111)

Add, Update, and Delete Users

You use the Server Console (see page 137) to add users, change the details for 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 window. Click the drop-down arrow in a column. Select Show/Hide Columns. Select or clear the appropriate check boxes.

To add a user:

1. In the Server Console, display the Administration panel.
2. Click the Users node.
3. At the bottom of the right panel, click Add User.

   The Add User dialog appears.
4. 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.
5. In the Password field, enter a password for the user.

   You can enter any combination of alphanumeric, hyphen (-), underscore (_), and ampersand (@) characters.
6. In the Re-type Password field, enter the password again.
7. 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.
8. In the Misc Info field, enter any additional information.

   The maximum number of characters is 600.
9. In the Roles for the User area, select one or more roles to assign to the user.
10. Click Add User.

    The User Added message appears.
11. Click OK.

To update a user:

1. Click the Show User Details icon to the right of the user ID.

   The User Details dialog appears.

2. Make the appropriate changes.

   You can view the permissions for a role by clicking the role name.

3. Click Save.

To delete a user:

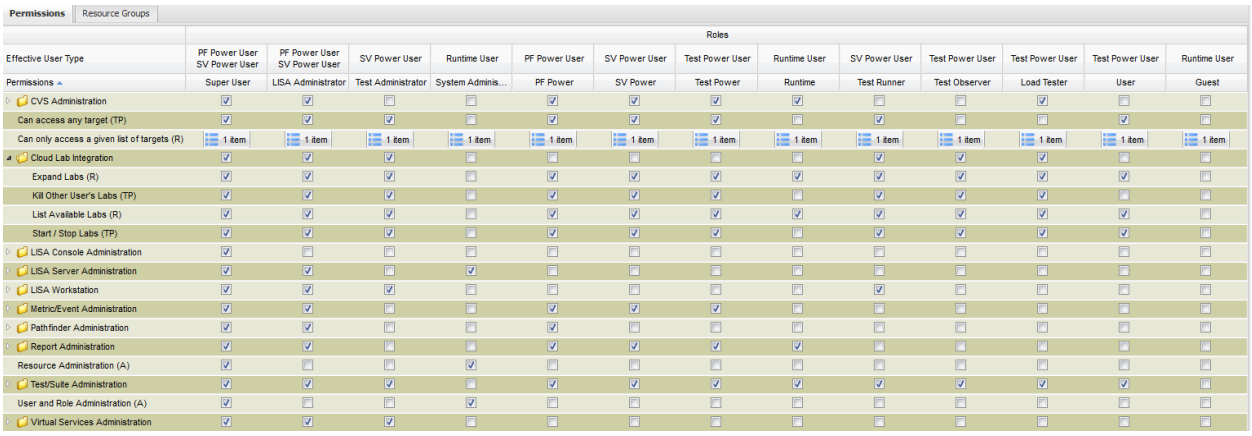
1. Select the check box to the left of the user ID.
2. Click Delete User(s).
3. Click Yes.

Add, Update, and Delete Roles

You use the Server Console (see page 137) to add roles, change the details for a role, and delete roles.

The Server Console displays the roles and permissions in a grid format.

- The permissions appear on the left as rows.
- The roles appear on the top as columns. The role on the left has the highest priority. The role on the right has the lowest priority. The order becomes important when a user has more than one role. The Server Console lets you change the priority order (see page 115).



If you select a parent permission, the child permissions are automatically selected. If you clear a parent permission, the child permissions are *not* automatically cleared. If you clear a child permission, the parent permissions are automatically cleared.

You can show or hide any of the columns on the Roles window. Click the drop-down arrow in the Permissions column. The drop-down arrow in the Permissions column appears when you place the mouse pointer over the column. Select Show/Hide Columns. Select or clear the appropriate check boxes.

To add a role:

1. In the Server Console, display the Administration panel.
2. Click the Roles node.
3. At the bottom of the right panel, click Add Role.

   The Add Role dialog appears.
4. 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.
5. In the Description field, enter the role description.

The maximum number of characters is 200.

6. In the Permissions for new Role area, assign permissions to the role.

   You can use the drop-down box to select an existing role on which to base the new role.

7. Click Add Role.

   A column is added for the new role. The column appears to the right of the existing roles.

To update a role:

1. Locate the role column in the roles and permissions grid.
2. To update a Boolean (see page 83) permission, select or clear the check box.
3. To update a numeric limit (see page 83) permission:
   a. Click the Numeric Limits icon.
   b. For the Maximum number of ... field, select Unlimited, None, or enter a value in the numeric limit.
   c. Click Save.
4. To update a list item (see page 83) permission:
   a. Click the List Items icon.
   b. Add, edit, and delete the list items as necessary.
   c. Click Save.
5. Click Save.

To delete a role:

1. In the roles and permissions grid, select the role name and click the drop-down arrow.
2. Click Delete Role.
3. Click Yes.

Change the Priority Order of Roles

The Server Console displays the roles in order of their priority. The order becomes important when a user has more than one role.

Follow these steps:

1. In the Server Console, display the Administration panel.
2. Do one of the following actions:
   - Click the Roles node.
   - Click Display Roles.
3. Click Reorder Priority.

   The Reset Role Priority Order dialog appears.
4. To change the role order, select roles and drag them to the target destination.
5. Click Save Order.

View the Audit Log

The audit log contains a series of entries for activities that ACL controls. Each entry includes the following information:

Timestamp

When the entry was created.

User ID

The unique ID of the user.

Role

The role that is assigned to the user.

Permission ID

A numeric identifier for the permission.

Permission Name

The permission that controls the activity.

Status

Whether the activity was allowed or denied.

Acted On

The name of the server component, test case, or virtual service model (if applicable).

Other Details

More information (if applicable).

Follow these steps:

1. In the Server Console, display the Administration panel.
2. Do one of the following actions:
   - Click the Audit Log node.
   - Click Display Audit Log.
3. Click Refresh.

adduser Command-Line Utility

You can 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 will have no permissions until a role is assigned from the Server Console.

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 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 ACL to Use LDAP Authentication

You can configure access control (ACL) so that user authentication is based on the information in an LDAP server, instead of the DevTest database. The authorization process continues to use the DevTest database. The ACL administrator needs to consult with your LDAP administrator for configuration and implementation that is based on the property items discussed below.

Note: When you configure ACL to use the LDAP server, users can log in only with accounts that are in their LDAP database. Users will not be able to use any standard user (see page 105) credentials to log in to a DevTest UI or CLI.

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

During the configuration process, you add the following properties:

lisa.acl.ldap.ldapUrl

The URL of the LDAP server.

lisa.acl.ldap.securityPrincipal

The distinguished name of the security principal.

lisa.acl.ldap.securityCredential

The password of the security principal. When the registry starts, it adds the string **_enc** to the property name and encrypts the value.

lisa.acl.ldap.securityAuthentication

The security level to use. The valid values are **none** and **simple**. If you set the value to **none**, the LDAP authentication call ignores the password that the user provided and simply validates the user name. In addition, you do not need to include the **lisa.acl.ldap.securityPrincipal** and **lisa.acl.ldap.securityCredential** properties. If you set the value to **simple**, the user name and password (which is passed as clear text) are validated.

lisa.acl.ldap.baseContext

The distinguished name of the node where the user search begins.

lisa.acl.ldap.userSearchFilter

The search filter that specifies the object class for user entries, for example, **(objectClass=user)**.

lisa.acl.ldap.usernameAttribute

The attribute that specifies the user name, for example, **sAMAccountName**.

lisa.acl.ldap.userSearchAllDepths

Indicates whether to search all the subnodes. The valid values are **true** and **false**.

lisa.acl.ldap.lisaDefaultRole

The default role that is assigned to a user that is added to the DevTest database after being successfully authenticated. If you do not include this property, the default role is Guest.

lisa.acl.ldap.referralSupport

If the LDAP server uses referrals, you can use this property to specify the type of referral. The valid values are **follow**, **ignore**, and **throw**. If you do not include this property, the default value is **follow**.

Note: If the properties file also includes the **lisa.acl.auth.enabled** property and the value is **true**, LDAP authentication does not work properly. Remove or comment out the **lisa.acl.auth.enabled** property.

LDAP authentication uses the same login dialogs that are provided for the default ACL module.

Follow these steps:

1. Open the **local.properties** or **site.properties** file on the computer where the registry is located.
2. Add the following line:

   ```
   lisa.acl.auth.module.impl=com.itko.lisa.acl.custom.BaseLDAPAuthenticationModule
   ```
3. Add the **lisa.acl.ldap.*** properties described earlier in this topic. For example:

   ```
   lisa.acl.ldap.ldapUrl=ldap://172.24.255.255:389
   lisa.acl.ldap.securityPrincipal=CN=admin,OU=users,DC=example,DC=com
   lisa.acl.ldap.securityCredential=adminpwd
   lisa.acl.ldap.securityAuthentication=simple
   lisa.acl.ldap.baseContext=OU=users,DC=example,DC=com
   lisa.acl.ldap.userSearchFilter=(objectClass=user)
   lisa.acl.ldap.usernameAttribute=sAMAccountName
   lisa.acl.ldap.userSearchAllDepths=true
   lisa.acl.ldap.lisaDefaultRole=DevTest Administrator
   ```
4. Save the **local.properties** or **site.properties** file.
5. Start the registry.

Authorize Users Authenticated by LDAP

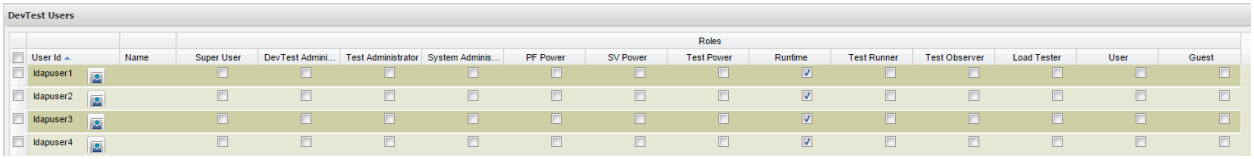
When you configure ACL to user LDAP, you specify a default role with the following setting:

lisa.acl.ldap.lisaDefaultRole

> The default role that is assigned to a user that is added to the DevTest database after being successfully authenticated. If you do not include this property, the default role is Guest.

When a user logs in with valid LDAP credentials, LDAP authenticates the user and if the user does not exist in the DevTest database, a row is automatically added to the database. The row includes the user ID and the role that is currently specified for **lisa.acl.ldap.lisaDefaultRole.**

There are thirteen roles. You could, for example, set the default role to Runtime. When each user logs onto any DevTest UI or CLI the first time, a row is added to the DevTest Users table with their User ID and the default role.



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 Console and log in.

   `http://hostname:1505`
3. Click Server Console and then click the Administrative tab in the left navigation pane.
4. Click Users.

   The DevTest Users table opens.
5. For each user, clear the default role and select the appropriate role.
6. (Optional) To view permissions for a role in the right pane of the User Details dialog, click the Show User Details icon and click the role name.
7. Click Save.

To update the DevTest Users table automatically

1. Set the default to a given role.
2. Ask all users to whom you want to assign that role to log in to DevTest Solutions and then log out.
3. Change the default to another role.
4. Ask all users to whom you want to assign that role to log in to DevTest Solutions and then log out.
5. Repeat Steps 3 and 4 for each role.

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.

This section contains the following topics:

Manage Resource Groups

Use the Server Console 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 window. Click the drop-down arrow in a column. Select Show/Hide Columns. Select or clear the appropriate check boxes.

To add a resource group:

1. In the Server Console, display the Administration panel.
2. Click the Resource Groups node.
3. At the bottom of the right panel, click Add Resource Group. The Add Resource Group dialog appears.
4. In the Name field, enter a unique name for the resource group. The name must contain only alphanumeric characters, spaces, hyphens, or underscores.
5. In the Description field, enter a description of the resource group.
6. Select a resource group or groups by selecting the check box beside the group.
7. Click Add.

   The Resource Groups panel shows the resource with the associated resource group displayed in the Resource Groups column.

To display resource groups:

1. To refresh the display, click the Refresh button at the upper right corner of the window.
2. Resources that are inactive, but still associated with a resource group, display with their labels using "strikethrough" text.

To delete a resource group:

1. Select the black triangle next to the name of the resource group.
2. From the drop-down list, select Delete Resource Group.
3. Click Yes in the Deleting Resource Group dialog.

To remove a resource from a resource group:

Note: You cannot remove a resource from a resource group when the resource is active.

Clear the check box next to the resource group and click Save.

Grant Roles to Resource Groups

To grant roles to resource groups, use the Server Console.

Follow these steps:

1. Select Administration, Security, Resource Groups.
2. Select the check boxes for the roles to associate with each resource group.
3. 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 (see page 123).
2. To open a configuration file in DevTest Workstation, double-click it from the Project Panel.
3. Click the Add icon 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.

Chapter 5: Logging

This section contains the following topics:

Log File Overview

This section describes the following log files and where they are located.

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 the 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.

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 18), 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, CA does 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** file in the **LISA_HOME** directory lets you configure the logging behavior.

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

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

This 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
log4j.logger.org.apache=ERROR
log4j.logger.com.smardec=ERROR
log4j.logger.org.apache.http=ERROR
log4j.logger.org.apache.http.header=ERROR
log4j.logger.org.apache.http.wire=ERROR
log4j.logger.com.mchange.v2=ERROR
log4j.logger.org.hibernate=WARN
log4j.logger.org.jfree=ERROR
log4j.logger.com.jniwrapper=ERROR
log4j.logger.sun.rmi=INFO
```

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
```

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 130).

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.

```
2012-03-05 12:48:01,136 [Event Sink Thread Pool Thread 1] INFO
com.itko.lisa.coordinator.TestRegistryImpl -
Coordinator Servers: 0 Simulator Servers: 0 VSEs: 0 Running vusers: 0 Labs: 1
Memory used 83mb, allocated 158mb, max 227mb (36%) labSims: 0 labVSEs: 0 labCoords:
0
Our cpu usage 0%, system cpu used 16%
```

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.

Test Step Logger

As described in Elements of a Test Step in *Using CA Application Test,* 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.

Chapter 6: Monitoring

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

This section contains the following topics:

Service Manager

The ServiceManager command-line utility lets you perform various actions on a registry, coordinator, simulator, or VSE server.

```
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=tc;"//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
```

Open the Server Console

You can open the Server Console from DevTest Workstation or from a web browser.

To open the Server Console from DevTest Workstation:

Select View, Server Console from the main menu.

To open the Server Console from a web browser:

1. Ensure the registry is running.
2. Enter **http://localhost:1505/** in a web browser.

   If the registry is on a remote computer, replace **localhost** with the name or IP address of the computer.

   The DevTest Console appears.
3. Click Server Console.

View the Component Health Summary

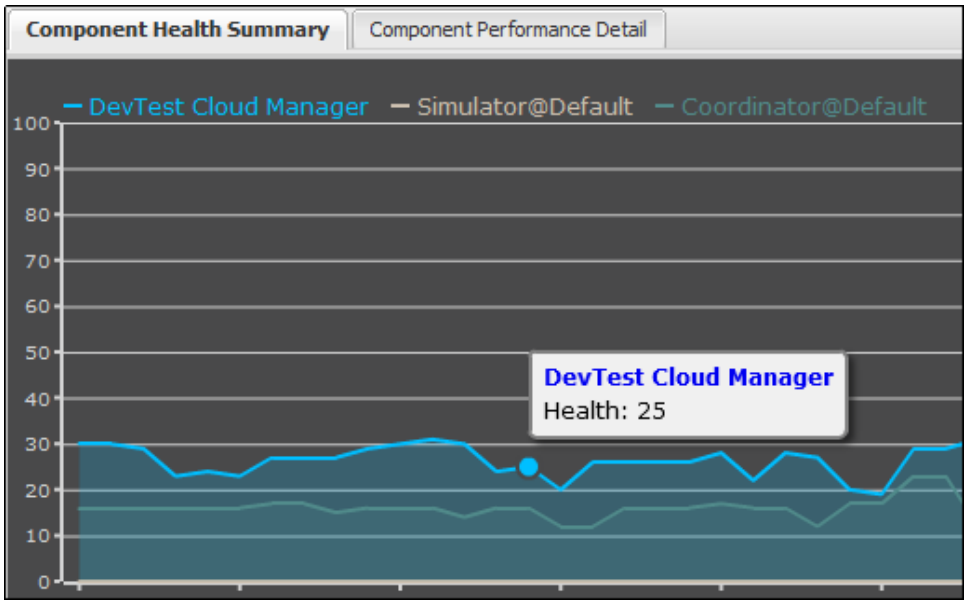
The Component Health Summary tab in the Server Console provides an indicator of the overall health of running components.

The health indicator appears on a scale from 0 through 100. The value is derived from the following statistics:

- Overall CPU load
- Amount of CPU the server is using
- How much of its own heap the server is using

A value of 0 represents an idle system. A value of 100 represents a system that is maxed out.

The following image shows the Component Health Summary tab. The X-axis represents the time. The Y-axis represents the value of the health indicator.



To view the component health summary:

1. Go to the Server Console.
2. Click the DevTest Cloud Manager node in the DevTest Network panel.

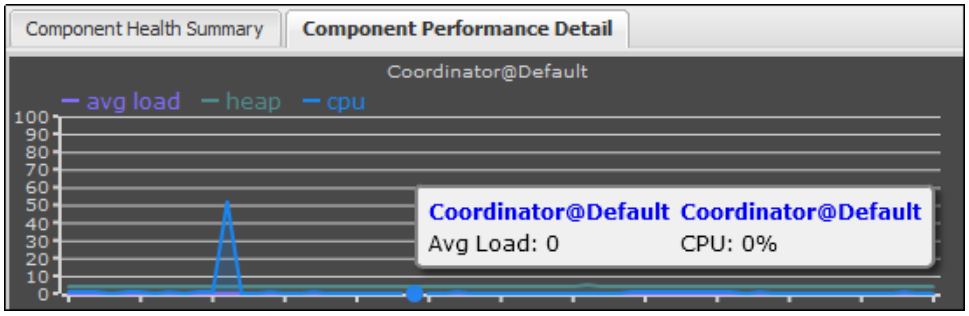
   The data appears in the Component Health Summary tab.

View the Component Performance Detail

The Component Performance Detail tab in the Server Console lets you view the following performance statistics about running lab members:

- **Average load**: The value is shown as an integer.
- **Heap:** The value is shown as a percentage.
- **CPU:** The value is expressed as a percentage.

The following image contains performance data for a coordinator in the Default lab. Tooltips for the average load and CPU metrics are shown.



To view the component performance detail:

1. Go to the Server Console.
2. Click the DevTest Cloud Manager node in the DevTest Network panel.
3. Click the Component Performance Detail tab.
4. Right-click a lab member in the network graph and select View Performance Data.

   The lab member is added to the Component Performance Detail tab. The metrics appear with different colors. Each metric provides a tooltip.

Create Heap Dumps and Thread Dumps

The Server Console lets you create heap dumps and thread dumps for a server component.

Typically, you perform these procedures only when requested to do so by customer support.

An alternate way to create a heap dump is by using the Service Manager command-line utility.

To create a heap dump:

1. Go to the Server Console.
2. Right-click a server component in the network graph and select Dump Heap.
3. When prompted, save the file.
4. Open the file.

   The file contains the fully qualified path name of the .hprof file, for example, **C:\Users\myusername\lisatmp_6.0.7\Coordinator_Server_HeapDump_2012-02-28_08-36-55.hprof**.

Note: This functionality is available only on Sun Java 6 Java runtimes. This diagnostic tool can help Support determine the causes of OutOfMemory conditions. The heap is automatically dumped if an OutOfMemory condition occurs; this button is for manually triggering a heap dump.

To create a thread dump:

1. Go to the Server Console.
2. Right-click a server component in the network graph and select Dump Threads.
3. When prompted, save the thread dump file.
4. Open the thread dump file and view the contents.

Force Garbage Collection

The Server Console lets you force a server component to perform garbage collection.

In addition to performing the garbage collection, this procedure creates a file that contains memory statistics. For example:

```
Before: Memory used 35mb, allocated 97mb, max 227mb (15%) ;
After: Memory used 19mb, allocated 95mb, max 227mb (8%)
```

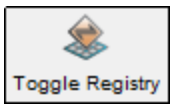
Typically, you perform this procedure only when requested to do so by customer support.

To force garbage collection:

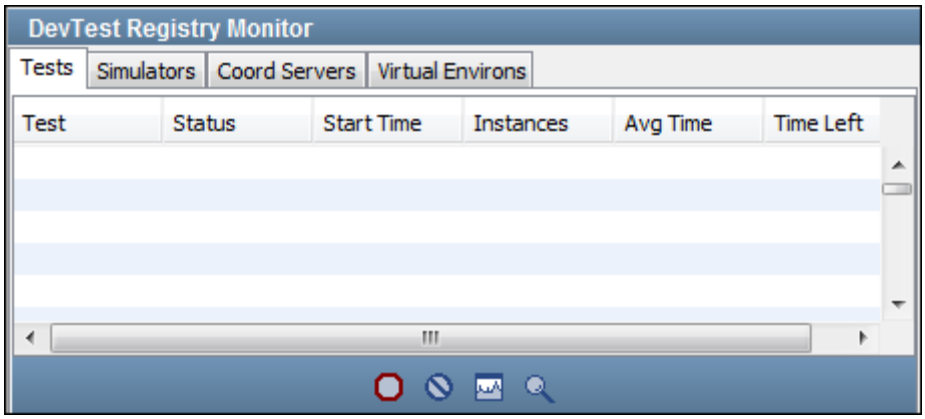
1. Go to the Server Console.
2. Right-click a server component in the network graph and select Force GC.
3. When prompted, save the statistics file.
4. Open the statistics file and view the contents.

Use the Registry Monitor

The Registry Monitor lets you monitor the test cases, simulators, coordinators, and virtual environments for a test suite.

To open the Registry Monitor, click Toggle Registry 

on the main toolbar of DevTest Workstation.



The Registry Monitor has the following tabs:

- Tests Tab (see page 143)
- Simulators Tab (see page 143)
- Coordinator Servers Tab (see page 143)
- Virtual Environments Tab (see page 143)

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 in *Using CA Application Test.*
- 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 topics:

Open the Enterprise Dashboard

To open the Enterprise Dashboard from a web browser:

1. Ensure that the registry is running.
2. Navigate to the directory where you installed the Enterprise Dashboard.
3. Navigate to the **bin** directory, and run **EnterpriseDashboard.exe**, or select Enterprise Dashboard from the Start, Programs menu.
4. Enter **http://localhost:1506/** in a web browser.

   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 the following performance statistics about running registries:

Display Name

The Display Name (also called the "registry name" on the configure window) is a name that you assign when you configure the registry.

Name

The URL of the registry.

Status

Values can be *Running* or *Stopped*.

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.

DevTest Console URL

The address of the DevTest Console.

To display a Registry Summary window with more information, hover your mouse over the Display Name of a registry.

Display Name

The Display Name (also called the "registry name" on the configure window) is a name that you assign when you configure the registry.

URL

The URL of the registry.

DevTest Solutions Version

Version of DevTest running on the computer where the registry runs.

Java Version

Version of Java running on the computer where the registry runs.

Operating System

Version of the operating system running on the computer where the registry runs.

Security

Enabled or Disabled, depending on whether the registry uses ACL.

Status

Values can be *Running* or *Stopped*.

Uptime

The number of days, hours, minutes, and seconds since the registry was started.

Last Start Time

The date and time the registry was last started.

To display information about the database that the Enterprise Dashboard uses, click 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.

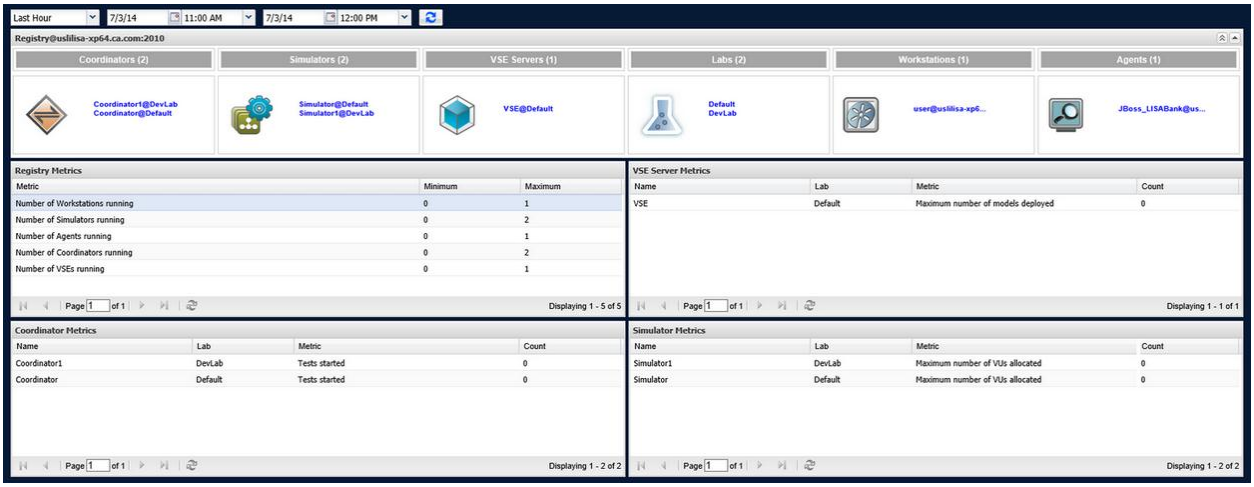
Note: All information is not available for releases earlier than 7.1. The following table shows what feature information displays for earlier releases.

Feature	Available in Registries Earlier than 7.1	Available for 7.1 Registries	New for 7.5 Registries
Registry display name		x	
URL	x		
Status	x		
Uptime		x	
Security	x		
DevTest version		x	
Java version		x	
Uptime		x	
Operating system		x	
DevTest Console URL		x	
Coordinator names	x		
Simulator names	x		
VSE server names	x		
Lab names	x		
Metrics			x
Historical Data			x

Enterprise Dashboard Registry Details Window

The registry details window in the Enterprise Dashboard lets you view details about registries.

Note: This window only displays data for registries running version 7.5 or later.



The top part of the window lists all active coordinators, simulators, VSE servers, labs, Workstations, and agents for the registry.

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 the Refresh icon 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 largest and smallest number of workstations, simulators, coordinators, and VSEs running in the designated time period.

The VSE Server 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 Go back to registry view  or click Overview in the breadcrumbs in the top left corner of the page.

Reactivate a Registry or Enterprise Dashboard

The DevTest Installation Setup Wizard for 8.0 and above configures new registries. When you start the Enterprise Dashboard and the registries, the registries are automatically activated. However, if there are subsequent changes to any of the following, reactivation is needed.

- Host name of the server where the registry is installed
- Registry name
- Registry port
- Enterprise Dashboard URL

Follow these steps:

1. If the host name or port of the Enterprise Dashboard changes:
   a. Log on to the computer with the Enterprise Dashboard.
   b. Navigate to LISA_HOME and open local.properties.
   c. Update the following line if the host name or port of the Enterprise Dashboard changes.

      ```
      lisa.enterprisedashboard.service.url=tcp://somehost:2003/EnterpriseDashboard
      ```

   d. Save the file.
   e. Restart the Enterprise Dashboard.
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.

Maintain Registries

Maintaining registries involves the following procedures:

- Add the registry of a DevTest Server and then validate the registry.

  Note: See Configure Existing Registries for information about adding a registry from a release earlier than DevTest 7.5. For versions beginning with 8.0, the installation process automatically configures new registries.

- Update an existing registry and then validate the registry.
- Delete a registry from an DevTest Server that is no longer in service.
- Periodically, validate each registry that is listed in the Enterprise Dashboard.

To add a registry (DevTest 7.5.x):

1. Navigate to the LISA_HOME directory of a newly installed DevTest Server.
2. Copy _local.properties and rename the copy to local.properties.
3. Open local.properties for edit. Locate Section 1 - Enterprise Dashboard.
4. Uncomment the following line and substitute localhost or the valid hostname for somehost.

   ```
   lisa.enterprisedashboard.service.url=tcp://localhost:2003/EnterpriseDashboard
   ```

5. Save local.properties and exit.

To delete a registry:

1. Open the Enterprise Dashboard (see page 144).
2. Select Configure from the Options menu.
3. Click the registry that you want to delete from the list of registries in the Registry column.
4. Click Delete.

   Note: If you delete a registry that runs on version 7.5 or later, you must also update the **lisa.enterprise.dashboard.url** property in the **local.properties** file for that registry.

   Note: Deleting a registry through the Registry Configuration page removes the registry from the dashboard. However, the historical data for that registry is not automatically purged from the database. For more information about purging historical data, see Purge Dashboard Data (see page 154).

To update a registry:

1. Open the Enterprise Dashboard (see page 144).
2. Select Configure from the Options menu.
3. Click the registry that you want to update from the list of registries in the Registry column.
4. Edit the Registry Name and Display Registry fields.
5. Click Save.

To validate a registry:

1. Open the Enterprise Dashboard (see page 144).
2. Select Configure from the Options menu.
3. Click the registry that you want to validate from the list of registries in the Registry column.
4. Click Validate.

   The dashboard server queries the selected registry and displays the status, for example, is running.
5. Click OK.

Export Dashboard Data

The Enterprise Dashboard allows you to export current and historical dashboard data in an Excel format. You can export data from either the main window (see page 145) or the Registry Details window (see page 147).

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 Registry Details window, skip to step 4.

   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. Click OK.

   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.

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:

- PF Power User
- SV Power User
- Test Power User
- Runtime User

The report can include the Admin user, which is displayed in the Roles grid in the Service Console as a combination of PF Power User and SV Power User. Licensing ignores this user type combination.

You can generate the Usage Audit Report on demand. The default is every three months.

Follow these steps:

1. Browse to the Enterprise Dashboard.

   `http://hostname:1506`

2. Click the Options drop-down list and select Export Usage Audit Data.

   The Export Usage Audit Data dialog opens.

3. To select a start date and 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 57) 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. You can also purge registries from the database that you deleted through the Registry Configuration page. For more information, see Configure Registries.

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. Select the **Purge registries marked for deletion** check box to permanently delete all registries from the database that you deleted through the Registry Configuration page.

   Note: Deleting a registry through the Registry Configuration page removes the registry from the dashboard. However, the historical data for that registry is not automatically purged from the database. Selecting this check box permanently removes deleted registries from the database and all event logs, metrics, and components that are associated with the registry.

4. Click OK.

   A confirmation dialog opens.

5. Click Yes.

   The selected data is purged from the Enterprise Dashboard.

Glossary

assertion

An *assertion* is an element that runs after a step and all its filters have run. An assertion verifies that the results from running the step match the expectations. An assertion is typically used to change the flow of a test case or virtual service model. Global assertions apply to each step in a test case or virtual service model. For more information, see Assertions in *Using CA Application Test.*

asset

An *asset* is a set of configuration properties that are grouped into a logical unit. For more information, see Assets in *Using CA Application Test*.

audit document

An *audit document* lets you set success criteria for a test, or for a set of tests in a suite. For more information, see Building Audit Documents in *Using CA Application Test.*

companion

A *companion* is an element that runs before and after every test case execution. Companions can be understood as filters that apply to the entire test case instead of to single test steps. Companions are used to configure global (to the test case) behavior in the test case. For more information, see Companions in *Using CA Application Test.*

configuration

A *configuration* is a named collection of properties that usually specify environment-specific values for the system under test. Removing hard-coded environment data enables you to run a test case or virtual service model in different environments simply by changing configurations. The default configuration in a project is named project.config. A project can have many configurations, but only one configuration is active at a time. For more information, see Configurations in *Using CA Application Test.*

Continuous Service Validation (CVS) Dashboard

The *Continuous Validation Service (CVS) Dashboard* lets you schedule test cases and test suites to run regularly, over an extended time period. For more information, see Continuous Validation Service (CVS) in *Using CA Application Test.*

conversation tree

A *conversation tree* is a set of linked nodes that represent conversation paths for the stateful transactions in a virtual service image. Each node is labeled with an operation name, such as withdrawMoney. An example of a conversation path for a banking system is getNewToken, getAccount, withdrawMoney, deleteToken. For more information, see *Using CA Service Virtualization.*

coordinator

A *coordinator* receives the test run information as documents, and coordinates the tests that are run on one or more simulator servers. For more information, see Coordinator Server in *Using CA Application Test.*

data protocol

A *data protocol* is also known as a data handler. In CA Service Virtualization, it is responsible for handling the parsing of requests. Some transport protocols allow (or require) a data protocol to which the job of creating requests is delegated. As a result, the protocol has to know the request payload. For more information, see Using Data Protocols in *Using CA Service Virtualization.*

data set

A *data set* is a collection of values that can be used to set properties in a test case or virtual service model at run time. Data sets provide a mechanism to introduce external test data into a test case or virtual service model. Data sets can be created internal to DevTest, or externally (for example, in a file or a database table). For more information, see Data Sets in *Using CA Application Test.*

desensitize

Desensitizing is used to convert sensitive data to user-defined substitutes. Credit card numbers and Social Security numbers are examples of sensitive data. For more information, see Desensitizing Data in *Using CA Service Virtualization.*

event

An *event* is a message about an action that has occurred. You can configure events at the test case or virtual service model level. For more information, see Understanding Events in *Using CA Application Test.*

filter

A *filter* is an element that runs before and after a step. A filter gives you the opportunity to process the data in the result, or store values in properties. Global filters apply to each step in a test case or virtual service model. For more information, see Filters in *Using CA Application Test.*

group

A *group*, or a *virtual service group,* is a collection of virtual services that have been tagged with the same group tag so they can be monitored together in the VSE Console.

Interactive Test Run (ITR)

The *Interactive Test Run (ITR)* utility lets you run a test case or virtual service model step by step. You can change the test case or virtual service model at run time and rerun to verify the results. For more information, see Using the Interactive Test Run (ITR) Utility in *Using CA Application Test.*

lab

A *lab* is a logical container for one or more lab members. For more information, see Labs and Lab Members in *Using CA Application Test.*

magic date

During a recording, a date parser scans requests and responses. A value matching a wide definition of date formats is translated to a *magic date*. Magic dates are used to verify that the virtual service model provides meaningful date values in responses. An example of a magic date is {{=doDateDeltaFromCurrent("yyyy-MM-dd","10");/*2012-08-14*/}. For more information, see Magic Strings and Dates in *Using CA Service Virtualization.*

magic string

A *magic string* is a string that is generated during the creation of a service image. A magic string is used to verify that the virtual service model provides meaningful string values in the responses. An example of a magic string is {{=request_fname;/chris/}}. For more information, see Magic Strings and Dates in *Using CA Service Virtualization.*

match tolerance

Match tolerance is a setting that controls how CA Service Virtualization compares an incoming request with the requests in a service image. The options are EXACT, SIGNATURE, and OPERATION. For more information, see Match Tolerance in *Using CA Service Virtualization.*

metrics

Metrics let you apply quantitative methods and measurements to the performance and functional aspects of your tests, and the system under test. For more information, see Generating Metrics in *Using CA Application Test.*

Model Archive (MAR)

A *Model Archive (MAR)* is the main deployment artifact in DevTest Solutions. MAR files contain a primary asset, all secondary files that are required to run the primary asset, an info file, and an audit file. For more information, see Working with Model Archives (MARs) in *Using CA Application Test.*

Model Archive (MAR) Info

A *Model Archive (MAR) Info* file is a file that contains information that is required to create a MAR. For more information, see Working with Model Archives (MARs) in *Using CA Application Test.*

navigation tolerance

Navigation tolerance is a setting that controls how CA Service Virtualization searches a conversation tree for the next transaction. The options are CLOSE, WIDE, and LOOSE. For more information, see Navigation Tolerance in *Using CA Service Virtualization.*

network graph

The network graph is an area of the Server Console that displays a graphical representation of the DevTest Cloud Manager and the associated labs. For more information, see Start a Lab in *Using CA Application Test*.

node

Internal to DevTest, a test step can also be referred to as a *node*, explaining why some events have node in the EventID.

path

A *path* contains information about a transaction that the Java Agent captured. For more information, see *Using CA Continuous Application Insight.*

path graph

A *path graph* contains a graphical representation of a path and its frames. For more information, see Path Graph in *Using CA Continuous Application Insight.*

project

A *project* is a collection of related DevTest files. The files can include test cases, suites, virtual service models, service images, configurations, audit documents, staging documents, data sets, monitors, and MAR info files. For more information, see Project Panel in *Using CA Application Test.*

property

A *property* is a key/value pair that can be used as a run-time variable. Properties can store many different types of data. Some common properties include LISA_HOME, LISA_PROJ_ROOT, and LISA_PROJ_NAME. A configuration is a named collection of properties. For more information, see Properties in *Using CA Application Test.*

quick test

The *quick test* feature lets you run a test case with minimal setup. For more information, see Stage a Quick Test in *Using CA Application Test.*

registry

The *registry* provides a central location for the registration of all DevTest Server and DevTest Workstation components. For more information, see Registry in *Using CA Application Test.*

service image (SI)

A *service image* is a normalized version of transactions that have been recorded in CA Service Virtualization. Each transaction can be stateful (conversational) or stateless. One way to create a service image is by using the Virtual Service Image Recorder. Service images are stored in a project. A service image is also referred to as a *virtual service image* (VSI). For more information, see Service Images in *Using CA Service Virtualization.*

simulator

A *simulator* runs the tests under the supervision of the coordinator server. For more information, see Simulator Server in *Using CA Application Test.*

staging document

A *staging document* contains information about how to run a test case. For more information, see Building Staging Documents in *Using CA Application Test.*

subprocess

A *subprocess* is a test case that another test case calls. For more information, see Building Subprocesses in *Using CA Application Test.*

test case

A *test case* is a specification of how to test a business component in the system under test. Each test case contains one or more test steps. For more information, see Building Test Cases in *Using CA Application Test.*

test step

A *test step* is an element in the test case workflow that represents a single test action to be performed. Examples of test steps include Web Services, JavaBeans, JDBC, and JMS Messaging. A test step can have DevTest elements, such as filters, assertions, and data sets, attached to it. For more information, see Building Test Steps in *Using CA Application Test.*

test suite

A *test suite* is a group of test cases, other test suites, or both that are scheduled to execute one after other. A suite document specifies the contents of the suite, the reports to generate, and the metrics to collect. For more information, see Building Test Suites in *Using CA Application Test.*

think time

Think time is how long a test case waits before executing a test step. For more information, see Add a Test Step (example) and Staging Document Editor - Base Tab in *Using CA Application Test.*

transaction frame

A *transaction frame* encapsulates data about a method call that the DevTest Java Agent or a CAI Agent Light intercepted. For more information, see Business Transactions and Transaction Frames in *Using CA Continuous Application Insight*.

Virtual Service Environment (VSE)

The *Virtual Service Environment (VSE)* is a DevTest Server application that you use to deploy and run virtual service models. VSE is also known as CA Service Virtualization. For more information, see *Using CA Service Virtualization.*

virtual service model (VSM)

A *virtual service model* receives service requests and responds to them in the absence of the actual service provider. For more information, see Virtual Service Model (VSM) in *Using CA Service Virtualization.*