DevTest Solutions

Installing

Version 8.0

This Documentation, which includes embedded help systems and electronically distributed materials, (hereinafter referred to as the “Documentation”) is for your informational purposes only and is subject to change or withdrawal by CA at any time.

This Documentation may not be copied, transferred, reproduced, disclosed, modified or duplicated, in whole or in part, without the prior written consent of CA. This Documentation is confidential and proprietary information of CA and may not be disclosed by you or used for any purpose other than as may be permitted in (i) a separate agreement between you and CA governing your use of the CA software to which the Documentation relates; or (ii) a separate confidentiality agreement between you and CA.

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

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

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

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

The manufacturer of this Documentation is CA.

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

Copyright © 2014 CA. All rights reserved. All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

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: Preinstallation

This section contains the following topics:

System Requirements

This section lists the requirements for the various DevTest components. These general requirements could change, depending on the scope of your project.

Operating System Requirements

The following operating systems are supported:

- Microsoft Windows:
  - Windows Server 2012
  - Windows 7
  - Windows 8 (with latest service pack and all critical updates applied)
- Linux and UNIX
  - Fedora 19
  - Red Hat Enterprise Linux 6.3
  - SUSE Linux 10 SP2, 11.x
  - Ubuntu 11.04, 12.04, 13.x
  - Oracle Solaris 10 and 11
  - IBM AIX 6.1, 7.0
- Mac OS X 10.9 and 10.10

We recommend a 64-bit operating system and JRE, especially for DevTest Server.

DevTest Server JVM System Requirements

We recommend a 64-bit Java 7 for DevTest Server when requiring heap sizes above 2 GB.

Java JDK

Products composing the DevTest Solutions are Java applications. An Oracle JRE is included with each operating system-specific installer (1.7.0_17 JRE including a **tools.jar** from the JDK) other than the generic UNIX installer. IBM JRE Version 7, Release 1 is also supported if you supply your own JVM (see page 12).

The minimum supported Java version for DevTest Workstation, DevTest Server, and VSE is Java 1.7 update 6.

This requirement is a DevTest-side requirement only. DevTest running in a Java 1.7 virtual machine (VM) can be used to test applications on application servers running older or newer JREs.

The following table lists the support that is provided for various JDKs:

	DevTest Workstation	DevTest Server - Coordinator, Simulator, Registry	DevTest Server - VSE	DevTest Java Agent (CAI)
JDK 1.5	Not supported	Not supported	Not supported	Supported
JDK 1.6	Not supported	Not supported	Not supported	Supported
JDK 1.7	Required for run time	Required for run time	Required for runtime	Supported

Note: DevTest does not support IBM JRE Version 8, Oracle JRE 1.8, or OpenJDK.

Kerberos authentication is not supported on IBM JREs.

DevTest Workstation cannot test web services and messaging backbones on different versions of Java.

The following information provides general guidance on JDK support in DevTest Java Agent (CAI):

- CAI supports systems that are based on JDK 1.5, JDK 1.6, and JDK 1.7.
- Only JDKs from Oracle are fully supported. IBM JDKs have limited support only.

DevTest Locks Folder

The following folder in the LISA_HOME directory requires read/write permissions:

- locks

In general, the rest of the LISA_HOME directory can be restricted with read-only permissions:

The lisatmp Folder

The following folder in the user home directory (on UNIX, Linux, and OS X) or Documents and Settings (on Windows) requires read/write permissions:

- lisatmp_x.x (if it exists)

DevTest CICS Agent

The DevTest CICS Agent supports the following CICS versions: 3.2, 4.1, and 4.2.

Supplying Your Own JVM

The generic UNIX installer does not include a JRE. If you are using the generic UNIX installer, you *must* supply your own JVM.

Optionally, you can use this procedure with an installer for another platform to override the included JRE. See Using DevTest Workstation with Your Java Environment (see page 67).

Note: DevTest supports IBM JRE Version 7, Release 1. DevTest does not support IBM JRE Version 8, Oracle JRE 1.8, or OpenJDK.

Follow these steps:

1. Download the Java SE Development Kit (JDK) 7 package for your platform from the Oracle website.
2. Download the Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files 7 from the Oracle website.
3. Install JDK 7 for your operating system on the computer where you plan to install DevTest. If there is no **java** directory, create one (mkdir java).

   For example, install JDK 7 in \usr\java, which creates the \usr\java\jdk1.7.0_67 directory (JDK_HOME).
4. After you copy the jdk-7u67-*platform*. tar.gz file, enter this command:
   `tar zxvf jdk-7u67-platform-x64.tar.gz`
5. Set environment variables:
   - Set the **JDK_HOME** environment variable to the directory where you installed JDK 7.
   - Set the **JAVA_HOME** environment variable to point to the **JDK_HOME** directory.

   For example:

   ```
   cd \usr\java\jdk1.7.0_67
   pwd
   export JAVA_HOME=$PWD
   export JDK_HOME=$PWD
   ```
6. Extract the UnlimitedJCEPolicy folder from the UnlimitedJCEPolicyJDK7.zip file. Move the following JAR files from this folder to the **JDK_HOME\jre\lib\security** directory:
   - local_policy.jar
   - US_export_policy.jar

   This action replaces the existing JAR files with the same names.

7. Copy the **tools.jar** file from the **JDK_HOME\lib** directory to the **JDK_HOME\jre\lib\ext** directory.

   ```
   cd $JDK_HOME\jre\lib\ext
   cp $JDK_HOME\lib\tools.jar .
   ```

8. If you are overriding the included JRE, perform the following steps after you install DevTest:

   a. Set the **LISA_JAVA_HOME** environment variable. For example:

      ```
      cd \usr\java\jdk1.7.0_67
      pwd
      export LISA_JAVA_HOME=$PWD
      ```

   b. Rename the **jre** directory under the DevTest installation directory to jre_default as described in Using DevTest Workstation with Your Java Environment (see page 67).

DevTest Server System Requirements

CA Application Test, CA Service Virtualization, and CA Continuous Application Insight require the registry in DevTest Server.

The minimum requirements for DevTest Server are:

- **CPU:** 2 GHz or faster, 4 cores minimum
- **RAM:** 4 GB
- **Disk Space:** 50 GB
- **Database:** See Database System Requirements (see page 17). The database can reside on a different system and must have at least 200 GB of storage.

The recommended requirements for DevTest Server are:

- **CPU:** 2 GHz or faster, 8 cores minimum
- **RAM:** 8 GB
- **Disk Space:** 50 GB
- **Database:** See Database System Requirements (see page 17). The database can reside on a different system and must have at least 500 GB of storage.

For load and performance testing, the following resources are recommended:

- 250 virtual users for each simulator
- 1 processor core and 2-GB RAM for each simulator

Example for 4000 concurrent virtual users: 16 simulators; 16 processor core; 32-GB RAM (for DevTest)

For each data center:

- 1 test registry and 1 coordinator
- 1 processor core/process = 2 processor cores
- 2 GB RAM/each = 4 GB (for DevTest)

A basic DevTest Server configuration has one enterprise dashboard, one registry, and one portal, which are required for all products. CA Application Test requires one coordinator server and one simulator server. CA Service Virtualization requires one virtual server environment. CA Continuous Application Insight requires one broker. Only one enterprise dashboard server is required for any given configuration.

Note: The requirements that are listed here are intended as a guideline. For large load environments, we recommend contacting Professional Services to assist with developing a load generation environment to suit your needs.

DevTest Workstation for CA Application Test System Requirements

The minimum requirements for DevTest Workstation are:

- **CPU:** 2 GHz or faster, 2 cores minimum
- **RAM:** 4 GB
- **Disk Space:** GB free space

CA Service Virtualization System Requirements

The CA Service Virtualization component, VSE, is required for maintaining a virtualization environment. VSE is a server-level service and can coexist with a registry that has a coordinator and a simulator that is attached to it. The simulator and coordinator are not mandatory to run VSE.

The following requirements are baseline requirements only:

- **CPU:** 2 GHz or faster, 2 cores minimum
- **RAM:** 2 GB for VSE, in addition to the RAM requirement for DevTest Workstation, DevTest Server, or CAI
- **Disk Space:** 50 GB of free space
- **Database:** See Database System Requirements (see page 17). The database can reside on a different system and must have at least 10 GB of storage.

CAI System Requirements

The minimum requirements for CA Continuous Application Insight are as follows:

- **CPU:** 2 GHz or faster, 8 cores minimum
- **RAM:** 8 GB
- **Disk Space:** 50 GB of local disk storage
- **Database:** See Database System Requirements (see page 17). The database can reside on a different system and must have at least 500 GB of storage.

Communication Requirements

Make sure that your firewall allows DevTest Solutions to send and receive network transmissions. The functionality that DevTest Solutions provides requires access to network resources and will not work properly if blocked by a firewall. Authorize DevTest Solutions applications.

Note: To implement secure communication, see Using SSL to Secure Communication and Using HTTP/S Communication with the DevTest Console in *Administering*.

Communications to and from the DevTest ports must be opened in any relevant firewall. See the following topics in *Administering*.

- DevTest Server Default Port Numbers
- DevTest Workstation Default Port Numbers
- Demo Server Default Port Numbers

Database System Requirements

The following components store information in a database:

- **DevTest Server:** The database is used for report results, which can be exported to other formats as needed. The database is also used for access control (ACL).
- **VSE:** The database is used for usage counts and legacy virtual service images.
- **CAI:** The database is used for paths, including request and response data, SQL statements, and application logs. The database is also used for tickets.
- **Enterprise Dashboard:** The database is used for the DevTest Solutions Usage Audit Report data, other registry information, historical event logs, and metrics.

  Important! Enterprise Dashboard requires its own unique large database. The database can reside on a different system and must have at least 50 GB of storage. IBM DB2 is not supported as the database for Enterprise Dashboard.

By default, these components use an Apache Derby database that is included with DevTest. This database is adequate only for small deployments that do not require load and performance testing, and is not supported. For all other scenarios, configure DevTest to use an external database.

To ensure correct performance when using an external database, the database server and DevTest server should have high network bandwidth and low latency.

DevTest, when run in a distributed configuration, strongly depends upon any server components having a high-bandwidth, low latency connection to a well-maintained enterprise class database.

All DevTest server components communicate directly with the database to record their actions, and any restriction to the flow of this data has adverse effects.

In order to ensure that your DevTest functions correctly, no DevTest server components should have a Round Trip Time (RTT) of greater than 20 ms to the database host. If the network latency exceeds this 20 ms value, you can expect performance problems.

Important! Use a clean database schema for any new installation. Data from the same DevTest version can be restored into the clean schema before you install. Do not use data from other versions.

The following external databases are supported:

- **IBM DB2 10.1 or 10.5** (not supported for Enterprise Dashboard) - The code page of the database must be 1208. In addition, the page size must be at least 8 KB.
- **MySQL 5.5 or 5.6 -** 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 must be UTF-8; it is not enough only to define your database as UTF-8.
- **Oracle 11g Release 2 or 12c** - The character set must be Unicode set.

- **Microsoft SQL Server 2008 R2 or 2012**

The schema is automatically created in the external database when the registry starts for the first time. Before the schema is created, ensure that the DevTest user has DBA privileges. After the schema is created, you can remove the DBA privileges from the user.

If your security policy does not permit this approach, the database administrator can manually create the schema. The DDL files in the **LISA_HOME\database** directory contain SQL statements for creating the reporting tables and indexes, and for creating the agent database schema that CA Continuous Application Insight uses. Provide this information to your database administrator.

Note: For more information about the configuration of an external database, see *Administering*. For information about obtaining the SQL statements for the agent database schema, see *Agents.*

Important! Registries and Enterprise Dashboards must each have a unique schema. Do not point multiple registries at the same database schema.

For load and performance testing, tune the external database to ensure that it can support the amount of data storage that DevTest requires.

The registry, coordinator, simulators, and any virtual service environments require high-performance database access. Performance data is recorded directly to the database by these components. CA recommends that the database be present within the same data center. Databases hosted within a virtual machine are not recommended for general availability use.

Supported Browsers

DevTest includes the following web-based portals, consoles, and dashboards:

- Enterprise Dashboard (http://*hostname*:1506/)
- DevTest Portal (http://*hostname*:1507/devtest)
- DevTest Console (http://*hostname*:1505/)
  - Reporting Portal
  - Continuous Validation Service
  - Server Console
  - VSEasy
  - CAI Console (redirects to DevTest Portal)
- Demo Server (http://*hostname*:8080/lisabank/)

The DevTest Portal requires HTML 5. Therefore, the latest internet browser technology is required to use the new web-based UI. The following internet browser versions support HTML 5:

- Google Chrome 36
- Mozilla Firefox 30
- Apple Safari 7.0
- Microsoft Internet Explorer 11

Note: Browser support for recording Selenium Integration tests is limited to Mozilla Firefox. After you import these tests to DevTest, running the test cases has been verified on Google Chrome, Mozilla Firefox 24, or Internet Explorer 10 and 11.

Note: Internet Explorer browser support for VSEasy is limited to version 10. All other listed browsers are supported.

Planning Your DevTest System

In order to successfully install DevTest Solutions, consider the following decision points prior to the installation:

- What supported operating system do you plan to use?
- Are you performing a shared or local installation?
- Is the installation targeted for workstation or server use?
- Which enterprise database do you plan to use?

  If the installation is a permanent installation or you intend to use it in an environment that is generally available, the supplied Derby database is not supported.
- Do you plan to install the Demo Server?

  The demo server is supplied separately from the product installer.

Before beginning an installation, the license file (devtestlic.xml) must be available and present on your target system.

Additional considerations regarding the ideal layout of the various services and processes within the target systems:

- The Enterprise Dashboard must be reachable by the registry, but it does not need to be local to the registry. The Enterprise Dashboard requires its own database instance.
- The registry and the coordinator should reside on the same machine, wherever possible.
- The simulators should be on the same network segment as the coordinator, if practicable.
- The registry, coordinator, simulators, and any virtual service environments require high-performance database access. Performance data is recorded directly to the database by these components. CA recommends that the database be present within the same data center. Databases hosted within a virtual machine are not recommended for general availability use.

For more information about database requirements, see Database System Requirements (see page 17).

Before you begin planning your DevTest system, review the Release Notes for the release you install. If you are migrating a system, review the *Migration Guide*.

Part of planning a DevTest system is deciding how many DevTest Servers to install and what processes or services to run on each host computer containing a DevTest Server.

When you download the DevTest Solutions installers, you can select one of three products, all of which point to the same installers. DevTest Components (see page 22) depicts graphically these three products with product-specific UIs and processes and the common UI and processes shared by all products. About DevTest Server Component (see page 23)s describes the purpose of each process and where you can find more details.

Example Relationships in a Distributed System (see page 24) introduces major concepts concerning what processes you can run on each host in a distributed system. The Post-Installation (see page 58) section addresses topics that can help you tailor your distributed system; specifically:

- Running components on different systems (see page 64)
- Calculate simulator instances (see page 65)
- Load and performance server sizing (see page 66)

Another aspect of planning is designing naming conventions for your project directory structure (see page 68).

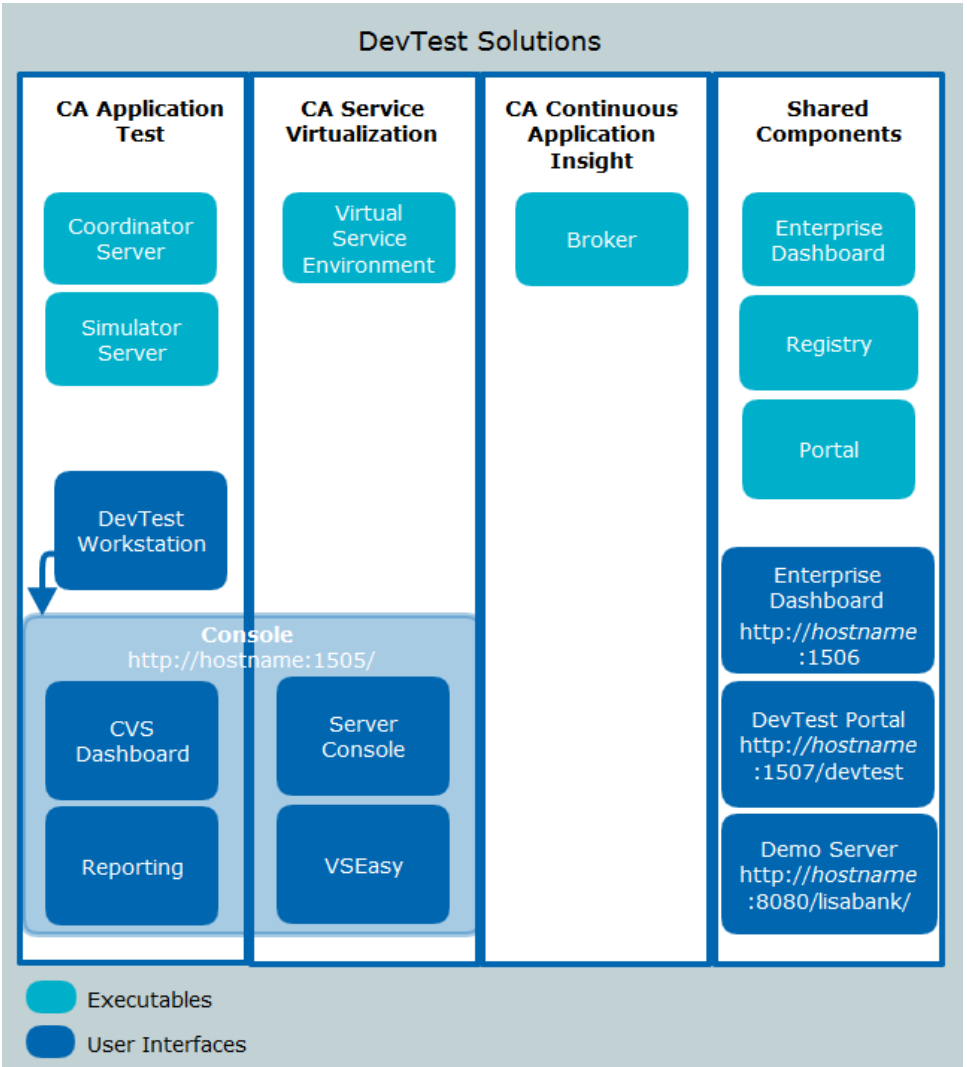
You can use the JRE that comes with DevTest or you can install your own. For more information, see Using DevTest Workstation with your Java environment (see page 67).

DevTest Components

The DevTest Solutions installer installs the following products when you install a DevTest Server (named *Server* in the Setup wizard):

- CA Application Test
- CA Service Virtualization
- CA Continuous Application Insight

These three products each have their own executables (and corresponding services). They are not independent products, however, because they all require the same registry executable. Previously, each product had its own UI. Now, the DevTest Portal is becoming the shared user interface for all products. (Some functionality can be accessed only from the DevTest Workstation or the DevTest Console.)



About DevTest Server Components (see page 23) describes each executable component with where to find more information.

Start the Server Components (see page 84) specifies the order in which to start the executables.

Log in to the User Interfaces (see page 91) includes procedures for accessing each UI.

For the details on setting up and maintaining DevTest Solutions, see *Administering*.

For product-specific details, see:

- *Using CA Application Test*
- *Using CA Service Virtualization*
- *Using CA Continuous Application Insight*

About DevTest Server Components

You start DevTest Server by starting a series of processes or services. The three components you start first (Enterprise Dashboard, registry, and portal) support the DevTest Solutions. The other components are product-specific. For startup details, see Start the Server Components (see page 84).

Name	Required By	Required To
Enterprise Dashboard	Registries (All DevTest Solutions products)	Start DevTest Solutions with the product license. Monitor enterprise activity. Generate the Usage Audit Report.
Registry	All DevTest Solutions products	Register DevTest Server and DevTest Workstation components. The registry is the central hub or engine for all processes. Collect usage data for Usage Audit Report.
Portal	All DevTest Solutions products	Start the DevTest Portal UI.
Broker	CA Continuous Application Insight (CAI)	Coordinate Java agents and CA Continuous Application Insight consoles.
Coordinator	CA Application Test	Coordinate tests run on simulators.
Simulator	CA Application Test	Run tests.

Name	Required By	Required To
Virtual Service Environment (VSE)	CA Service Virtualization	Create and deploy virtual services.

The referenced topics provide more information about each process supporting DevTest Server.

Process	Reference
Enterprise Dashboard	Enterprise Dashboard Main Window in *Administering*
Registry	Registry in *Using CA Application Test*
Portal	Using the DevTest Portal in *Getting Started*
Coordinator	Coordinator Server in *Using CA Application Test*
Simulator	Simulator Server in *Using CA Application Test*
Workstation	DevTest Workstation in *Using CA Application Test*
Virtual Service Environment	Preparing for Virtualization in *Using CA Service Virtualization*
Broker	Start the Broker in *Agents*

DevTest Process Relationships

The registry is at the center of all DevTest systems. Typically, the installer is run on multiple computers with different selections. Consider the following example:

- Computer 1: Enterprise Dashboard and server components, where the embedded workstation may be unused.
- Computer 2: Server where the registry component links to Enterprise Dashboard on Computer 1. Typically, only one registry is needed for a distributed system. This second registry is added here simply to demonstrate that multiple registries are supported.
- Computer 3: Workstation and Demo Server with links to the registry on Computer 1. This computer represents user machines.
- Computer 4, which can also represent user machines, particularly CA Continuous Application Insight users who need only portal access.
- Also, you can install the DevTest server on a computer to run only one product-specific service, such as the simulator executable or service.

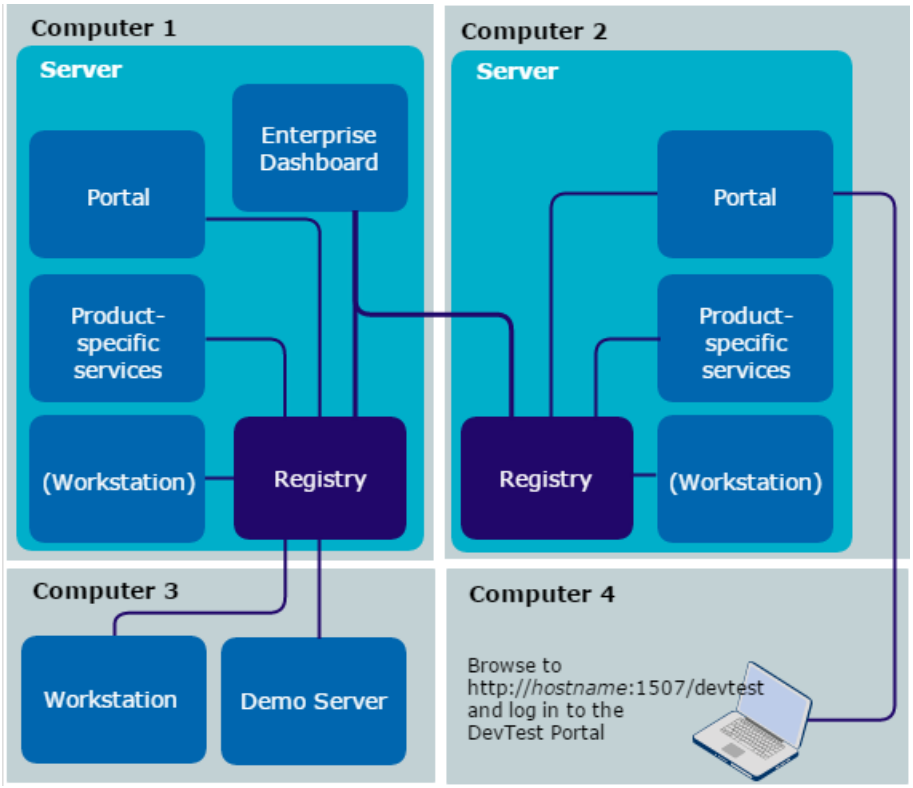
This example system shares the following characteristics with any DevTest system:

- Each DevTest Solutions system has only one Enterprise Dashboard.

  Note: You must have one Enterprise Dashboard per accessible network. If you have closed networks (networks that cannot reach each other), you need an Enterprise Dashboard for each network where registries are running.

- Each Enterprise Dashboard is connected to one or more registries. Typically, one registry is sufficient.
- Each DevTest Server installation installs one registry, the component that audits all user activity.
- Each DevTest Server installation installs one local workstation. This embedded workstation is used in a standalone installation. When the DevTest Server is installed in a distributed environment, the local workstation may be unused.
- Each DevTest Server registry in a distributed system can connect to one or more remote Workstations.
- A web browser can access the DevTest Portal by browsing to a DevTest Server on port 1507 where the URL ends with devtest (all lower case). The portal can be accessed from any web browser without needing other DevTest Solutions software to be installed on the computer.

The following diagram highlights the relationships among the Enterprise Dashboard, the registries, the workstation, and the Demo Server in a distributed system. It also shows that users with no local DevTest component can browse to the DevTest Portal.



DevTest Solutions Architecture

More information:

DevTest Server Components (see page 33)
Data Flow in DevTest Server for CA Application Test and CA Service Virtualization (see page 35)

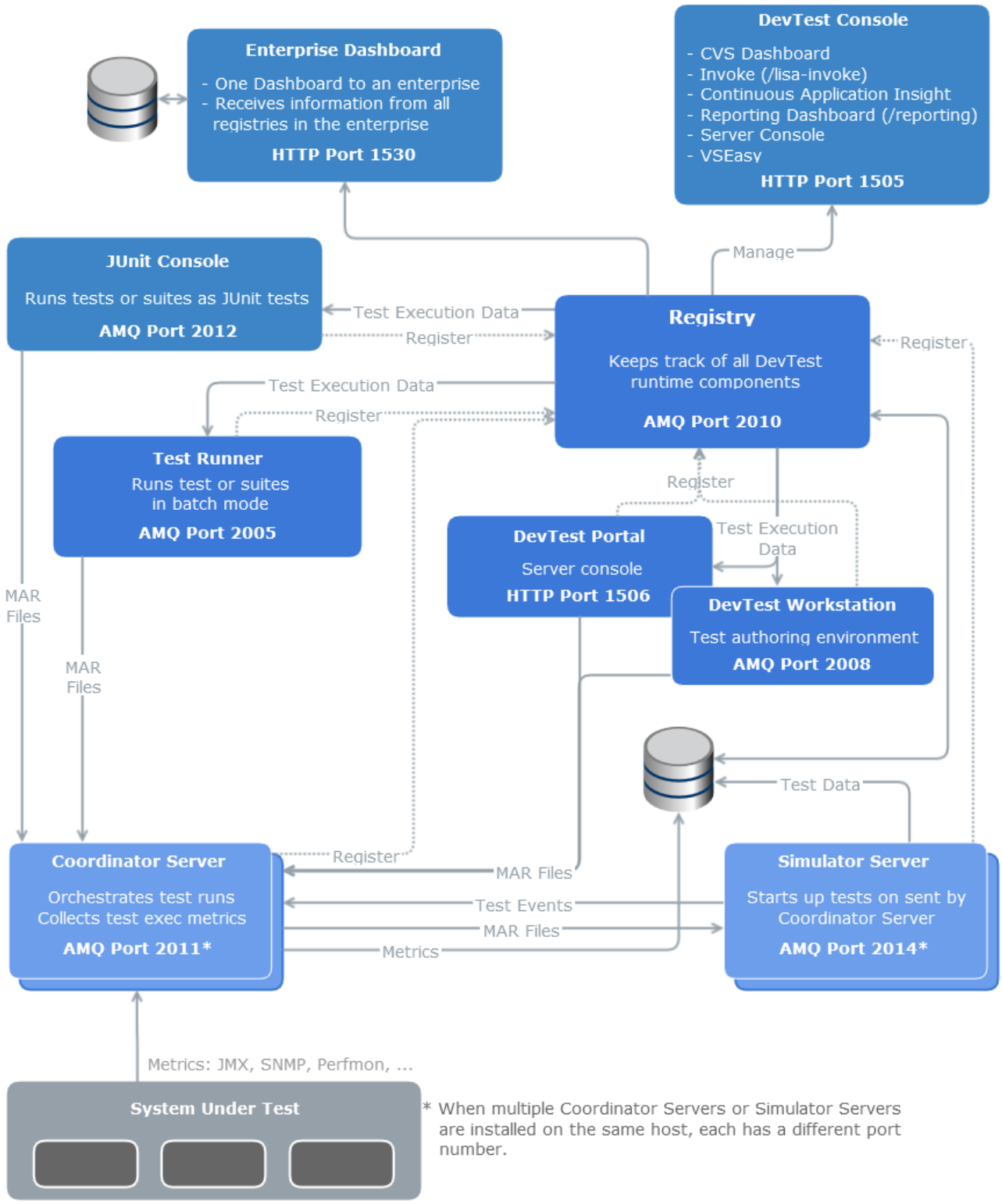
CA Application Test Architecture

CA Application Test uses the following DevTest Server components:

- Registry
- DevTest Workstation
- DevTest Portal
- Coordinator server
- Simulator server

The DevTest Portal and DevTest Workstation are used to create and monitor the tests, but the test cases are run in the DevTest Server environment. A coordinator server and a simulator server are embedded in DevTest Workstation.

The following diagram shows the CA Application Test architecture.


Enterprise Dashboard
- One Dashboard to an enterprise
- Receives information from all registries in the enterprise
HTTP Port 1530
DevTest Console
- CVS Dashboard
- Invoke (/lisa-invoke)
- Continuous Application Insight
- Reporting Dashboard (/reporting)
- Server Console
- VSEasy
HTTP Port 1505
Manage
JUnit Console
Runs tests or suites as JUnit tests
AMQ Port 2012
Test Execution Data
Register
Registry
Keeps track of all DevTest runtime components
AMQ Port 2010
Register
Test Execution Data
Register
Test Runner
Runs test or suites in batch mode
AMQ Port 2005
Register
DevTest Portal
Server console
HTTP Port 1506
Test Execution Data
DevTest Workstation
Test authoring environment
AMQ Port 2008
MAR Files
MAR Files
Test Data
Coordinator Server
Orchestrates test runs
Collects test exec metrics
AMQ Port 2011*
Register
MAR Files
Test Events
MAR Files
Metrics
Simulator Server
Starts up tests on sent by Coordinator Server
AMQ Port 2014*
Metrics: JMX, SNMP, Perfmon, ...
System Under Test
* When multiple Coordinator Servers or Simulator Servers are installed on the same host, each has a different port number.

All tests are run by the virtual users (or simulators) spawned by the simulator server under the supervision of a coordinator in the coordinator server.

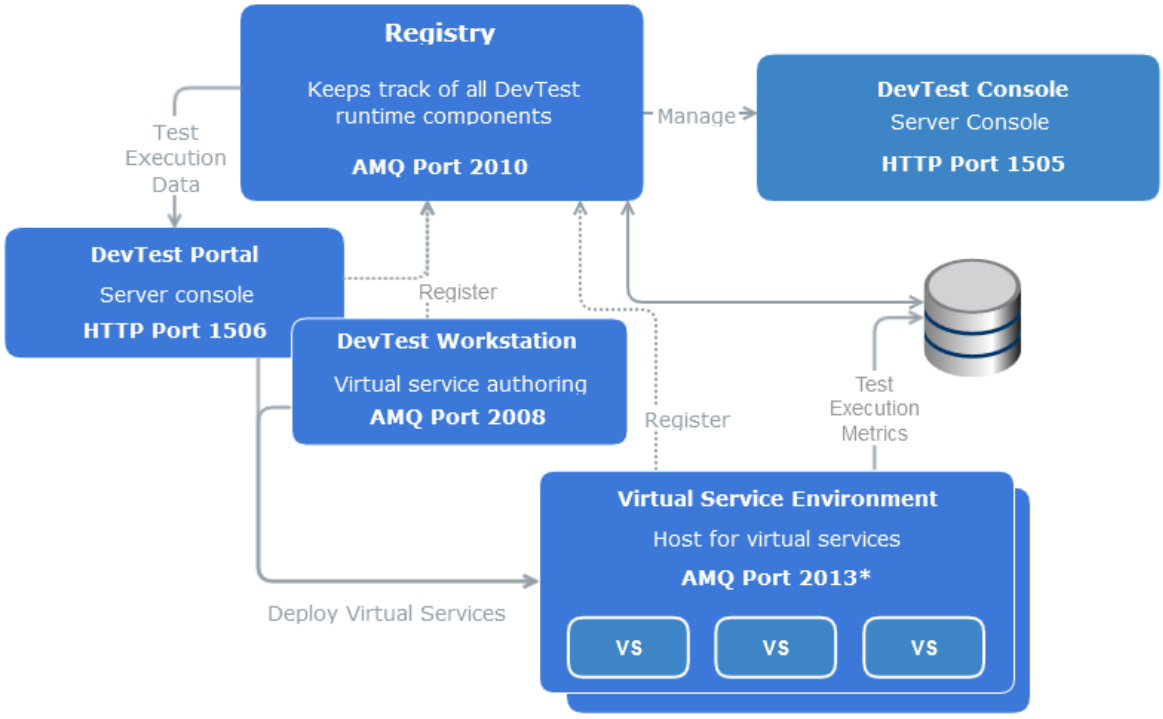
Each simulator connects to and invokes actions on the system under test. A load test results when the virtual users are running in a parallel mode.

An embedded instance of the DevTest Workstation runs on the same computer that the DevTest Server is running on. You can also configure standalone DevTest Workstations to run on separate computers in a large distributed environment. Your testing requirements dictate the server architecture to be used. DevTest Solutions can be scaled for large testing environments by distributing the different components onto different hardware/operating systems.

DevTest products include CA Application Test, CA Service Virtualization, and CA Continuous Application Insight, each of which connects to a central DevTest registry. CA Application Test users use an embedded DevTest Workstation with its optional coordinator server and simulator servers and optionally and all remotely installed DevTest Workstations. Components of all DevTest products connect to a central registry.

CA Service Virtualization Architecture

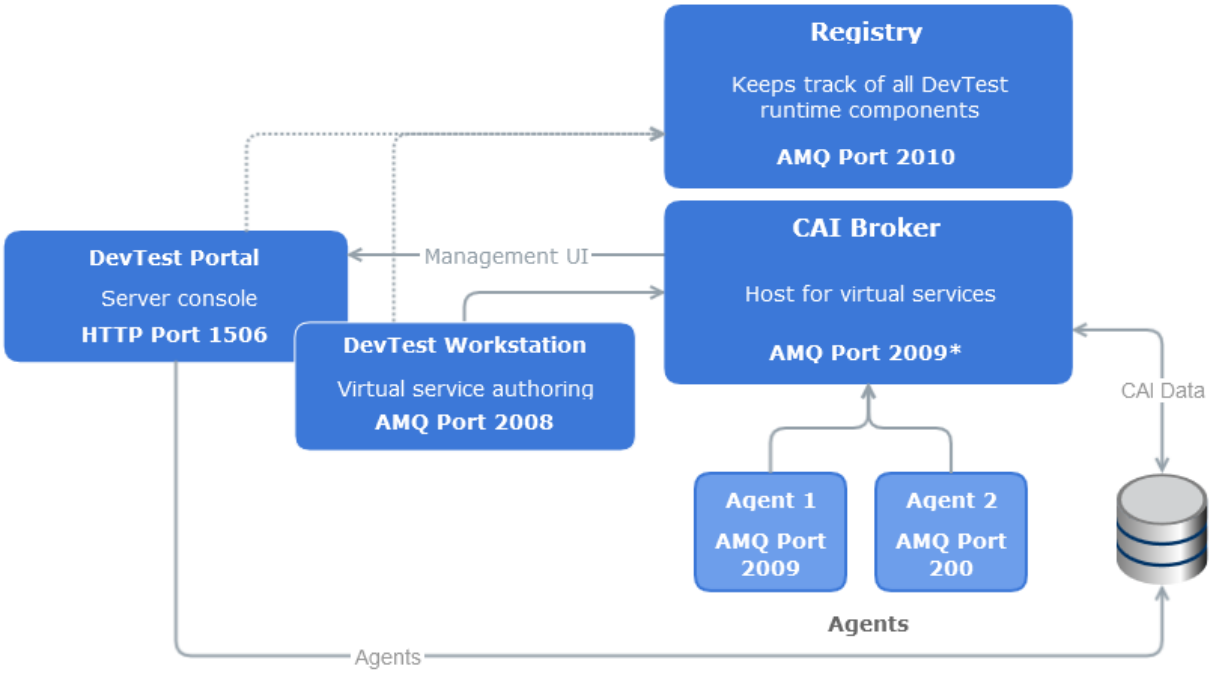
The following diagram shows the CA Service Virtualization architecture.



*Multiple Coordinators, Simulators, and VSEs on the same machine will have different port numbers.

CAI Architecture

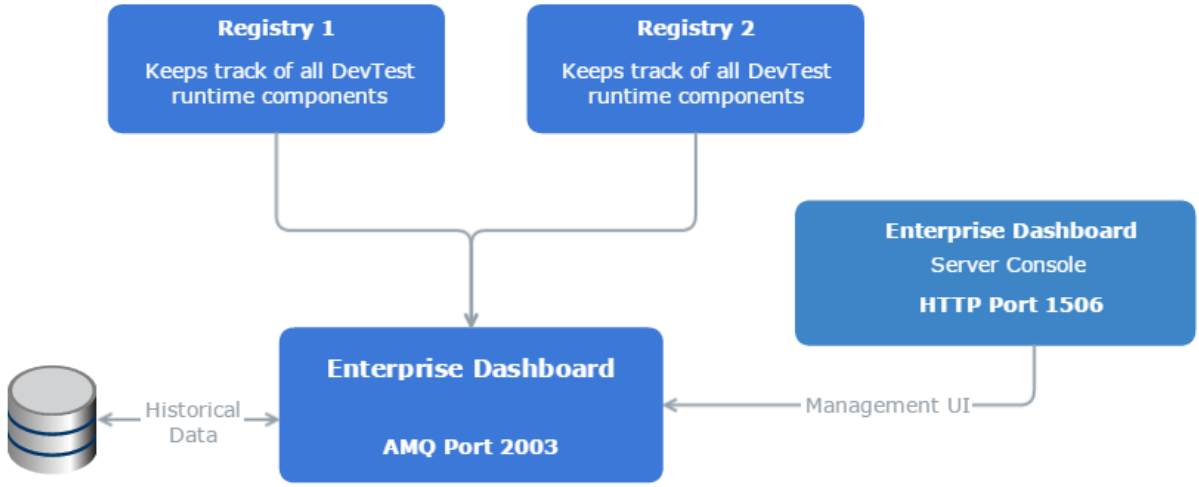
The following diagram shows the CAI architecture.



*Multiple VSEs on the same machine will have different port numbers.

Enterprise Dashboard Architecture

The following diagram shows the Enterprise Dashboard architecture.

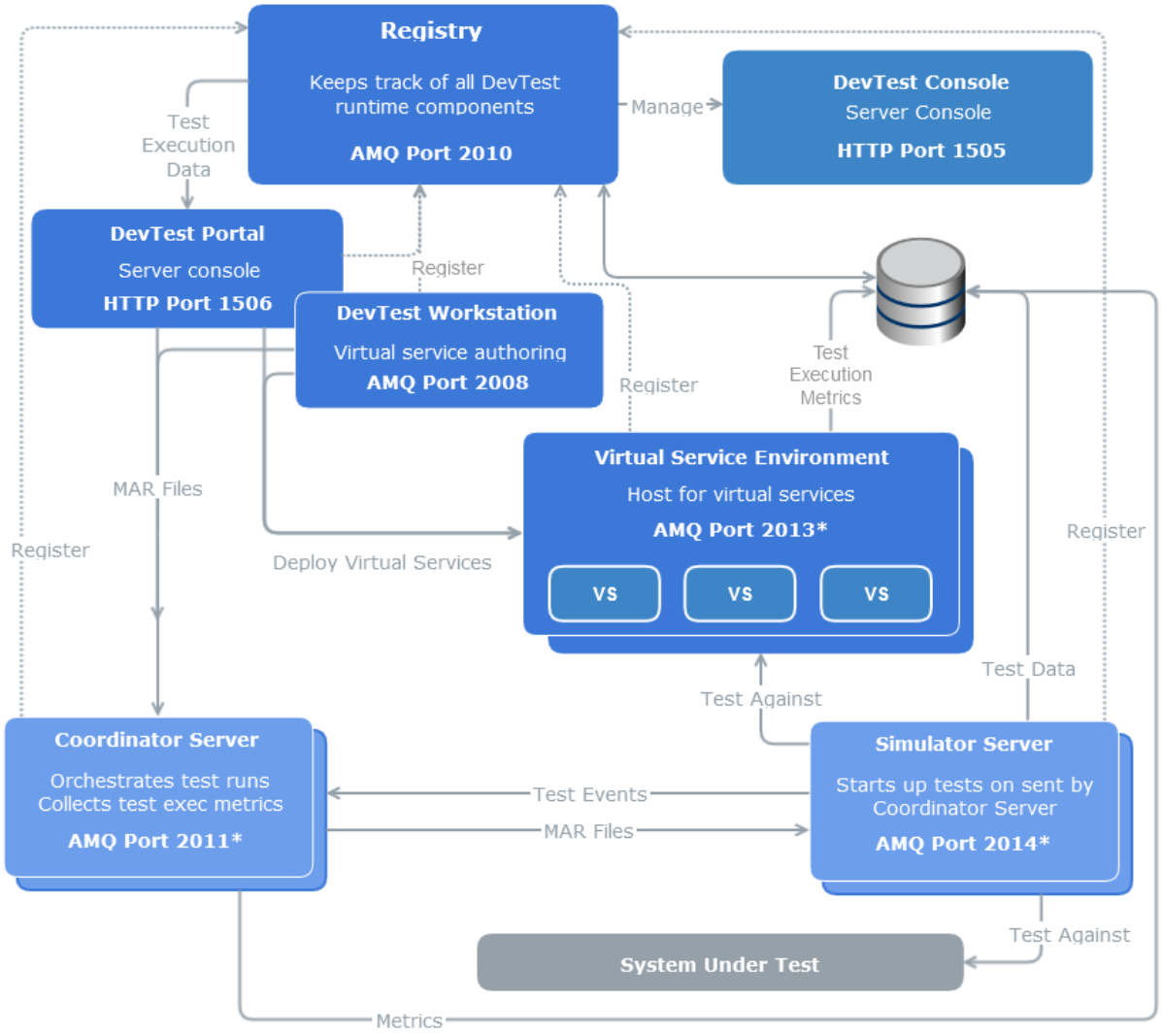


DevTest Server Components

In CA Application Test, the tests are run in the DevTest Server environment. DevTest Workstation connects to the DevTest Server to deploy and monitor tests that were developed in DevTest Workstation.

CA Application Test and CA Service Virtualization use the following DevTest Server components:

- **DevTest Workstation:** An integrated development environment (IDE) where test case assets and virtual service models are created and edited. You can run test cases and models locally in the workstation or you can stage them for a remote execution. DevTest Workstation must be installed on desktop computers for users who author test and virtual model assets. Any number of workstations can attach to the registry and can share the server environment. For more information, see DevTest Workstation in *Using CA Application Test.*
- **Registry:** A central location for the registration of all DevTest Server and DevTest Workstation components. For more information, see Registry in *Using CA Application Test.*
- **DevTest Console:** The Console includes links to the Server Console (including VSE), CVS Dashboard, Reporting, and VSEasy.
- **Coordinator Server:** The coordinator receives the test run information as MAR files, and coordinates the tests that are run on one or more simulator servers. For more information, see Coordinator Server in *Using CA Application Test.*
- **Simulator Server:** The simulator runs the tests under the supervision of the coordinator server. For more information, see Simulator Server in *Using CA Application Test.*
- **VSE:** Used to deploy and run virtual service models. For more information, see *Using CA Service Virtualization.*



*Multiple Coordinators, Simulators, and VSEs on the same machine will have different port numbers.

The registry, coordinator server, simulator server, and VSE are "headless" Java applications that run in separate virtual machines.

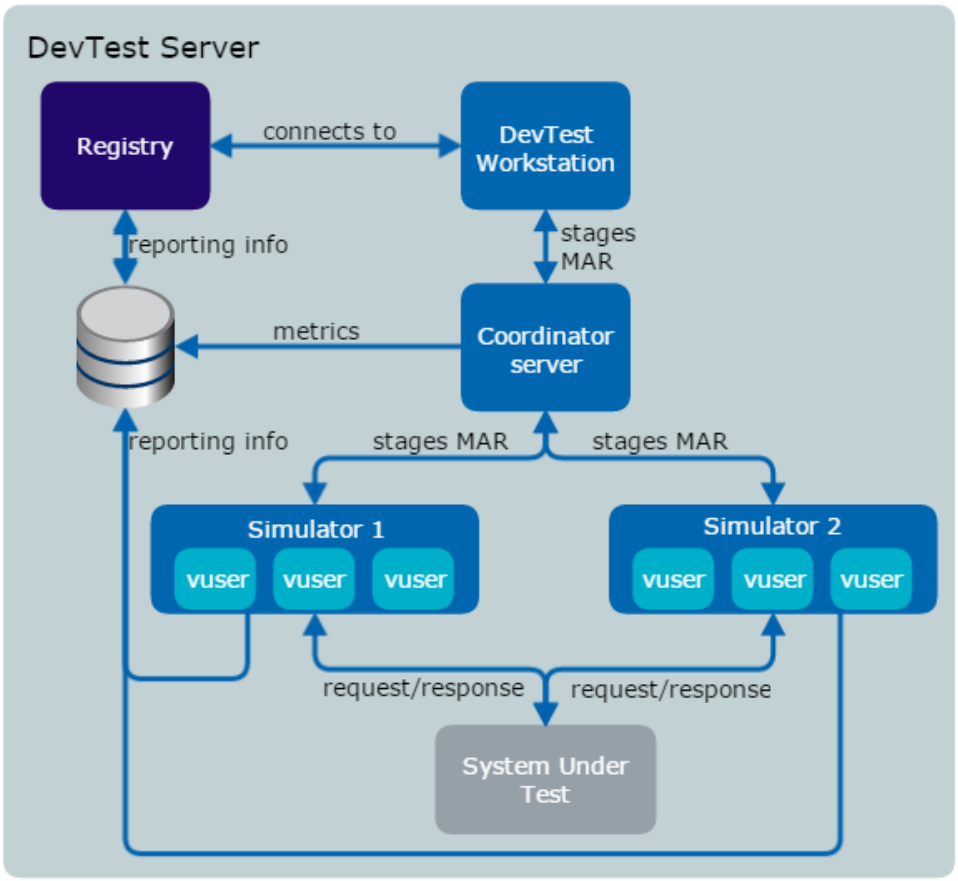
A minimal DevTest Server configuration for CA Application Test and CA Service Virtualization must include at least one of each of these components. There can be as many instances of each type as is needed for a specific testing environment.

Typically, a DevTest Server configuration for CA Application Test has one registry, one coordinator server, and multiple simulator servers.

VSE is a server-level service. The service can coexist with a registry that has a coordinator and a simulator that are attached to it. The simulator and coordinator are not mandatory to run VSE.

Data Flow in DevTest Server for CA Application Test and CA Service Virtualization

The following graphic shows the data flow among the registry, DevTest Workstation, coordinator server, simulator servers, and database.



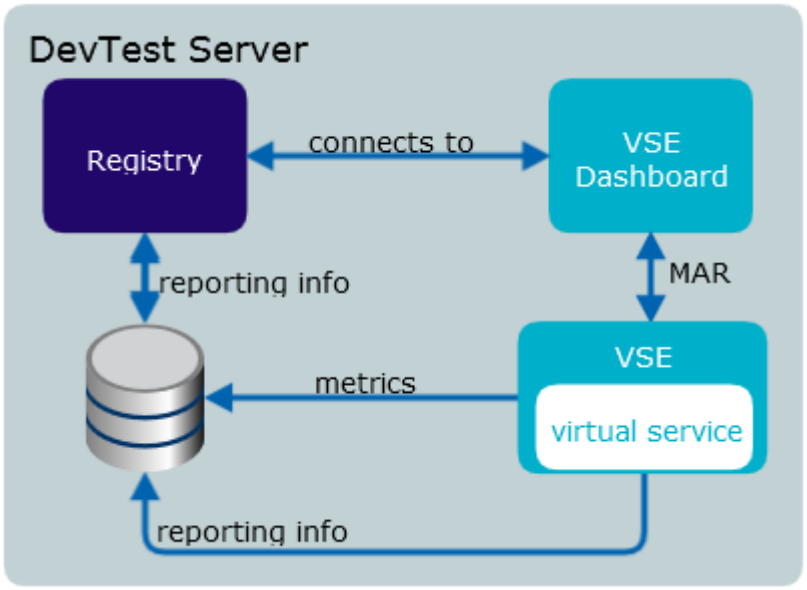
The coordinator server sends test cases to one or more simulator servers in the form of Model Archives (MARs).

The simulator servers interact with the system under test. The types of data that are exchanged between these components can vary enormously. The data could be simple HTTP requests with HTML responses, web service calls, database calls, and so on.

The various components send metrics and reporting information to the database.

The registry hosts the reporting portal, so it communicates with the database to retrieve reporting data.

The following graphic shows the data flow among the registry, VSE, VSE Dashboard, and database.


DevTest Server
Registry
connects to
VSE
Dashboard
reporting info
MAR
VSE
metrics
virtual service
reporting info

Download DevTest Solutions Installers

The Download Center on CA Support Online lets you download one or more platform-specific installers and an optional related file for this release of DevTest Solutions.

First, download the platform-specific installers. After the download is complete, repeat the process to download the Demo Server with examples. You can use the demo server to practice with examples or to work through the tutorials that are included in the CA Application Test documentation.

Follow these steps:

1. Go to support.ca.com.
2. Click Login and log in with your CA Support Online user name and password.
3. Hover over Download Center and select Download Products.

   The Download Center page opens, with the All Products option selected.
4. Type one of the following DevTest product names and the drop-down list displays relevant options.
   - CA Service Virtualization for Performance – MULTI-PLATFORM
   - CA Service Virtualization Power User – MULTI-PLATFORM
   - CA Service Virtualization Runtime User – MULTI-PLATFORM
   - CA Service Virtualization Virtual User – MULTI-PLATFORM
   - CA Continuous Application Insight Power User – MULTI-PLATFORM
   - CA Continuous Application Insight Runtime User – MULTI-PLATFORM
   - CA Application Test – MULTI-PLATFORM
5. Select the product to download. Each of these selections lets you download the DevTest Solutions Installation Wizard for your operating system. With this installer, you can install the DevTest Server, which includes all three products, and the DevTest Workstation.
6. Select the specific release to download.
7. Accept the default values for all other fields.
8. Click Go.

   A list of product components opens.
9. Select the check box in the Add to Cart column for each component to download.
   - (Required) The installer for each of your operating systems
   - (Optional) DevTest Demo Server
10. Click My Download Cart at the top of the window.

The My Download Cart page opens.

11. Enter your email address in the Checkout section, and click Checkout.

    The Download Method page opens.

12. Click Download and save the file.

Chapter 2: DevTest Installation Overview

This installing documentation covers a single, all-inclusive DevTest Solutions installation. There is no breakout of per-product sections. CA Application Test, CA Service Virtualization, and CA Continuous Application Insight are installed together as one solution set.

This section contains the following topics:

Installation Options

Those users without system administrator privileges can install the Server (DevTest Server), the DevTest Workstation, and the Demo Server. However, only system administrators can install the Server components as Windows services.

Options of Components to Install

The DevTest Solutions Setup Wizard presents you with three dialogs that control the options that are installed on the local host. You can select only one option from each dialog. The default option is shown in italics:

- **Select Components**
  - *Server* (with Workstation) also known as DevTest Server
  - Workstation (only)
- **Enterprise Dashboard**
  - *New Enterprise Dashboard*
  - Existing Enterprise Dashboard Service
    This option requires the devtestlic.xml license file to be available.
- **Demo Server**
  - Install
    This option requires the Demo Server installation file to be downloaded and present.
  - *Do not install*

The installer presents a dialog requesting a configuration type choice. The two configuration types are Shared and Local, where the Shared configuration is suited to multiple users on a single system. The Local configuration is suited to an installation where users may access the installation from other systems or where components are controlled and run on multiple systems (a distributed configuration).

Typical Configurations

Standalone (Single computer that is used to host DevTest components)

If you are installing a new standalone DevTest Solutions on one host, install all three components:

- Server (with Workstation)
- New Enterprise Dashboard
- Demo Server

If multiple users will use the embedded DevTest Workstation on a single computer, select the Shared option for Installation Type. For more details, see Shared Installation Type.

Distributed Servers (Multiple computers that are used to host DevTest components)

If you are installing a distributed DevTest Solutions system on multiple hosts, consider the following approach:

1. For the *first installation*, from a server that meets system requirements, select the following options:
   - Server
   - New Enterprise Dashboard, where you browse to the license file, **devtestlic.xml**. The installer validates the license file and its content for signature, version, and expiration.
2. For *more installations of servers*, select the following options:
   - Server
   - Existing Enterprise Dashboard

   Note: You install the server even if you plan to use only one component from it. For example, if you are collecting many metrics or requesting many reports, you can run a coordinator server on a computer with no other service running.
3. To install components on individual hosts for CA Application Test and CA Service Virtualization users, select the following options:
   - Workstation

     When users log in to the workstation, they link to the registry installed with one of the servers.
   - Demo Server - Install (for new users who could benefit from tutorials)
4. For CA Continuous Application Insight users, ensure that a supported browser (see page 19) is installed. All work is done from a web-based portal.

Options for Installing Required Processes (Windows)

Up to seven processes must be started to use DevTest Solutions. The installation wizard lets you select how to perform this task:

- Create a Start menu folder
- Install Services

  Note: You can select one of these options, both options, or neither option. If you select neither, you can start the process executables from the bin folder under the DevTest server installation directory (LISA_HOME\bin) or from a command prompt.

How to Install and Set up DevTest Solutions

This installation guide includes system requirements, conceptual background, and procedures that are required only under certain circumstances. Use the following guidelines to install DevTest Solutions for the first time:

1. Download DevTest Solutions installers (see page 37).
2. If you are installing the Enterprise Dashboard, store your **devtestlic.xml** license file on the target system.

   Note: Record the location of the license file for future use by CA Product Support.
3. If you are using the generic UNIX installer, supply your own JVM (see page 12).
4. Install DevTest Server on as many systems as you need. When you install the first DevTest Server, select New Enterprise Dashboard and browse to the **devtestlic.xml** license file.
   - Windows (see page 45)
   - UNIX (see page 50)
   - Mac (see page 52)

Post-Installation Steps

1. Configure external databases.
   - Configure an external database for Registries. (DB2, MySQL, Oracle, or SQL Server).
   - Configure an external database for the Enterprise Dashboard. (MySQL, Oracle, or SQL Server)

   Note: There is no migration path from the Apache Derby database that is installed with DevTest Solutions to an enterprise-grade database. If you use Derby, you must re-enter the mandatory user authentication (role) data when your system becomes generally available, where an enterprise-grade database is required.
2. Activate the Enterprise Dashboard and all registries.
   a. If you are upgrading, configure existing registries (see page 59). Registry configuration is automated for releases 8.0.0 and above.
   b. Activate the registries (see page 56).

      Note: This process also activates the Enterprise Dashboard with a product key.
   c. Verify registry activations (see page 57). (New or upgrade)
3. Perform post-installation (see page 58) tasks.
4. (Optional) Install more DevTest Workstations with the Demo Server for use with each registry.

- Windows (see page 72)
- UNIX (see page 74)
- Mac (see page 76)

5. Verify the DevTest Solutions installation (see page 83).
6. To prevent unauthorized use, change the passwords for standard users.
7. Install the integration tools (see page 93) that you need.
8. Set up the mobile testing environment (see page 127).

When the installation is complete, see Administering for details on setting up ACL-enforced security and honoring your license agreement.

How License Activation Works

One DevTest Solutions license (**devtestlic.xml**) is issued for each enterprise. This file unlocks all functionality of DevTest Solutions. When you install your first DevTest Server with the DevTest Solutions Setup wizard, you navigate to the license file. The Setup process then installs the Enterprise Dashboard and places that file in the DevTest Server installation directory (LISA_HOME). When you install more DevTest registry servers, you provide the URL to the Enterprise Dashboard Server.

Note: To change the location of the Enterprise Dashboard database, you must rerun the installer while retaining the current database schema. Back up customized properties files before you reinstall. These include local.properties, site.properties, and the various vmoptions files.

Important! If the URL to the Enterprise Dashboard changes, you must update the **local.properties** file in the LISA_HOME directory for each registry server that reports to the Enterprise Dashboard, setting the **lisa.enterprisedashboard.service.url** property with the new URL.

When you start the Enterprise Dashboard process and each registry process, the Enterprise Dashboard process reads the registry settings and it activates the registries. The activated registries are displayed on the Enterprise Dashboard UI for your verification.

Important! If the host name or port of a registry changes, you must restart the registry so that the Enterprise Dashboard can reactivate it.

Note: See Reactivate a Registry or Enterprise Dashboard in *Administering*.

Chapter 3: Installing DevTest Server and Post-Installation

This section describes how to install and configure DevTest Server.

This section contains the following topics:

Install DevTest Server on Windows

This topic describes how to install the DevTest Solutions Server components in a Windows environment.

To preview installation choices, see What You Can Install When You Run the Installer (see page 40).

Before you start the installation procedure, download the following files from the Download Center (see page 37):

- The installer for your platform
- (Optional) The demo server zip file

  Note: Typically, the Demo Server is installed with the Server components only in a standalone system. In a distributed system, the Demo Server is usually installed with the DevTest Workstation for users who are new to DevTest Solutions. The Demo Server is used for tutorials and for many of the artifacts in the Examples project.

Follow these steps:

1. Run the installer file, for example, devtest_win_x64.exe.

   The Welcome to the DevTest Solutions Setup Wizard step opens.

2. Click Next.

   The CA End User License Agreement step opens.

3. Read the license agreement, scrolling to the end, select the I accept the terms of the License Agreement option, and click Next.

   The Select Destination Directory step opens.

4. Enter the path and folder name for the installation directory (LISA_HOME). Consider a name that includes the release identifier, if you want to keep the current release separate from older releases. For example, enter: C:\DevTestServer_8.0. Or, accept the default (C:\Program Files\CA\DevTestSolutions). If you browse to a path and you enter the name of a new folder, the installation wizard creates the folder.

5. Click Next.

   The Installation Type step opens.

6. Select one of the following options and click Next.

   Local

   Installs all DevTest Solutions components into a single directory on the local computer. By default, all data is stored in this directory, and each user has a personal temp directory. Local is the most common installation type that is used in most environments.

   Shared

   Used by administrators to install all of the DevTest Solutions components to a shared location where multiple users can log in and can use the DevTest Workstation. All data and temporary files are stored in user-specified directories. Each user has personal data, but they share a common DevTest Solutions installation. With a shared installation, users only need read access to the DevTest Solutions programs directory. This installation type is a good option for a standalone installation.

   The Select Components step opens.

7. Ensure that the Server check box is selected; an embedded Workstation is installed with the Server. Click Next.

The Enterprise Dashboard step opens.

8. Specify one of the following options, and click Next.

   New Enterprise Dashboard (Specify location of license file)

   If this is the first Server you are installing, click Browse, navigate to the location of the license file, select **devtestlic.xml** and click Open. The installer copies the **devtestlic.xml** file to the specified installation directory (LISA_HOME) on the local host. The new Enterprise Dashboard process is installed in the LISA_HOME\bin directory. This option specifies that all registries are to connect to the new Enterprise Dashboard.

   Existing Enterprise Dashboard Service

   If this is not the first Server that is installed in this network, enter the URL for the existing Enterprise Dashboard. This option specifies that the registry installed with this Server is to connect to the existing Enterprise Dashboard. Replace *localhost* with the appropriate host name:

   `tcp://localhost:2003/EnterpriseDashboard`

   The Demo Server step opens.

9. If you want the installer to unzip the DevTestDemoServer.zip into the LISA_HOME directory, select the Install demo server option. Then, accept the default path (the Downloads directory) or specify another fully qualified path.

   Note: In a distributed system where Workstations are installed on user computers, the demo server is typically installed with the Workstations for new users; not with the Server.

10. Click Next.

11. If you chose the Local installation type, skip the next step, which applies to a Shared installation type.

12. Specify the data directory, clicking Next after each step.

    The Select Start Menu Folder step opens.

13. (Optional) Specify whether the DevTest Solutions processes can be started from the Start menu. (You can start the executable for each process manually from the bin directory under the installation directory.) The advantage of starting the executables, as opposed to the associated services, is that you can monitor messages that are displayed in the command-line interface (CLI).
    - To create a Start menu folder with shortcuts for all users, accept all defaults. Optionally, enter a new folder name.
    - To have no Start menu folder, clear the Create a Start Menu folder check box.
    - To create a Start menu folder and restrict the shortcut display to your Start menu:
        - Accept the selection of Create a Start Menu folder.
        - Accept the default name or enter another name.
        - Clear the Create shortcuts for all users check box.
14. Click Next.

    The Desktop Icons step opens.
15. (Optional) If you do not want to create desktop icons for DevTest Enterprise Dashboard UI, DevTest Portal UI, and DevTest Workstation, clear the Create a desktop icon check box. Click Next.
    - If you selected a Local installation type and you are an administrator, the Windows Services step opens.
    - If you selected a Local installation type and you are a nonadministrator, skip the next step.
    - If you selected a Shared installation type, skip to the information step.

16. (Optional) Select Install Services to create the following Windows services.
    - DevTest Broker Service (for CAI)
    - DevTest Coordinator Service
    - DevTest Enterprise Dashboard Service
    - DevTest Portal Service
    - DevTest Registry Service
    - DevTest Simulator Service
    - DevTest VSE Service

    This selection adds the services to Administrative Tools, Component Services, Services. The advantage to starting services as opposed to the associated executables is that you reduce the number of icons that are displayed on the system tray. If you want the Startup type to be defined as Automatic so that the services start when the host computer is restarted, select the Start on bootup check box. You can configure Automatic startup within Services, if you do not select this check box here.
17. Click Next.

    The Select File Associations step opens. All associations are selected by default.
18. Leave all associations selected or clear the file extensions that you do not want to associate with DevTest Solutions. The file extensions that you can associate with the DevTest Solutions include:
    - *.tst -- Select this extension to create test cases with CA Application Test.
    - *.vsm and *.vsi -- Select these extensions to create virtual services with CA Service Virtualization.
    - *.ste -- Select this extension to run a suite with Test Runner in CA Application Test.
    - *.stg -- Select this extension to run a test case as staging document in CA Application Test.
19. Click Install to start the installation.

    The Installing step opens. When the installation is finished, the Information step opens.
20. Read the information and click Next.

    The Completing the DevTest Solutions Setup Wizard step opens.
21. Click Finish.

Continue as described in How to Install DevTest Solutions (see page 43).

Install DevTest Server on UNIX

This topic describes how to install DevTest Server in a UNIX or Linux environment.

Before you start the installation procedure, download the following files from the Download Center (see page 37):

- The installer for your platform
- (Optional) The demo server zip file

  Note: Typically, the Demo Server is installed with the Server components only in a standalone system. In a distributed system, the Demo Server is usually installed with the DevTest Workstation for users who are new to DevTest Solutions.

The following procedure is based on the graphical version of the installer. To use the command-line version of the installer, add the **-c** option. For example:

```
./devtest_platform_x64.sh -c
```

Note: If you are using the generic UNIX installer, ensure that a Java virtual machine (JVM) (see page 12) is on the same computer. The version of the JVM must be 1.7. You can specify a specific JVM by setting the **JAVA_HOME** environment variable. If the installer cannot find a JVM, the installer displays a message and exits.

Follow these steps:

1. In a terminal window, navigate to the directory where the installer file is located.
2. Ensure that the installer file has the execute permission.

   ```
   chmod 777 devtest_platform_x64.sh
   ```

   This command gives rwxrwxrwx permissions on the file.
3. Run the installer file. For example:

   ```
   ./devtest_platform_x64.sh
   ```

   The Welcome to the DevTest Solutions Setup Wizard opens.
4. Click Next.

   The CA End User License Agreement step opens.
5. Read the license agreement, select the I accept the terms of the License Agreement check box, and click Next.

   The Select Destination Directory step opens.
6. Specify the directory where you want to install DevTest Solutions, for example, /opt/CA/DevTest. Do not use a path with a directory name that contains spaces.
7. Click Next.

The Installation Type step opens.

8. Select one of the following options and click Next.

   Local

   Installs all DevTest Solutions components into a single directory on the local computer. By default, all data is stored in this directory, and each user has a personal temp directory. Local is the most common installation type that is used in most environments.

   Shared

   Used by administrators to install all of the DevTest Solutions components to a shared location that multiple users from multiple computers can access. All data and temporary files are stored in user-specified directories. Each user has personal data, but they share a common DevTest Solutions installation. With a shared installation, users only need read access to the DevTest Solutions programs directory.

   The Select Components step opens.

9. Ensure that the Server check box is selected and click Next.

   The Enterprise Dashboard step opens.

10. Specify one of the following options, and click Next.

    New Enterprise Dashboard (Specify location of license file)

    If this is the first DevTest Server you are installing, click Browse, navigate to the location of the license file, select **devtestlic.xml** and click Open. When you click Next, the installer copies the devtestlic.xml file to the installation directory (LISA_HOME) on the local host. The Enterprise Dashboard process is installed in the LISA_HOME/bin directory.

    Existing Enterprise Dashboard Service

    If the Registry in this DevTest Server connects to the Enterprise Dashboard installed with the first DevTest Server you installed, enter the URL for the existing Enterprise Dashboard (ED).

    The Specify Demo Server step opens.

11. If you want the installer to unzip the demo server into the same directory where you are installing the DevTest Server, select the Install demo server check box and browse to the DevTestDemoServer.zip file.

12. Click Next.

If you chose the shared installation type, the following steps prompt you to specify the data directory and the temporary files directory.

13. Specify the directories, clicking Next after each step.

    The Select Directory for Symlinks step opens.

14. Click Browse and navigate to the directory where DevTest creates symbolic links to the executable files. The default is /usr/local/bin. You must have the required permissions to write to the directory. If you do not want symbolic links to be created, clear the Create symlinks check box.

15. Click Next.

    The Desktop Icons step opens.

16. (Optional) If you do not want to create desktop icons for, DevTest Enterprise Dashboard, DevTest Portal UI, and DevTest Workstation, clear the check box.

17. Click Install to start the installation.

    When the installation finishes, the Information step opens.

18. Read the information and click Next.

    The Completing the DevTest Solutions Setup Wizard step opens.

19. Click Finish.

Continue as described in How to Install DevTest Solutions (see page 43).

Install DevTest Server on a Mac

This topic describes how to install the DevTest Solutions Server components on a Mac.

To preview installation choices, see What You Can Install When You Run the Installer (see page 40).

Before you start the installation procedure, download the following files from the Download Center (see page 37):

- The installer for your platform
- (Optional) The demo server zip file

  Note: Typically, the Demo Server is installed with the Server components only in a standalone system. In a distributed system, the Demo Server is usually installed with the DevTest Workstation for users who are new to DevTest Solutions. The Demo Server is used for tutorials.

Follow these steps:

1. Run the installer file, for example, devtest_osx_x64.dmg.

   The Welcome to the DevTest Solutions Setup Wizard step opens.

2. Click Next.

   The CA End User License Agreement step opens.

3. Read the license agreement, scrolling to the end, select the I accept the terms of the License Agreement option, and click Next.

   The Select Destination Directory step opens.

4. Specify the folder where you want to install one or more components of the DevTest Solutions. If you specify a folder that does not exist, the Setup wizard creates it.

5. Click Next.

   The Installation Type step opens.

6. Select one of the following options and click Next.

   Local

   Installs all DevTest Solutions components into a single directory on the local computer. By default, all data is stored in this directory, and each user has a personal temp directory. Local is the most common installation type that is used in most environments.

   Shared

   Used by administrators to install all of the DevTest Solutions components to a shared location where multiple users can log in and can use the DevTest Workstation. All data and temporary files are stored in user-specified directories. Each user has personal data, but they share a common DevTest Solutions installation. With a shared installation, users only need read access to the DevTest Solutions programs directory. This installation type is a good option for a standalone installation.

   The Select Components step opens.

7. Ensure that the Server check box is selected; an embedded Workstation is installed with the Server. Click Next.

The Enterprise Dashboard step opens.

8. Specify one of the following options, and click Next.

   New Enterprise Dashboard (Specify location of license file)

   If this is the first Server you are installing, click Browse, navigate to the location of the license file, select **devtestlic.xml** and click Open. The installer copies the **devtestlic.xml** file to the specified installation directory (LISA_HOME) on the local host. The new Enterprise Dashboard process is installed in the LISA_HOME/bin directory. This option specifies that the registry installed with this Server is to connect to the new Enterprise Dashboard.

   Existing Enterprise Dashboard Service

   If this is not the first Server that is installed in this network, enter the URL for the existing Enterprise Dashboard. This option specifies that the registry installed with this Server is to connect to the existing Enterprise Dashboard. Replace *localhost* with the appropriate host name:

   `tcp://localhost:2003/EnterpriseDashboard`

   The Demo Server step opens.

9. If you want the installer to unzip the demo server into the LISA_HOME directory, select the Install demo server option. Then, accept the default path (the Downloads directory) or specify another fully qualified path.

   Note: In a distributed system where Workstations are installed on user computers, the demo server is typically installed with the Workstations for new users; not with the Server.

10. Click Next.

11. If you chose the Local installation type, skip the next step, which applies to a Shared installation type.

12. Specify the data directory, clicking Next after each step.

13. Click Next.

    The Desktop Icons step opens.

14. (Optional) If you do not want to create desktop icons for, DevTest Enterprise Dashboard, DevTest Portal UI, and DevTest Workstation, clear the check box. Click Next.

15. Click Next.

The Select File Associations step opens. All associations are selected by default.

16. Leave all associations selected or clear the file extensions that you do not want to associate with DevTest Solutions. The file extensions that you can associate with the DevTest Solutions include:
    - *.tst -- Select this extension to create test cases with CA Application Test.
    - *.vsm and *.vsi -- Select this extension to create virtual services with CA Service Virtualization.
    - *.ste -- Select this extension to run a suite with Test Runner in CA Application Test.
    - *.stg -- Select this extension to run a test case as staging document in CA Application Test.
17. Click Install to start the installation.

    The Installing step opens. When the installation is finished, the Information step opens.
18. Read the information and click Next.

    The Completing the DevTest Solutions Setup Wizard step opens.
19. Click Finish.

Continue as described in How to Install DevTest Solutions (see page 43).

Activate the Registries

After you install all instances of DevTest Server, you start the Enterprise Dashboard process on the first DevTest Server you installed. Then you start the Registry process on that DevTest Server and every other DevTest Server. The activation of the registries does not occur immediately. For example, if you restart the registries between 10:15 and 10:30, the activation occurs at 11:00.

Verify that a registry from each new and existing DevTest Server displays on the Enterprise Dashboard.

Follow these steps:

1. Start the Enterprise Dashboard Server.
   a. Log on to the host where the Enterprise Dashboard is installed.
   b. Start the Enterprise Dashboard Server.

      Windows users can select Enterprise Dashboard Server from the Enterprise Dashboard option in the Start menu. Alternatively, navigate to the LISA_HOME\bin directory and start EnterpriseDashboard.exe.
2. If you have existing registries, configure the existing registries (see page 59).
3. Start each registry.
   a. Log on to the host where a DevTest Server is installed.
   b. Start the registry.

      Windows users start the registry from the Start menu (under DevTest Solutions) or from the Registry.exe in the bin directory under the DevTest Server installation directory.
   c. Repeat these steps for each registry.

      Note: After this property is set, the registry does not push data to the Enterprise Dashboard until the top of the next hour. For example, if you set this property at 1:35pm, the registry data does not appear until 2:00pm.

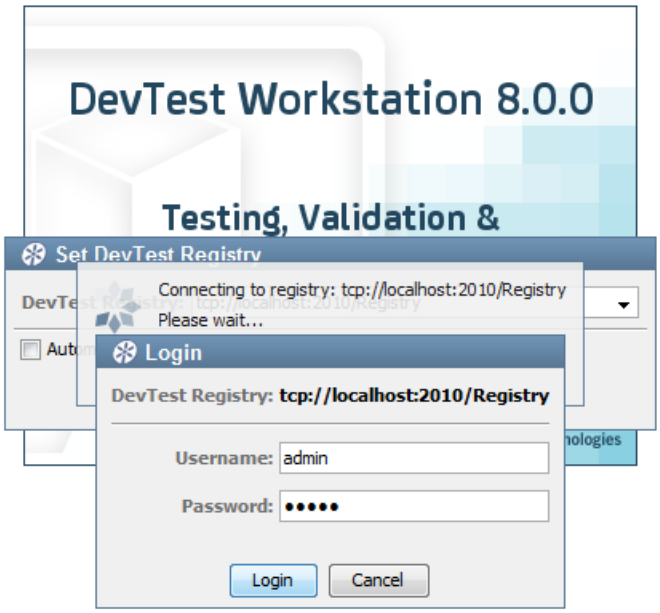
Verify Registry Activation

After starting each registry, wait until the top of the next hour before confirming registry activation.

Follow these steps:

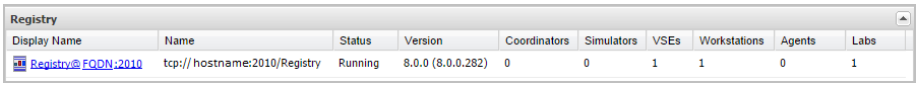
1. Open the Enterprise Dashboard UI in one of the following ways:
   - Browse to the Enterprise Dashboard. Specify the IP address or host name if installed remotely or specify **localhost** if installed locally.

     `http://hostname:1506`
   - From the computer where the Enterprise Dashboard is installed, Windows users select the Start menu option, Enterprise Dashboard, Enterprise Dashboard UI.
2. Log in.



   To log in, type *admin* in the Username field, type *admin* in the Password field, and click Login. The Enterprise Dashboard opens.
3. Examine the registry configurations on the Enterprise Dashboard.

   A list of your registries appears in a format similar to the following example:



Registry									
Display Name	Name	Status	Version	Coordinators	Simulators	VSEs	Workstations	Agents	Labs
Registry@ FQDN :2010	tcp:// hostname:2010/Registry	Running	8.0.0 (8.0.0.282)	0	0	1	1	0	1

4. Verify that a registry from each new and existing DevTest Server displays on the Enterprise Dashboard.

Note: If existing registries are not displayed, configure existing registries (see page 59).

Post-Installation

This section contains the following topics:

Configure Existing Registries

Each DevTest Server you install has one registry. Product licensing requires that the registries are configured. The configuration process varies by release:

- **Version 8.0 and above:** The installation process automatically configures new registries.
- **Releases 7.5.x:** Follow the procedure "To configure the registries (DevTest 7.5.x)"
- **Releases before 7.5:** Follow the procedure "To add a registry from a release earlier than DevTest 7.5"

To configure the registries (DevTest 7.5.x):

1. Log on to a computer where a DevTest Server is installed and navigate to the installation directory.
2. Open local.properties and locate Section 1 - Enterprise Dashboard. (If the local.properties file does not exist, copy _local.properties and rename the copy to local.properties.)
3. Uncomment the following line and replace *somehost* with the host name where Enterprise Dashboard is installed.

   `lisa.enterprisedashboard.service.url=tcp://somehost:2003/EnterpriseDashboard`
4. Save the local.properties file, and then close the file.
5. (Optional) If you plan to use DevTest Workstation from a remote computer, you can specify a property that lets you connect to a specific registry automatically.
   a. Log on to the computer where Workstation is installed.
   b. Navigate to LISA_HOME.
   c. Open local.properties and locate Section 2 - Autoconnection.
   d. Uncomment the following line and replace *somehost* with the host name of the computer where you are logged on.

      `lisaAutoConnect=tcp://somehost:2010/Registry`
   e. Save the local.properties file, and then close the file.
6. Log on to the computer with the target registry
7. Start or restart the registry.
   a. Navigate to the LISA_HOME\bin directory.
   b. Run registry.exe.
8. Repeat Steps 1-7 for each registry.
9. To verify registry configurations, browse to the Enterprise Dashboard.

   `http://hostname:1506`

Important! If there is a change to the host name of the system on which you install DevTest Solutions, repeat this procedure.

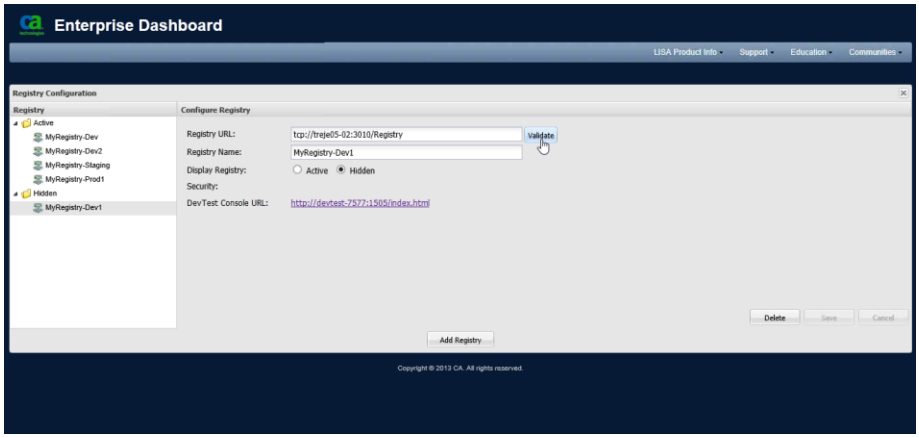
The Registry Configuration window in the Enterprise Dashboard lets you add and configure older registries and also delete unused registries that display in the dashboard.

To add a registry from a release earlier than DevTest 7.5

1. Open the Enterprise Dashboard.
2. Click Options in the upper right corner of the Enterprise Dashboard main window.
3. Select Configure.

   The Registry Configuration window opens with the Configure Registry fields blank.

   Note: 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.



4. Click Add Registry in the Registry Configuration window. Click Yes to confirm.
5. Complete the following fields:

   Registry URL

   The URL of the registry. This field is required.

   Example:

   `tcp://hostname:2010/Registry`

   To validate the URL, click Validate.

   Registry Name

   The common name or nickname for this registry.

   Example:

   `MyRegistry@hostname:2010`

   Display Registry

   Select Active to display this registry in the main panel of the dashboard, or Hidden to hide this registry. This parameter also controls which folder on the left side contains this registry.

   The Registry Configuration window also includes the following fields:

   Security

   Whether ACL security is enabled or disabled.

   DevTest Console URL

   The URL of the DevTest Console for this registry. Example:

   `http://hostname:1505/index.html`
6. Click Save to add the registry to the dashboard.

To validate a registry:

1. Select Configure from the Options menu.
2. Click the registry that you want to validate from the list of registries in the Registry column.
3. Click Validate.

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

Using an HTTP/S Proxy Server - DevTest Server

If you are using a plain HTTP proxy server or an SSL-secured HTTP proxy server, define that proxy server and any hosts to exclude in the **local.properties** file in the LISA_HOME directory.

Follow these steps:

1. Log on to the host where the DevTest Server is installed.
2. Navigate to LISA_HOME.
3. If local.properties does not exist, copy **_local.properties** and save the copy as local.properties (without the underscore).
4. Open local.properties for edit and locate the section header for either HTTP Proxy Server or HTTPS Proxy Server, depending on whether your proxy server uses plain HTTP or is SSL-secured.
5. Identify your proxy server by FQDN or IP address and port:
   - For an HTTP server, use the lisa.http.webProxy.host and lisa.http.webProxy.port properties
   - For an HTTPS server, use the lisa.http.webProxy.ssl.host and lisa.http.webProxy.ssl.port properties
6. Identify any hosts to exclude from going through your proxy server:
   - For an HTTP server, use the lisa.http.webProxy.nonProxyHosts property
   - For an HTTPS server, use the lisa.http.webProxy.ssl.nonProxyHosts property
7. Verify that you removed the comment symbols in front of the properties.
8. Save the file and exit.

Example:

The first two lines of the following example specify that the URL for the HTTP proxy server is **http://192.168.24.242:49185**.

The third line specifies that the hosts that will not go through this proxy include the loopback address for the localhost (127.0.0.1) and IP addresses in the range 192.168.32.0 through 192.168.32.255. Notice that the pipe symbol (|) is used as the delimiter between the IP addresses to exclude. Notice also that the wildcard (*) represents any valid value, where valid values for an IP address node range from zero to 255. The wildcard character can also be used with FQDNs and host names, if hosts to exclude share a standard naming convention.

```
lisa.http.webProxy.host=192.168.24.242
lisa.http.webProxy.port=49185
lisa.http.webProxy.nonProxyHosts=127.0.0.1|192.168.32.*
```

HTTP/S Proxy Server settings in local.properties

```
## ============================================
## HTTP Proxy Server
## ============================================
#lisa.http.webProxy.host=<machine name or ip>
##list of excluded machine names or ip addresses delimited by pipes, * wildcard
accepted <machine name or ip>[|<machine name or ip>]*
lisa.http.webProxy.nonProxyHosts=127.0.0.1
#lisa.http.webProxy.port=

## ============================================
## HTTPS Proxy Server
## ============================================
#lisa.http.webProxy.ssl.host=<machine name or ip>
##list of excluded machine names or ip addresses delimited by pipes, * wildcard
accepted <machine name or ip>[|<machine name or ip>]*
lisa.http.webProxy.ssl.nonProxyHosts=127.0.0.1
#lisa.http.webProxy.ssl.port=

## === Leave blank to use integrated NTLM authentication
#lisa.http.webProxy.host.domain= used for NTLM authentication
#lisa.http.webProxy.host.account=
#lisa.http.webProxy.host.credential=

## === Exclude simple host names from proxy use - default value is true
#lisa.http.webProxy.nonProxyHosts.excludeSimple=false

## === Preemptively send authorization information rather than waiting for a
challenge
## ===== valid values are basic or ntlm
#lisa.http.webProxy.preemptiveAuthenticationType=ntlm
```

Running Components on Different Systems

If the server components are on different systems, be sure to use the following properties correctly:

- The registry uses the **lisa.registryName** property to name itself to something other than the default.
- The nonregistry server components use the **lisa.registry.url** property as the locator.

In the **local.properties** file for the nonregistry server component, you specify the registry with the **lisa.registry.url** property.

```
lisa.registry.url=tcp://registry-hostname-or-ip:port/registry-name
```

For example:

```
lisa.registry.url=tcp://myserver.example.com:2010/Registry
```

Do not use the **lisa.registryName** property for this purpose.

Another option for specifying the registry is to pass **-m** as an argument when starting the nonregistry server component.

```
./CoordinatorServer -m tcp://registry-hostname-or-ip:port/registry-name
```

For example:

```
./CoordinatorServer -m tcp://myserver.example.com:2010/Registry
```

Calculate Simulator Instances

To calculate the number of instances for a specific simulator, do the following analysis:

1. Start DevTest Workstation, select the registry, and log in. Note the memory usage from Help, DevTest Runtime Info.
2. Run the test suite locally and note the memory usage from Help, DevTest Runtime Info.
3. Take the difference between the memory usage in step 2 and step 1.
4. Multiply your available RAM by 60 percent.
5. Divide the available RAM in step 4 by the memory usage in step 3.

The result of step 5 is a good starting estimate of the number of virtual users (instances) that you need configured in your simulator server.

If the coordinator server and the registry both run on the same server as the simulator server, then multiply available RAM by 40 percent. Use 40 percent instead of 60 percent because the coordinator server collects all reports and metrics and therefore consumes RAM.

This technique provides a starting point. To get to the correct number of instances for each simulator, use several iterations and other intuitive methods.

Note: You can set the number of concurrent instances for a simulator with a command-line option. Open a command prompt, navigate to the LISA_HOME\bin directory, and type **simulator --help** for details.

Load and Performance Server Sizing

It is not easy to calculate how many simulation servers are needed for a specific load test. How many servers are required depends on many factors, including:

- Server host configuration (number of CPUs, amount of RAM)
- Test case footprint (number of test steps, type of test steps)
- Other test requirements (number of reports, size of data sets)

We recommend making several test runs of your performance test. These test runs allow you to collect data that can be helpful in determining the configuration of your DevTest Server environment. Collecting metrics and monitoring memory and CPU usage is invaluable for estimating the number of virtual users you can use on a given simulator server.

The registry is lightweight and requires few computing resources. The registry can be run from virtually any computer in your network.

The coordinator server requires resources. Although the coordinator server does not require its own computer, installing it on a separate computer is a common practice. Follow this practice if you are collecting many metrics, requesting many reports, or both.

Simulator servers are used to simulate thousands of virtual users. We recommend running one simulator server per physical server. Technically, a single simulator server can be started with as many instances as you want. However, server memory size and speed typically limit the number of instances for each simulator. A good upper limit is around 250 virtual users.

Vertical or horizontal scaling can be used for sizing the server. In vertical scaling, you increase CPU speed and available memory, which are typically limited. In horizontal scaling, you add more servers. To increase the number of virtual users, horizontal scaling is recommended.

The number of instances per simulator depends on many factors. A simple rule is not available for calculating the maximum number of instances.

Network latency impacts load and performance. We recommend that the database is housed on a server within the same data center as the major DevTest components.

Using DevTest Workstation with Your Java Environment

You can replace the default JRE used by the DevTest installation with your own Java environment. To do so, it is important to understand how DevTest Workstation selects the JRE to use.

The following priority is used to select what Java VM to use when starting DevTest Workstation:

1. The DevTest Workstation-installed JRE in the LISA_HOME\jre directory
2. LISA_JAVA_HOME environment variable
3. JAVA_HOME environment variable
4. JDK_HOME environment variable

Follow these steps:

1. Rename the LISA_HOME\jre directory (for example, rename jre to jre_default).
2. Point the LISA_JAVA_HOME environment variable to your Java installation directory.

See also Supplying Your Own JVM (see page 12).

Change the Default Project Home

By default, projects are saved in the **LISA_HOME\Projects** directory.

This procedure describes how to change the default location for the DevTest Portal. You cannot change the default location for DevTest Workstation.

Follow these steps:

1. Navigate to the **LISA_HOME** directory.
2. Create a text file with the name **res-hub-config.properties**.
3. Add the **resHub.projects.dir** property to the file. Be sure to use forward slashes in the directory path, even on Windows platforms. For example:

   ```
   resHub.projects.dir=C:/MyNewProjectHome
   ```

4. Save and close the file.
5. If the portal server component is running, restart it.

Project Directory Structure

As a best practice, make test assets (for example, projects) available to the server components using them.

To manage access to the test assets, the requirements are as follows:

- Use naming standards. Multiple teams can use the same server environment. To differentiate ownership and purpose and to maintain order, use naming standards.
- The project names must be unique. On the server environment, if two deployed projects have the same name, unexpected things can happen.

Uninstall DevTest Server

The top level of the LISA_HOME directory includes an uninstall application. Use the DevTest Solutions Uninstall wizard to uninstall the DevTest Server components.

Follow these steps:

1. Stop DevTest Solutions.
   a. Verify that all users have logged off DevTest workstations, the Portal, the DevTest Console, and the Enterprise Dashboard.
   b. Verify that no DevTest command-line utilities are running.
   c. Close the executables or stop the services in this order: Simulator, Coordinator, VSE, Broker, Portal, Registry, and Enterprise Dashboard.
2. (Optional) Remove the directory where the main log files are located.
   a. If the logs are stored in a custom directory, open the DevTest Workstation and select DevTest Runtime Info from the Help menu, and scroll to lisa.tmpdir. The path is the value for lisa.tmpdir.
   b. Navigate to the lisatmp_*release-number* directory. The default location of each release-specific log directory (lisatmp_*release-number*) is the USER_HOME directory.
   c. Right-click the directory and select Delete. Click Yes to the confirmation prompt.
3. If you are uninstalling a Shared installation type, manually delete the lisa.user.properties file. The default location is the USE_HOME directory for the user who installed DevTest.

   Important! If you do not delete this file, future installations of a Local installation type will not install correctly.
4. Start the uninstall process.
   - Windows: You can start this application from:
     - The Windows Start menu option
     - DevTest Solutions Uninstaller
     - The Control Panel, Programs, Programs and Features, Uninstall or change a program window.
   - UNIX or Linux: Open DevTest, click uninstall, and select Run.

   The DevTest Solutions Uninstall step opens.
5. Click Next.
6. To delete folders for the database, hotDeploy, lib/core, and related user preferences, select the Delete all files check box.
7. Click Next.

If you did not choose to delete all files, a Results of Uninstaller step opens with the list of files that could not be deleted.

8. Click Finish.

Chapter 4: Installing DevTest Workstation

You can install DevTest Server with an embedded DevTest Workstation or you can install DevTest Workstation as a standalone application.

This section describes how to install and configure DevTest Workstation as a standalone application.

After you install and configure DevTest Workstation, you can log in by following the steps in Open DevTest Workstation in *Using CA Application Test.* If your computer has an installation of DevTest Workstation without DevTest Server, specify a registry that is running on a remote computer. DevTest Workstation installations do not include a local registry.

For more information, see Using DevTest Workstation with Your Java Environment (see page 67).

This section contains the following topics:

Install DevTest Workstation on Windows

This topic describes how to install DevTest Workstation with or without the Demo Server in a Windows environment.

Before you start the installation procedure, download the following files from the Download Center (see page 37):

- The installer for your platform
- (Optional) The demo server zip file

Follow these steps:

1. Run the installer file, for example, devtest_win_x64.exe.

   The Welcome to the DevTest Solutions Setup Wizard opens.

2. Click Next.

   The CA End User License Agreement step opens.

3. Read the license agreement, select the I accept the terms of the License Agreement option, and click Next.

   The Select Destination Directory step opens.

4. Specify the folder where you want to install one or more components of the DevTest Solutions. You can use a directory that contains spaces, such as **C:\Program Files\CA\DevTestSolutions**. If you specify a folder that does not exist, the Setup wizard creates it.

5. Click Next.

   The Installation Type step opens.

6. Select one of the following options and click Next.

   Local

   Installs all DevTest Solutions components into a single directory on the local computer. By default, all data is stored in this directory, and each user has a personal temp directory. Local is the most common installation type that is used in most environments.

   Shared

   Used by administrators to install all of the DevTest Solutions components to a shared location that multiple users from multiple computers can access. All data and temporary files are stored in user-specified directories. Each user has personal data, but they share a common DevTest Solutions installation. With a shared installation, users only need read access to the DevTest Solutions programs directory.

   The Select Components step opens.

7. Clear the Server check box, ensure that the Workstation check box is selected, and click Next.

   If you chose the shared installation type, the following steps prompt you to specify the data directory and the temporary files directory.

8. Specify the directories, clicking Next after each step.

   The Demo Server step opens.

9. If you want the installer to unzip the demo server into the LISA_HOME directory, select the Install demo server option and specify the fully qualified path of the demo server zip file.

10. Click Next.

    The Select Start Menu Folder step opens.

11. Specify whether to add DevTest Workstation to your Start menu and if so, whether to add DevTest Workstation to the Start menu of other users.

    - To create a Start menu folder with shortcuts for all users, accept all defaults. Optionally, enter a new folder name.
    - To have no Start menu folder, clear the Create a Start Menu folder check box.
    - To create a Start menu folder and restrict the shortcut display to your Start menu:
      - Accept the selection of Create a Start Menu folder.
      - Accept the default name or enter another name.
      - Clear the Create shortcuts for all users check box.

12. Click Next.

    The Select File Associations step opens.

13. Select all the extensions to work with the Examples Project tutorials provided in *Using CA Application Test*. The file extensions that you can associate with the DevTest Solutions include:

    - *.tst -- Select this extension to create test cases with CA Application Test.
    - *.vsm and *.vsi -- Select this extension to create virtual services with CA Service Virtualization
    - *.ste -- Select this extension to run a suite with Test Runner in CA Application Test.
    - *.stg -- Select this extension to run a test case as staging document in CA Application Test

14. Click Install to start the installation.

    When the installation finishes, the Information step opens.

15. Read the information and click Next.

    The Completing the DevTest Solutions Setup Wizard step opens.

16. Click Finish.

Install DevTest Workstation on UNIX

This topic describes how to install DevTest Workstation in a UNIX or Linux environment.

Before you start the installation procedure, download the following files from the Download Center (see page 37):

- The installer for your platform
- (Optional) The demo server zip file

The following procedure is based on the graphical version of the installer. To use the command-line version of the installer, add the **-c** option. For example:

```
./devtest_linux_x64.sh -c
```

Note: If you are using the generic UNIX installer, ensure that a Java virtual machine (JVM) (see page 12) is on the same computer. The version of the JVM must be 1.7. You can specify a specific JVM by setting the **JAVA_HOME** environment variable. If the installer cannot find a JVM, the installer displays a message and exits.

Follow these steps:

1. In a terminal window, navigate to the directory where the installer file is located.
2. Ensure that the installer file has the execute permission.

   `chmod 777 devtest_`*`platform`*`_x64.sh`

   This gives rwxrwxrwx permissions on the file.
3. Run the installer file. Double-click the icon or enter a command similar to the following command from a terminal window:

   `./devtest_`*`platform`*`_x64.sh`

   The DevTest Solutions Setup wizard opens.
4. Click Next.

   The CA End User License Agreement step opens.
5. Read the license agreement, select the I accept the terms of the License Agreement check box, and click Next.

   The Select Destination Directory step opens.
6. Specify the directory where you want to install DevTest Workstation. Do not use a directory that contains spaces. (The default is /opt/CA/DevTest.)
7. Click Next.

The Installation Type step opens.

8. Select one of the following options and click Next.

   Local

   Installs all DevTest Solutions components into a single directory on the local computer. By default, all data is stored in this directory, and each user has a personal temp directory. Local is the most common installation type that is used in most environments.

   Shared

   Used by administrators to install all of the DevTest Solutions components to a shared location that multiple users from multiple computers can access. All data and temporary files are stored in user-specified directories. Each user has personal data, but they share a common DevTest Solutions installation. With a shared installation, users only need read access to the DevTest Solutions programs directory.

   The Select Components step opens.

9. Clear the Server check box, ensure that the Workstation check box is selected, and click Next.

   If you chose the shared installation type, the following steps prompt you to specify the data directory and the temporary files directory.

10. Specify the directories, clicking Next after each step.

    The Specify Demo Server step opens.

11. If you want the installer to unzip the demo server into the **LISA_HOME** directory, select the Install demo server check box and specify the fully qualified path of the demo server zip file.

12. Click Next.

    The Select Additional Tasks step opens.

13. If you do not want to create a desktop icon for DevTest, clear the check box.

14. Click Install.

    When the installation is finished, the Information step opens.

15. Read the information and click Next.

    The Completing the DevTest Solutions Setup Wizard step opens.

16. Click Finish.

Install DevTest Workstation on a Mac

This topic describes how to install DevTest Workstation with or without the Demo Server on a Mac.

Before you start the installation procedure, download the following files from the Download Center (see page 37):

- The installer for your platform
- (Optional) The demo server zip file

Follow these steps:

1. Run the installer file, for example, devtest_osx_x64.dmg.

   The Welcome to the DevTest Solutions Setup Wizard opens.

2. Click Next.

   The CA End User License Agreement step opens.

3. Read the license agreement, select the I accept the terms of the License Agreement option, and click Next.

   The Select Destination Directory step opens.

4. Specify the folder where you want to install one or more components of the DevTest Solutions. If you specify a folder that does not exist, the Setup wizard creates it.

5. Click Next.

   The Installation Type step opens.

6. Select one of the following options and click Next.

   Local

   Installs all DevTest Solutions components into a single directory on the local computer. By default, all data is stored in this directory, and each user has a personal temp directory. Local is the most common installation type that is used in most environments.

   Shared

   Used by administrators to install all of the DevTest Solutions components to a shared location that multiple users from multiple computers can access. All data and temporary files are stored in user-specified directories. Each user has personal data, but they share a common DevTest Solutions installation. With a shared installation, users only need read access to the DevTest Solutions programs directory.

   The Select Components step opens.

7. Clear the Server check box, ensure that the Workstation check box is selected, and click Next.

The Demo Server step opens.

8. If you want the installer to unzip the demo server, select the Install demo server option and specify the fully qualified path of the demo server zip file.
9. Click Next.

   The Select Additional Tasks step opens.
10. (Optional) If you do not want to create a DevTest Workstation desktop icon, clear the check box.
11. Click Next.

    The Select File Associations step opens.
12. Select all the extensions to work with the Examples Project tutorials provided in *Using CA Application Test*. The file extensions that you can associate with the DevTest Solutions include:
    - *.tst -- Select this extension to create test cases with CA Application Test.
    - *.vsm and *.vsi -- Select this extension to create virtual services with CA Service Virtualization
    - *.ste -- Select this extension to run a suite with Test Runner in CA Application Test.
    - *.stg -- Select this extension to run a test case as staging document in CA Application Test
13. Click Install to start the installation.

    When the installation finishes, the Information step opens.
14. Read the information and click Next.

    The Completing the DevTest Solutions Setup Wizard step opens.
15. Click Finish.

Using an HTTP/S Proxy Server - DevTest Workstation

If you are using a plain HTTP proxy server or an SSL-secured HTTP proxy server, define that proxy server and any hosts to exclude in the **local.properties** file in LISA_HOME.

Follow these steps:

1. Log on to the host where DevTest Workstation is installed.
2. Navigate to LISA_HOME.
3. If local.properties does not exist, copy **_local.properties** and save the copy as local.properties (without the underscore).
4. Open local.properties for edit and locate the section header for either HTTP Proxy Server or HTTPS Proxy Server, depending on whether your proxy server uses plain HTTP or is SSL-secured.
5. Identify your proxy server by FQDN or IP address and port:
   - For an HTTP server, use the lisa.http.webProxy.host and lisa.http.webProxy.port properties
   - For an HTTPS server, use the lisa.http.webProxy.ssl.host and lisa.http.webProxy.ssl.port properties
6. Identify any hosts to exclude from going through your proxy server:
   - For an HTTP server, use the lisa.http.webProxy.nonProxyHosts property
   - For an HTTPS server, use the lisa.http.webProxy.ssl.nonProxyHosts property
7. Verify that you removed the comment symbols in front of the properties.
8. Save the file and exit.

Example:

The first two lines of the following example specify that the URL for the HTTP proxy server is **http://192.168.24.242:49185**.

The third line specifies that the hosts that will not go through this proxy include the loopback address for the localhost (127.0.0.1) and IP addresses in the range 192.168.32.0 through 192.168.32.255. Notice that the pipe symbol (|) is used as the delimiter between the IP addresses to exclude. Notice also that the wildcard (*) represents any valid value, where valid values for an IP address node range from zero to 255. The wildcard character can also be used with FQDNs and host names, if hosts to exclude share a standard naming convention.

```
lisa.http.webProxy.host=192.168.24.242
lisa.http.webProxy.port=49185
lisa.http.webProxy.nonProxyHosts=127.0.0.1|192.168.32.*
```

HTTP/S Proxy Server settings in local.properties

```
## ============================================
## HTTP Proxy Server
## ============================================
#lisa.http.webProxy.host=<machine name or ip>
##list of excluded machine names or ip addresses delimited by pipes, * wildcard
accepted <machine name or ip>[|<machine name or ip>]*
lisa.http.webProxy.nonProxyHosts=127.0.0.1
#lisa.http.webProxy.port=

## ============================================
## HTTPS Proxy Server
## ============================================
#lisa.http.webProxy.ssl.host=<machine name or ip>
##list of excluded machine names or ip addresses delimited by pipes, * wildcard
accepted <machine name or ip>[|<machine name or ip>]*
lisa.http.webProxy.ssl.nonProxyHosts=127.0.0.1
#lisa.http.webProxy.ssl.port=

## === Leave blank to use integrated NTLM authentication
#lisa.http.webProxy.host.domain= used for NTLM authentication
#lisa.http.webProxy.host.account=
#lisa.http.webProxy.host.credential=

## === Exclude simple host names from proxy use - default value is true
#lisa.http.webProxy.nonProxyHosts.excludeSimple=false

## === Preemptively send authorization information rather than waiting for a
challenge
## ===== valid values are basic or ntlm
#lisa.http.webProxy.preemptiveAuthenticationType=ntlm
```

Environment Settings

DevTest documentation mentions a token that is named **%LISA_HOME%** (for Windows) or **$LISA_HOME** (for OSX or UNIX). This token indicates the location where the DevTest Solution was installed.

On all supported operating systems, an environment variable is set with this name automatically from the launch scripts or programs.

For example, if you installed DevTest Server into **C:\DevTest**_*release_number*, that is the value of **%LISA_HOME%**. DevTest Workstation also has access to the value of this variable in a property named LISA_HOME.

To put more JARs, zips, or directories in the DevTest classpath, you have two options:

- Define the environment variable **LISA_POST_CLASSPATH** and set the resources that you want there.
- Put them in the **%LISA_HOME%/hotDeploy** directory.

Note: For more information about environment settings, see Common Properties and Environment Variables in *Using CA Application Test.*

Chapter 5: Installing the Demo Server

The optional demo server is a JBoss 4.2.3 application server that has several applications for demonstrating DevTest features.

- The **examples** project contains test cases that use the demo server.
- Some of the tutorials in *Using CA Application Test* use the demo server.

Note: The demo server uses port 1529, which cannot be in use by any other application. If this port is not available the demo server does not start successfully.

Follow these steps:

1. Download the demo server as described in Download DevTest Installers (see page 37).
2. Select one of the following approaches to installing:
   - (Preferred) Install the DevTest Solutions (Server or Workstation) and have the setup wizard unzip the file into the same installation directory (**LISA_HOME**). The setup wizard also creates a desktop icon for the demo server.
   - Unzip the **DevTestDemoServer.zip** file on your computer. (In Windows, this file is downloaded to your Downloads folder.) Go to the **lisa-demo-server** folder and follow the instructions in the README file. The README file contains platform-specific instructions for starting the demo server.
3. If you select the second approach, be aware of the following information:
   - You must have Java 7 installed separately on your system.
   - Set the environment variable **JAVA_HOME**. This variable is required for JBoss to compile and run JSP files.
   - Do not put the JBoss Server directory on your desktop or any path that contains spaces. JBoss cannot compile the JSPs if there is a space in the path to its directory.

Notes:

- To start the demo server from the command line, go to the **LISA_HOME\DemoServer\lisa-demo-server** directory and start the script for your operating system:
  - (Windows) **start-windows.bat**
  - (UNIX or Linux) **start-unix-linux.sh**
  - (OS/X) **start-osx.command**
- To run the demo server on UNIX or Linux, use the **/bin/bash** shell.
- The demo server runs the DevTest Java Agent by default, reporting as much information as possible back to CA Continuous Application Insight. To turn off this reporting, use the **-noagent** flag. To turn off heap / stack information only, use the **-noheapss** flag.
- If native agent binaries are present on the widely used UNIX and Linux distributions, the demo server launches the native agent instead. To make the demo server use the pure Java agent in this case, pass the **--javaagent** parameter to the startup command. The native agent binaries are only intended for use with Java 1.4 and earlier.
- The demo server database is created when the demo server is started for the first time. This database is located in the **LISA_HOME\DemoServer\lisa-demo-server\jboss\server\default\data\lisa-demo-server.db** directory.
- After the demo server is started, you can access the server with a browser on port 8080.

Chapter 6: Verifying the DevTest Solution Installation

To verify successful installation of the DevTest Solution, start the DevTest Server, open each user interface and log in.

This section contains the following topics:

Start DevTest and Log In to UIs

Follow these steps:

1. Start the server components (see page 84).
2. Prepare to log in to the user interfaces. You can:
   - Log in as the standard DevTest Super User (see page 88).
   - Create a DevTest user with the Super User role (see page 90) for yourself and log in with your own credentials.
3. Access each user interface (see page 91) and log in. Examine the UIs.
4. (Optional) Examine DevTest Server directories.

Start the DevTest Processes or Services

This section explains the process of starting the DevTest Server with all components available. Use the sequence that is shown to ensure that all components start. Some of the ways you can start the DevTest processes or services are described following the start process.

Note: About DevTest Server Components (see page 23) describes the processes.

Start Order Sequence

Start the DevTest Server processes (or services) in the following sequence. (The Start menu shortcuts are shown.)

Follow these steps:

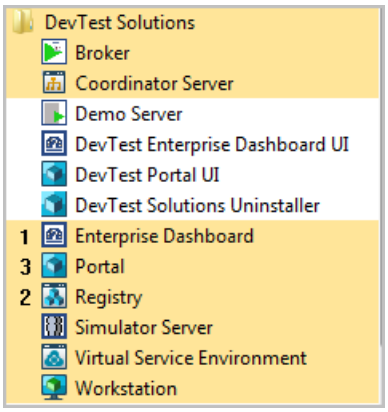
1. Start the Enterprise Dashboard Server.
2. Start each registry.
3. Start the Portal.
4. Start the following components in any order:
   - Broker
   - Coordinator Server
   - Simulator Server associated with each coordinator.
   - Virtual Service Environment
5. If used on the DevTest Server:
   - Workstation
   - Demo Server

Note: To shut down the server components, use the reverse order.

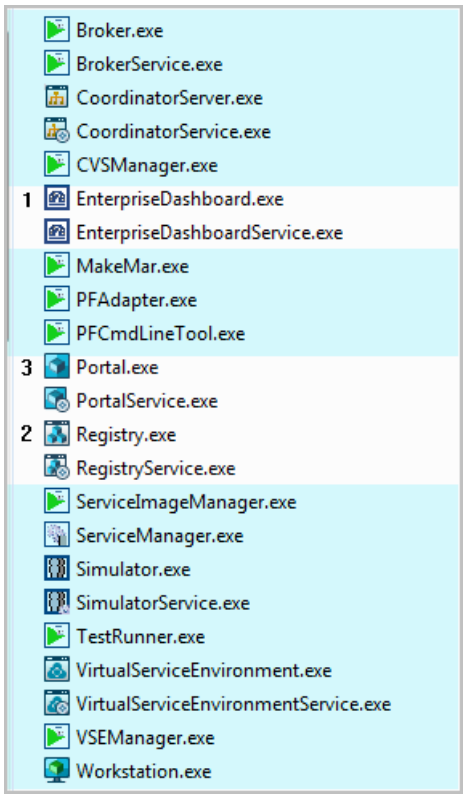
Ways You Can Start DevTest Server

Access the DevTest Server processes in one of the following ways:

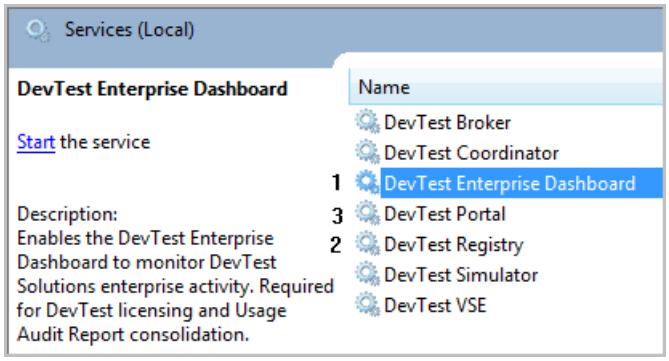
- (Windows) Click the Start menu and expand DevTest Solutions. Start the processes in start order sequence.


DevTest Solutions
Broker
Coordinator Server
Demo Server
DevTest Enterprise Dashboard UI
DevTest Portal UI
DevTest Solutions Uninstaller
1 Enterprise Dashboard
3 Portal
2 Registry
Simulator Server
Virtual Service Environment
Workstation

- Go to the LISA_HOME\bin folder. Start the executables in start order sequence.

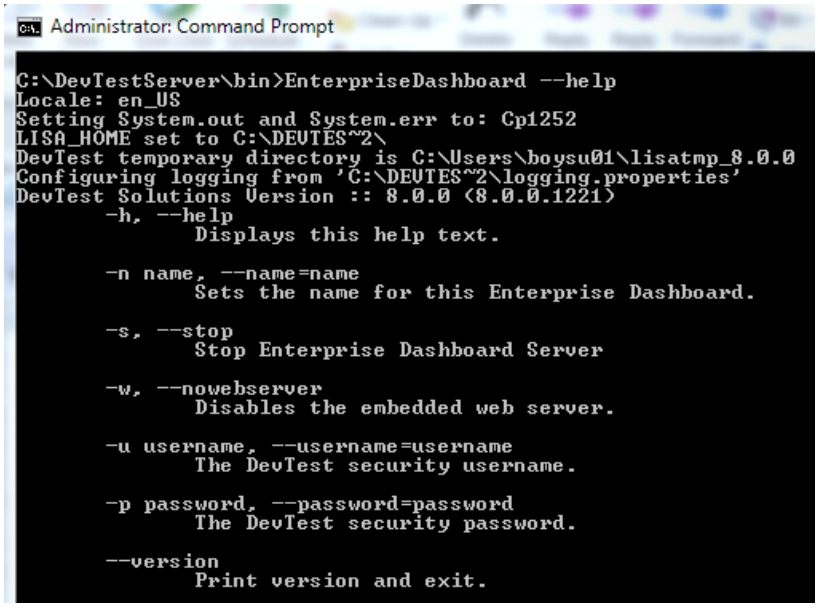


- (Windows) Go to Services and start the services in start order sequence.



- Open a command prompt as an Administrator or open a terminal window. Go to LISA_HOME\bin, and enter the command to start each process in start order sequence. You can also start the respective services or start the associated vmoptions file.

  Get help by entering the command name followed by --help. For example:

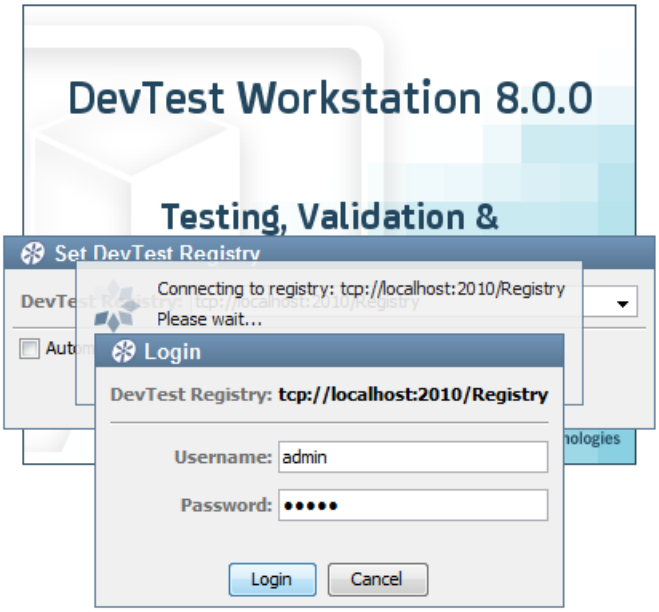

Administrator: Command Prompt
C:\DevTestServer\bin>EnterpriseDashboard --help
Locale: en_US
Setting System.out and System.err to: Cp1252
LISA_HOME set to C:\DEVTES~2\
DevTest temporary directory is C:\Users\boysu01\lisatmp_8.0.0
Configuring logging from 'C:\DEVTES~2\logging.properties'
DevTest Solutions Version :: 8.0.0 (8.0.0.1221)
-h, --help
Displays this help text.
-n name, --name=name
Sets the name for this Enterprise Dashboard.
-s, --stop
Stop Enterprise Dashboard Server
-w, --nowebserver
Disables the embedded web server.
-u username, --username=username
The DevTest security username.
-p password, --password=password
The DevTest security password.
--version
Print version and exit.

Log in as the Standard Super User

Important! Access Control List (ACL) is mandatory. No one can use DevTest Solutions without first logging in with a valid name and password.

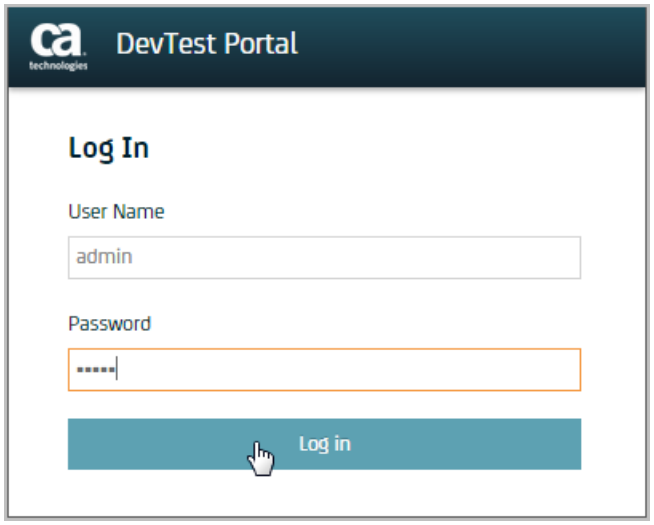
Setting up security involves creating role-based user accounts and specifying the ACL with LDAP or with credentials that you define. If you plan to use LDAP for authentication, you do not add your own user name and password as described in Create a User with the Super User Role (see page 90). To access the UIs before you configure LDAP, use the standard user that is defined with the Super User role.

When you access the DevTest Workstation, a login dialog resembling the following graphic opens:



To log in, type *admin* in the Username field, type *admin* in the Password field, and click Login. The DevTest Workstation opens.

When you browse to the DevTest Portal, the login area resembles the following example.



To log in, type **admin** in the User Name field, type **admin** in the Password field, and click Log in.

The DevTest Console presents a similar login dialog. You can log in to that browser with the same standard user.

Create a User with the Super User Role

Grant yourself full access to DevTest Solutions.

Follow these steps:

1. Browse to the Server Console.

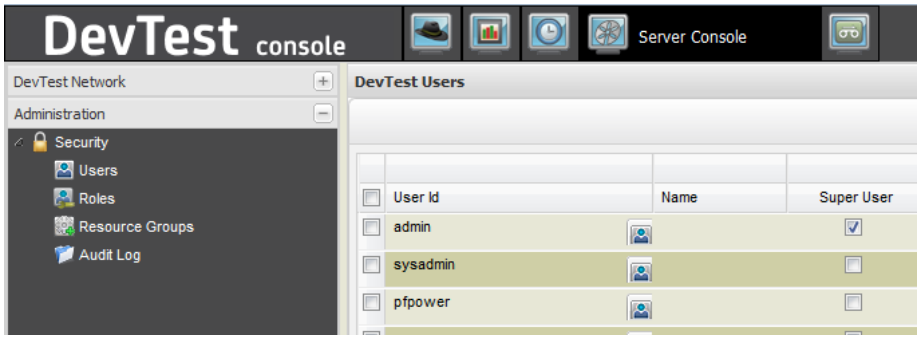
   `http://localhost:1505`

2. Log in as the standard administrator user.

   a. Enter *admin* for the user name.

   b. Enter *admin* for the password.

3. 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.



4. Create a user account for yourself with a password you will not forget. Assign a Super User role for full access to DevTest Solutions.

   a. At the bottom of the right panel, click Add User.

   b. In the User ID field, enter a unique ID that is composed of any combination of alphanumeric, hyphen (-), underscore (_), period (.), and ampersand (@) characters.

   c. In the Password field, enter a password that is composed of any combination of alphanumeric, hyphen (-), underscore (_), and ampersand (@) characters.

   d. In the Re-type Password field, enter the password again.

   e. In the Name field, enter the user name (alphanumeric,hyphen (-), underscore (_), and space characters).

   f. In the Roles for the User area, select Super User.

   g. Click Add User.

   h. Click OK.

Now, you can access any DevTest user interface or command-line interface and log in with the user ID and password that you defined.

Access the DevTest User Interfaces

The Enterprise Dashboard, the registry, and the Portal you are using must be running for all UIs. See Start the Server Components (see page 84).

Access the UIs as described here. Then, log in as the standard Super User (see page 88) or log in with your own credentials, where you defined yourself with the Super User Role (see page 90).

Demo Server

To learn about DevTest Solutions, browse to the demo server on your local host, and follow the tutorials in *Using CA Application Test*.

- Browse to the Demo Server and log in.

  `http://localhost:8080/lisabank`

- (Windows) From the Start menu, expand DevTest Solutions and select Demo Server.

Enterprise Dashboard

To view the list of registries that are connected to the Enterprise Dashboard:

- Browse to the Enterprise Dashboard UI and log in.

  `http://hostname:1506`

- (Windows) Log in to the server where the Enterprise Dashboard is installed. From the Start menu, expand DevTest Solutions and select DevTest Enterprise Dashboard UI.

Portal

Before you use CA Continuous Application Insight, install the DevTest Java agent. Browse to the following URL and log in to the DevTest Portal to access CAI. Specify the host name of a computer with a running registry. See the *Using CAI* documentation.

CAI requires the Portal to be running. The UI requires the Broker to be running. CA Application Test and CA Service Virtualization use the Portal for some functionality. See *Using CA Application Test* and *Using CA Service Virtualization*.

- Browse to the Portal UI and log in.

  `http://hostname:1507/devtest`

- (Windows) From the Start menu, expand DevTest Solutions and select DevTest Portal UI.

Workstation

Start DevTest Workstation on your local host. Workstation is where you create and edit tests and also stage tests. Some alternative methods include TestRunner and junitlisa ant task, for example. See the *Using CA Application Test* documentation. The UI requires a coordinator server and a simulator server to be running.

- (Windows) From the Start menu, expand DevTest Solutions and select Workstation, specify the registry, and log in.
- Go to LISA_HOME\bin and run the Workstation.exe.

DevTest Console

The Console includes links to the Server Console (including VSE), CVS Dashboard, Reporting, and VSEasy. Specify the host name of a computer with a running registry. For information about using the Administrative tab in the Server Console, see *Administering DevTest Solutions*. For other information, see *Using CA Service Virtualization*. The UI for CA Service Virtualization requires VSE to be running.

- Browse to the DevTest Console and log in.

  `http://hostname:1505`

- Open Workstation and click the Server Console toolbar button.

Chapter 7: Installing Integration Tools

This section describes how to install and configure third-party tools that can be used with DevTest.

This section contains the following topics:

Install Performance Monitor (Perfmon)

Performance Monitor (Perfmon) is a utility that demonstrates monitoring the performance of the local or remote system. Perfmon demonstrates how to monitor system performance using performance counters.

To use Perfmon to monitor the performance of a Windows system:

- You must have .NET Framework 2.0 compatibility.
- From a command prompt, run the **setup-wperfmon.bat** file that is located in the **LISA_HOME\bin** directory.
- On Windows, the command prompt must be "Run as Administrator".

Note: If you are running Windows 2012, you can achieve .NET Framework 2.0 compatibility by installing .NET Framework 3.5.

In addition, ensure:

- The user ID is the same on both computers.
- The user ID has administrator privileges on both computers.
- File and Printer sharing is turned ON.
- Simple File sharing is turned OFF.
- The default C$ share, ADMIN$ share, or both are enabled.

Sometimes the firewall on the computer to be monitored must be stopped.

To verify that remote monitoring is working:

1. Select Start**,** Control Panel**,** Administrative Tools**,** Performance.
2. Add a monitor to the computer that you want to observe.

DevTest and Windows use the same technology to do remote monitoring. If the Windows monitoring works, then DevTest monitoring typically works.

To use Perfmon to gather metrics in DevTest, see Windows Perfmon Metrics in *Using CA Application Test.*

Install and Configure SNMP

The Microsoft Windows implementation of the Simple Network Management Protocol (SNMP) is used for the following tasks:

- Configure remote devices
- Monitor the network performance
- Audit network usage
- Detect network faults or inappropriate access.

SNMP support on Windows

Windows 7 provide an agent that is able to answer SNMP requests and send traps.

- Install Microsoft SNMP Agent (see page 95)
- Configure Microsoft SNMP Agent (see page 97)

SNMP support on UNIX

SNMP support is available from your operating system vendor, or you can try the Net-SNMP open source SNMP package. See its accompanying documentation for installation and configuration directions.

Install Microsoft SNMP Agent

Install Microsoft SNMP Agent on Windows 7

You can install the Microsoft SNMP Agent on a Windows 7 operating system.

Follow these steps:

1. From the Start menu, click Control Panel.
2. Click Programs.
3. Under Programs and Features, select Turn Windows features on or off.

   The Windows Features window opens.
4. Select the Simple Network Management Protocol (SNMP) check box and click OK.
5. Wait for SNMP to be installed.

Install Microsoft SNMP Agent on Earlier Windows Versions

Follow these steps:

1. Open the Windows Control Panel.
2. Double-click the Add or Remove Programs icon.

   The Add or Remove Programs window opens.
3. Click Add/Remove Windows Components on the left side of the window.

   The Windows Component wizard appears.
4. Select Management and Monitoring Tools in the Components list and click Details.

   The Management and Monitoring Tools window opens.
5. Select Simple Network Management Protocol from the Subcomponents of Management and Monitoring Tools list and click OK.
6. Click Next.

   The Windows Components Wizard installs the Microsoft SNMP agent.
7. When complete, click Finish.

Configure Microsoft SNMP Agent

This procedure describes how to configure the SNMP Agent.

Follow these steps:

1. Open the Windows Control Panel.
2. Double-click Administrative Tools.

   The Administrative Tools window opens.
3. Double-click Services.

   The Services window opens.
4. Double-click the SNMP service.

   The SNMP Service Properties window opens.
5. Change Startup Type to **Automatic** on the General tab.

   This action configures the SNMP service to start the Microsoft SNMP agent on system startup.
6. Click the Traps tab.
7. Type the community name that your computer sends trap messages to in the Community Name field.
8. Click Add to list.
9. Click Apply and OK.
10. Click OK.

To use Windows SNMP to gather metrics, see *Using CA Application Test*.

Run TCPMon

TCPMon is a utility that lets you monitor the messages that are passed in a TCP-based conversation.

TCPMon consists of the following elements:

- For Windows: A .jar file, a .bat file
- For UNIX: A shell script

To run TCPMon:

Double-click the .bat file on Windows or execute the shell script on UNIX.

You can find a **tcpmon.bat** file in the **LISA_HOME\bin** directory. You can get the latest version of TCPMon from: http://ws.apache.org/commons/tcpmon/.

Note: This section documents the TCPMon version from Apache. This TCPMon version contains a **Sender** tab that is not available in the TCPMon version that is distributed with DevTest.

More information:

Using TCPMon as an Explicit Intermediary (see page 99)
Using TCPMon as a Request Sender (see page 99)

Using TCPMon as an Explicit Intermediary

The most common usage pattern for the TCPMon is as an intermediary. This usage is specified as explicit because the client has to point to the intermediary, rather than the original endpoint, to monitor the messages.

To start the TCPMon in this configuration:

1. Provide the listen port, the host name, and the port for the listener in the Admin tab for TCPMonitor.
2. To open up a new tab (Port 8000) that allows the messages to be seen, click Add.

Requests now point to the listener port of the TCPMon instead of the original endpoint.

All messages that are passed to and from localhost:8080 are monitored.

- We set the listener to port 8000 which can be any unused port in the local computer.
- We added a listener with host as **localhost** and port as 8080.
- Point the browser to **localhost:8000** instead of localhost:8080.

Using TCPMon as a Request Sender

TCPMon can also be used as a request sender for web services.

- The request SOAP message can be pasted into the Sender window and then sent directly to the server.
- The web service endpoint is entered in the connection Endpoint text box.

Install the HP ALM - Quality Center Plug-in

The HP ALM - Quality Center plug-in lets you load and run a DevTest test case as a Quality Center test from the HP ALM - Quality Center suite. You can import into and can run DevTest tests from Quality Center. This integration allows you to take advantage of all Quality Center features while harnessing the power of DevTest testing. By loading a DevTest test case into Quality Center, you get a real-time execution of DevTest tests. You also get the full capture of the test results and DevTest callbacks returning from any system under test. DevTest tests are executable in the workflow of Quality Center, and they report back results to maintain the context and status of the testing process.

The following software must be installed:

- DevTest 7.0 or later
- DevTest HP ALM - Quality Center Plug-in
- HP Quality Center 10 or 11
- HP Quality Center Client Side components
- .NET 2.0 Runtime

When you install the plug-in on a computer that has DevTest installed, the **LisaQCRunner.exe** file is added to the **LISA_HOME\bin** directory. **LisaQCRunner.exe** is the COM server that gets registered as part of the install process. DevTest is invoked using the Quality Center VAPI-XP interface. The VAPI-XP script creates an instance of our COM object. That in turn stages the test to be executed and listens for the test results. Finally, the COM object takes the results and updates the Quality Center instance with the results from DevTest.

The following files are also installed:

- Script_Template_js.txt
- Script_Template_vbs.txt

These files contain scripts that are used during the process of setting up DevTest tests in Quality Center.

Note: For information about using the HP ALM - Quality Center plug-in, see *Using CA Application Test*.

Follow these steps:

1. Ensure that the .NET runtime is installed on the client computer.
2. Install the HP Quality Center Client Side components.
3. Install DevTest.
4. Go to the **LISA_HOME\addons\qc** directory and run the **td_plugin.exe** file.
5. Complete the wizard steps.

Install IBM Rational Quality Manager

IBM Rational Quality Manager is a web-based centralized test management environment for test planning, workflow control, tracking, and metrics reporting. Rational Quality Manager is extensible in a number of ways including the ability to use "connectors" or plug-ins to bridge between RQM and external systems. The DevTest RQM plug-in lets you reuse or create DevTest test cases and tie them into RQM. The DevTest test cases can then be executed from the RQM interface. Results from test runs are aggregated into the RQM test execution and reporting history.

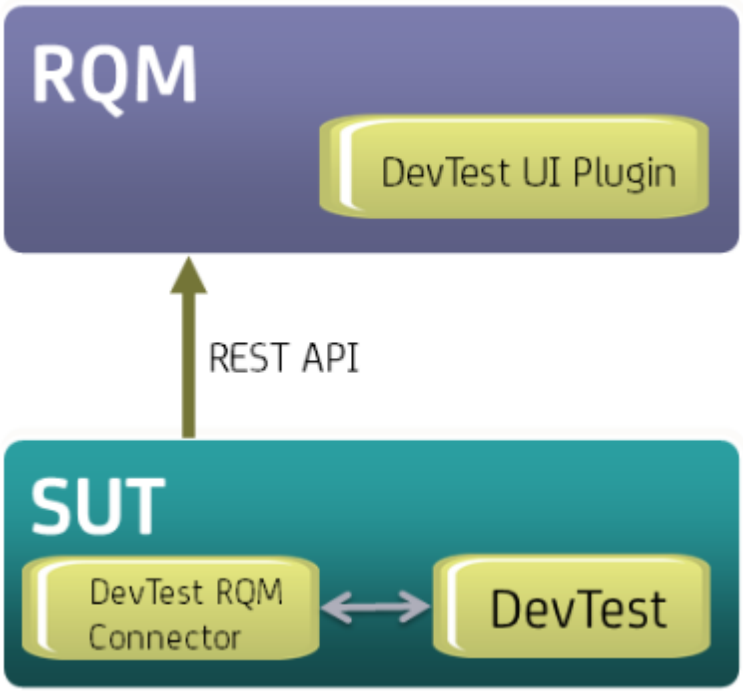
This section contains the following topics:

Implementation

The complete DevTest RQM solution is implemented as two components:

- The DevTest RQM Connector
- The web UI extension.

The web UI allows for the extension of the out-of-the-box UI. This UI also allows the necessary parameters to be passed and displayed between DevTest and RQM. The DevTest RQM Connector is used to respond to work tasks that the RQM schedules. The task information that the UI collects is passed to the connector, and the connector invokes TestRunner with the necessary parameters.



The connector takes advantage of the TestRunner -html switch and uses it to generate the HTML output that is later uploaded back to RQM. After the test completes, the results of the test are uploaded to RQM and associated with the test run. The user can then inspect the results through the RQM interface.

Install the Adapter UI

Follow these steps:

1. Go to the **LISA_HOME\addons** directory and locate the zip file named **rqm-adapter.zip**.
2. Unpack the file and locate the file **com.itko.lisa.integration.ibm.rqm.update.site.zip** in the folder **lisa-adapter-ui**.
3. On the RQM server, unzip the file **com.itko.lisa.integration.ibm.rqm.update.site.zip**.
4. Open the file **com.itko.lisa.integration.ibm.rqm.adapter.web.update.ini** and update the reference to point to the location of the update site.
5. Copy the .ini file that you modified to the following location: **<RQM install root>\server\server\conf\jazz\provision_profiles**.
6. Update the server configuration by using the Rational Quality Manager Server Reset service. The reset utility is found at the following URL:

   [https://<hostname>:<portnumber>/jazz/admin?internal#action=com.ibm.team.repository.admin.serverReset|https://%3chostname%3e%3cportnumber%3e]

   The service prompts you to log in to the IBM Rational Quality Manager Admin console. The user ID must have administrator privileges.
7. Stop the RQM server.
8. Restart the Rational Quality Manager server.

Run Command Line Adapter

Follow these steps:

1. Go to the **LISA_HOME\addons** directory and locate the zip file named **rqm-adapter.zip**.
2. Unpack the file and locate the child folder **lisa-adapter2.0** in the folder.
3. From a command line, start the adapter. The following parameters are needed:
   - -repository [https://*rqmserver*:*port*/jazz|https://%3crqmserver%3e%3cport%3e]
   - [https://%3crqmserver%3e%3cport%3e]-user *userid*
   - -password *password*
   - -adapter *adapter*(com.itko.lisa.integration.ibm.rqm)
   - -adapterName *adapterName* (LISA RQM Adapter)## -LISA_HOME *lisaHomePath* (d\:/lisa)
   - [-projectArea *project area* (Quality%20Manager)]
   - [-sleepTime *sleep time*(5)]

General Usage Workflow

Create a test script in RQM.

1. Associate a DevTest test case, staging document, and optional config document with the test script.
2. Associate one or more test scripts with a test case, and the test case is ready to run.
3. Select to run the test case, select a running adapter, and click OK.
4. The server contacts the system under test through the RQM Connector plug-in, which in turn invokes DevTest.
5. Wait for the test to run, and then verify the run results in RQM.

Configure Integration with CA APM

DevTest Solutions is available in two versions, Workstation and Server. The Workstation version is designed to create and verify DevTest test cases and virtual simulations of server components. The Workstation runs in a single Java process (the DevTest Workstation) and is therefore limited in the size of such tests and simulations. The Server version breaks out a number of functions into one or more separate Java processes that can be run on multiple systems, if necessary.

All DevTest Java processes can be instrumented with CA Application Performance Management (APM) Introscope agents. This instrumentation allows generic metrics that describe the state of the process to be reported to the Introscope Enterprise Manager. These metrics include:

- CPU and memory usage
- Garbage collection operations
- Use of sockets and backend databases

In addition, the DevTest TestEvent tracer can instrument the DevTest coordinator to generate a series of metrics describing the execution of DevTest test cases. This instrumentation is available regardless of whether the coordinator is run as a separate process or in DevTest Workstation. DevTest Workstation is an interactive process with a Windows-based UI. Other than DevTest Workstation, all the other DevTest processes can be run either as regular Windows processes or as Windows services. Both kinds of process can be instrumented with Introscope agents.

This section contains the following topics:

Instrumenting DevTest Processes

The **LISA_HOME\bin** directory contains a Windows executable file for each DevTest process. This folder includes separate versions for those processes that can be run as regular Windows processes and as Windows services. The list of processes and their executable file names is:

DevTest Process Name	Process Type	Executable Name
Coordinator	Regular	CoordinatorServer.exe
Coordinator	Service	CoordinatorService.exe
Enterprise Dashboard	Regular	EnterpriseDashboard.exe
Enterprise Dashboard	Service	EnterpriseDashboardService.exe
Portal	Regular	Portal.exe
Portal	Service	PortalService.exe
Registry	Regular	Registry.exe
Registry	Service	RegistryService.exe
Simulator	Regular	Simulator.exe
Simulator	Service	SimulatorService.exe
Virtual Service Environment	Regular	VirtualServiceEnvironment.exe
Virtual Service Environment	Service	VirtualServiceEnvironmentService.exe
Workstation	Regular	Workstation.exe

Note: This list excludes executables in the bin directory that are not relevant for the Introscope instrumentation.

To instrument any DevTest process with an Introscope agent, create a file with the same name as the executable and the extension .**vmoptions**. For example, to instrument the DevTest Workstation, create a **Workstation.vmoptions** file in the **bin** directory under the DevTest Server installation directory. The .vmoptions file contains the following two lines:

```
-javaagent:<AGENT_HOME>/Agent.jar
-Dcom.wily.introscope.agentProfile=<AGENT_HOME>/core/config/Int
roscopeAgent.profile
```

<AGENT_HOME> is the path to a DevTest-specific Introscope agent installation. Typically this path is an absolute path, but it can also be a path relative to the current directory in which the DevTest process runs.

To have the Introscope agent report under a name other than the default DevTest Agent, add a line defining the **Java com.wily.introscope.agent.agentName** system property. For example, to call the agent DevTest Workstation 6.0.5.87 Agent, add the following line:

```
-Dcom.wily.introscope.agent.agentName=DevTest Workstation
6.0.5.87 Agent
```

To set other JVM command-line options or system properties as required, add extra lines. Each option must be specified in a separate line.

The Introscope agent installation that you use can be specific to the DevTest installation, or even to the DevTest process. You can create variations in agent configurations by varying contents of the **IntroscopeAgent.profile** in those separate agent installations. Or, to create multiple agent profiles in a single agent installation, modify the **com.wily.introscope.agentProfile** system property value that is defined in each **.vmoptions** file. This approach allows different levels of tracing or logging (or even different Introscope Enterprise Managers) to be used to monitor different DevTest processes.

For example, a site can include a production use of DevTest that uses a coordinator node that is run as a Windows service. This site can also include a test/experimental use that uses a coordinator that is run as a regular Windows process. This coordinator could even be the one that is built into the DevTest Workstation. The production metrics are reported to the production EM, while the test/experimental metrics are reported to the test EM. This can be achieved by using multiple profiles and nominating the correct profile in the relevant. vmoptions file. It can also be achieved by overriding the value of **introscope.agent.enterprisemanager.connectionorder** and **introscope.agent.enterprisemanager.transport.*** properties that are defined in the profile by setting Java system properties of those names in the .vmoptions file. Consult the Introscope Java Agent Guide for the full range of Introscope Agent configuration options.

Introscope Agent Files

The DevTest-specific Introscope agent files are shown in the following table. For more details on the contents of these files and how they can be modified to customize the instrumentation of DevTest, see Tracer Configuration (see page 111).

File	Location relative to <AGENT_HOME>	Contents
Lisa.jar	core\ext	DevTest-specific tracers, name formatters.
IntroscopeAgent.profile	core\config	DevTest-specific agent profile. By default, references **lisa-typical.pbl** to define a set of tracers used to instrument DevTest processes.
lisa.pbd	core\config	Tracer and instrumentation point definitions.
lisa-full.pbl	core\config	List of PBD files to be used for "full" DevTest instrumentation. References **lisa-toggles-full.pbd**.
lisa-toggles-full.pbd	core\config	List of toggles used to turn on specific instrumentation features when instrumenting in "full" mode.
lisa-typical.pbl	core\config	List of PBD files to be used for "typical" DevTest instrumentation. References **lisa-toggles-typical.pbd**.
lisa-toggles-typical.pbd	core\config	List of toggles used to turn on specific instrumentation features when instrumenting in "typical" mode.

Metrics Reported by Tracers

DevTest tracers generate metrics at one or more of the four levels:

- DevTest
- Test Case
- Simulator
- Test Step

Metrics that are created at the simulator and test step levels are also aggregated to a user-configurable maximum level (test case, by default). If a test case executes on two simulators and it contains two test steps, the "Responses Per Interval" metric is generated for each test step under each simulator. Then (depending on the configuration) the values are aggregated to generate the "Responses Per Interval" metrics for each simulator. Finally (depending on the configuration) the values are aggregated again to generate the "Responses Per Interval" metric for the test case. The highest level to which any metric can be rolled up is the test case level. Thus, the only metrics that are generated at the DevTest level are listed in the following table with a level of DevTest.

Data from all events of relevant type in each agent reporting interval of 15 seconds are combined as shown.

Metric	Level	Source
Average Response Time (ms)	Test Step	Average of "Long Description" value (converted to integer) from all "Step response time" events (id 18)
Responses Per Interval	Test Step	Count of all "Step response time" events (id 18)
Errors Per Interval	Test Step	Count of all "Step error" events (id 20).
Failures Per Interval	Test Step	Count of all "Abort" (id 50) and "Cycle failed" (id 13) events.
Tests Running	Test Case	Count of test cases running at the end of agent reporting interval. Count is incremented for each "Test started" event (id 4) and decremented for each "Test ended" event (id 5).
Virtual Users Running	Simulator	Count of virtual users running at the end of agent reporting interval. Count is incremented for each "Cycle started" event (id 11) and decremented for each "Cycle ending" event (id 24).
Test Runner Errors Per Interval	DevTest	Count of all "Cycle runtime error" events (id 25)
Staging Errors Per Interval	DevTest	Count of all "Model definition error" events (id 23)

The Errors Per Interval metrics for any test step, simulator, or test case are reported only when the first error event is detected. The same is true for Failures Per Interval.

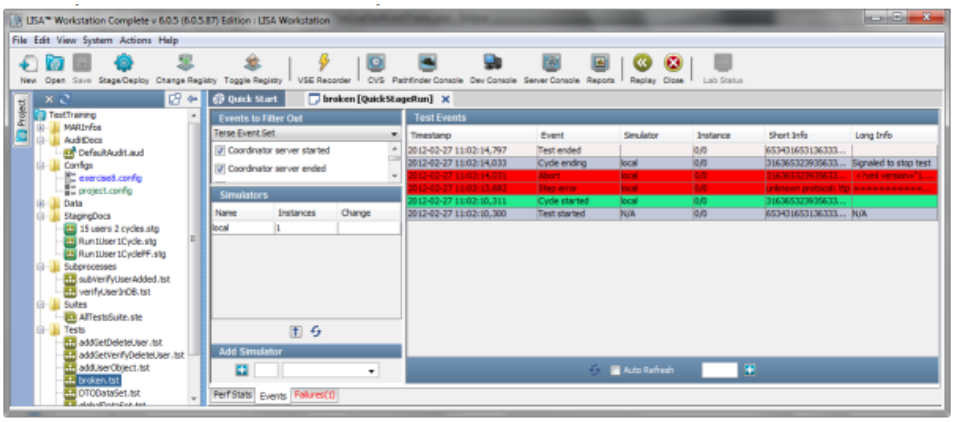
If only a few, short test cases run, the Tests Running and Virtual Users Running metrics can continue to report 0 for each agent reporting cycle. This happens when both start and stop events occur in the same 15-second period, and the count at the end of the period is 0.

When the metrics are being aggregated, the lower-level metric can be created in one agent time reporting period. The aggregated metrics can be created in the following period. Occasionally, there are points where lower level and rolled up metrics within a time period do not agree in value. This situation is unavoidable.

Deriving DevTest Metrics

The DevTest-specific Test Event tracer is a "method" tracer. A method tracer means that it contains code that is inserted immediately before and after a specific method call. The Test Event tracer is designed to instrument the method that is the constructor of the BaseCoordinator class. After the constructor completes, the tracer creates a Test Event Listener and registers this listener with the coordinator. The listener then receives each TestEvent object of the types that were shown previously. The listener extracts relevant information from these TestEvent objects and sends that information to the Introscope Enterprise Manager in the form of Introscope metric values.

The DevTest Workstation screen shot that follows shows the TestEvents generated by one run of a test case named "broken," which contains an error. To understand which of the Introscope metrics are created, look up each value in the Event column, in Metrics Reported by Tracers (see page 109).



Tracer Configuration

All DevTest tracing can be disabled by removing all references to DevTest-specific .pbl files (**lisa-full.pbl** or **lisa-typical.pbl**) in the agent profile (IntroscopeAgent.profile by default). You can also delete **Lisa.jar**. However, if there are any references remain to the tracers defined in **lisa.pbd**, the agent fails to start correctly. This failure is due to an attempt to use a tracer that cannot be found in any agent extension.

The Test Event tracer can be disabled by commenting out the "TurnOn: LisaTestEventTracing" directive in the active DevTest toggles file (**lisa-toggles-full.pbd** or **lisa-toggles-typical.pbd**).

You can use a number of other parameters to limit the number of metrics that reported. These parameters are defined in the **lisa.pbd** file. You can control the following parameters:

- **minMetricLevel:** Parameter that controls the lowest level at which metrics are reported. The parameter can take any of the following values:
  - Lisa
  - TestCase (Default Value)
  - Simulator
  - TestStep
- **maxRollupLevel:** Parameter that controls the maximum level to which metrics can be aggregated. The parameter can take any of the following values:
  - TestCase (Default Value)
  - Simulator
  - TestStep.

Metrics that are generated below the current setting of **minMetricLevel** are aggregated until they reach **minMetricLevel**. Those metrics are then reported from that level up to **maxRollupLevel**, unless **minMetricLevel** is above the **maxRollupLevel.** If **minMetricLevel** is above the **maxRollupLevel**, metrics are not reported at all. Any metrics that are generated at a level above the **maxRollupLevel** are reported only at the generated level, not at the aggregated level.

In addition, six more parameters allow the Test Event tracer to restrict metrics to selected combinations of test cases, simulators, or test steps. You can configure the tracer to include or exclude based on the name of the test case, simulator, or test step. You can match the name against regular expressions or a pair of regular expressions. All names that match the inclusion regular expression and that do not match the exclusion regular expression are selected to report metrics. If the inclusion regular expression is not defined, all names are included. If the exclusion regular expression is not defined, no names are excluded.

By default, all possible test cases, simulators, and test steps are included. Only the internal test steps named "abort" and "end" are excluded. If the regular expression contains certain special characters (for example, the | symbol that is used to specify alternative patterns), the entire regular expression must be in double quotation marks. The quotation marks are optional otherwise. The following shows the default inclusion and exclusion patterns that are defined in **lisa.pbd**.

```
SetTracerParameter: LisaCoordinatorTracer includeTestCasesRegExp
""
SetTracerParameter: LisaCoordinatorTracer excludeTestCasesRegExp
""
SetTracerParameter: LisaCoordinatorTracer
includeSimulatorsRegExp ""
SetTracerParameter: LisaCoordinatorTracer
excludeSimulatorsRegExp ""
SetTracerParameter: LisaCoordinatorTracer includeTestStepsRegExp
""
SetTracerParameter: LisaCoordinatorTracer excludeTestStepsRegExp
"abort|end"
```

You can configure the metric path under which metrics are reported. The DevTest level part of the metric path is the last value that is specified in the **TraceOneMethodWithParametersIfFlagged** directive for the [set the init variable for your book] method. This defaults to "DevTest". For the remaining levels, the following parameters define the parts of the metric path for the levels Test Case, Simulator, and Test Step:

- **pathComponentForTestCase**
- **pathComponentForSimulator**
- **pathComponentForTestStep**

Avoid the following character usage in the values for these parameters:

- The colon character (:) is not valid in metric paths.
- Metric paths cannot begin or end with the metric path node separator (|).
- Metric paths cannot contain two adjacent metric path node separators (|).

An underscore (_) replaces each occurrence of a colon or vertical bar character. Leading and trailing spaces are removed from these names. The value "Unknown" replaces any name with a null value or an empty string.

The default values for these parameters are applied if the parameter is commented out or is an empty string. Each value must contain the relevant placeholder to indicate the place at which the name of the relevant level is inserted into the metric path. The following shows the default values defined in **lisa.pbd**:

```
SetTracerParameter: LisaCoordinatorTracer
pathComponentForTestCase "Test Case|{TestCase}"
```

```
SetTracerParameter: LisaCoordinatorTracer
pathComponentForSimulator "Simulator|{Simulator}"
SetTracerParameter: LisaCoordinatorTracer
pathComponentForTestStep "Step|{TestStep}"
```

Install Introscope Agent

You can install the Introscope agent for DevTest by extracting the contents of the distribution file **CALISAIntegrationNoInstaller9.1.1.0.zip** (on Windows) or **CALISAIntegrationNoInstaller9.1.1.0.tar** (on UNIX/Linux).

To avoid profile file name conflicts, extract this file to a directory that does not contain any of the other Introscope Java agent distributions. Then create one or more vmoptions files to instrument the set of DevTest processes to instrument. The only DevTest processes that report DevTest-specific metrics are the DevTest Workstation and the coordinator. You can instrument other processes. However, the other processes report a minimal set of metrics. These metrics are typically limited to CPU usage, memory usage, and agent and JVM identity.

Configuring Tracer Logging

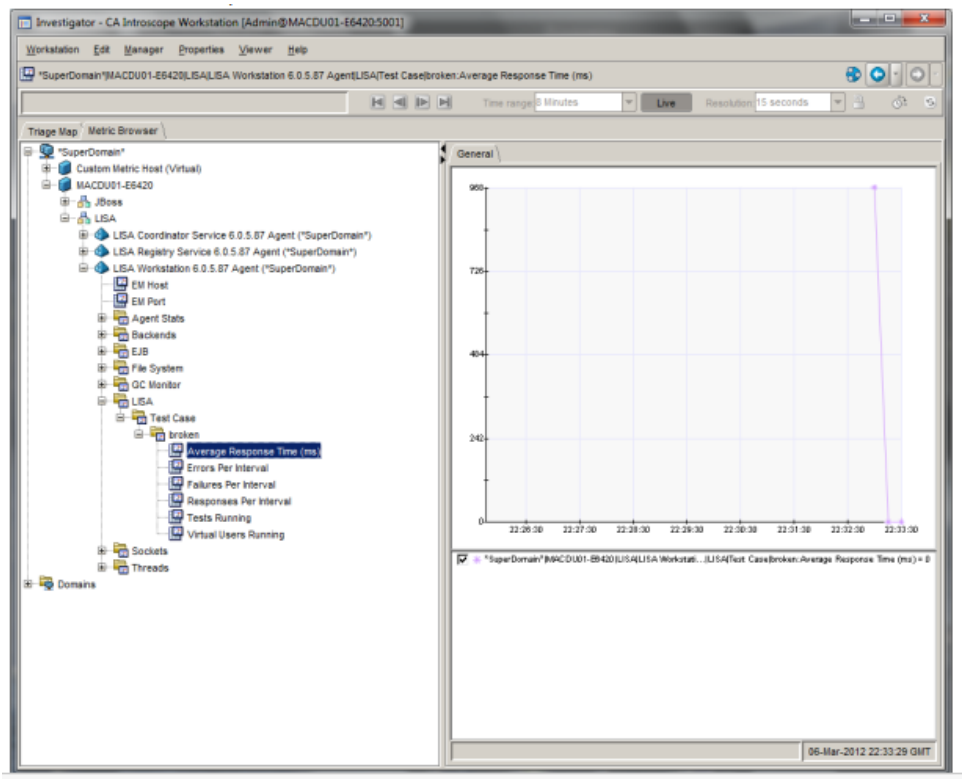
To write more information about the operation of the DevTest-specific tracer to the agent log file, add the following lines to the agent profile:

```
log4j.additivity.IntroscopeAgent.LisaCoordinatorTracer=false
log4j.logger.IntroscopeAgent.LisaCoordinatorTracer=DEBUG, logfile
```

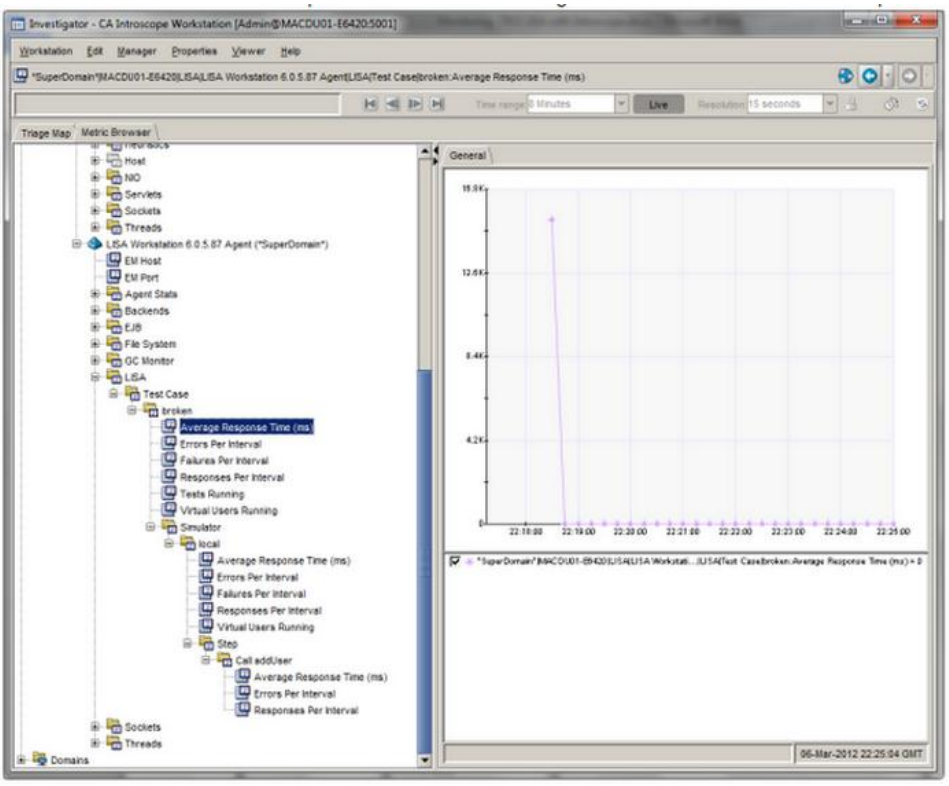
If the agent logging level is set to DEBUG, these lines are not required. However, logging at that level writes a large amount of information to the agent log. This amount of information can make it difficult to find the lines from the DevTest tracer.

Typical Metrics Reported

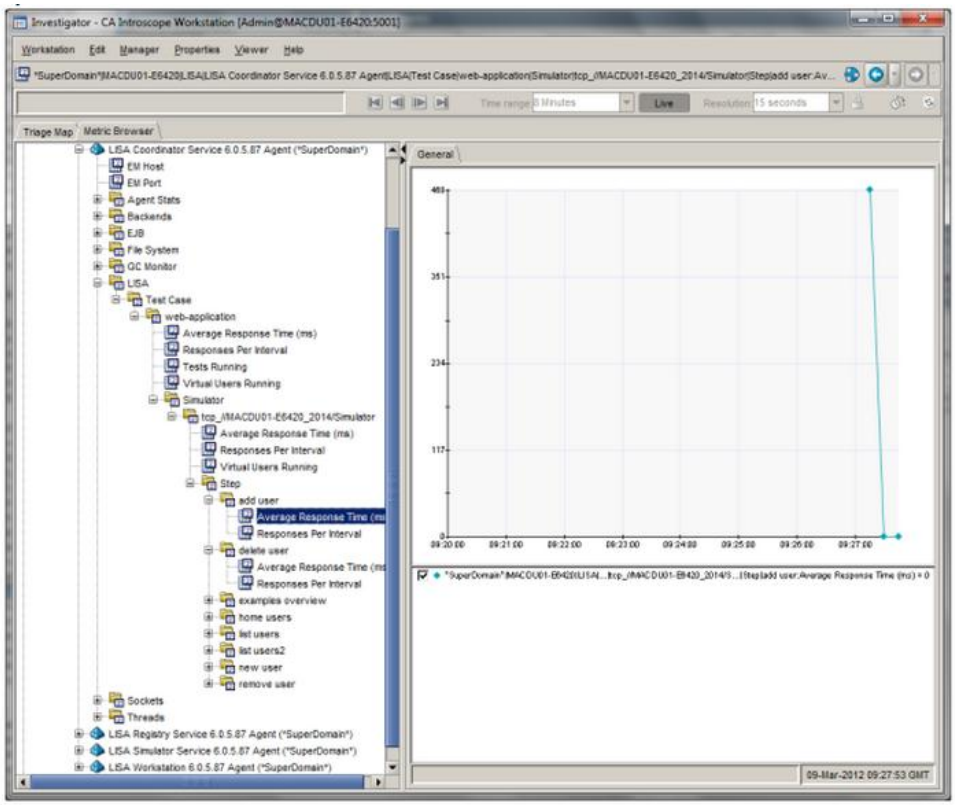
The following graphic shows typical sets of metrics that the Test Event tracer reports. In the following example, the simple test case named "broken" was executed. This test case contains only one test step that contains a configuration error. This error causes the test case to fail. The failure causes the "Errors Per Interval" and "Failures Per Interval" metrics to be reported as shown.



The following graphic shows the same test case that was run after the **minMetricLevel** parameter was changed from the default of TestCase to TestStep. Metrics are generated for simulator and test step levels.



The graphic that follows shows a more complex test case that contains eight test steps and does not generate any errors when run. In addition, the test has been staged using separate coordinator and simulator processes rather than the processes built into the DevTest Workstation. This window shows the additional test step nodes, but no "Errors Per Interval" or "Failures Per Interval" metrics. The metrics are reported under a Coordinator Service Agent node (whose name has been customized to include the DevTest version number). Also notice that because the simulator name contained colon characters that are invalid in metric path names, the colons were replaced by underscores.



Displaying Metrics

The Introscope Enterprise manager ships with a DevTest-specific management module that contains metrics groupings, alerts, dashboards, and reports relevant to monitoring the DevTest environment. Two dashboards are provided:

- Overview Dashboard (see page 117)
- System Under Test Overview Dashboard (see page 123)

Overview Dashboard

The Overview dashboard gives a high-level overview of all instrumented processes and of the tests being run in the DevTest installation.

The Tests section of the Overview dashboard includes four simple alert icons:

Failures

Set to caution if any test has reported a failure in the last reporting interval. Set to danger if any test has reported two or more failures in the last reporting interval.

Errors

Set to caution if any test has reported an error in the last reporting interval. Set to danger if any test has reported two or more errors in the last reporting interval.

Runner Errors

Set to caution if the Test Runner Errors Per Interval metric has a value of 1 for any reporting interval. Set to danger if it has a value of 2 or higher for any reporting interval.

Staging Errors

Set to caution if the Staging Errors Per Interval metric has a value of 1 for any reporting interval. Set to danger if it has a value of 2 or higher for any reporting interval.

This section also includes the following elements:

Overall alert

A summary alert that is set to caution if any of the preceding four alerts is set to caution. This alert is set to danger if any of the preceding three alerts is set to danger.

Test Average Response Time (ms) graph

Shows up to ten Average Response Time (ms) metric graphs for test case, simulator, and test steps. The ten graphs that are chosen have the top ten metric values in the time period displayed.

Test Responses Per Interval Graph

Shows up to ten Responses Per Interval metrics graphs for test case, simulator, and test steps. The ten graphs that are chosen have the top ten metric values in the time period displayed.

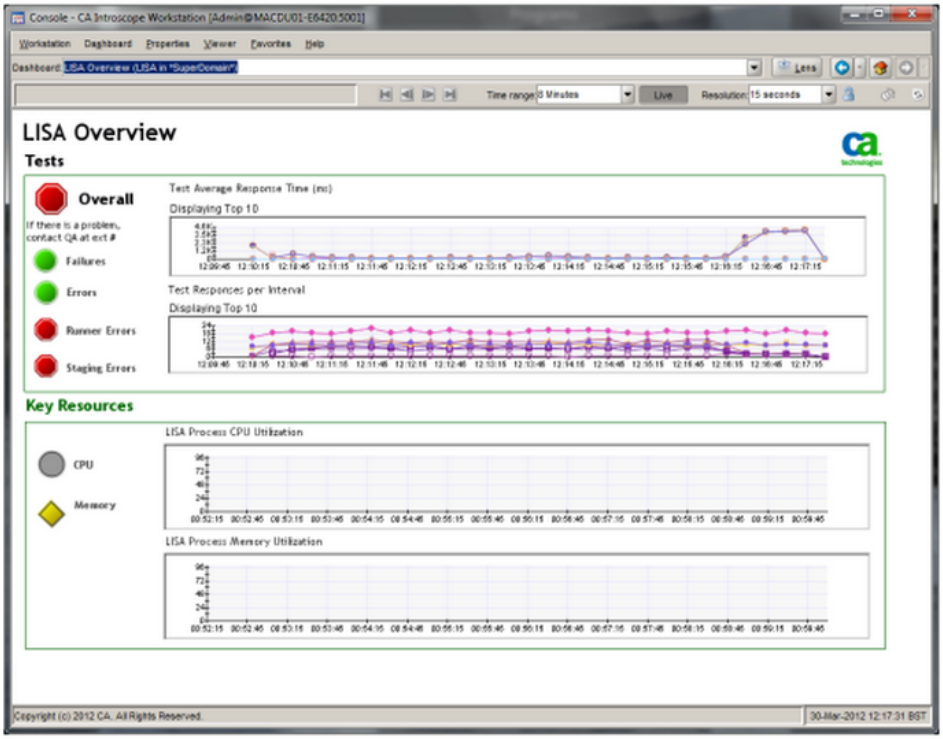
LISA Process CPU Utilization Graph

Shows a graph of **CPU:Utilization Percent** (process) metrics for all instrumented processes.

LISA Process Memory Utilization Graph

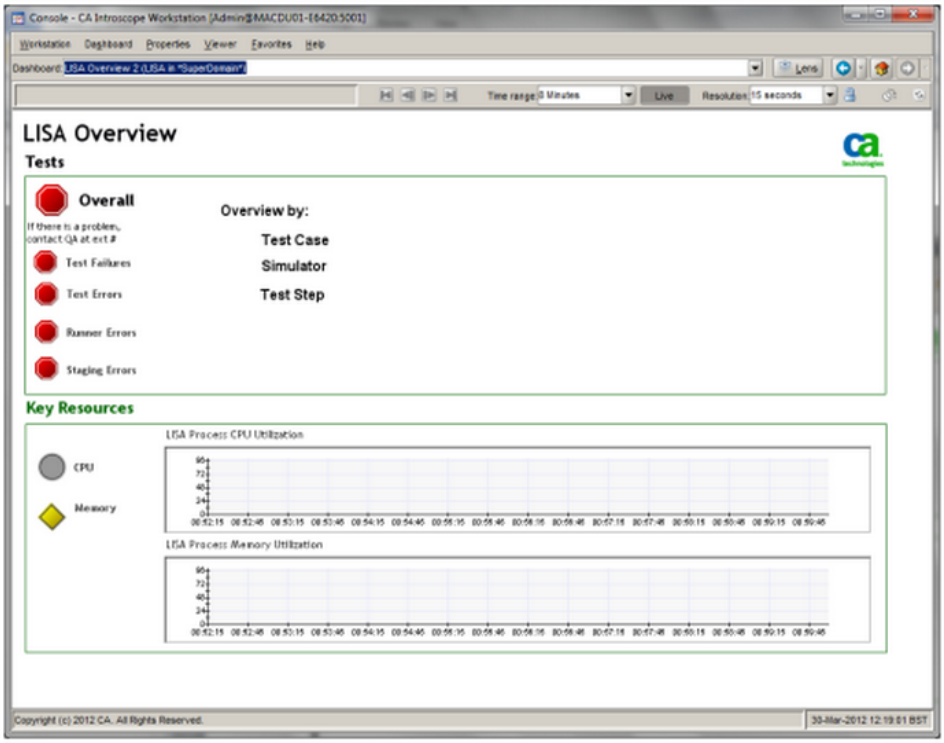
Shows a graph of the **GC Heap:Bytes In Use** metrics for all instrumented processes.

The following graphic shows the LISA Overview Dashboard.



An alternate version of the Overview dashboard replaces the graphs in the top left panel with links to three new dashboards. These dashboards provide overviews of the following metrics:

- Test case metrics
- Simulator metrics
- Test step metrics

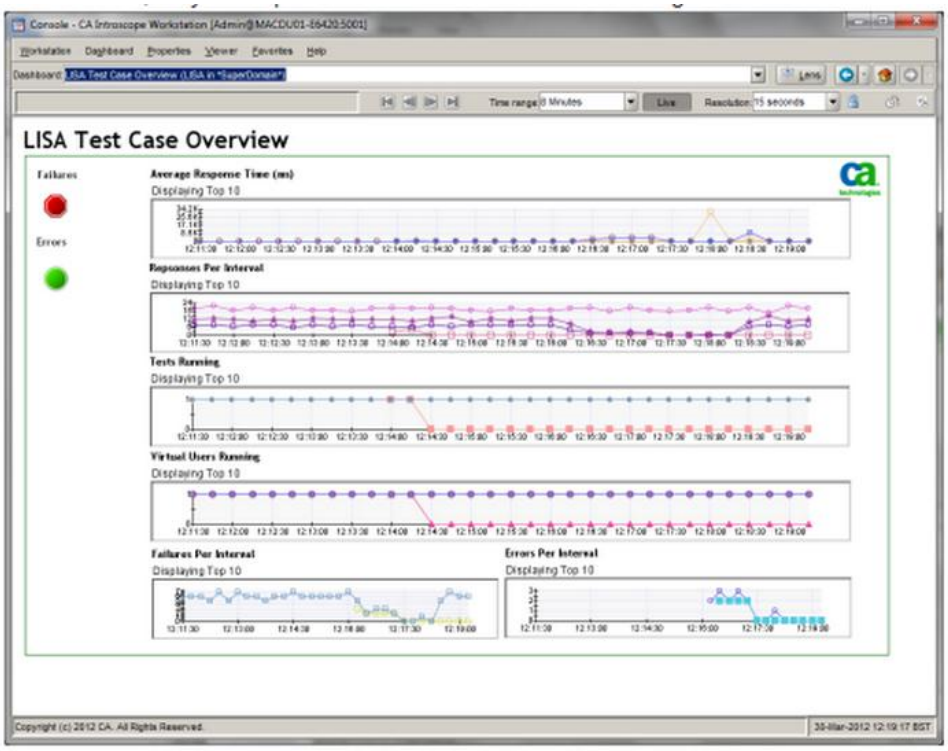


Test Case Overview **Dashboard**

The Test Case Overview dashboard shows graphs for the following metrics:

- Average Response Time (ms)
- Responses Per Interval
- Tests Running
- Virtual Users Running
- Failures Per Interval
- Errors Per Interval

In each case, only the top ten metrics are shown.

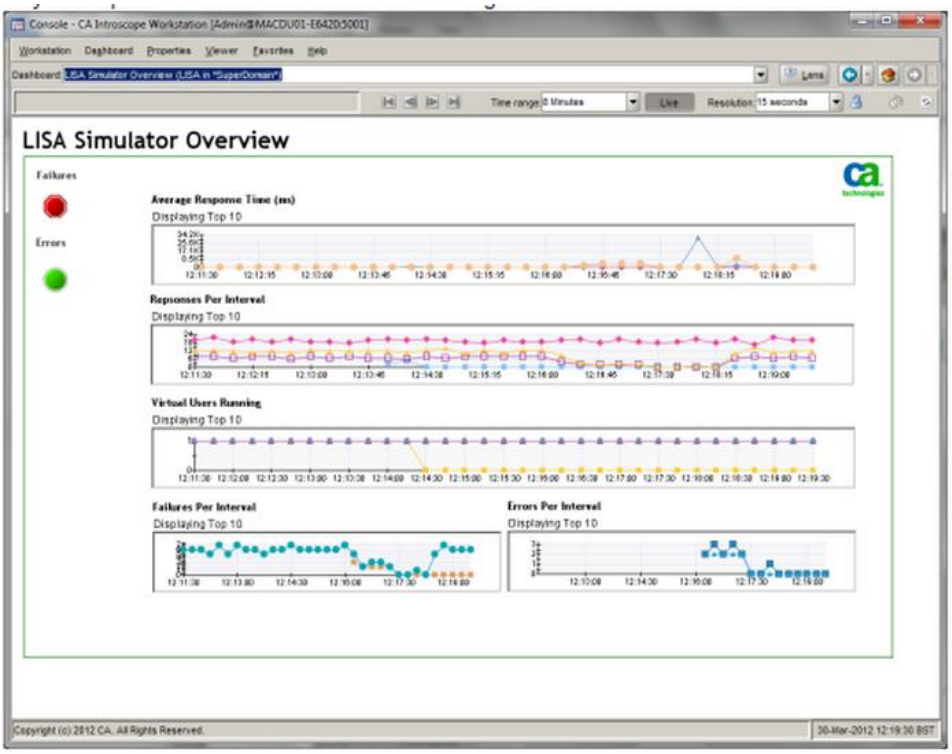


Simulator Overview **Dashboard**

The Simulator Overview dashboard shows graphs for the following metrics:

- Average Response Time (ms)
- Responses Per Interval
- Virtual Users Running
- Failures Per Interval
- Errors Per Interval

In each case, only the top ten metrics are shown.

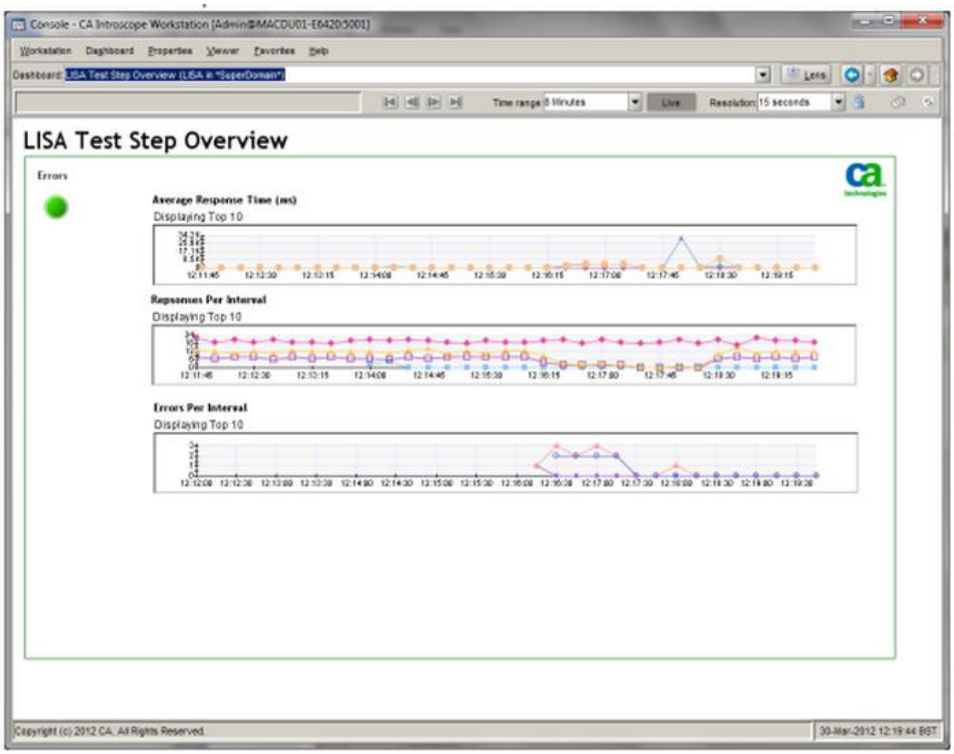


Test Step Overview **Dashboard**

The Test Step Overview dashboard shows graphs for the following metrics:

- Average Response Time (ms)
- Responses Per Interval
- Errors Per Interval

In each case, only the top ten metrics are shown.


Console - CA Introscope Workstation [Admin@MACDU01-E64205001]
Workstation
Dashboard
Properties
Viewer
Favorites
Help
Dashboard: LISA Test Step Overview (LISA in "SuperDomain")
Lens
Time range: 5 Minutes
Live
Resolution: 15 seconds
LISA Test Step Overview
Errors
Average Response Time (ms)
Displaying Top 10
Reponses Per Interval
Displaying Top 10
Errors Per Interval
Displaying Top 10
Copyright (c) 2012 CA. All Rights Reserved.

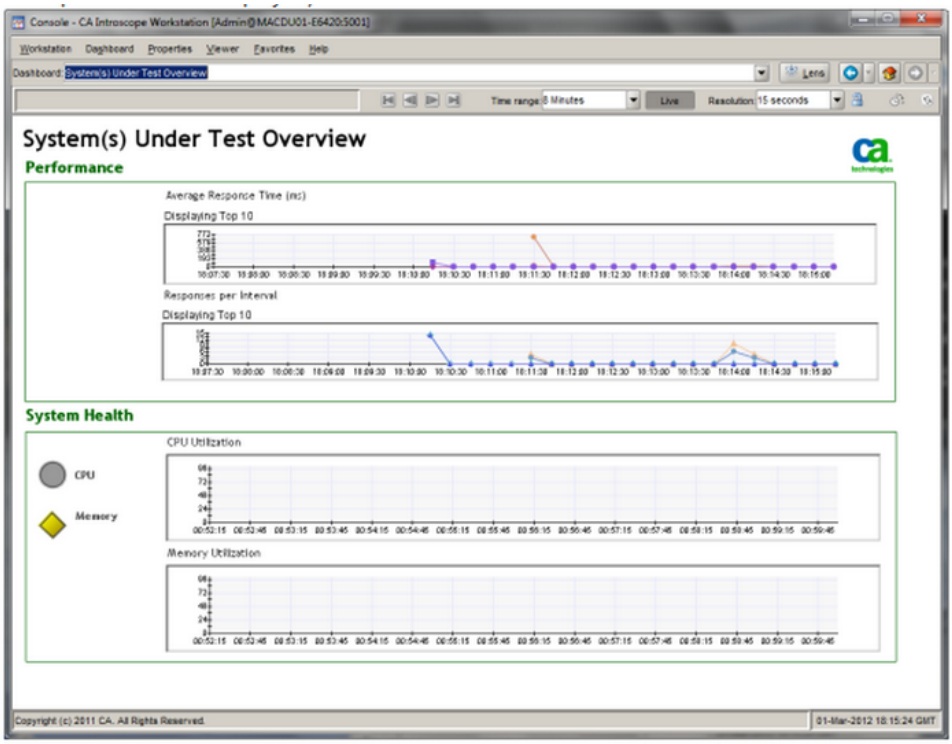
SUT Dashboard

The System(s) Under Test Overview gives a high-level overview of systems that are executing transactions that are sent from DevTest tests. Identifying the systems that DevTest is testing is difficult. You can edit the metric groupings to define which of the systems that Introscope instruments are also systems that DevTest is testing. By default, the grouping contains the demo server.

The System(s) Under Test dashboard displays metrics from systems under test by DevTest. To define which agents instrument systems under test, edit the Metric Grouping Agent Expression for all metric groups that are defined in the Management Module with names starting with "Systems Under Test". By default this expression is set to **(.)|(.)|JBoss LISA Demo Server(.*)**, which selects the demo server.

The dashboard is divided into two sections. The top section shows metrics on the top ten front-end Average Response Time (ms) and Responses Per Interval metrics from all application servers under test.

The bottom section shows CPU and memory utilization for all application servers under test. This section also contains two alert icons that indicate alerts relating to CPU and memory utilization.



Reporting

The management module contains a report named DevTest Report. This report contains a series of graphs describing the state of the systems under test by DevTest. The report also includes test metrics that DevTest gathers from the execution of those tests. You can duplicate and refine the report to focus it upon specific tests. To focus on specific tests, modify the duplicate metric grouping expressions to select only those metrics relevant to the specific report. You can also drop individual graphs from the duplicate report.

Set Up the SAP System Landscape Directory

This topic describes how to set up the SAP System Landscape Directory (SLD) for CA Service Virtualization.

Follow these steps:

1. Obtain the following information for the SLD server:
   - Host
   - Port
   - User
   - Password
2. Log on to the System Landscape Directory.

   Note: you must have a LcrInstanceWriterLD role to perform this task.
3. Perform one of the following actions:
   - Manually Register DevTest with SLD (see page 125)
   - Import the DevTest SLD XML File (see page 126)

Manually Register DevTest with SLD

This topic describes how to manually register DevTest with the SAP System Landscape Directory (SLD). This process creates a product name/version and software component name/version in the SLD software catalog.

Follow these steps:

1. In the SLD software catalog, select Products, New Product Version.
2. Enter **CA LISA SV** in the Product Name field.
3. Enter **CA Technologies** in the Vendor Name field.
4. Enter the version number; for example, **V7.5**, in the Product Version field.
5. Click Create.
6. Enter **CA LISA SV** in the Technical Name field.
7. Enter **CA LISA** in the Software Component Name field.
8. Enter the version number; for example, **V7.5**, in the Software Component Version field.
9. Enter **released** in the Production State field.
10. Click Create.

Import the DevTest SLD XML File

This topic describes how to set up the SAP System Landscape Directory (SLD) by importing the DevTest SLD XML file.

Follow these steps:

1. In the SLD software catalog, select Administration, Content, Import.
2. Enter the path of the provided SLD data zip file in the Selected File field.

   This file is located in **LISA_HOME\addons\sap\CALISA.xml**
3. Click Import Selected File.

After you install DevTest, refer to the following paths:

- Default installation path: **C:\lisa\bin**
- Default log file path: **C:\Users\<userID>\lisatmp_<version number>**

  For detailed information about each log file and its meaning, refer to Log File Overview in *Administering*.

  All log files are human readable.
- Default configuration file path: **C:\LISA_HOME**
- Default configuration files: **lisa.properties** and **local.properties**

  For detailed information about the order of properties files and setting values in the log files, refer to Property Files in *Using CA Application Test*.

  All property files are human readable java name=value entries.

Chapter 8: Setting Up the Mobile Testing Environment

This section contains the following topics:

System Requirements for Testing Mobile Applications

This section describes the system requirements for testing mobile applications in DevTest Solutions.

Supported Operating Systems for Mobile Testing

The following operating systems are supported:

Mac OSX 10.8 - Mountain Lion

VM Image is supported.

Mac OSX 10.9 - Mavericks

VM Image is supported.

Windows 7 or Later

Only remote iOS tests are supported. Local iOS tests are not supported.

Supported Mobile Operating Systems

The following mobile operating systems are supported:

- iOS 6.1 and later
- Android 4.2 and later
- Native, Hybrid, and Web

Note: SauceLabs Android support is limited to version 4.2.

Hardware Requirements for Mobile Testing

The following hardware is required to support mobile testing:

CPU

- Intel Core i5 2.0 GHZ or AMD equivalent

Memory

- Cloud or Remote Mobile Tests: 4GB
- Local Simulators: 8GB

Preinstallation Steps for Mobile Testing

Complete the following tasks before installing DevTest Solutions for mobile testing.

Preinstallation Steps for Mobile Testing-Macintosh

Follow these steps:

1. Install Xcode 6.1 (see page 129).

   Xcode is available free from the Mac App Store.

2. Download the latest Android SDK (ADT Bundle) from:

   http://developer.android.com/sdk/index.html#download

   Note: The minimum version that mobile testing supports is SDK Platform 19 rev 3. The ADT bundle contains Build-tools, and mobile testing requires at least version 19.0.1, 19.1.0, or 20.0.0. If these versions are not installed with your ADT bundle, and you encounter an error message in DevTest Workstation regarding zipalign or aapt, see Android SDK Build-tools (see page 134).

3. Define your ANDROID_HOME property (see page 132).
4. Set up your Android SDK (see page 133).

Note: Available mobile device simulators are dependent on your installed version of Xcode or SDK.

Install Xcode Version 6.1

Mobile Testing requires Xcode version 6.1 (later versions are not supported).

Download a New Version

Follow these steps:

1. Go to https://developer.apple.com/support/xcode.
2. Download and install Xcode 6.1.

Note: Running ios 7 tests from Xcode 6 is not supported. Xcode 5 must be installed to run iOS 7 tests.

IPA Signing

To test iOS apps on devices, you must get SSL certificates and provision your device for testing. This can be done at the Apple Developer Member Center.

You can then provision your device with the mobile provision file to identify the apps that are signed with a certificate. Adding the mobile provision file to an .ipa file (through IPA signing) ensures that the signatures on the certificate match the signatures of those who can run the apps on their devices.

Mobile Testing can sign iOS .ipa files before deploying them to your test device. Some configuration is required for it to function properly.

Follow these steps:

1. Log in to the Apple Developer Member Center.
2. Click Certificates, Identifiers, & Profiles.
3. Connect your iOS device with a USB cable.
4. Under iOS Apps (on the left), click Devices.
5. To add your device, click the + icon.

   Note: You can also add your device through iTunes.
6. Confirm that your device is included in the list of iOS devices.
7. Under Certificates, click Development.
8. Click the certificate in the list.
9. Click Download, and save it to your hard drive.
10. Install this certificate into Keychain Access.
11. Under Provisioning Profiles, click Development.
12. Select the iOS Team Provisioning Profile in the list.
13. Download this file and save it on your hard drive.

    The file name is similar to this example:
    iOS_Team_Provisioning_Profile_.mobileprovision.
14. Open DevTest.
15. In the lisa.properties property file, set the following project properties:

    `MOBILE_PROVISION = path to iOS_Team_Provisioning_Profile_.mobileprovision`

    `IOS_CERTIFICATE = name of the certificate as displayed in Keychain Access (not the certificate filename itself)`

Note: You do not need to type the exact string, as the codesign utility can match it. You can use a name such as "iPhone Developer" and that would be enough to be a unique name. However, if you have other iOS certificates, it is good practice to use the entire certificate name.

Enable UIAutomation

UIAutomation should be enabled in iOS 8 devices. Enabling this mode lets Appium detect instruments.

The UIAutomation option can be found under Settings, Developer, Enable UI Automation.

Preinstallation Steps for Mobile Testing-Windows

Follow these steps:

1. Download and install the latest Android SDK.

   http://developer.android.com/sdk/index.html#download

   Note: The minimum version DevTest supports is SDK Platform 19 rev 3. The ADT bundle contains Build-tools, and mobile testing requires at least version 19.0.1, 19.1.0, or 20.0.0. If these versions are not installed with your ADT bundle, and you encounter an error message in DevTest Workstation regarding zipalign or aapt, see Android SDK Build-tools (see page 134).

2. Define your ANDROID_HOME property (see page 132).
3. Set up your Android SDK (see page 133).

Define ANDROID_HOME

For your Android Virtual Devices (AVDs) to function properly, you must define an ANDROID_HOME property.

Follow these steps:

1. In DevTest Workstation, click System, Edit Properties.

   System Properties open.

2. Insert the following property:

   ```
   ANDROID_HOME= <The fully qualified path to your ADT Bundle>
   ```

   For example:

   ```
   ANDROID_HOME= C:\ADT
   Bundle\adt-bundle-windows-x86_64-20131030\sdk
   ```

3. Click Save.
4. Close the System Properties window.

Set Up the Android SDK

The DevTest Solutions installer includes an Android Developer Tools (ADT) Bundle that provides the tools that are used to develop Android applications. The ADT bundle includes a version of the Eclipse IDE with built-in ADT. Use the ADT Bundle to create one or more Android Virtual Devices (AVDs).

Follow these steps:

1. Download the latest Android SDK (ADT Bundle).

   http://developer.android.com/sdk/index.html#download

2. Unzip the ADT Bundle.
3. Run **eclipse.exe** in the eclipse folder of the ADT bundle.

   The Workspace Launcher opens.

4. Select the workspace for storing your projects and click OK.

   The Java ADT window opens.

5. Click Window, Android Virtual Device Manager.

   The Android Virtual Device Manager opens.

6. Click New.

   The Create New Android Virtual Device (AVD) page opens.

7. Complete the following fields:

   AVD Name

   Name of the Android Virtual Device that to create. Use this name in the Android AVD field when creating an Emulator Session Asset.

   Device

   Select the type of device to test.

   Target

   Select the target for your AVD.

   Memory Options

   Set the RAM value to 512 and the VM Heap value to 32.

8. Accept the default values for the remaining fields, or customize them for your AVD.
9. Click OK.

   Your new device is added to the list of devices in the Android Virtual Device Manager.

Android SDK Build-tools

The Android SDK bundle contains a component called Build-tools. Mobile testing requires at least version 19.0.1, 19.1.0, or 20.0.0.

If these versions are not installed with your ADT bundle, you can encounter an error message when creating a mobile asset in DevTest Workstation:

```
Cannot find the aapt command. Please install Android SDK Build-tools Rev 20.
```

Or

```
Cannot find the zipalign command. Please install Android SDK Build-tools Rev 20.
```

If you encounter this error message, update your Android SDK Build-tools version.

Follow these steps:

1. Run **android sdk** from the folder that contains the unzipped ADT bundle.

   The Android SDK Manager opens.

2. Under the Tools folder, ensure that 19.0.1, 19.1.0, or 20.0.0 is installed and selected. Version 20.0.0 is the optimal version to use.

Once you install the correct Build-tools version, you can continue creating your mobile asset.

Enable USB Debugging

USB Debugging Mode can be enabled in Android after connecting your device directly to a computer with a USB cable. Enabling this mode lets you facilitate a connection between an Android device the Android SDK.

USB Debugging Mode can be found in your device's Developer Options under the Settings menu.

Using Genymotion

Genymotion is another emulator that uses VirtualBox as an Android emulator. Genymotion is faster and provides overall better performance than the Android emulator provided with the Android Software Development Kit (ASDK). This tool requires additional setup.

Follow these steps:

1. Download and install both VirtualBox and Genymotion from www.genymotion.com.

   Note: Installing Genymotion requires you to create a Genymotion user account, but the basic download is free. The Windows platform has a combined Virtualbox/Genymotion installer.

2. Add your virtual devices.

   a. Start Genymotion.

      The first time you start Genymotion, you are prompted to add a virtual device.

   b. Click Yes.

      The Create a New Virtual Device window opens.

   c. Click Connect and enter your Genymotion user ID and password to view available devices.

      A list of available virtual devices displays. You can filter the list of devices by Android version or device model at the top of the page.

   d. Select the device you want to add, then click Next.

   e. Review the details for the selected device.

      Note: The name of the device is the name you enter in the asset dialog in DevTest Solutions. You can change the default Genymotion name to a simpler name.

   f. Click Next.

      The download of your virtual device begins. These download images are typically around 200MB in size.

   g. Click Finish when the download is complete.

   h. To add another virtual device, click Add in the Genymotion main window and repeat the previous steps.

3. Create a config file and asset in DevTest for your new virtual device.

   a. Create a standard Simulator Session asset for an Android virtual device. For more information, see Mobile Assets in *Using CA Application Test*.

      For more information, see Mobile Tutorial 2 - Record an Android Test Case in *Getting Started*.

b. In the Android AVD field, enter the name of your Genymotion device, preceded by **genymotion:**. For example:

```
genymotion:Galaxy Nexus - 4.3 - API 18 - 720x1280
```

Managing Genymotion Devices

You can manage your Genymotion devices with the VBoxManage tool that is installed with VirtualBox.

The following is a list of useful command-line commands:

vboxmanage list avd

Lists all available virtual devices. The output from this command resembles the following:

```
Nexus 7 - 4.3 - API 18 - 1280x800"
{144161dd-750e-4fff-8d46-0da8bc0c226b}
GalaxyNexus4.2.2-API17" {d740a4fa-df15-4768-aee1-ffebfb883dc1}
```

Each line is a pair of identifiers. The first identifier is the name you gave the device when you created it. The second identifier is the UID. You can use either identifier when passing devices as targets on the command line. The UID does not need the braces when you are using it on the command line.

vboxmanage list runningvms

Lists the virtual devices that are running.

You can perform the same function with the ADK command, **$ANDROID_HOME/platform-tools/adb devices.** The ADK command lists both Genymotion and ADK emulated devices.

The output from this command resembles the following:

```
emulator-5554   device
192.168.0.56:5555   device
```

ADK devices are prefixed with "emulator." Genymotion devices begin with the IP address. The next number is the port that the device is using.

player --vm-name <UID or name>

Navigate to the path where the device is installed, and use this command to start a virtual device. You can also start a device from the Genymotion interface.

Define VBOXMANAGE_CMD

If you install VirtualBox in a non-default location, define the VBOXMANAGE_CMD property. This property lets DevTest find the vboxmanage tool, so your Genymotion simulators function efficiently.

You can define this property in three ways:

- Define an environment variable in Windows
- Edit the lisa.properties file (to affect all projects)
- Add a property to the project property file in the Properties Editor (to affect a single project)

Define an Environment Variable in Windows

In the System Properties for your machine, open the Environment Variables dialog and add the following command:

Windows

```
VBOXMANAGE_CMD=c:\program
files\oracle\virtualbox\vboxmanage.exe
```

Mac

```
export VBOXMANAGE_CMD=/usr/bin/vboxmanage
```

Edit the LISA Property File

Editing the lisa.properties file affects all of the projects that are defined in DevTest Workstation.

Follow these steps:

1. Select System, Edit LISA Properties from the main menu.
2. To define the VBOXMANAGE_CMD in the system file, add the following text:

   ```
   VBOXMANAGE_CMD=c:\program
   files\oracle\virtualbox\vboxmanage.exe
   ```
3. Save the file.

Add a Property to the Project Property File

Adding a property to the property file in the Properties Editor affects only that project.

Follow these steps:

1. In the Project panel, double-click your **project** in the Configs folder.

The Properties Editor opens.

2. To add a row, click  Add at the bottom of the Properties Editor.
3. In the Key field, type:

   `VBOXMANAGE_CMD`
4. In the Value field, type:

   `c:\program files\oracle\virtualbox\vboxmanage.exe`
5. From the main toolbar, click 

Save.

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.*