LISA User Guide

LISA^{TM} is a complete and collaborative automated testing solution.

LISA provides complete test coverage, with the ability to invoke and verify the behavior of each component across the end-to-end application. LISA provides automated testability for all the components in the technology stack.

LISA also builds portable, executable test cases that are easy to extend, easy to chain into workflows with other tests, and simple to integrate with existing test repositories. LISA test cases are designed to be shared across different teams and environments, with the ability to easily attach prior results and artifacts to extend them, and the ability to readily execute with different underlying data.

Several components require parameters that must be obtained from people knowledgeable about the System Under Test (SUT). This information is identified throughout the guide. You will rarely be able to proceed with building and running the component without this information.

Getting Started: This section provides an overview of the registry, coordinator server, simulator server, LISA Workstation, the LISA Console, and command-line utilities. In addition, this section contains a series of tutorials that illustrate various features of the product.

Building Test Cases: This section contains information about the important building blocks of LISA test cases, such as properties, configurations, filters, assertions, and data sets.

Building Documents: This section describes details of building various documents such as staging documents and audit documents.

Working with Model Archives (MARs): The main deployment artifact in LISA is a type of file referred to as a Model Archive (MAR).

Running Test Cases and Suites: There are some utilities within the Workstation that facilitate running test cases, not as production tests, but more to validate and tune your test case. This section has information about staging and running individual tests and test suites, in the workstation environment and on LISA Server.

Cloud DevTest Labs: You can use use cloud-based infrastructure to provision development and test environments.

Continuous Validation Service (CVS): The Continuous Validation Service (CVS) lets you schedule tests and test suites to run on a regular basis over an extended time period.

Reports: There are a wealth of features related to the generation and capture of data for the purpose of reporting results.

Recorders and Test Generators: In addition to building test cases from the ground up, there are many tools to automate or semi-automate the creation of a test case. These tools range from recorders that can follow your actions through a system and produce a test case for you, to smart test generators that can build a test case for you based on some basic information. For example, if you have the Web Service Definition Language (WSDL) file from a web service, LISA can build a test case to test that web service.

Access Control (ACL): LISA provides role-based access control. This feature is also known as ACL.

Advanced Features: This section deals with advanced features and explains ways for Java developers to customize and extend LISA, and use it in ways that require knowledge of Java.

Appendix A - LISA Property File (lisa.properties): This appendix lists properties in the LISA property file.

Appendix B - Custom Property Files (local.properties, site.properties): This appendix lists properties included in the two custom property files.

Getting Started

This section provides an overview of the registry, coordinator server, simulator server, LISA Workstation, LISA Console, and command-line utilities.

This section also contains a series of tutorials that illustrate various features.

In this section, the following topics are covered:

Registry

The registry provides a central location for the registration of all LISA Server and LISA Workstation components.

The registry keeps track of the locations of any LISA runtime components and provides lookup to their locations for each registered component. The registry also provides the web consoles for server administration, reporting, CVS, and Pathfinder. The common JMS provider used for component-to-component communication is started and run in the registry process. The broker for LISA-deployed Java agents is also in the registry.

The fully qualified name of the registry is **tcp://*hostname-or-IP-address*:2010/*registry-name***. For example:

- **tcp://localhost:2010/Registry**
- **tcp://myserver:2010/Registry**
- **tcp://myserver.example.com:2010/Registry**
- **tcp://172.24.255.255:2010/Registry**

LISA Workstation includes the Registry Monitor, which lets you monitor the test cases, simulators, coordinators, and virtual environments for a test suite.

The registry is associated with at least one lab, named the Default lab. If you create a coordinator, simulator, or VSE server without specifying a lab, then the server belongs to the Default lab.

Starting the Registry

The procedure for starting the registry depends on whether access control (ACL) is enabled.

> ⚠ As of release 6.0.4, the second procedure is obsolete. You do not need to provide a user name and password when ACL is enabled.

To start the registry when ACL is not enabled

1. Do one of the following:
    - (Windows) Open a command prompt, navigate to the **LISA_HOME\bin** directory, and enter the following command:

      ```
      Registry
      ```

    - (Windows) Click **Start Menu > All Programs > LISA > Registry**.
    - (Windows) Double-click the **Registry.exe** file in the **LISA_HOME\bin** directory.
    - (UNIX) Open a terminal window, navigate to the **LISA_HOME/bin** directory, and enter the following command:

      ```
      ./Registry
      ```

2. Wait until the following message appears:

   ```
   LISA Registry Ready.
   ```

To start the registry when ACL is enabled

1. Do one of the following:
    - (Windows) Open a command prompt, navigate to the **LISA_HOME\bin** directory, and enter the following command:

      ```
      Registry -u username -p password
      ```

    - (UNIX) Open a terminal window, navigate to the **LISA_HOME/bin** directory, and enter the following command:

```
./Registry -u username -p password
```

2. Wait until the following message appears:

```
LISA Registry Ready.
```

Creating a Named Registry

To create a named registry, run the registry executable with the **-n** option:

```
LISA_HOME/bin/Registry -n RegistryName
```

The following examples create a registry named **registry1.**

This example is based on a Windows installation:

```
cd C:\Lisa\bin
Registry -n registry1
```

This example is based on a UNIX installation:

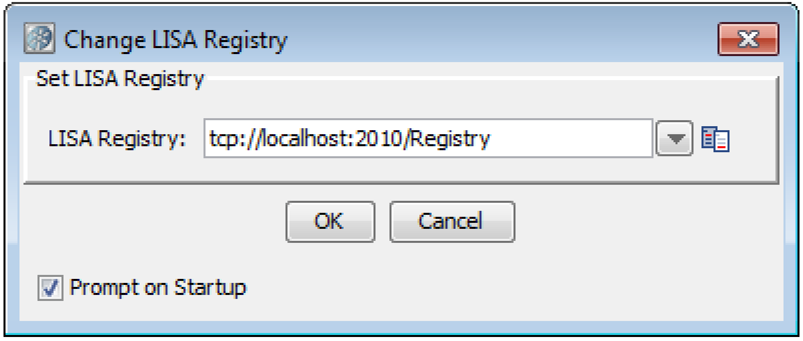
```
cd Lisa/bin
./Registry -n registry1
```

Changing the Registry

While you are working in LISA Workstation, you can switch to another registry.

To change the registry

1. From the main menu, select System > Registry > Change LISA Registry.
   The Change LISA Registry dialog appears.



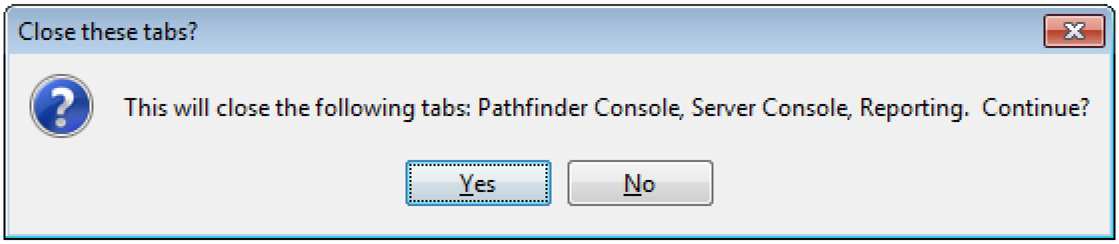
Change LISA Registry

Set LISA Registry

LISA Registry: tcp://localhost:2010/Registry

☑ Prompt on Startup

2. Enter the registry name, or open the drop-down list and select a previously used registry.
3. Select or clear the Prompt on Startup check box. If the check box is selected, the Set LISA Registry dialog opens when you start LISA Workstation. If the check box is cleared, LISA Workstation will connect to the last connected registry on start up.
4. Click OK.
   If one or more consoles are open within LISA Workstation, a dialog indicates that the consoles will be closed.


Close these tabs?

This will close the following tabs: Pathfinder Console, Server Console, Reporting. Continue?

5. Click Yes.

Coordinator Server

Coordinator servers receive the test run information in the form of documents, and coordinate the tests that are run on one or more simulator servers.

The coordinator server manages metric collection and reporting, and communicates the test data to LISA Workstation for monitoring purposes. A LISA server environment can have, and commonly does have, more than one coordinator server.

The coordinator server runs LISA tests when presented with a staging document, test case, default configuration and alternatively an alternate config.

The default name of a coordinator server is set by the **lisa.coordName** property.

```
lisa.coordName=Coordinator
```

The fully qualified name of a coordinator server is **tcp://*hostname-or-IP-address*:2011/*coordinator-name***.

Creating Coordinator Servers

To create a coordinator server, run the following command:

```
LISA_HOME/bin/CoordinatorServer -n CoordinatorServerName -m RegistryName
```

The following examples create a coordinator server named **coordinator1**.

This example is based on a Windows installation:

```
cd C:\Lisa\bin
CoordinatorServer -n coordinator1 -m tcp://localhost:2010/registry1
```

This example is based on a UNIX installation:

```
cd Lisa/bin
./CoordinatorServer -n coordinator1 -m tcp://localhost:2010/registry1
```

You can add the coordinator to a named lab by using the **-l** option. If you do not specify the **-l** option, then the coordinator is added to the Default lab.

If access control (ACL) is enabled, use the **-u** and **-p** options to specify your user name and password.

⚠ As of release 6.0.4, the **-u** and **-p** options are obsolete. You do not need to provide a user name and password when ACL is enabled.

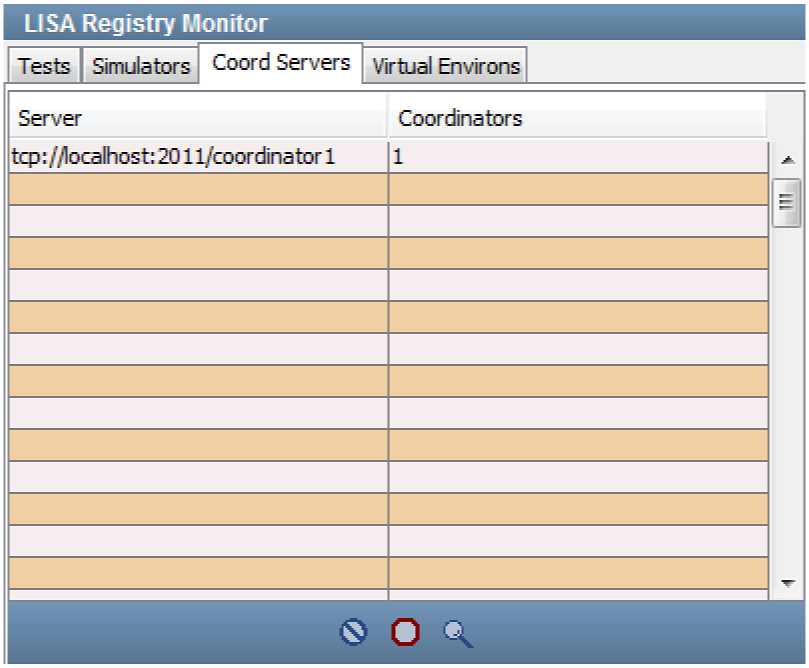
You can display the version number by using the **--version** option.

For information about running the coordinator server as a service, see Running Server Components as Services.

Monitoring Coordinator Servers

If LISA Workstation is attached to the associated registry, then a running coordinator server appears in the LISA Registry Monitor.

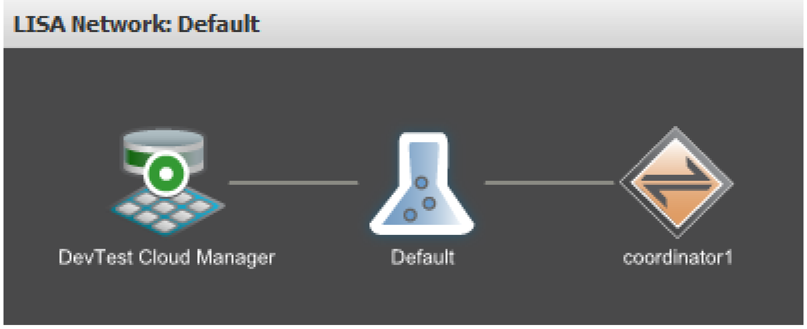
The following image shows the Coord Servers tab of the LISA Registry Monitor.



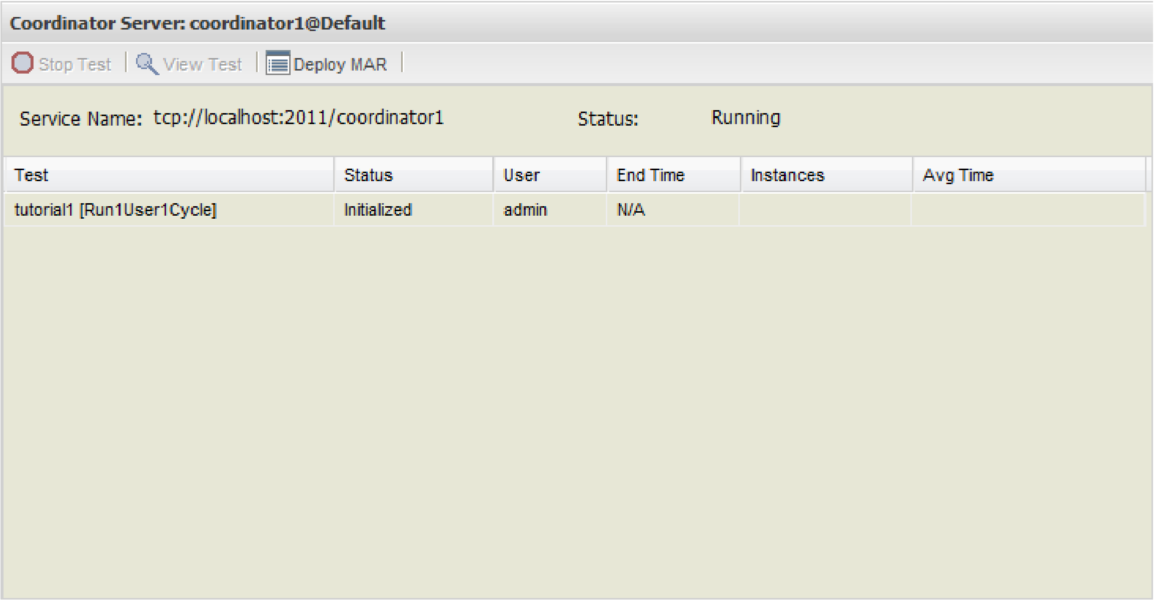
Server	Coordinators
tcp://localhost:2011/coordinator1	1

The coordinator server also appears in the network graph of the Server Console.

The following image shows an example of the network graph. The Default lab contains one coordinator.



If you click the coordinator server in the network graph, then a details window appears.



Service Name: tcp://localhost:2011/coordinator1 Status: Running

Test	Status	User	End Time	Instances	Avg Time
tutorial1 [Run1User1Cycle]	Initialized	admin	N/A		

Simulator Server

Simulator servers run the tests under the supervision of the coordinator server.

Virtual users or test instances are created and run on the simulator servers. The number of virtual users, and hence the number of simulator servers deployed, depend on the nature of the tests being performed. Each virtual user is in communication with the client system.

For large tests with many virtual users, virtual users can be distributed among several simulator servers.

The default name of a simulator server is set by the **lisa.simulatorName** property.

```
lisa.simulatorName=Simulator
```

The fully qualified name of a simulator server is **tcp://*hostname-or-IP-address*:2014/*simulator-name***.

Creating Simulator Servers

To create a simulator, run the following command:

```
LISA_HOME/bin/Simulator -n SimulatorName -m RegistryName
```

The following examples create a simulator named **simulator1**.

This example is based on a Windows installation:

```
cd C:\Lisa\bin
Simulator -n simulator1 -m tcp://localhost:2010/registry1
```

This example is based on a UNIX installation:

```
cd Lisa/bin
./Simulator -n simulator1 -m tcp://localhost:2010/registry1
```

You can add the simulator to a named lab by using the **-l** option. If you do not specify the **-l** option, then the simulator is added to the Default lab.

You can specify the number of virtual users for this simulator by using the **-i** option. For example:

```
Simulator -n simulator1 -m tcp://localhost:2010/registry1 -i 100
```

If access control (ACL) is enabled, use the **-u** and **-p** options to specify your user name and password.

⚠ As of release 6.0.4, the **-u** and **-p** options are obsolete. You do not need to provide a user name and password when ACL is enabled.

You can display the version number by using the **--version** option.

When creating a simulator, you can override the default port number by adding a colon and the non-default port number to the **-n** option. For example:

```
Simulator -n testSim1:35001
```

To run multiple simulators, use the command prompt to create the simulators as shown earlier.

Simulators will try port 2014 first if the port number is not specified as part of the name. If 2014 is taken, it will try 2015, 2016, and so on, up until port 2024 before giving up.

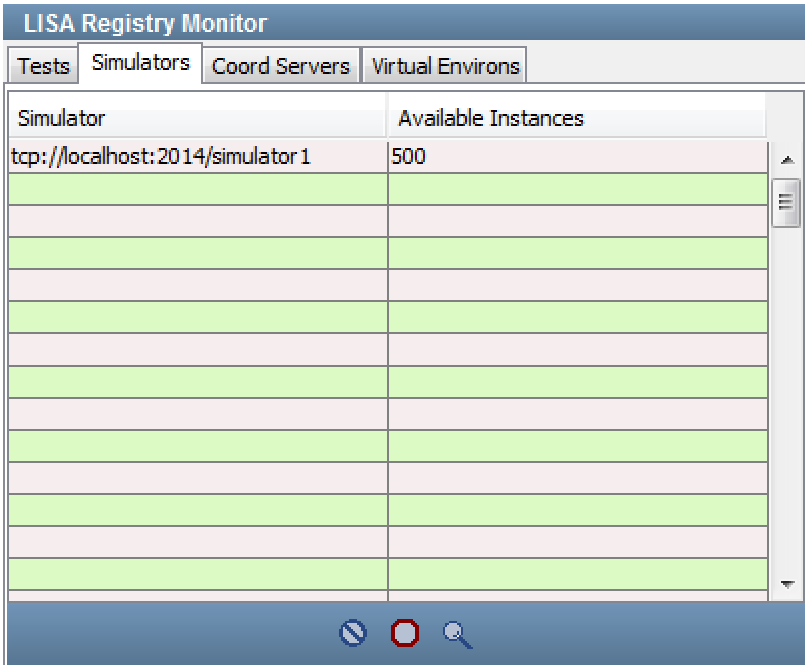
For more information about port usage, see Default Port Numbers.

For information about running the simulator as a service, see Running Server Components as Services.

Monitoring Simulator Servers

If LISA Workstation is attached to the associated registry, then a running simulator appears in the LISA Registry Monitor.

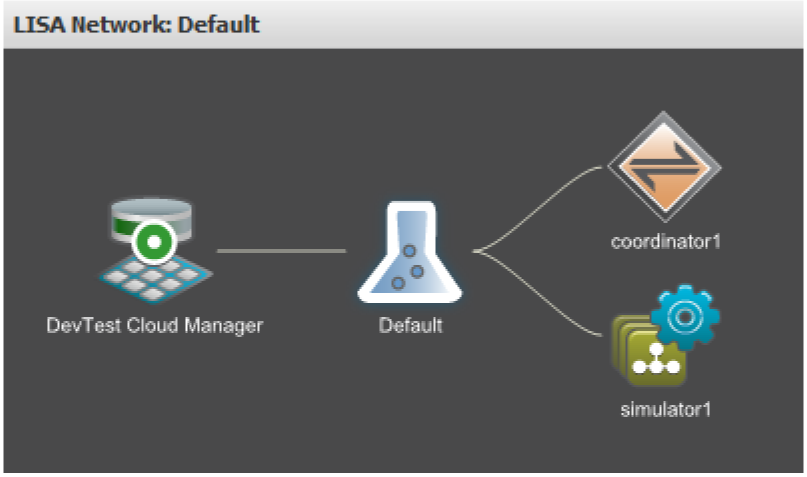
The following image shows the Simulators tab of the LISA Registry Monitor.



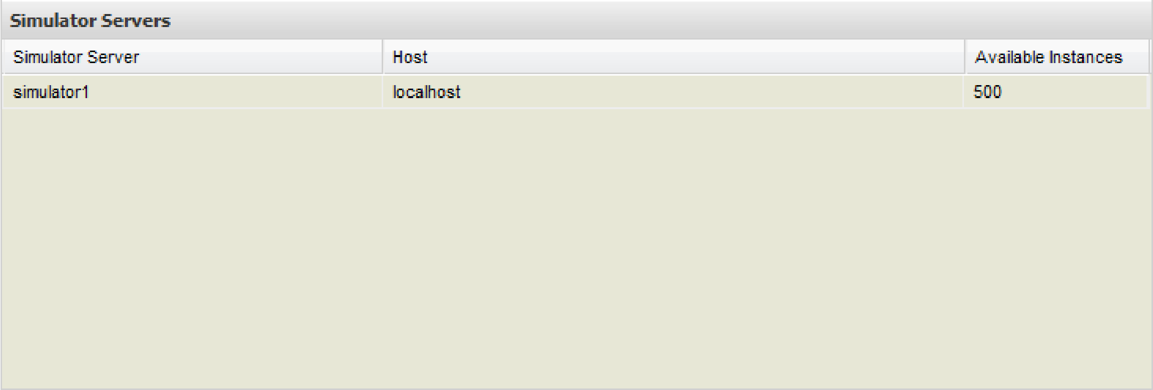
Simulator	Available Instances
tcp://localhost:2014/simulator1	500

The simulator also appears in the network graph of the Server Console.

The following image shows an example of the network graph. The Default lab contains one coordinator and one simulator.



If you click the simulator in the network graph, then a details window appears.



Simulator Servers

Simulator Server	Host	Available Instances
simulator1	localhost	500

LISA Workstation

LISA Workstation is a single, easy-to-use, integrated environment for developing, staging, and monitoring tests.

You can work in a LISA Workstation version or in a LISA Server environment.

- In LISA Workstation, the tests are managed and run within the Workstation environment. LISA Workstation is a test client used by QA/QE, development, and business analysis teams to test rich browser and web user interfaces, and the building blocks below the user interface. LISA Workstation is used to build and stage, all in a code-less manner: unit, functional, integration, regression, and business process testing. It requires a LISA registry to run. Tests are managed and run within the LISA Workstation environment (test authoring IDE), which runs embedded coordinator and simulator servers.

- In LISA Server, the tests are also managed and run within the Workstation environment. The workstation then connects to the server to deploy and monitor tests that were developed in LISA Workstation.

The server-side engine for LISA test cases and virtual services (test and virtual service model authoring IDE) is used to manage, schedule, and orchestrate LISA test cases for unit, functional, load, and performance tests on a continuous basis.

Opening LISA Workstation

When you open LISA Workstation, you are prompted to specify a registry.

If your computer has an installation of LISA Server, then you can use either of the following:

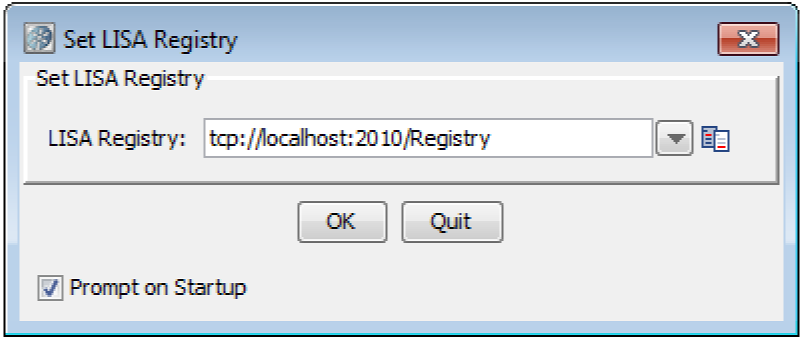
- A registry that is running on your local computer
- A registry that is running on a remote computer

If your computer has an installation of LISA Workstation, then you must use a registry that is running on a remote computer.

For information about how to specify the registry when SSL is enabled, see Using SSL to Secure Communication Between Components.

To open LISA Workstation

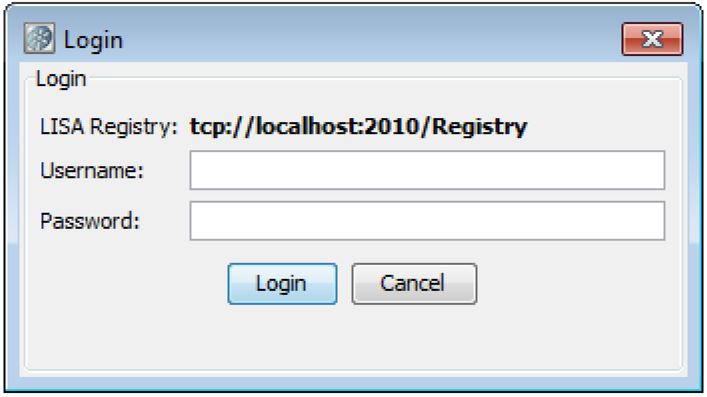
1. Do one of the following:
   - Open a command prompt, go to the **LISA_HOME/bin** directory, and run the LISA Workstation executable.
   - If you have a LISA application icon on your desktop, double-click the application icon.
   - (Windows) Click **Start Menu > All Programs > LISA > LISAWorkstation**.
     The Set LISA Registry dialog appears.



LISA Registry: tcp://localhost:2010/Registry

☑ Prompt on Startup

2. Accept the default registry, or specify a different registry.
3. Click OK.
   If access control (ACL) is enabled, then the Login dialog appears.



LISA Registry: **tcp://localhost:2010/Registry**

Username:

Password:

4. Enter your user name and password, and click Login.

Main Menu

The main menu includes menu options for all the major functions available in LISA Workstation. The main menu options available are dynamic to the selections at times. They are covered in detail here for the most common choices available.

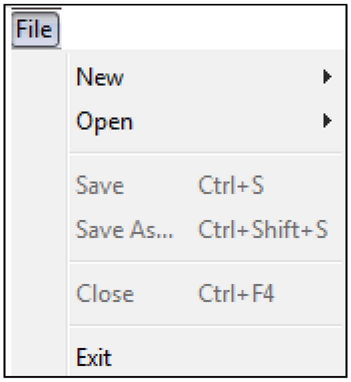
There are some varying menus and also drop-down menus and toolbars available throughout LISA Workstation.

The items on this menu are dynamic. The items available can be controlled by your LISA administrator as part of your security profile.

File Menu

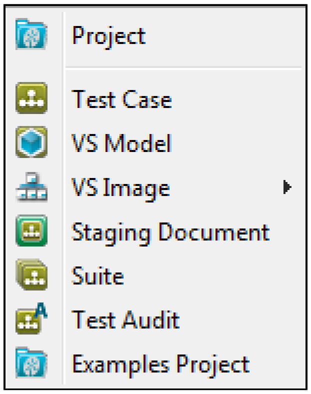
> The items on this menu are dynamic. The items available can be controlled by your LISA administrator as part of your security profile.

The File menu has the standard items related to file manipulation.



File > New

File > New shows a secondary menu where you can create a LISA project or create one of the test documents: test case, VS model, VS image, staging document, suite or test audit.

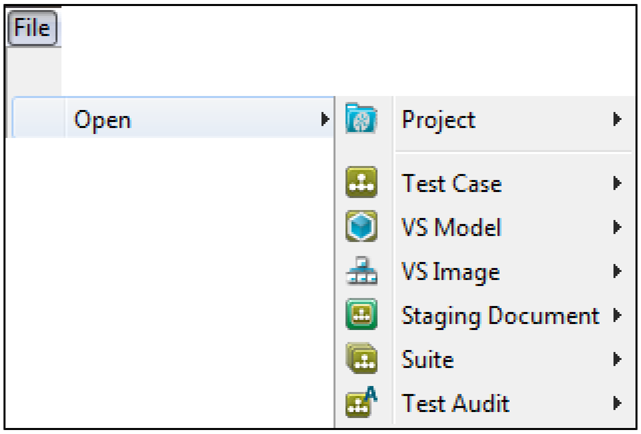


From this menu, you can create LISA documents within the project that is open in LISA Workstation or outside of the project. By default, the current directory selection is the **Tests** folder of the open project.

- **Project**: Creates a new project. For more information see Creating A New Project.
- **Test Case**: For additional information see Creating Test Cases.
- **VS Model**: ***Requires an additional license***. For additional information see the *LISA Virtualize Guide*.
- **VS Image**: ***Requires an additional license***. For additional information see the *LISA Virtualize Guide*.
- **Staging Document**: For additional information see Building Staging Documents.
- **Suite:** For additional information see Building Test Suites.
- **Test Audit:** For additional information see Building Audit Documents.
- **Examples Project**: Creates a new examples project. Specify a location for the new project in the dialog, click OK, and the LISA_HOME/examples contents will be copied into a new project.

File > Open

File > Open shows a secondary menu where you can open either an existing LISA project or one of the six major LISA documents: test case, VS model, VS image, staging document, suite or test audit.



You can browse for the document to open by looking in the file system, on the LISA classpath, or as a URL. LISA keeps a record of the most recent documents you have opened, and lists them on the Quick Start Window - Open Recent.

See File > New for additional information on the choices available, as they are the same as File > Open.

You can open any LISA document either from within a LISA project or from outside a project by using the main menu File > Open > menu option. If you select a document from a project that is different from the currently open project, the current project will close, and the project that the selected document resides in will open, then the document will open.

File > Save

File > Save: Saves the current document.

File > Save As

File > Save As...: Saves the current document under a different name.

File > Close

File > Close: Closes the active tab.

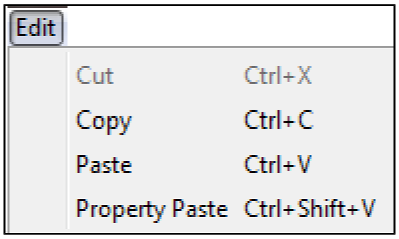
File > Exit

File > Exit: Exits LISA Workstation.

Edit Menu

⚠ The items on this menu are dynamic. The items available can be controlled by your LISA administrator as part of your security profile.

The Edit menu has the normal set of editing options.



Edit > Cut: Cuts the selected text.

Edit > Copy: Copies the selected text.

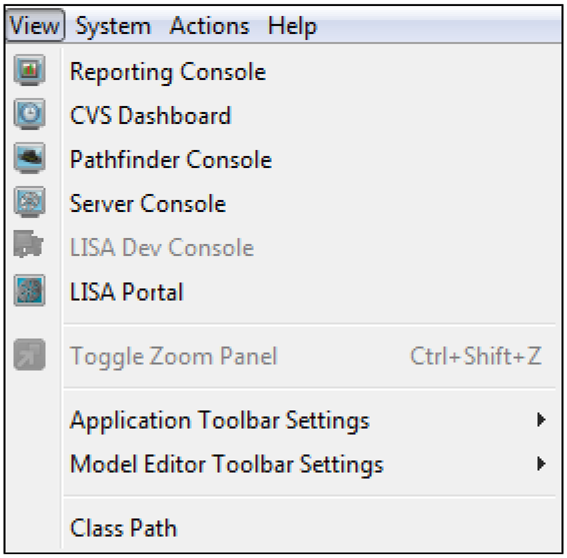
Edit > Paste: Pastes the selected text in the selected area.

Edit > Property Paste: Pastes the selected text in the selected area, and surrounds the pasted text with a set of double curly braces.

View Menu

⚠ The items on this menu are dynamic. The items available can be controlled by your LISA administrator as part of your security profile.

The View menu contains a list of menus that deal with the viewing of the Reporting Console, CVS Dashboard, Pathfinder Console, Server Console, LISA Dev Console, and others. You can also set the look and feel of LISA Workstation here and visit the LISA portal for more information related to LISA.

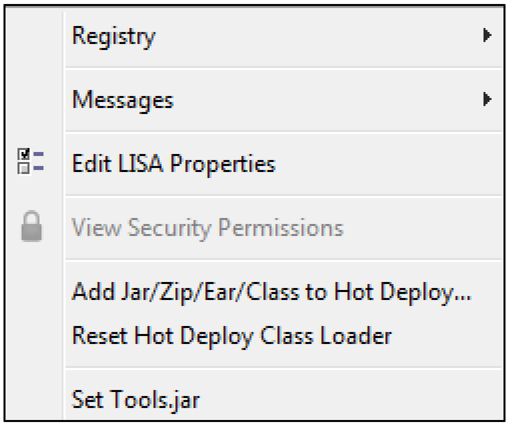


- **Reporting Console**: For additional information see Reporting Console.
- **CVS Dashboard**: For additional information see CVS Dashboard.
- **Pathfinder Console**: Invokes the Pathfinder Console. For more information, see the *Pathfinder Guide*.
- **Server Console**: Provides a visual representation of the LISA network, starting with the registry, showing the connected coordinators and their simulators, and VSE. The VSE Dashboard is accessed through the Server Console.
- **Dev Console**: Lets you view detailed information about the workings of the LISA Agent. It also gives you the opportunity to add extensions.
- **LISA Portal**: Invokes the LISA Console.
- **Toogle Zoom Panel**: Lets you toggle the Zoom panel.
- **Application Toolbar Settings:** Lets you choose the interface of the Application Toolbar. You can select the toolbar settings to display Icons and Labels, Icons only, Small Icons or no toolbar at all.
- **Model Editor Toolbar Settings:** Lets you choose between large icons or smaller icons to be displayed on the LISA Model Editor menu.
- **Class Path**: Lets you view and set the classpath.

System Menu

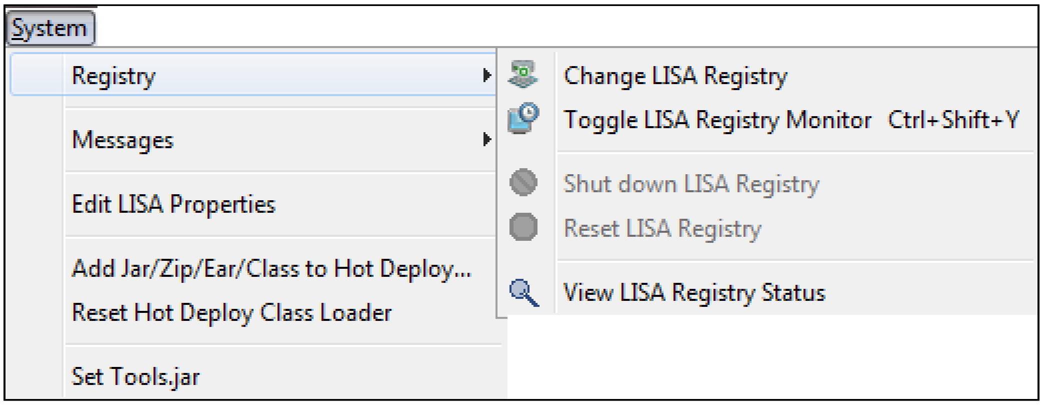
⚠ The items on this menu are dynamic. The items available can be controlled by your LISA administrator as part of your security profile.

The System menu has a menu to manage the system registries and system messages. You can also edit the LISA properties, add JAR or zip files to Hot Deploy or reset Hot Deploy Class Loader and set the tools.jar file here.


Registry
Messages
Edit LISA Properties
View Security Permissions
Add Jar/Zip/Ear/Class to Hot Deploy...
Reset Hot Deploy Class Loader
Set Tools.jar

System > Registry

- **Registry**: You can do all activities related to the LISA registry in this menu.

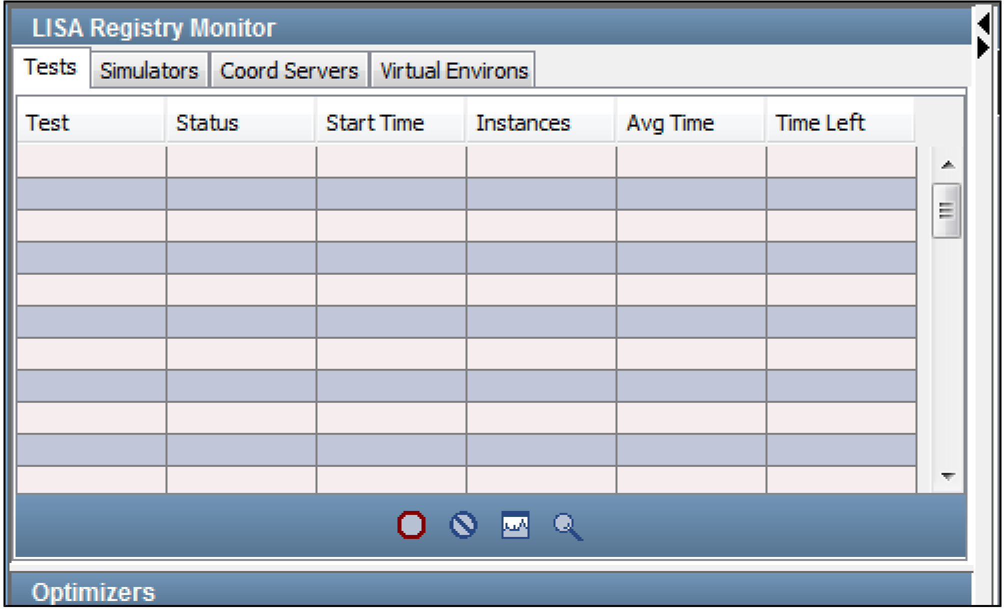

System
Registry
Messages
Edit LISA Properties
Add Jar/Zip/Ear/Class to Hot Deploy...
Reset Hot Deploy Class Loader
Set Tools.jar
Change LISA Registry
Toggle LISA Registry Monitor Ctrl+Shift+Y
Shut down LISA Registry
Reset LISA Registry
View LISA Registry Status

System > Registry > Change LISA Registry

- **Change LISA Registry:** See Registry for more information about this menu.

System > Registry > Toggle LISA Registry Monitor

- **Toggle LISA Registry Monitor**: You can toggle the LISA Registry Monitor, so that it can be viewed in LISA Workstation.



System > Registry > Shut Down LISA Registry

- **Shut Down LISA Registry**: Shuts down the LISA registry.

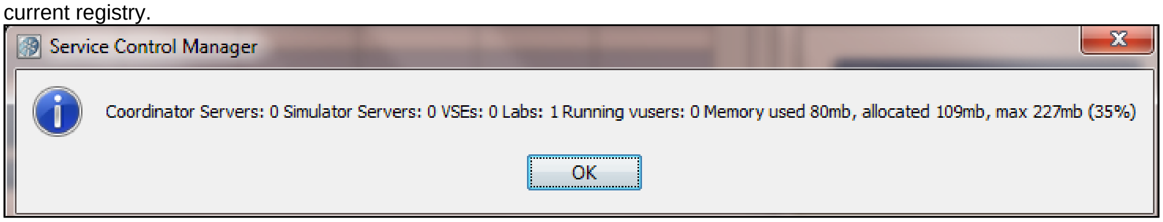
> ⚠ A LISA registry is mandatory for LISA Workstation to remain open. You will be prompted to select a registry if the registry is closed.

System > Registry > Reset LISA Registry

- **Reset LISA Registry**: Resets the LISA registry, which will reset the registry information in the system.

System > Registry > View LISA Registry Status

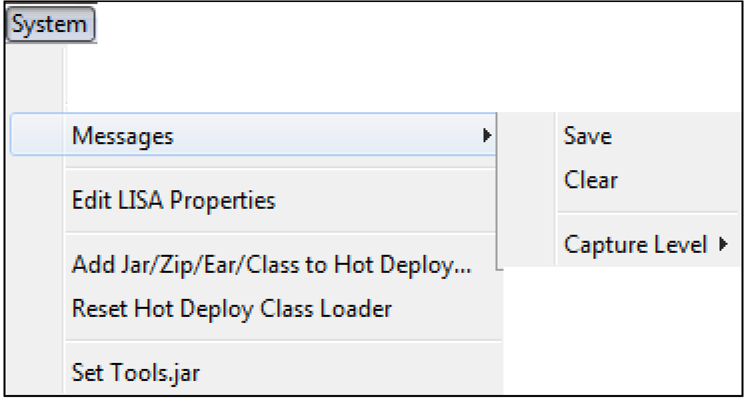
- **View LISA Registry Status:** Displays a dialog listing the number of coordinator servers and simulator servers that are attached to the current registry.



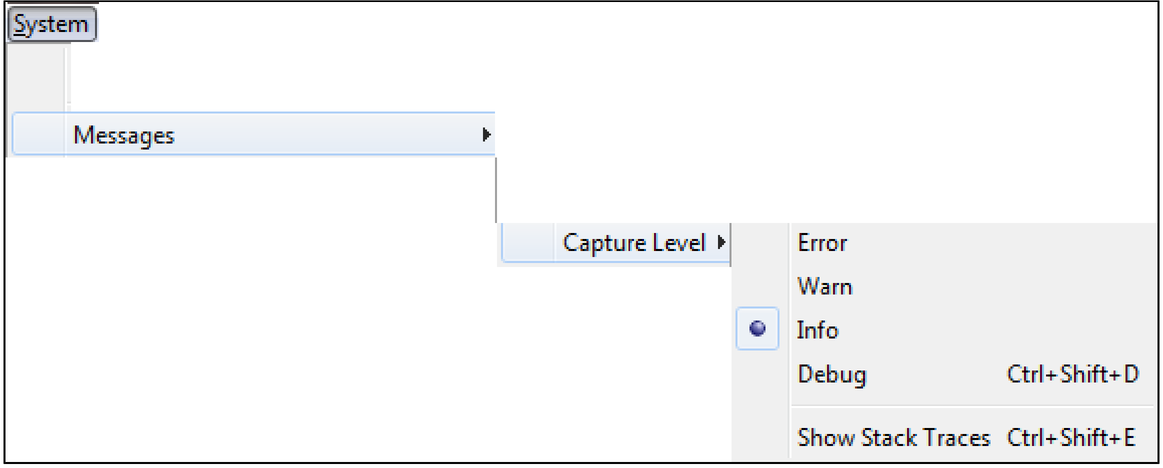
Coordinator Servers: 0 Simulator Servers: 0 VSEs: 0 Labs: 1 Running vusers: 0 Memory used 80mb, allocated 109mb, max 227mb (35%)

System > Messages

- **Messages**: This menu is for the System Message settings.

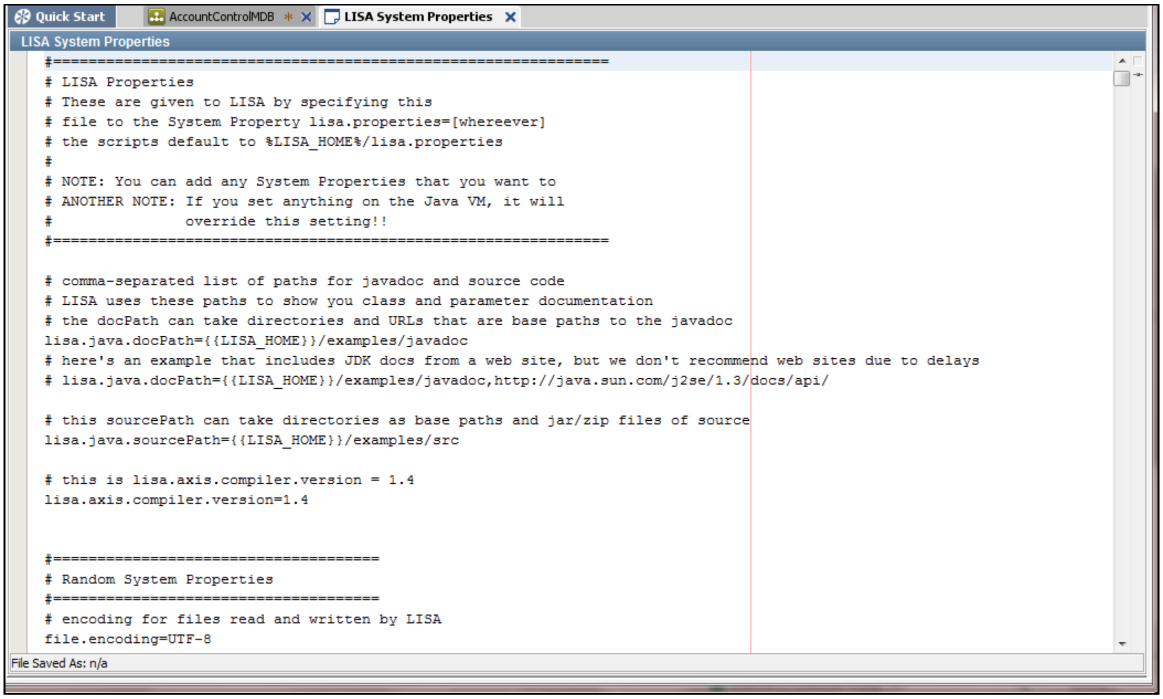


- **Save**: Save the system messages.
- **Clear**: Clear the system messages.
- **Capture Level**: Select the type of system messages to view: Error, Warn, Info (selected by default), Debug, or Show Stack Traces.



System > Edit LISA Properties

- **Edit LISA Properties:** Opens the **lisa.properties** file in an editable window.



```
#===============================================================
# LISA Properties
# These are given to LISA by specifying this
# file to the System Property lisa.properties=[whereever]
# the scripts default to %LISA_HOME%/lisa.properties
#
# NOTE: You can add any System Properties that you want to
# ANOTHER NOTE: If you set anything on the Java VM, it will
#                override this setting!!
#===============================================================

# comma-separated list of paths for javadoc and source code
# LISA uses these paths to show you class and parameter documentation
# the docPath can take directories and URLs that are base paths to the javadoc
lisa.java.docPath={{LISA_HOME}}/examples/javadoc
# here's an example that includes JDK docs from a web site, but we don't recommend web sites due to delays
# lisa.java.docPath={{LISA_HOME}}/examples/javadoc,http://java.sun.com/j2se/1.3/docs/api/

# this sourcePath can take directories as base paths and jar/zip files of source
lisa.java.sourcePath={{LISA_HOME}}/examples/src

# this is lisa.axis.compiler.version = 1.4
lisa.axis.compiler.version=1.4


#=====================================
# Random System Properties
#=====================================
# encoding for files read and written by LISA
file.encoding=UTF-8
```

File Saved As: n/a

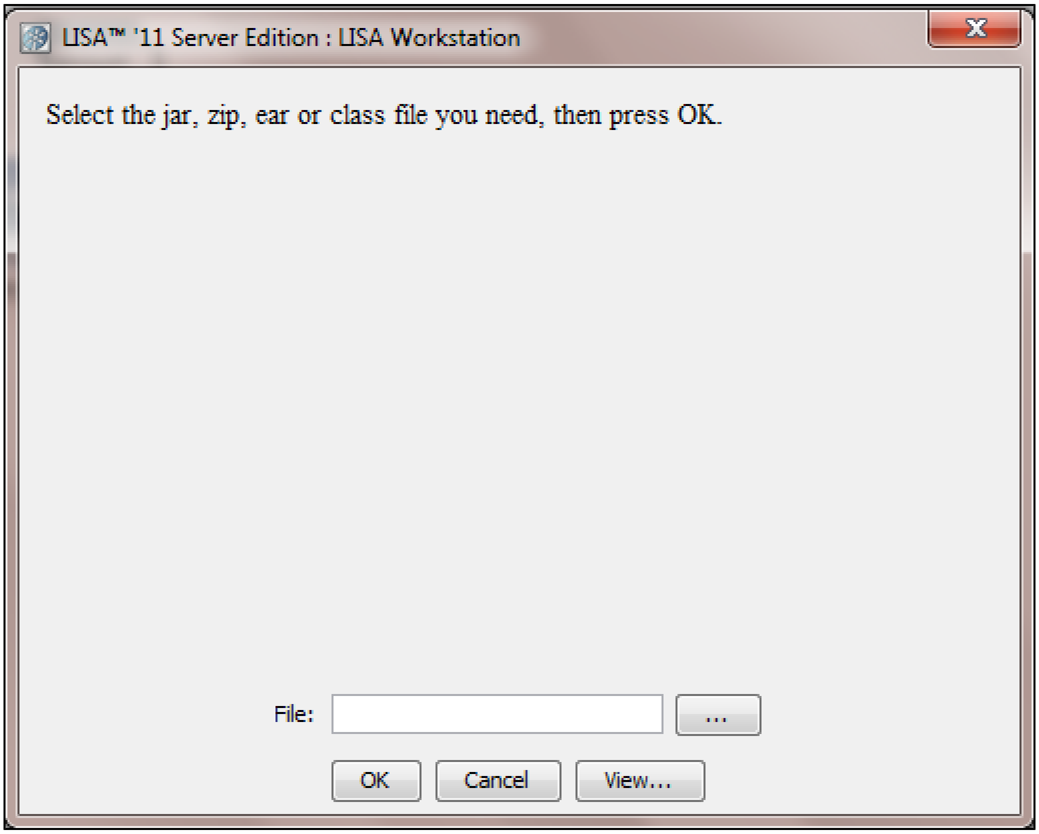
⚠ Use extreme caution if you choose to edit this file.

System > View Security Permissions

- **View Security Permissions**: If access control (ACL) is enabled, displays the unique ID of your user name, roles that are assigned to you, and permissions that you have.

System > Add Jar/Zip/Ear to Hot Deploy

- **Add Jar/Zip/Ear to Hot Deploy...**: Displays a dialog to enter the name of a file to upload into the hot deploy directory. The file you upload contains items that you want to add to LISA's classpath.

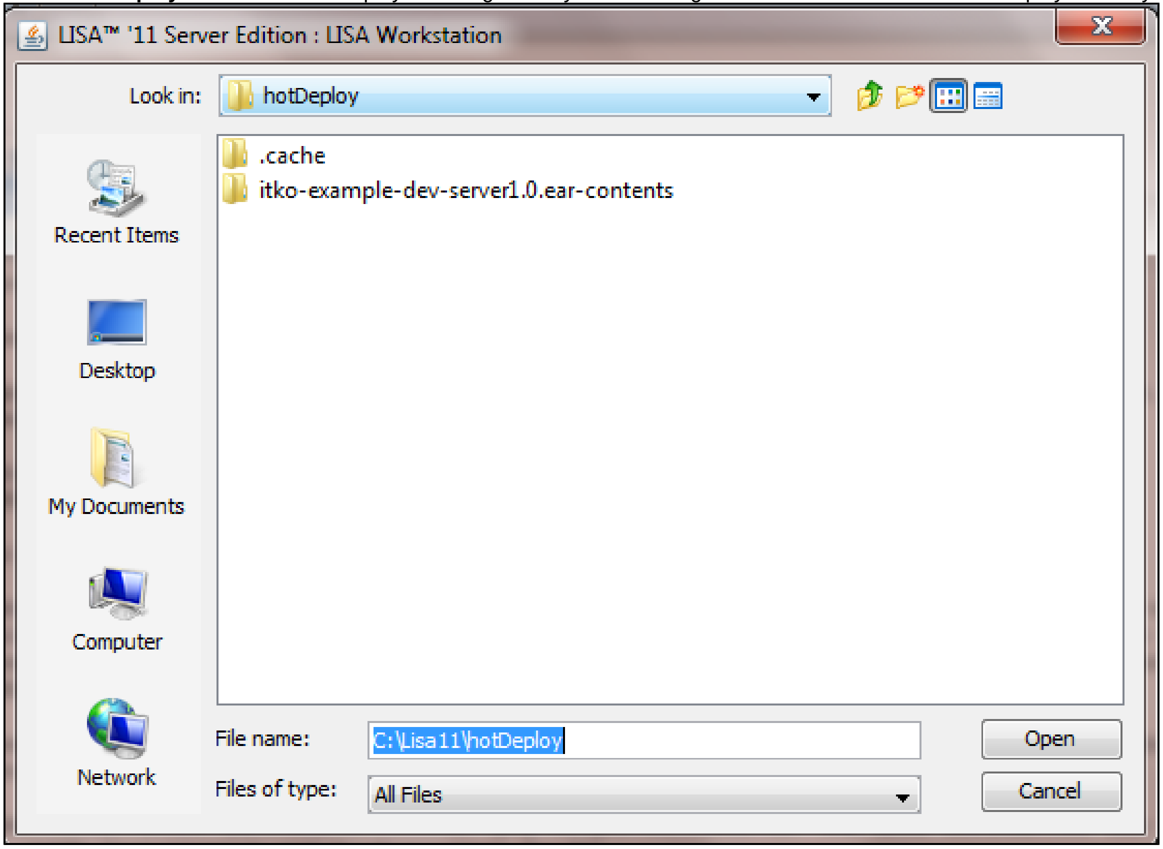


LISA™ '11 Server Edition : LISA Workstation
Select the jar, zip, ear or class file you need, then press OK.
File:
...
OK
Cancel
View...

For more information see the *Installation and Configuration Guide*.LISA Installation and Configuration Guide

System > Reset Hot Deploy Class Loader

- **Reset Hot Deploy Class Loader:** Displays a dialog where you can change the location of the current hot deploy directory.



LISA™ '11 Server Edition : LISA Workstation
Look in:
hotDeploy
Recent Items
Desktop
My Documents
Computer
Network
.cache
itko-example-dev-server1.0.ear-contents
Open
Cancel

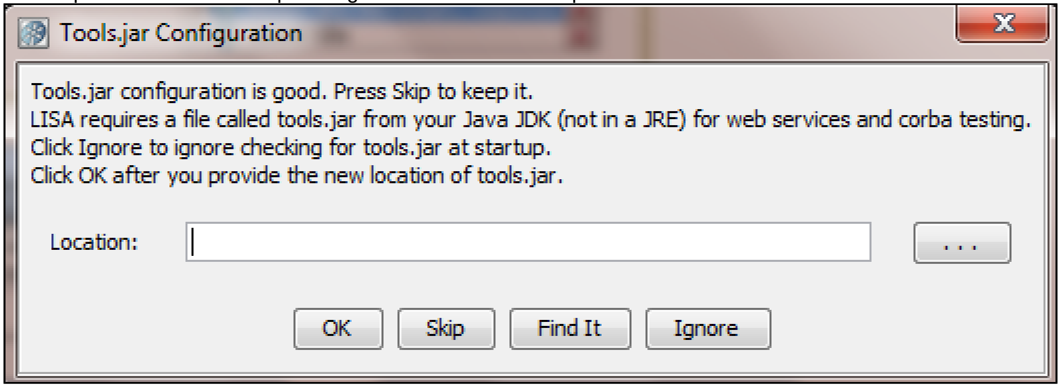
File name: C:\Lisa11\hotDeploy

Files of type: All Files

For more information see the *Installation and Configuration Guide*.

System > Set Tools.jar

- **Set Tools.jar:** Displays a dialog where you can enter, or browse to the location of a Tools.jar file. This file is required Java code needs to be compiled. Several test steps and generators need to compile code.



Tools.jar configuration is good. Press Skip to keep it.
LISA requires a file called tools.jar from your Java JDK (not in a JRE) for web services and corba testing.
Click Ignore to ignore checking for tools.jar at startup.
Click OK after you provide the new location of tools.jar.

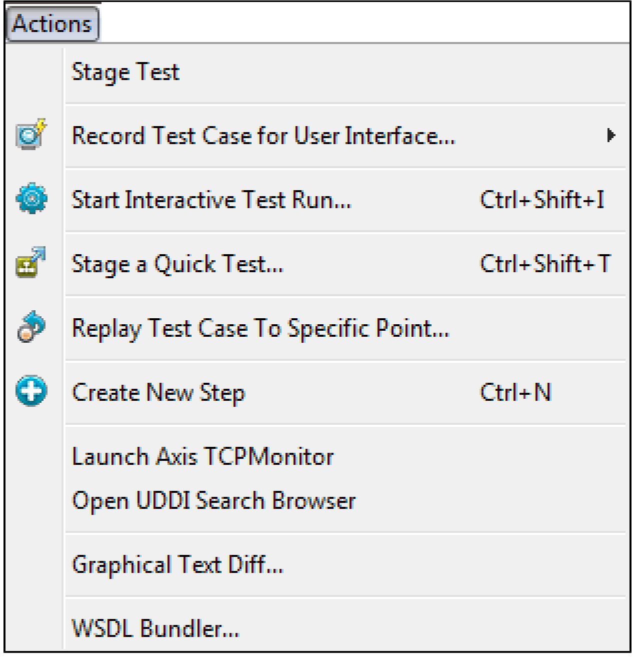
⚠ If a Tools.jar file is available, the dialog will indicate that.

For more information see the *Installation and Configuration Guide*.

Actions Menu

⚠ The items on this menu are dynamic. The items available can be controlled by your LISA administrator as part of your security profile.

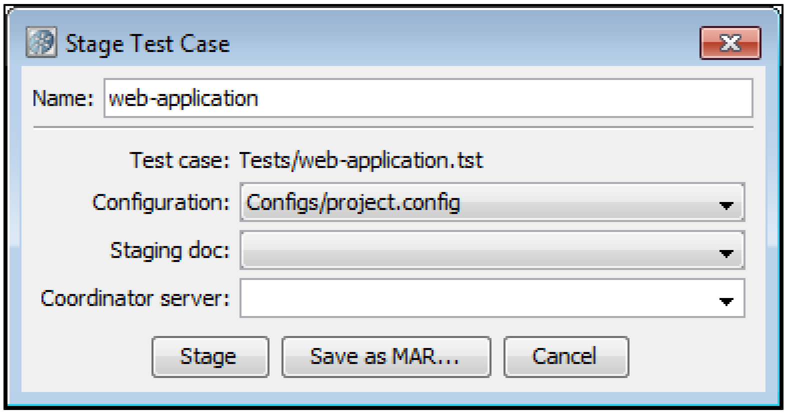
The Actions menu has items related to staging test cases, running suites, recording test cases, staging quick tests, adding test steps, and launching various tools. You will not see the full Actions menu as shown below unless you have a test case open.



Actions
Stage Test
Record Test Case for User Interface...
Start Interactive Test Run... Ctrl+Shift+I
Stage a Quick Test... Ctrl+Shift+T
Replay Test Case To Specific Point...
Create New Step Ctrl+N
Launch Axis TCPMonitor
Open UDDI Search Browser
Graphical Text Diff...
WSDL Bundler...

Actions > Stage Test

- **Stage Test:** This item is available for test cases. Opens a dialog where you can start the test case execution.



Stage Test Case

Name: web-application

Test case: Tests/web-application.tst

Configuration: Configs/project.config

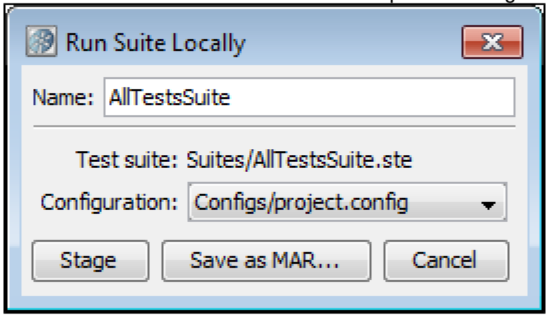
Staging doc:

Coordinator server:

For more information, see Staging Test Cases.

Actions > Run

- **Run:** This item is available for suites. It opens a dialog where you can configure and stage the suite.



Run Suite Locally

Name: AllTestsSuite

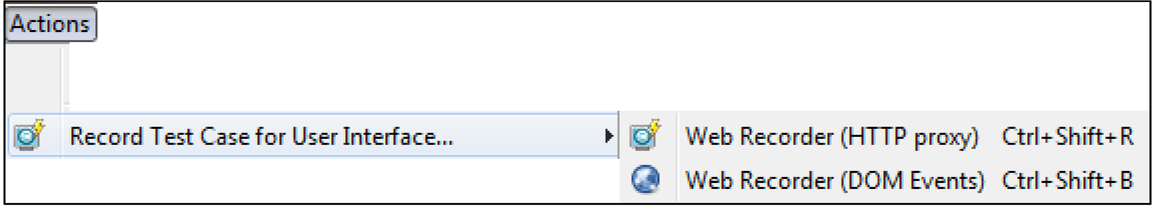
Test suite: Suites/AllTestsSuite.ste

Configuration: Configs/project.config

For more information, see Running Test Suites.

Actions > Record Test Case for User Interface

- **Record Test Case for User Interface:** Opens a new submenu to record test cases for HTTP Proxy or DOM Events.



Actions
Record Test Case for User Interface...
Web Recorder (HTTP proxy) Ctrl+Shift+R
Web Recorder (DOM Events) Ctrl+Shift+B

For more information, see Recorders and Test Generators.

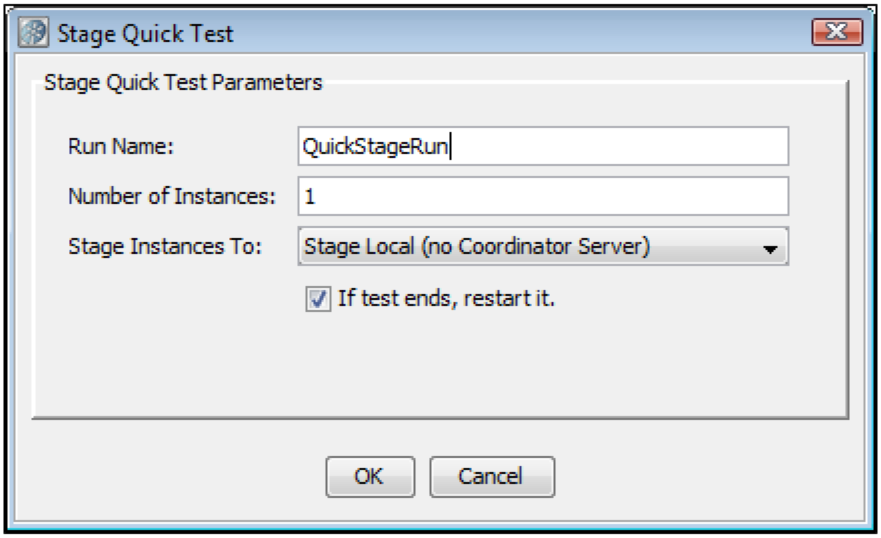
Actions > Start Interactive Test Run

- **Start Interactive Test Run (ITR):** Click to start the Interactive Test Run (ITR) utility.

For more information, see Using the Interactive Test Run (ITR) Utility.

Actions > Stage a Quick Test

- **Stage a Quick Test:** Click to stage a quick test.



Stage Quick Test

Stage Quick Test Parameters

Run Name: QuickStageRun

Number of Instances: 1

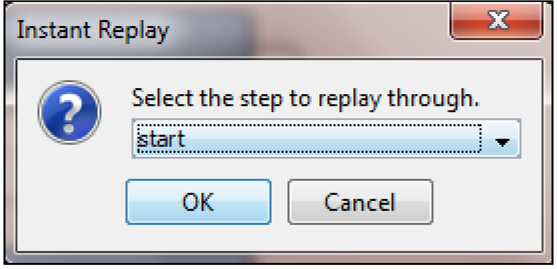
Stage Instances To: Stage Local (no Coordinator Server)

☑ If test ends, restart it.

For more information, see Staging Quick Tests.

Actions > Replace Test Case to Specific Point

- **Replay Test Case to Specific Point:** Click to run a test to a specific point in a test case. You can replay the test in the ITR utility to any given test step. A dialog is displayed, to let you enter the last step of the replay. This will populate the known state up to that point. This is particularly useful if you want to examine properties like LASTRESPONSE that store the value of the last step response.



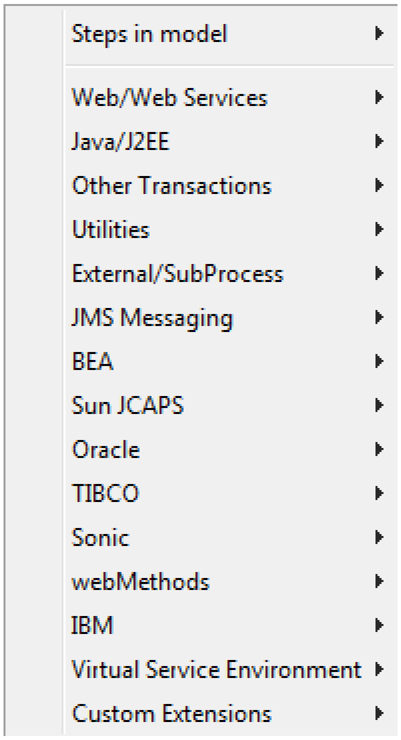
Instant Replay

Select the step to replay through.

start

Actions > Create New Step

- **Create New Step:** Opens a Steps panel with a list of all the available test steps.



The selected step will be added to the test case.

For more information, see Building Test Cases.

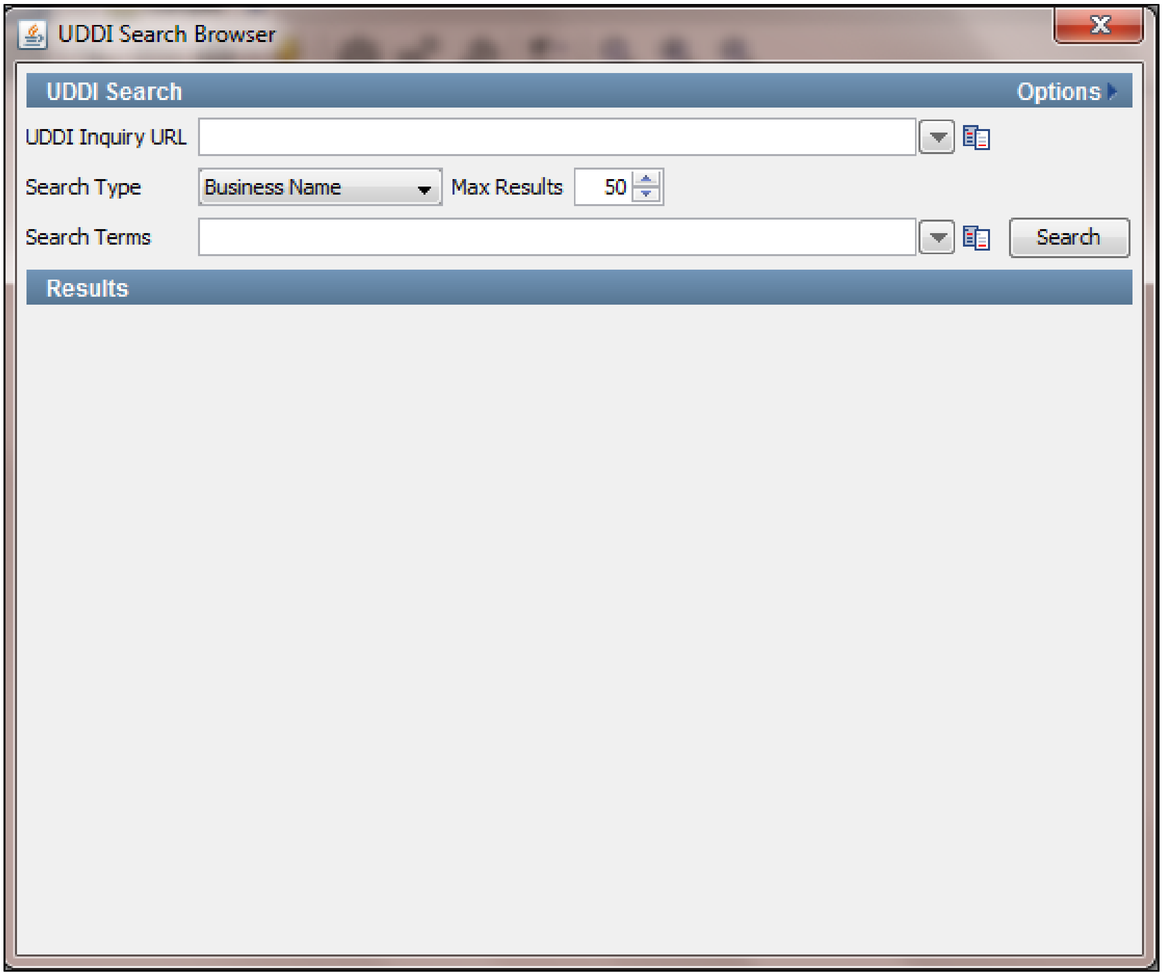
Actions > Launch Axis TCPMonitor

- **Launch Axis TCPMonitor:** Launches an external application named TCPMonitor, which lets you monitor client/server communications.

For more information, see http://ws.apache.org/axis/.

Actions > Open UDDI Search Browser

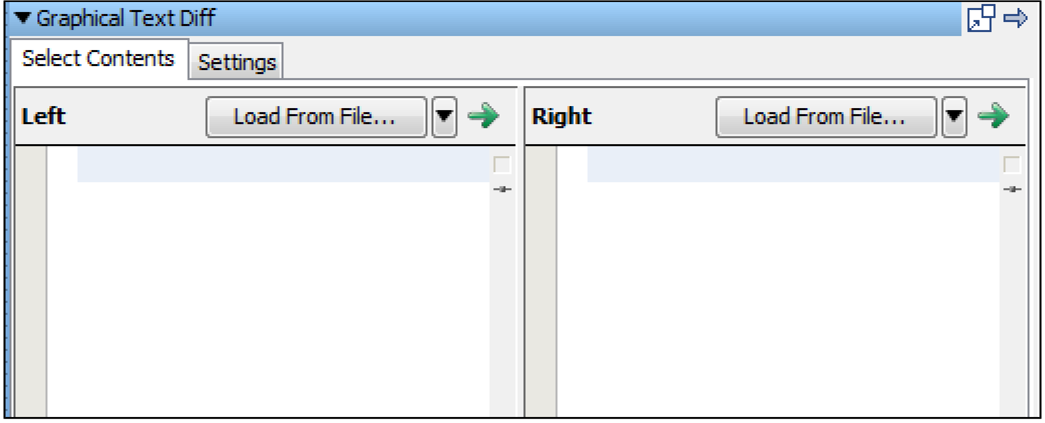
- **Open UDDI Search Browser:** Launches a UDDI search browser.



For more information, see the Web Service test step.

Actions > Graphical Text Diff

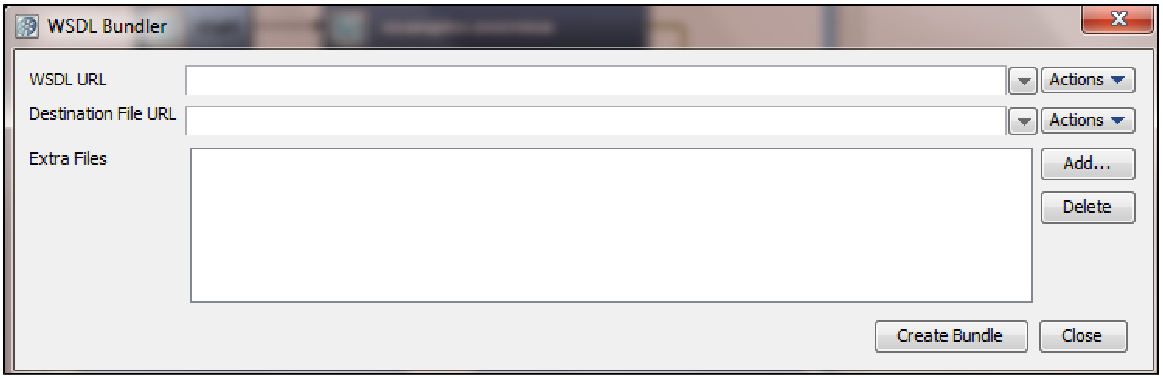
- **Graphical Text Diff...**: Opens a graphical XML diff engine and visualizer.



For more information, see Graphical Text Diff Utility.

Actions > WSDL Bundler

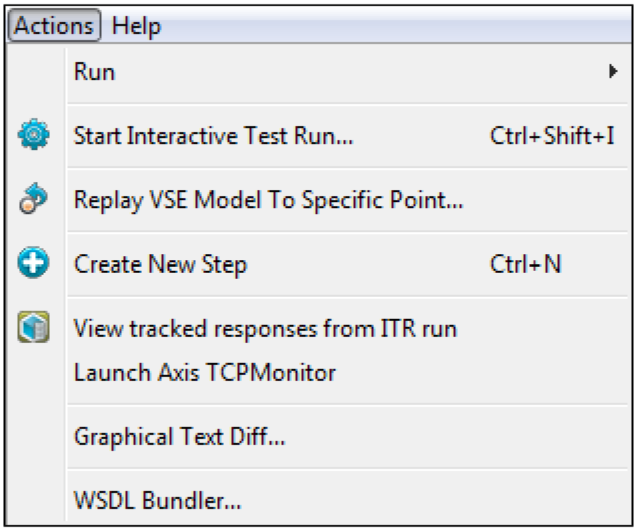
- **WSDL Bundler:** Lets you bundle a WSDL and any supporting documents into a single ZIP file. The ZIP file can be sent to ITKO Support if you are experiencing problems using the WSDL.



The Destination file is specified as a file URL.

VSE Actions Menu

When you are working with virtual service models or virtual service images, the Actions menu will change to display options related to LISA Virtualize.


Actions
Help
Run
Start Interactive Test Run... Ctrl+Shift+I
Replay VSE Model To Specific Point...
Create New Step Ctrl+N
View tracked responses from ITR run
Launch Axis TCPMonitor
Graphical Text Diff...
WSDL Bundler...

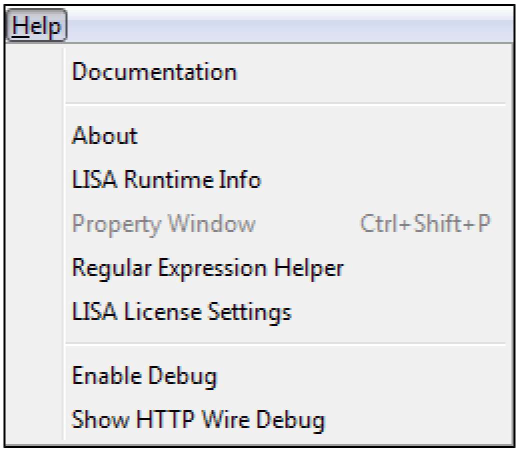
This menu contains additional options for VSE users:

- **Replay VSE Model to a specific Point**: Replays the VSE model to the point selected.
- **View Tracked Responses from the ITR run**: Enables you to see transactions tracked in the ITR.

Help Menu

> ⚠ The items on this menu are dynamic. The items available can be controlled by your LISA administrator as part of your security profile.

The Help menu includes information about Documentation, Runtime Information, LISA License settings, and debuggers.



Help > Documentation

- **Documentation:** Opens the LISA documentation page.

Help > About

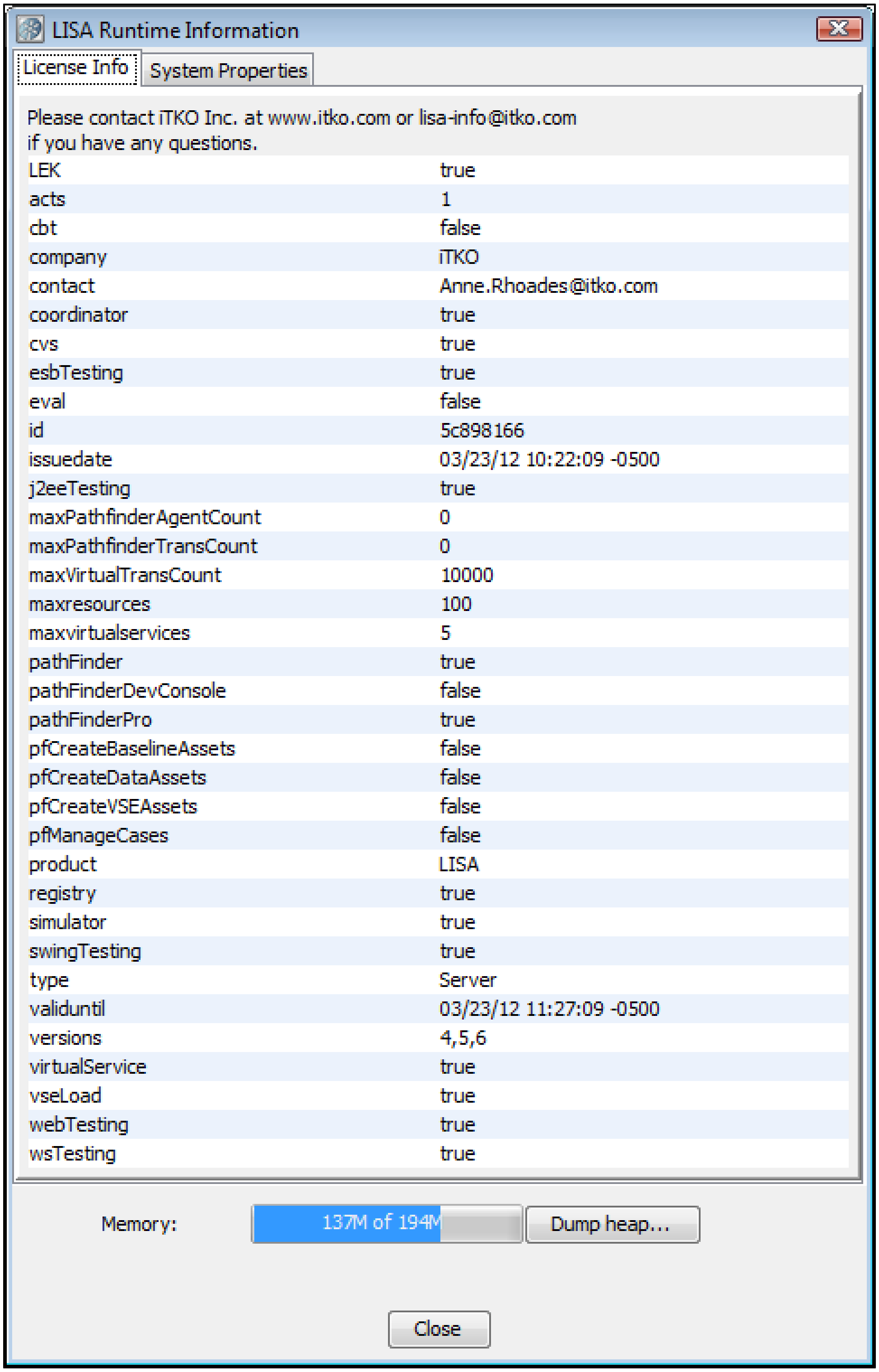
- **About:** Shows the LISA version and build numbers.

Help > LISA Runtime Info

- **LISA Runtime Info:** Shows current runtime information in two tabs, License Info (default) and System Properties.

LISA Runtime Info - License Info Tab

This tab gives information related to the LISA license.



Please contact iTKO Inc. at www.itko.com or lisa-info@itko.com
if you have any questions.

LEK	true
acts	1
cbt	false
company	iTKO
contact	Anne.Rhoades@itko.com
coordinator	true
cvs	true
esbTesting	true
eval	false
id	5c898166
issuedate	03/23/12 10:22:09 -0500
j2eeTesting	true
maxPathfinderAgentCount	0
maxPathfinderTransCount	0
maxVirtualTransCount	10000
maxresources	100
maxvirtualservices	5
pathFinder	true
pathFinderDevConsole	false
pathFinderPro	true
pfCreateBaselineAssets	false
pfCreateDataAssets	false
pfCreateVSEAssets	false
pfManageCases	false
product	LISA
registry	true
simulator	true
swingTesting	true
type	Server
validuntil	03/23/12 11:27:09 -0500
versions	4,5,6
virtualService	true
vseLoad	true
webTesting	true
wsTesting	true

Memory: 137M of 194M

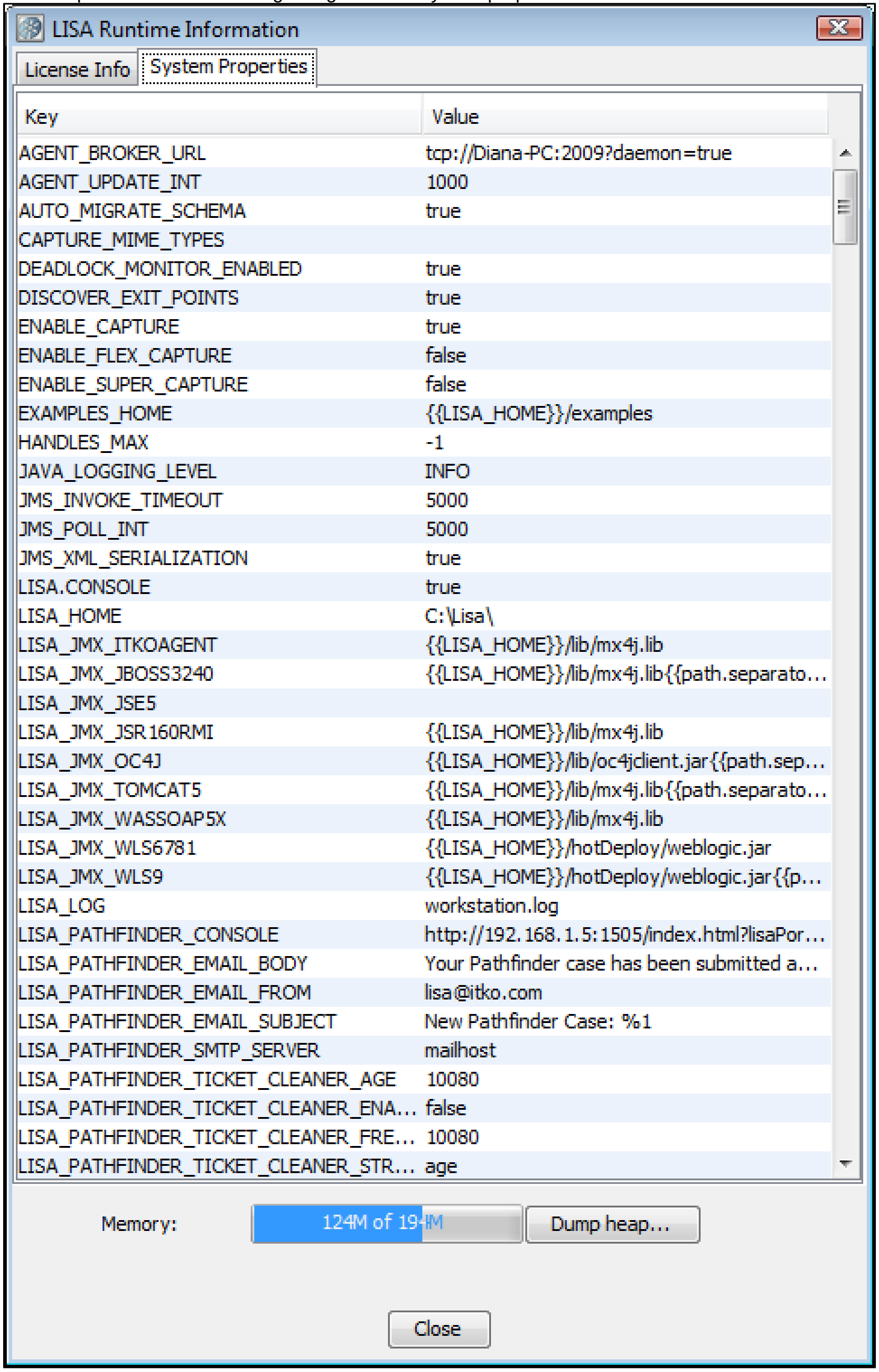
- **Dump Heap:** You can generate a heap dump file (HProf) through the LISA runtime panel. Click the Dump Heap button to produce the heap dump.

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

After the dump is created, a message indicates that the heap dump has been taken.

LISA Runtime Info - System Properties Tab

This tab provides information regarding the LISA system properties.

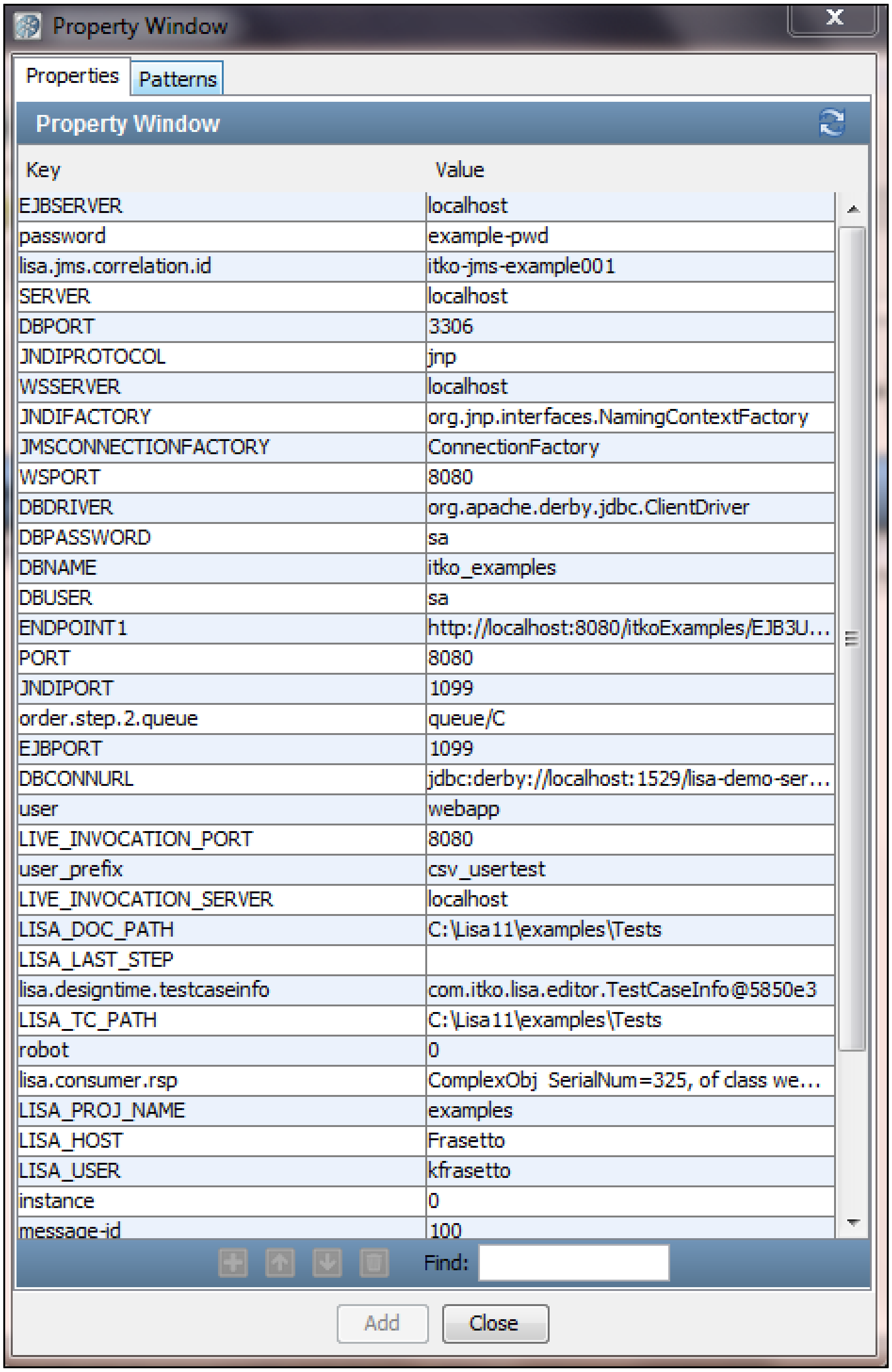


Key	Value
AGENT_BROKER_URL	tcp://Diana-PC:2009?daemon=true
AGENT_UPDATE_INT	1000
AUTO_MIGRATE_SCHEMA	true
CAPTURE_MIME_TYPES	
DEADLOCK_MONITOR_ENABLED	true
DISCOVER_EXIT_POINTS	true
ENABLE_CAPTURE	true
ENABLE_FLEX_CAPTURE	false
ENABLE_SUPER_CAPTURE	false
EXAMPLES_HOME	{{LISA_HOME}}/examples
HANDLES_MAX	-1
JAVA_LOGGING_LEVEL	INFO
JMS_INVOKE_TIMEOUT	5000
JMS_POLL_INT	5000
JMS_XML_SERIALIZATION	true
LISA.CONSOLE	true
LISA_HOME	C:\Lisa\
LISA_JMX_ITKOAGENT	{{LISA_HOME}}/lib/mx4j.lib
LISA_JMX_JBOSS3240	{{LISA_HOME}}/lib/mx4j.lib{{path.separato...
LISA_JMX_JSE5	
LISA_JMX_JSR160RMI	{{LISA_HOME}}/lib/mx4j.lib
LISA_JMX_OC4J	{{LISA_HOME}}/lib/oc4jclient.jar{{path.sep...
LISA_JMX_TOMCAT5	{{LISA_HOME}}/lib/mx4j.lib{{path.separato...
LISA_JMX_WASSOAP5X	{{LISA_HOME}}/lib/mx4j.lib
LISA_JMX_WLS6781	{{LISA_HOME}}/hotDeploy/weblogic.jar
LISA_JMX_WLS9	{{LISA_HOME}}/hotDeploy/weblogic.jar{{p...
LISA_LOG	workstation.log
LISA_PATHFINDER_CONSOLE	http://192.168.1.5:1505/index.html?lisaPor...
LISA_PATHFINDER_EMAIL_BODY	Your Pathfinder case has been submitted a...
LISA_PATHFINDER_EMAIL_FROM	lisa@itko.com
LISA_PATHFINDER_EMAIL_SUBJECT	New Pathfinder Case: %1
LISA_PATHFINDER_SMTP_SERVER	mailhost
LISA_PATHFINDER_TICKET_CLEANER_AGE	10080
LISA_PATHFINDER_TICKET_CLEANER_ENA...	false
LISA_PATHFINDER_TICKET_CLEANER_FRE...	10080
LISA_PATHFINDER_TICKET_CLEANER_STR...	age

Help > Property Window

Property Window - Properties Tab

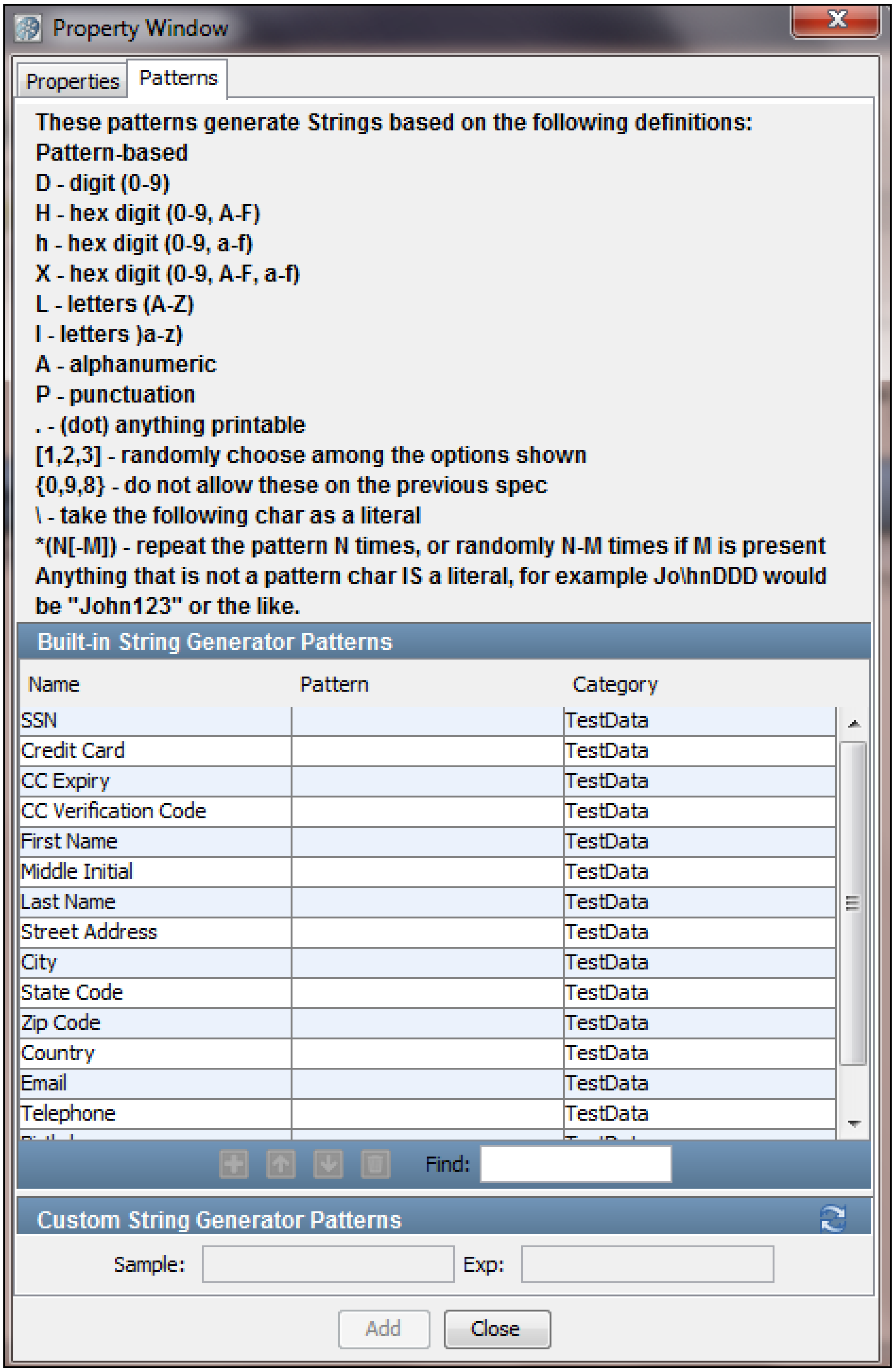
Opens up the LISA Property Window. You cannot update these values from this window.



Key	Value
EJBSERVER	localhost
password	example-pwd
lisa.jms.correlation.id	itko-jms-example001
SERVER	localhost
DBPORT	3306
JNDIPROTOCOL	jnp
WSSERVER	localhost
JNDIFACTORY	org.jnp.interfaces.NamingContextFactory
JMSCONNECTIONFACTORY	ConnectionFactory
WSPORT	8080
DBDRIVER	org.apache.derby.jdbc.ClientDriver
DBPASSWORD	sa
DBNAME	itko_examples
DBUSER	sa
ENDPOINT1	http://localhost:8080/itkoExamples/EJB3U...
PORT	8080
JNDIPORT	1099
order.step.2.queue	queue/C
EJBPORT	1099
DBCONNURL	jdbc:derby://localhost:1529/lisa-demo-ser...
user	webapp
LIVE_INVOCATION_PORT	8080
user_prefix	csv_usertest
LIVE_INVOCATION_SERVER	localhost
LISA_DOC_PATH	C:\Lisa11\examples\Tests
LISA_LAST_STEP	
lisa.designtime.testcaseinfo	com.itko.lisa.editor.TestCaseInfo@5850e3
LISA_TC_PATH	C:\Lisa11\examples\Tests
robot	0
lisa.consumer.rsp	ComplexObj SerialNum=325, of class we...
LISA_PROJ_NAME	examples
LISA_HOST	Frasetto
LISA_USER	kfrasetto
instance	0
message-id	100

Property Window - Patterns Tab

- All the LISA properties are listed here. You can also generate string patterns from the **Patterns** tab.



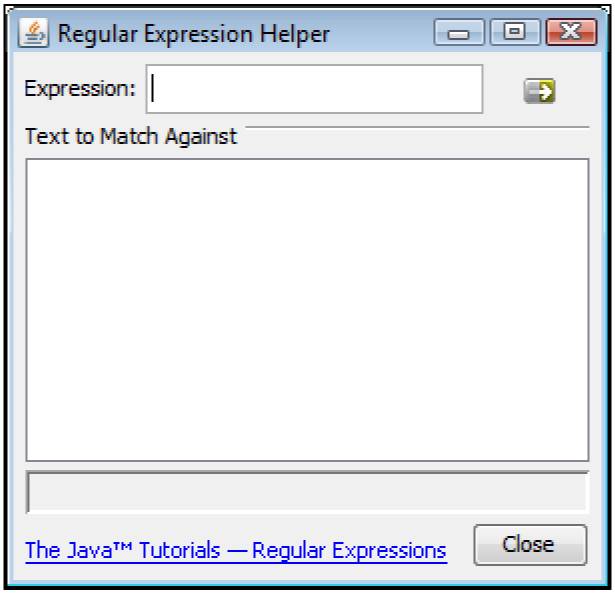
These patterns generate Strings based on the following definitions:
Pattern-based
D - digit (0-9)
H - hex digit (0-9, A-F)
h - hex digit (0-9, a-f)
X - hex digit (0-9, A-F, a-f)
L - letters (A-Z)
l - letters)a-z)
A - alphanumeric
P - punctuation
. - (dot) anything printable
[1,2,3] - randomly choose among the options shown
{0,9,8} - do not allow these on the previous spec
\ - take the following char as a literal
*(N[-M]) - repeat the pattern N times, or randomly N-M times if M is present
Anything that is not a pattern char IS a literal, for example Jo\hnDDD would be "John123" or the like.

Built-in String Generator Patterns

Name	Pattern	Category
SSN		TestData
Credit Card		TestData
CC Expiry		TestData
CC Verification Code		TestData
First Name		TestData
Middle Initial		TestData
Last Name		TestData
Street Address		TestData
City		TestData
State Code		TestData
Zip Code		TestData
Country		TestData
Email		TestData
Telephone		TestData

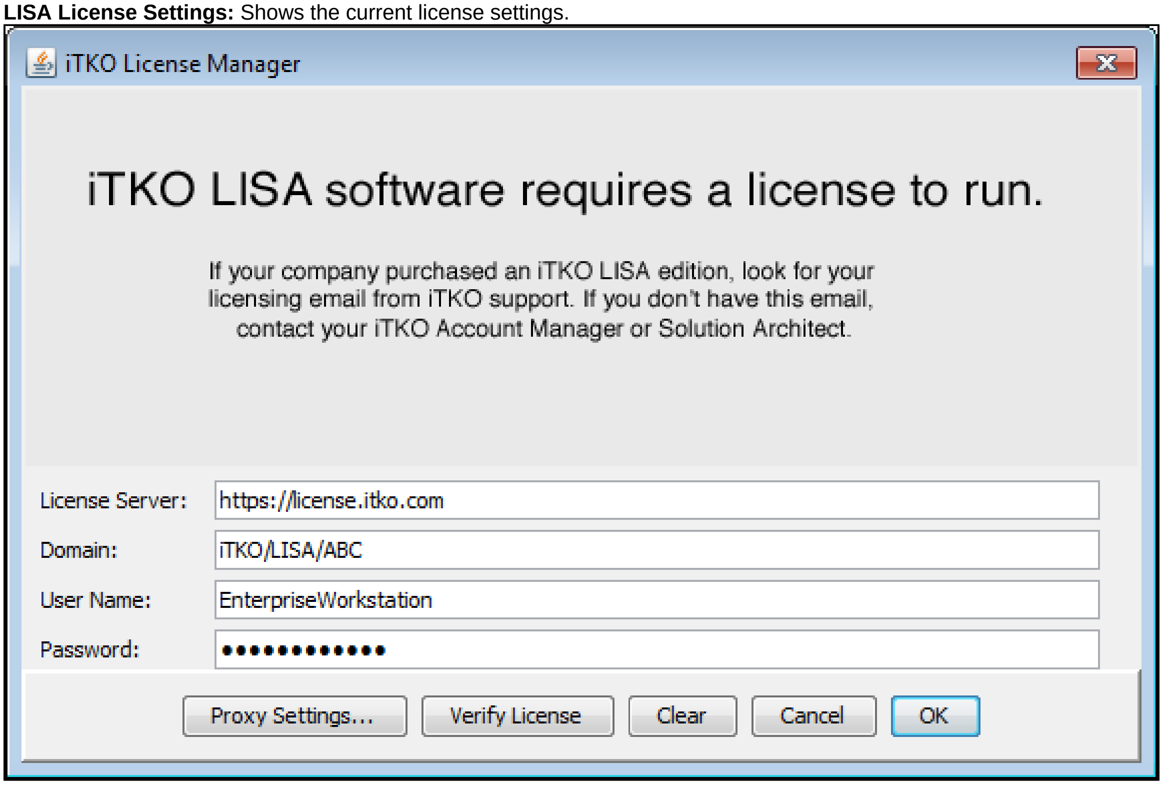
Help > Regular Expression Helper

This option displays a simple window that can be used to experiment and verify how regular expressions match against any piece of text. Here, you enter a Regex and a target string and the dialog highlights the match spans to verify that the Regex pattern itself is actually going to correctly match your input data.


Regular Expression Helper
Expression:
Text to Match Against
The Java™ Tutorials — Regular Expressions
Close

Help > LISA License Settings

- **LISA License Settings:** Shows the current license settings.



iTKO License Manager

iTKO LISA software requires a license to run.

If your company purchased an iTKO LISA edition, look for your licensing email from iTKO support. If you don't have this email, contact your iTKO Account Manager or Solution Architect.

License Server: https://license.itko.com

Domain: iTKO/LISA/ABC

User Name: EnterpriseWorkstation

Password: ●●●●●●●●●●●●

For more information on LISA licensing and license settings, see "Licensing Approaches."

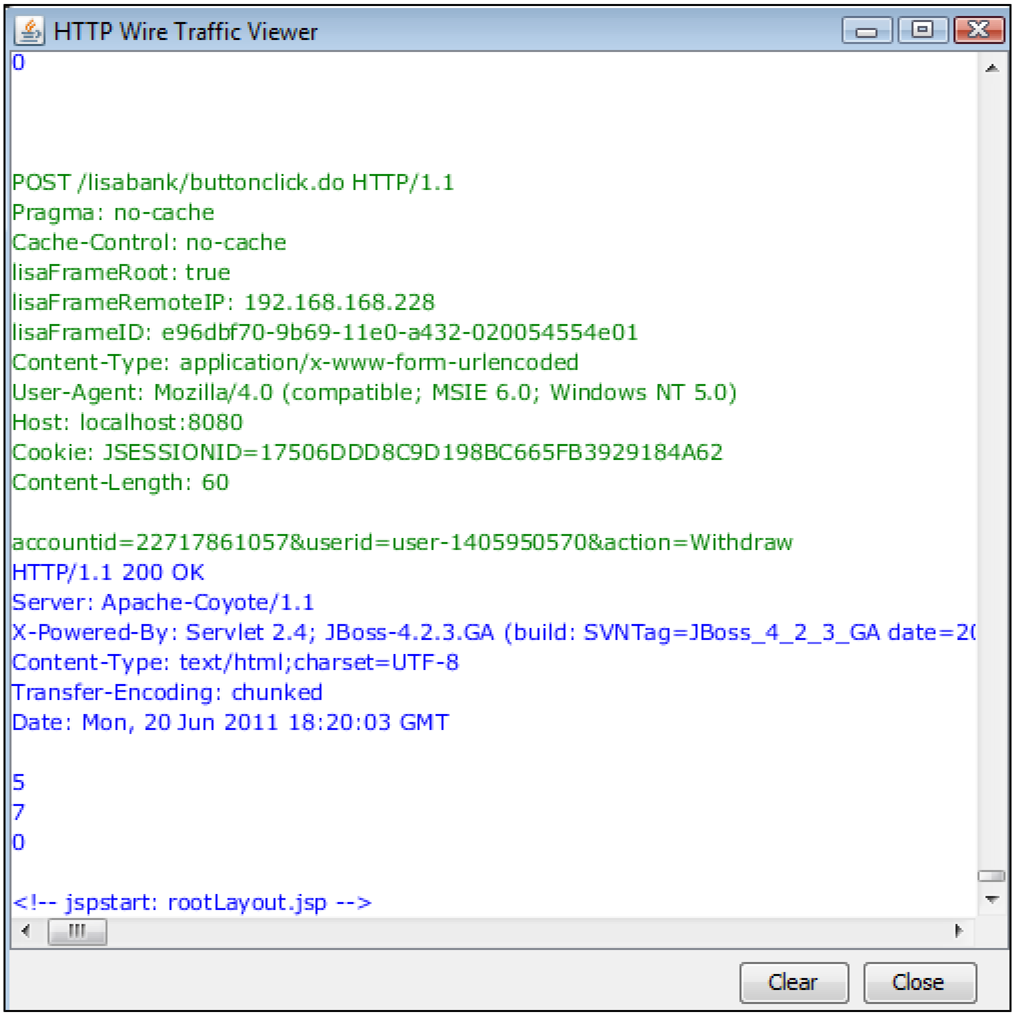
Help > Enable Debug

- **Enable Debug:** Turns on an extra level of debugging, and shows Java stack traces in the output log.

Help > Show HTTP Wire Debug

- **Show HTTP Wire Debug:** Turns on a debug mode that captures all the handshaking (multiple request/responses that happen during negotiation) to see what's happening during HTTP transactions.

The following image shows the HTTP Wire Traffic Viewer dialog.

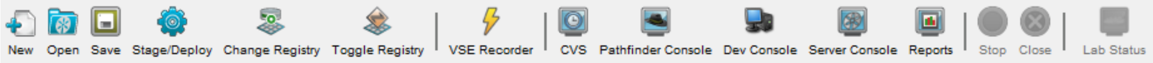


```
0



POST /lisabank/buttonclick.do HTTP/1.1
Pragma: no-cache
Cache-Control: no-cache
lisaFrameRoot: true
lisaFrameRemoteIP: 192.168.168.228
lisaFrameID: e96dbf70-9b69-11e0-a432-020054554e01
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)
Host: localhost:8080
Cookie: JSESSIONID=17506DDD8C9D198BC665FB3929184A62
Content-Length: 60

accountid=22717861057&userid=user-1405950570&action=Withdraw
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
X-Powered-By: Servlet 2.4; JBoss-4.2.3.GA (build: SVNTag=JBoss_4_2_3_GA date=2(
Content-Type: text/html;charset=UTF-8
Transfer-Encoding: chunked
Date: Mon, 20 Jun 2011 18:20:03 GMT

5
7
0

<!-- jspstart: rootLayout.jsp -->
```

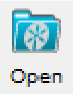
Main Toolbar

The main toolbar of LISA Workstation contains icons for common functions.

> The items on this toolbar are dynamic. The items available can be controlled by your LISA administrator as part of your security permissions.



The following table describes each icon.

Icon	Description	Main Menu Equivalent
	Creates a new project.	File > New > Project
	Opens an existing project.	File > Open > Project

Save	Saves the current document.	File > Save
Stage/Deploy	Lets you stage and deploy a test case or suite.	Actions > Stage Test and Actions > Run
Change Registry	Lets you switch to another registry.	System > Registry > Change LISA Registry
Toggle Registry	Shows and hides the LISA Registry Monitor.	System > Registry > Toggle LISA Registry Monitor
VSE Recorder	Opens the LISA Virtualize Recorder.	File > New > VS Image > By recording
CVS	Opens the CVS Dashboard.	View > CVS Dashboard
Pathfinder Console	Opens the Pathfinder Console.	View > Pathfinder Console
Dev Console	Opens the Dev Console.	View > LISA Dev Console
Server Console	Opens the Server Console.	View > Server Console
Reports	Opens the Reporting Console.	View > Reporting Console
Stop	Stops the test run. When you run a test case, this icon becomes a Play icon until the run finishes.	Not Applicable
Close	Closes the test run.	Not Applicable

Lab Status	Indicates when a cloud-based lab is being provisioned and then ready.	Not Applicable

Project Panel

The following topics are available.

Project Overview

A project is a collection of related LISA assets. The assets can include test cases, suites, virtual service models, service images, configurations, audit documents, staging documents, data sets, monitors, and MAR info files.

In LISA Workstation, only one project can be open at a time.

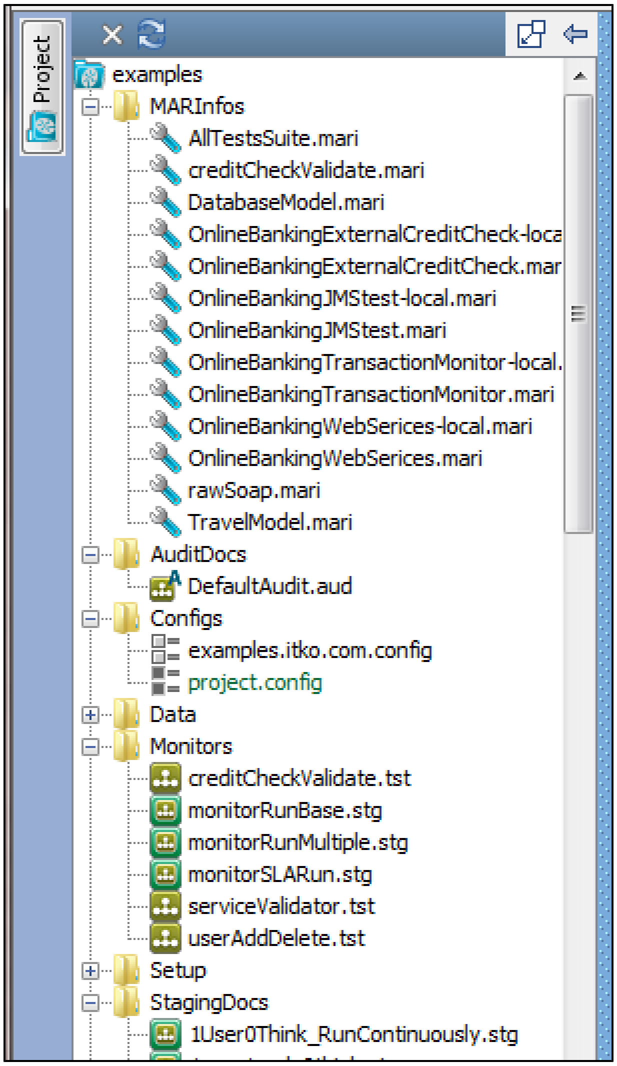
The **LISA_HOME/Projects** directory does not exist until you create a project for the first time.

All the folders in the Project panel of LISA Workstation are the same as those created in the file system. You can check the **LISA_HOME** directory to see the folder structure and files.

You can import files into a project.

Examples Project

When you start LISA Workstation for the first time, the Project panel displays a project named **examples**.



- examples
  - MARInfos
    - AllTestsSuite.mari
    - creditCheckValidate.mari
    - DatabaseModel.mari
    - OnlineBankingExternalCreditCheck-loca
    - OnlineBankingExternalCreditCheck.mar
    - OnlineBankingJMStest-local.mari
    - OnlineBankingJMStest.mari
    - OnlineBankingTransactionMonitor-local.
    - OnlineBankingTransactionMonitor.mari
    - OnlineBankingWebSerices-local.mari
    - OnlineBankingWebSerices.mari
    - rawSoap.mari
    - TravelModel.mari
  - AuditDocs
    - DefaultAudit.aud
  - Configs
    - examples.itko.com.config
    - project.config
  - Data
  - Monitors
    - creditCheckValidate.tst
    - monitorRunBase.stg
    - monitorRunMultiple.stg
    - monitorSLARun.stg
    - serviceValidator.tst
    - userAddDelete.tst
  - Setup
  - StagingDocs
    - 1User0Think_RunContinuously.stg

The **examples** project includes the following configurations:

- examples.itko.com.config
- project.config

The **project.config** file is the initial active configuration.

You can open any of the sample files by double-clicking them. The appropriate editor will open in the right panel.

The actual project files are located in the **LISA_HOME/examples** directory.

MARInfos

AllTestsSuite

Suite MAR that includes the test suite AllTestsSuite, with all the .tst files and accompanying data files to run the AllTestsSuite. It also includes the 1User1Cycle0Think staging document, the DefaultAudit audit document, and the project.config configuration file.

creditCheckValidate

Test-based Monitor MAR that includes the creditCheckValidate test case and the monitorRunBase staging document.

DatabaseModel

Virtual Service MAR that includes the DatabaseModel virtual service model and virtual service image.

OnlineBankingExternalCreditCheck-local

Test-based Monitor MAR that includes test case webservices-xml-fail, staging document Run1User1Cycle and project.config.

OnlineBankingExternalCreditCheck

Test-based Monitor MAR that includes test case webservices-xml-fail, staging document Run1User1Cycle, project.config and examples.itko.com.config.

OnlineBankingJMStest-local

Test-based Monitor MAR that includes the ansync-consumer-jms test case and Run1User1Cycle staging document.

OnlineBankingJMStest

Test-based Monitor MAR that includes the ansync-consumer-jms test case, Run1User1Cycle staging document, and project.config and examples.itko.com.config configuration files.

OnlineBankingTransactionMonitor-local

Test Based Monitor MAR that includes the multi-tier-combo test case, Run1User1Cycle staging document, all the data files to support the test case, and project.config**.**

OnlineBankingTransactionMonitor

Test Based Monitor MAR that includes the multi-tier-combo test case, Run1User1Cycle staging document, all the data files to support the test case, and project.config and examples.itko.com.config configuration files.

OnlineBankingWebServices-local

Test-based Monitor MAR that includes the webservices-xml test case, Run1User1Cycle staging document, and the project.config configuration file.

OnlineBankingWebServices

Test-based Monitor MAR that includes the webservices-xml test case, Run1User1Cycle staging document, and project.config and examples.itko.com.config configuration files.

rawSoap

Test MAR that includes the rawSoap test case, the 1User0Think_RunContinuously staging document, and the project.config configuration file.

Audit Docs

DefaultAudit

Configurations

examples.itko.com

The examples.itko.com config file should be the active configuration file when you are running your examples against the ITKO examples server at examples,itko.com. The only difference between it and the default project.config file is that instead of pointing to servers on localhost, it points to the ITKO examples website.

project

The project.config file contains intelligent defaults for many properties.

Data

The Data directory contains data sets, keystores, and WSDLs that you need to run some of the examples for the LISA Demo Server.

Monitors

creditCheckValidate.tst

Test case used for CVS monitor demos. Fails randomly on a specific cid.

monitorRunBase.stg

Staging document with one user, one cycle, and 100% think time, with Pathfinder not enabled and no maximum run time.

monitorRunMultiple.stg

Staging document with one user, run continously, 136% think time, with Pathfinder enabled and a maximum run time of 15 seconds.

monitorSLARun.stg

Staging document with one user, one cycle, and 100% think time, with Pathfinder not enabled and no maximum run time. JMX and JBOSS metrics are selected to record.

serviceValidator.tst

Used for CVS monitor demos. Fails randomly on a specific cid.

userAddDelete.tst

This test is used in the monitor setup for CVS demos.

Setup

The Setup directory in the Examples directory contains batch files to start all LISA components, stop all LISA components, and load CVS monitors. Because these files are not LISA assets, they do not appear in the Projects Panel listing of files.

If you want to use the stop script with access control (ACL) enabled, then you must add the user name and password options to the Service Manager commands in the script. The password will not be automatically encrypted, so be sure to protect the file by using the appropriate method for your operating system.

Staging Documents

1User0Think_RunContinuously

This staging doc runs a single virtual user with zero think time. It also runs the test(s) "continuously," which does not necessarily mean "forever".

If a test being run by this staging doc has ALL of the following conditions, the run will finish when the dataset runs out of data.

1. There is a dataset on the first step of the test.
2. The dataset "End of data" step is "End the test."
3. The data set is "global," not "local."

If there is more than one dataset matching these conditions, then the first dataset to expire will finish the run. For a good example of this, look at the multi-tier-combo.tst test case.

1user1cycle0think

This staging document runs a test with a single user one time with a 0% think time scale.

ip-spoofing

Use this staging document to test IP spoofing support with your LISA installation. IP spoofing is enabled in this staging document in the IP Spoofing tab. An IP spoofing test web page is available at http://localhost:8080/ip-spoof-test, if you are running the examples server.

jboss-jmx-metrics-localhost

This staging documents runs a test with three users, run continuously, with a maximum run time of 440 seconds and 100% think time. It has all four report generator parameter checkboxes selected, and specifies all JBOSS JMX metrics to be collected.

Run1User1Cycle

This staging document runs a test with a single user one time with 100% think time scale.

Run1User1CyclePF

This staging document runs a test with a single user one time with 100% think time scale, with Pathfinder enabled.

Run1User1CycleShowAll

This staging document runs a test with a single user one time with 100% think time scale. It also turns on all four checkboxes in the default report generator so more things will show in the web base model execution page.

Subprocesses

ws-sec-sub

This one-step test case is marked as a subprocess and can be called from any Execute Sub Process step.

Test Suites

AllTestsSuite

The AllTestsSuite runs all the tests in the LISA Tests directory, using the 1user1cycle0think staging document and a default audit document of AuditDocs/DefaultAudit.aud. For report metrics, it only records requests and responses, and produces the default metrics.

Test Cases

AccountControlMDB

A simple JMS test that adds a new user with an account to LisaBank. We expect and assert on patterns in the responses from the two steps.

async-consumer-jms

An example of an "async consumer" queue where the test case continually accepts messages from a response queue/topic and makes them available to the test case in the order that the messages came in.

The first step creates the queue (internal to the LISA test).

The second step fires three messages to a JMS queue on the demo server, which should cause three messages to be received on the async queue.

The third step validates that three messages were received by the async queue.

ejb3EJBTest

A pure EJB test of the LisaBank functionality. Usually you would test applications by recording a web browser or other UI. Those tests are "end to end" integration tests; these sorts of tests are "lower down the food chain" and require more technical authors (though you still do not need to write any code!)

These tests are good for the development team to use to constantly test and validate the code without having the need for a user interface, which may not exist at all or be changing so much that the tests cannot keep up.

ejb3WSTest

This model thoroughly tests the LisaBank web services. It is almost identical in functionality to the ejb3EJB test and useful for the same reasons (see that test case documentation).

ip-spoofing

This example test case demonstrates IP spoofing support in LISA.

It requests the URL "http://localhost:8080/ip-spoof-test" using a REST step, a web page that contains the IP address of the requesting client. It then makes a SOAP request to the URL http://localhost:8080/itko-examples/ip-spoof-test/webservice, a web service containing an operation that returns the IP address of the requesting client.

It executes both requests in a loop for 10 times. You can stage this test in conjunction with the IP Spoofing Test staging document "ip-spoofing.stg". With the correct network interface configuration, you should see different IP addresses used among virtual users as they make the HTTP and SOAP requests.

jms

Simple JMS example showing how to send XML/text messages and objects in native Java format.

Lisa_config_info

This test case fetches diagnostic information about the computer running LISA. The results can help ITKO support solve configuration issues.

load-csv-dataset-web

This test model uses a comma separated values (CSV) file as a data set to test a web application. The demo web app that ships with LISA lets us add and remove users from the database.

log-watcher

This example shows how to fail a test by watching a log file for ERROR or WARNing messages.

It uses a data set to feed the example AntiPattern bean two numbers to divide. About halfway through the data set we give 0 as the operand. This, of course, causes a divide-by-zero exception to occur on the server. The AntiPattern bean logs the exception and returns -1 as a result.

This is a common anti-pattern: internal errors occurring but external parties have no idea that the result is incorrect. It *looks* believable but it is wrong. What should happen is that the EJB propagates the exception back to the caller.

If we were using Pathfinder this test case would fail because the fact that an exception was logged will be recorded by the agent and LISA would determine that something is wrong.

An alternative to using Pathfinder/LISA Agent is to set up a Global Assertion to watch the server log file. We define what to look for as a regular expression: in this case simply the test "ERROR". The regular expressions can be as simple or as complicated as you want.

Usually you would set the assertion to fail the test immediately. In this case we step to the "Error detected in log file" step and end the test normally.

Applications under test should never pass if they are logging errors or warnings. Consider using an equivalent companion in your test cases by default.

main_all_should_fail

This test is an example of negative testing. We expect test steps to fail; that is, we feed a service data that should cause an error.

The test has a companion, the NegativeTestingCompanion, which fails the test if any steps succeed.

In this case we try to create users in the demo server that already exist. We get this data from the database itself (the username dataset queries the table directly). If any step passes, the overall test should fail.

The only step that does pass is the "quietly succeed" step; in other words you can put steps that you do *not* expect to fail in this sort of testing scenario but you should mark them as "quiet" in the editor so that they are not included by the NegativeTestingCompanion.

main_all_should_pass

This test calls a subprocess to insert a unique username into the demo server USERS table.

The data set is interesting in that it relies on a datasheet to draw values from a unique code generator. The same thing could have been done with a unique code generator in conjunction with a "counter" dataset but this example demonstrates how one data set can influence another.

Data sets are evaluated in the order they are specified; each time the step is executed, the UniqueUser property is assigned a new value and the data sheet refers to {{UniqueUser}} four times so we get five unique values.

If any of the steps fail the test fails immediately.

Compare this test to "main_all_should_fail," which is a similar test where we expect each step to fail and we fail the test if anything passes (this is known as negative testing).

multi-tier-combo-XML

The multi-tier-combo test uses a variety of service endpoints to validate the LisaBank example. It tests SOAP, EJB, JMS and web transactions and validates these transactions in a variety of ways including directly validating the demo server database.

The test also demonstrates how you can build complex SOAP objects from spreadsheets. The "User" dataset on the first step is backed by a spreadsheet named "multi-tier-users.xls" in the Data folder of the project.

If you run this test in the Interactive Test Run window (ITR) it will create a single user from the first row of the spreadsheet and then the test will finish.

If you stage the test with the example "1User0Think_RunContinuously" staging document, the test will be restarted until the end of the data set is reached. This is the preferred way to repeatedly iterate over a large data set. You could introduce a loop in the test case that is not as flexible.

If you let the staging document control the data set ending the test then you can spread the test over many virtual users (if you want to) or control the pacing of the test with think times and other parameters.

The staging document "end the continuous test run" behavior is only affected by global data sets that are set on the FIRST step in the test. If the data set is local to the test or declared elsewhere in the test, the "run continuously" behavior really does mean "run forever".

multi-tier-combo

The multi-tier-combo test uses a variety of service endpoints to validate the LisaBank example. It tests SOAP, EJB, JMS and web transactions and validates these transactions in a variety of ways including directly validating the demo server database.

The test also demonstrates how you can build complex SOAP objects from spreadsheets. The "User" dataset on the first step is backed by a spreadsheet named "multi-tier-users.xl'" in the Data folder of the project.

If you run this test in the Interactive Test Run window (ITR) it will create a single user from the first row of the spreadsheet and then the test will finish.

If you stage the test with the example "1User0Think_RunContinuously" staging document, the test will be restarted until the end of the data set is reached. This is the preferred way to repeatedly iterate over a large data set. You could introduce a loop in the test case that is not as flexible.

If you let the staging document control the data set ending the test then you can spread the test over many virtual users (if you want to) or control the pacing of the test with think times and other parameters.

The staging document "end the continuous test run" behavior is only affected by global data sets that are set on the FIRST step in the test. If the data set is local to the test or declared elsewhere in the test, the "run continuously" behavior really does mean "run forever".

rawSoap

The rawSoap step is a one-step test case demonstrating a simple raw SOAP request in the "listUsers" step.

rest-example

The rest-example test demonstrates how to execute RESTful services. The LISA Demo Server contains a JAX-RS example, and each step in this test shows how to interact with that service using both XML and JSON.

service-validation

This is a simple example of service validation. The test calls two services (one web service, one EJB service) and validates that they do what they claim to do by inspecting the underlying database with SQL.

web-application

This is a simple web test that was generated using the web recorder. It contains some basic assertions such as "assert non-empty response," which is automatically generated and some "assert title" assertions that were created by parsing the HTML responses for the <title> tag and helping to ensure the page we recorded is the same page when we play back the test.

webservices-xml

This test case will add, get, and delete a user from the EJB3 web services. It uses a unique code generator to create a number prefixed by a value {{user}} from the config file. The password is hard-coded in the config file.

webservices-xml-fail

This test case will add, get, and delete a user from the EJB3 web services. It uses a unique code generator to create a number prefixed by a value {{user}} from the config file. The password is hard-coded in the config file.

webservices

This test case is a very basic web service example.

ws_attachments

This test case tests our ability to send and receive inline base64 encoded data blobs and XOP/MTOM attachments. Filters and assertions on steps verify the requests and responses look correct.

ws_security

The ws_security test case shows how to use signed and encrypted SOAP messages. The first two steps should succeed and the last two steps should fail (the calls are the same but the web service will not accept messages that are not encrypted or signed).

Virtual Services

DatabaseModel.vsi

kioskV4ServiceImage.vsi

kioskV6.vsi

si-kioskV5-dynamic.vsi

si-kioskV5.vsi

WebAppModel.vsi

WebServicesModel.vsi

DatabaseModel.vsm

kioskV4model.vsm

kioskV5.vsm

kioskV6.vsm

statefullATM.tst

statelessATM.tst

web-app-proxy.tst

WebAppModel.vsm

webservices-vs.vsm

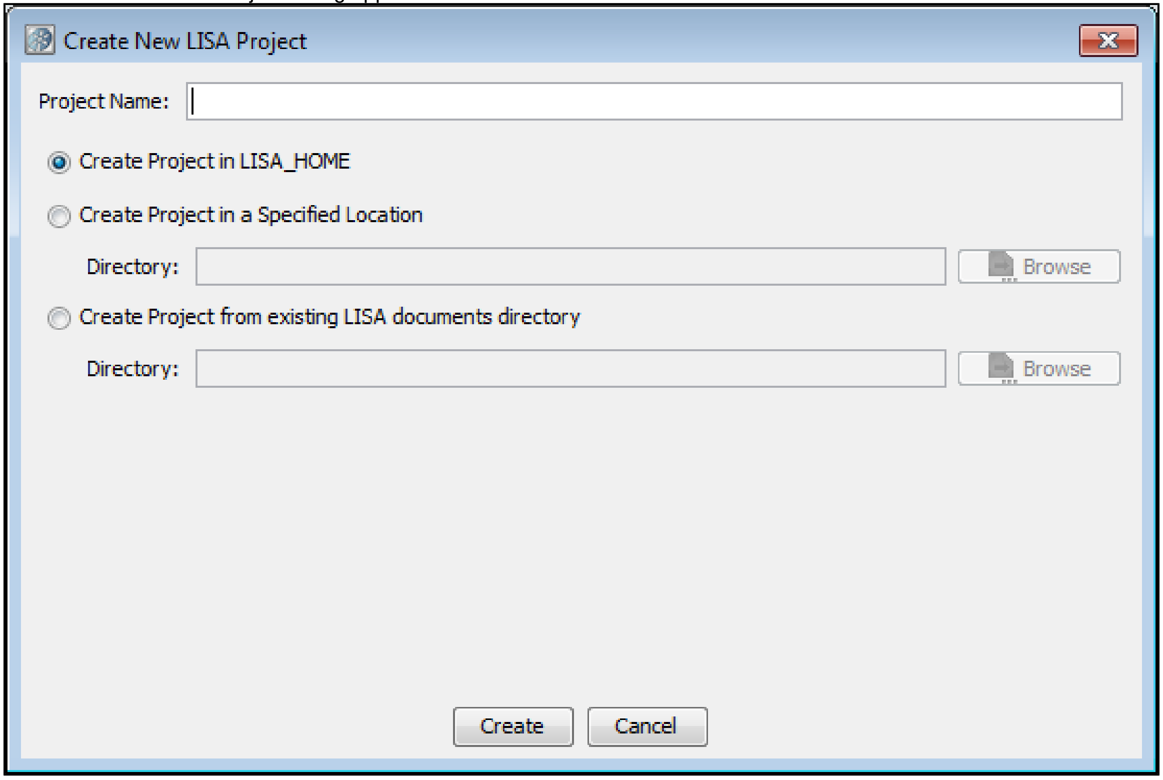
WebServicesModel.vsm

Creating a Project

You create a project from within LISA Workstation.

To create a project

1. From the main menu, select File > New > Project.
   The Create New LISA Project dialog appears.



Create New LISA Project

Project Name:

● Create Project in LISA_HOME

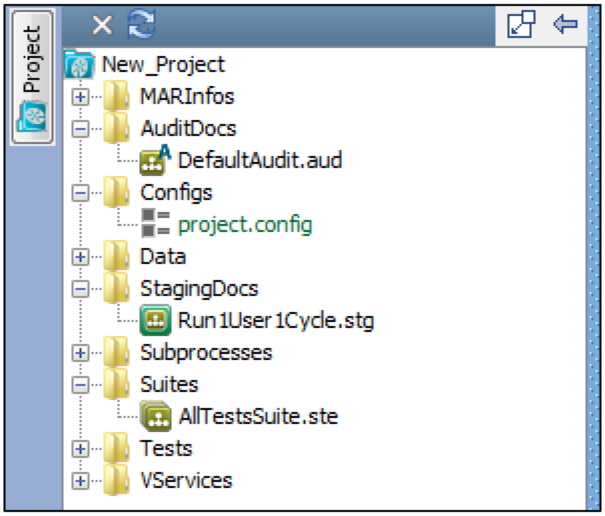
○ Create Project in a Specified Location

Directory:

○ Create Project from existing LISA documents directory

Directory:

2. In the Project Name field, enter the name of the project.
3. Select one of the following options:
   - **Create Project in LISA_HOME:** Click to create a project in the project subfolder of the LISA home directory.
   - **Create Project in a Specified Location:** Click to specify the location of the project. Click Browse to browse to the directory.
   - **Create Project from existing LISA documents directory:** Click to create a new project from an existing projects file. This option is used for importing a projects file from another directory.
4. Click Create. The new project appears in the Project panel.

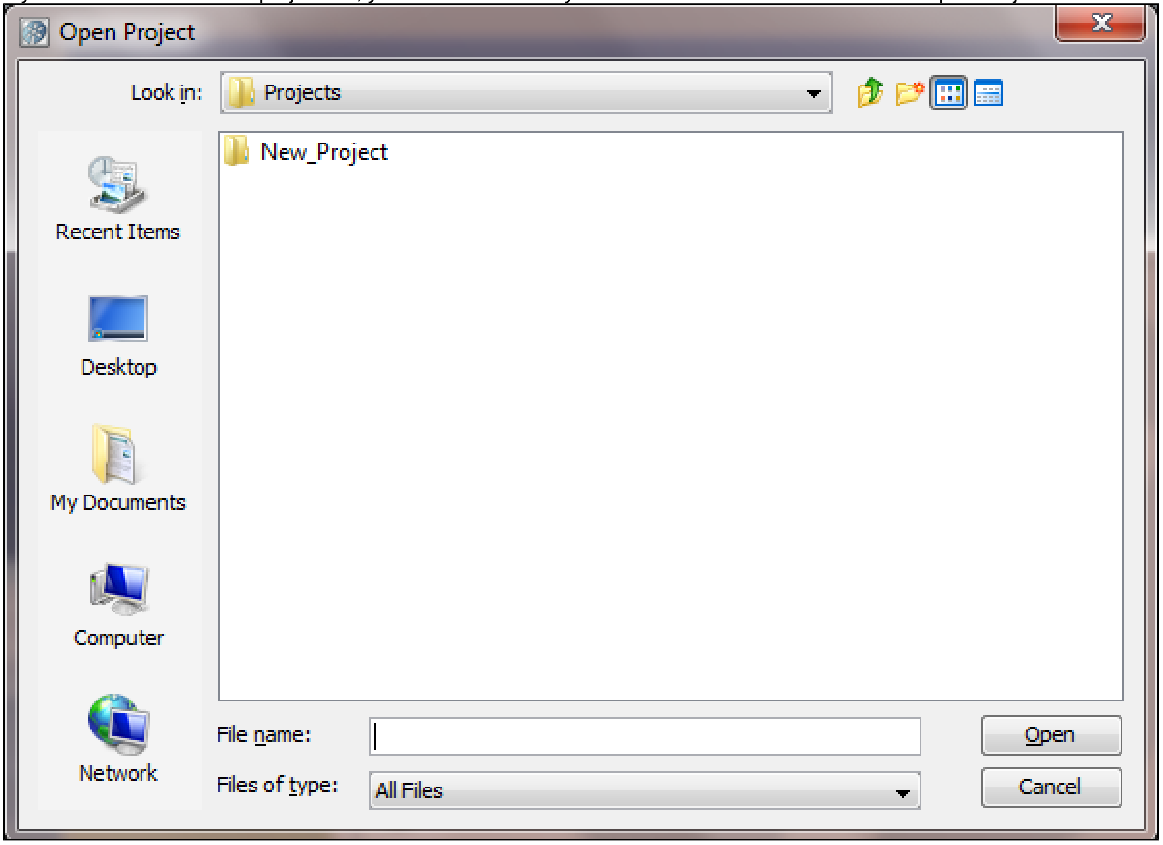


Opening a Project

When you open a project, it will have a list of folders and subfolders in the left panel.

After the project is opened, hovering on the Project icon on the left of the Project panel displays a tooltip that shows the complete project path.

To open a project from LISA Workstation

1. From the main menu, select File > Open > Project.
   If you have already opened some projects, you will see a list of recently opened projects that you can select from. After it is selected, the project will open.
   If you want to browse for a project file, you can select File System from the list of choices and the Open Project window will open.
   

2. Select the required project and click Open.

To open a project from the command line

1. Navigate to the **LISA_HOME/bin** directory.
2. Run the LISA Workstation executable and specify the project directory as an argument. The project directory can be an absolute path or

a relative path. The following example uses an absolute path:

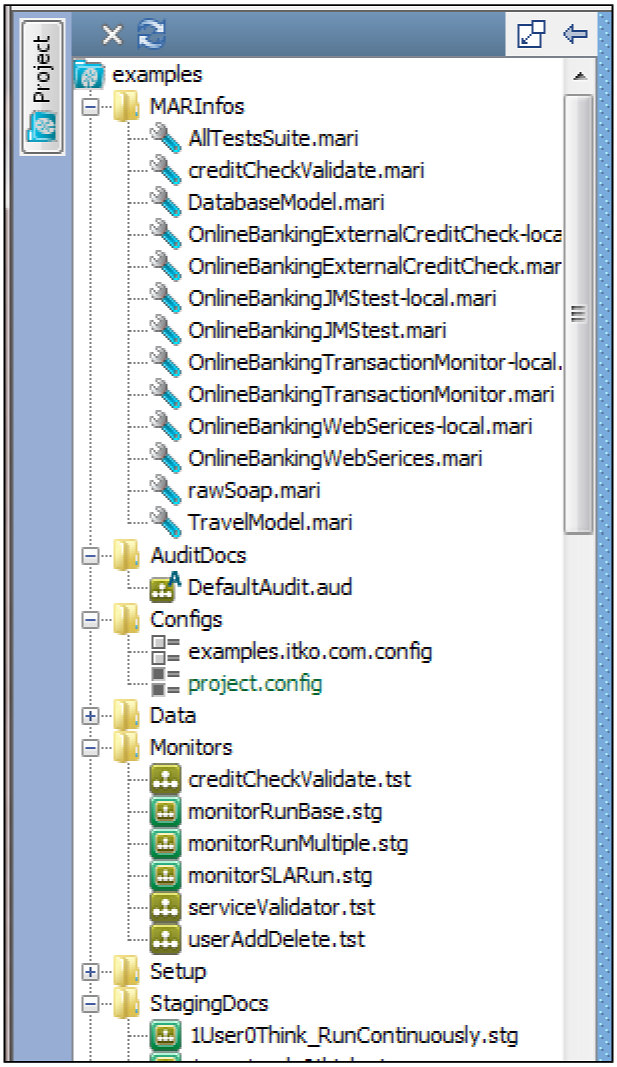
```
LISAWorkstation C:\Lisa\Projects\Project1
```

Project Panel Layout

The Project panel is dockable. You can open or close the Project panel by clicking the Project button.

The Project panel contains a project tree. For a new project created from scratch, the following folders are automatically created:

- MARInfos
- AuditDocs
- Configs
- Data
- StagingDocs
- Subprocesses
- Suites
- Tests
- VServices



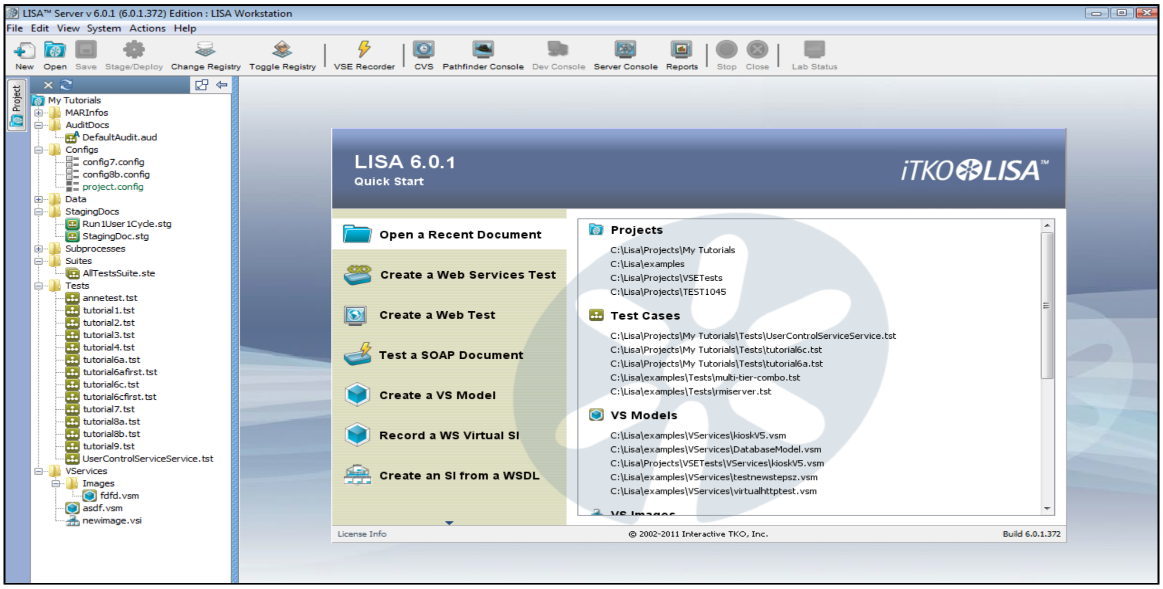
- examples
  - MARInfos
    - AllTestsSuite.mari
    - creditCheckValidate.mari
    - DatabaseModel.mari
    - OnlineBankingExternalCreditCheck-loca
    - OnlineBankingExternalCreditCheck.mar
    - OnlineBankingJMStest-local.mari
    - OnlineBankingJMStest.mari
    - OnlineBankingTransactionMonitor-local.
    - OnlineBankingTransactionMonitor.mari
    - OnlineBankingWebSerices-local.mari
    - OnlineBankingWebSerices.mari
    - rawSoap.mari
    - TravelModel.mari
  - AuditDocs
    - DefaultAudit.aud
  - Configs
    - examples.itko.com.config
    - project.config
  - Data
  - Monitors
    - creditCheckValidate.tst
    - monitorRunBase.stg
    - monitorRunMultiple.stg
    - monitorSLARun.stg
    - serviceValidator.tst
    - userAddDelete.tst
  - Setup
  - StagingDocs
    - 1User0Think_RunContinuously.stg

The Project toolbar has icons to close ✕ and refresh ↻ the project, and an icon to dock or undock ⧉ the Project panel.

The **project.config** configuration is included by default. This configuration can be overridden by another configuration.

The project has a **.settings** directory, which does not appear in the Project panel. The **.settings** directory is used for saving some settings internally and can be seen in the file system in the Project directory.

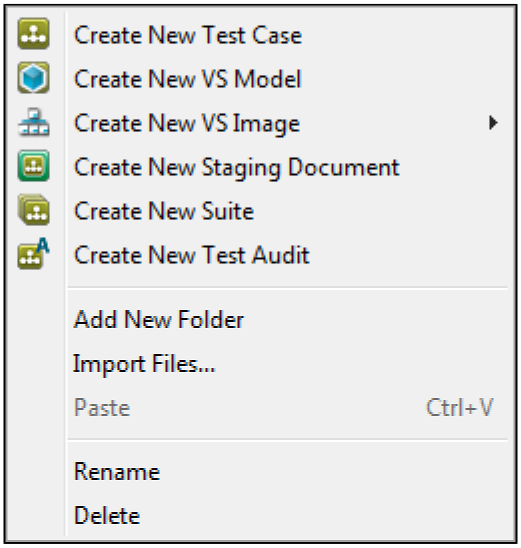
The Quick Start window appears in the right panel.



Project Panel Right-Click Menu

You can create various documents within a project from the Project panel. When you right-click a choice in the Project panel, the menu that appears is dynamic to the selection.

The following image includes all the available choices. The order of these choices will vary depending on what choice is selected when you right-click.



The right-click menu choices described here are also available from the main menu by selecting File > New.

- **Create New Test Case**: For detailed information, see Creating Test Cases.
- **Create New VS Model**: For detailed information, see Working with Virtual Service Models. This feature requires an additional license.
- **Create New VS Image**: For detailed information, see Creating Service Images. This feature requires an additional license.
- **Create New Staging Document**: For detailed information, see Building Staging Documents.
- **Create New Suite**: For detailed information, see Building Test Suites.
- **Create New Test Audit**: For detailed information, see Building Audit Documents.

One or more of the following choices may also be available:

- Add New Folder
- Import Files
- Rename
- Delete
- Paste
- View/Edit Project Metadata

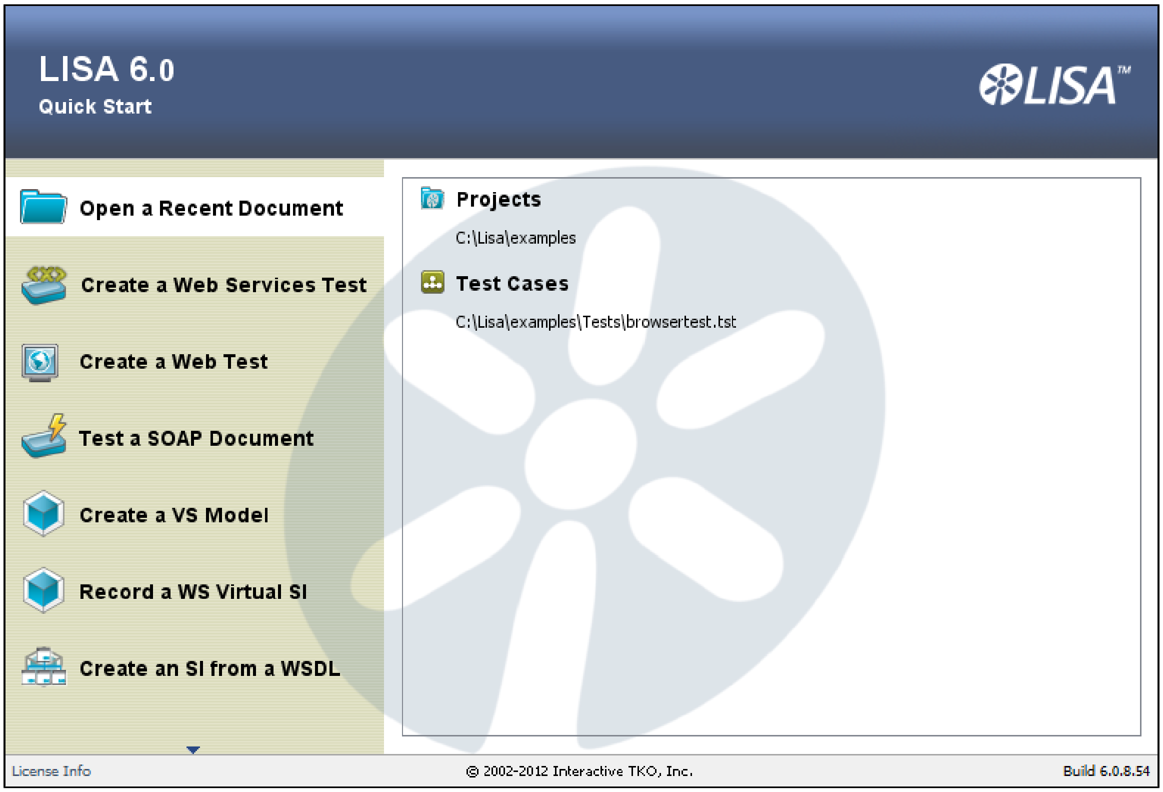
If you want the documents you create to default to the matching folder name, right-click that choice when adding. If you add a document without being on that actual selection in the Project panel, it will be added to the project, but it will appear at the bottom of the Project panel list. You will then have to go to the root directory and manually move the file into the correct folder and then reopen the project to correct.

You are not limited to keeping a certain kind of file (such as .tst) under the recommended folder (such as Tests). The file can stay anywhere within the project. The only exceptions are .config files and data resources; they must be located in the Configs and Data folders, respectively.

Quick Start Window

The Quick Start window is the first one you will see when you open LISA Workstation.

> ⚠ The Quick Start window is always available as a tab after additional tabs are opened.

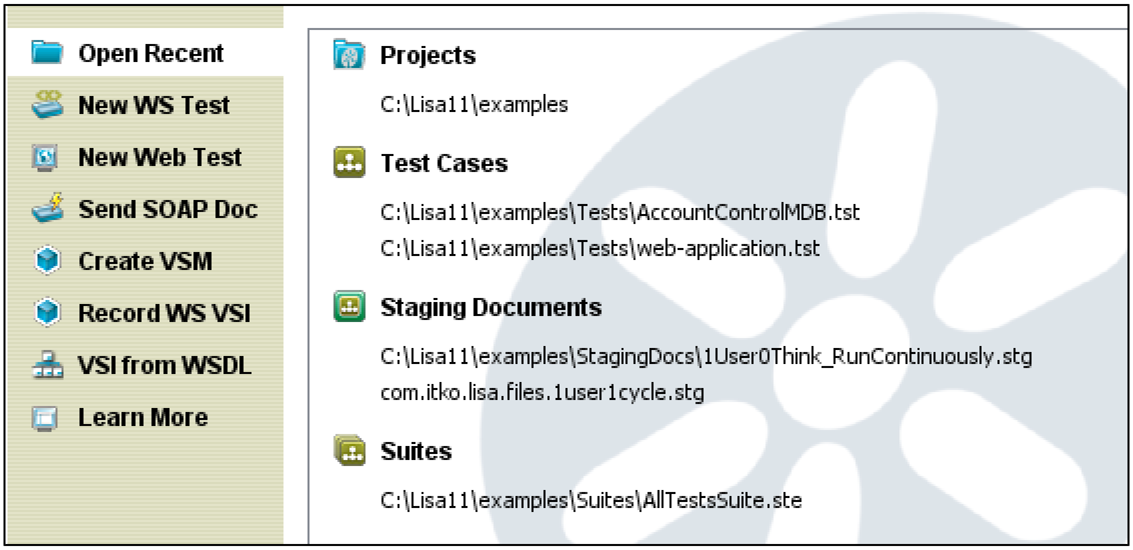


The Quick Start window has some of the most useful options listed within LISA Workstation. When you click an option, its parameters are listed on the right side of the window.

> The following choices are available on this window. Depending on your screen resolution, you may see the compact or the expanded wording for each menu choice. You may need to click the down arrow at the bottom of the screen to see all menu options.
>
> Open Recent or Open a Recent Document
> New WS Test or Create a Web Services Test
> New Web Test or Create a Web Test
> Send SOAP Doc or Test a SOAP Document
> Create VSM or Create a VS Model
> Record WS VSI or Record a WS Virtual SI
> VSI from WSDL or Create an SI from a WSDL
> Learn More or Learn More About LISA

Open Recent

When LISA Workstation opens, by default this option is selected. The right pane displays recently opened projects, test cases and suites, staging documents, VS models or VS images (if applicable).



Projects

C:\Lisa11\examples

Test Cases

C:\Lisa11\examples\Tests\AccountControlMDB.tst

C:\Lisa11\examples\Tests\web-application.tst

Staging Documents

C:\Lisa11\examples\StagingDocs\1User0Think_RunContinuously.stg

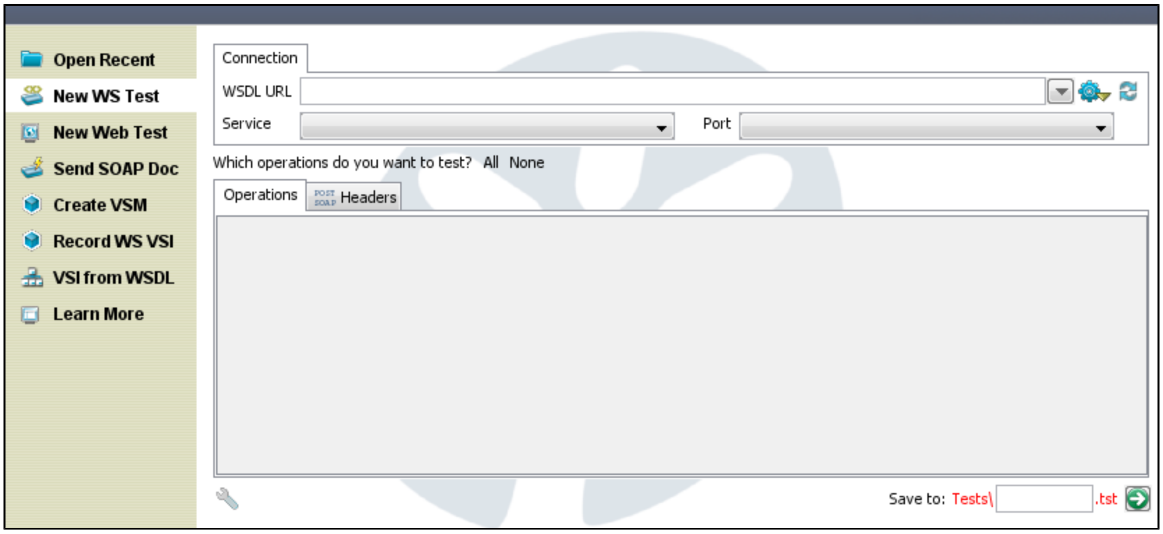
com.itko.lisa.files.1user1cycle.stg

Suites

C:\Lisa11\examples\Suites\AllTestsSuite.ste

New WS Test

The **New WS Test** option in the Quick Start window lets you create a Web Service test case. The parameters needed are displayed in the right pane.

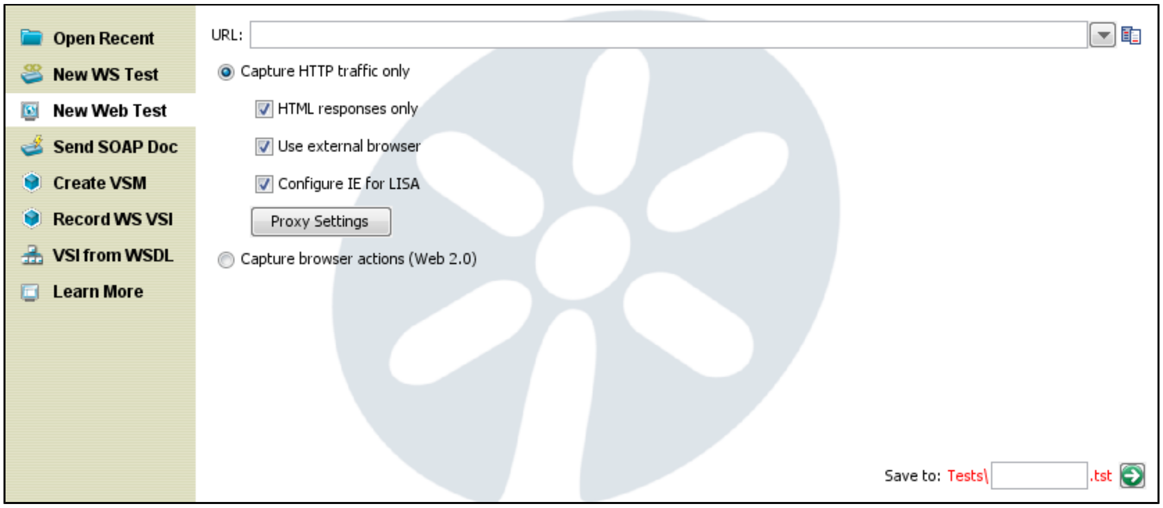


1. Enter the WSDL URL, Service and Port details.
2. Check the operations that you want to test from **All** or **None** or select each item individually.
3. In the right pane, enter the name of the test and select the path where you would like to save it within the project folder. When you select the path, it will be seen in the **Path** field. Within this pane, you can right-click on any folder and create a new one or rename or delete an existing one.
4. Click the green arrow to create the test case.

For more information, see Generating a Web Service.

New Web Test

The **New Web Test** option in the Quick Start window lets you create a web test.



URL:

● Capture HTTP traffic only

☑ HTML responses only

☑ Use external browser

☑ Configure IE for LISA

○ Capture browser actions (Web 2.0)

Save to: Tests\ .tst

1. Enter the **URL** for the web test.
2. Select if you want to capture **HTTP traffic** or capture **Browser actions**.

If you select **HTTP** traffic, check your options in the check boxes:

- **HTML Responses Only**: Will capture only the HTML responses.
- **Use External Browser**: Will open an external browser window.
- **Configure IE for LISA**: Will configure Internet Explorer for LISA.

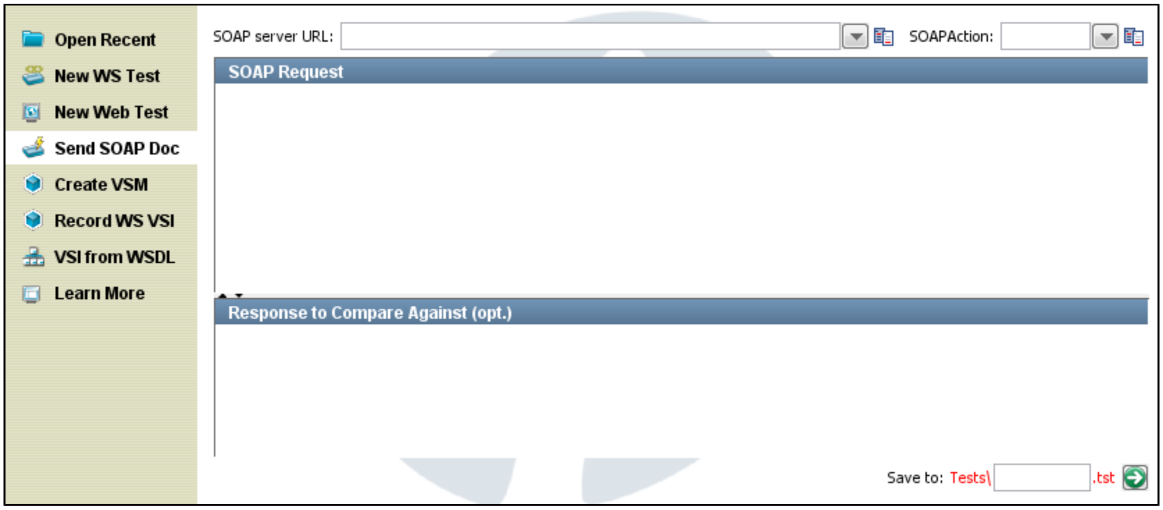
3. In the right pane, enter the name of the test and select the path where you would like to save it in the project folder. When you select the path, it will be seen in the Path field. Within this pane, you can right-click on any folder and create a new one or rename or delete the same as shown previously.

4. Click the green arrow ➔ to create the test case.

For more information, see Recording a Website.

Send SOAP Doc

The **Send SOAP Doc** option in the Quick Start window lets you create a SOAP Request test case.



SOAP server URL:

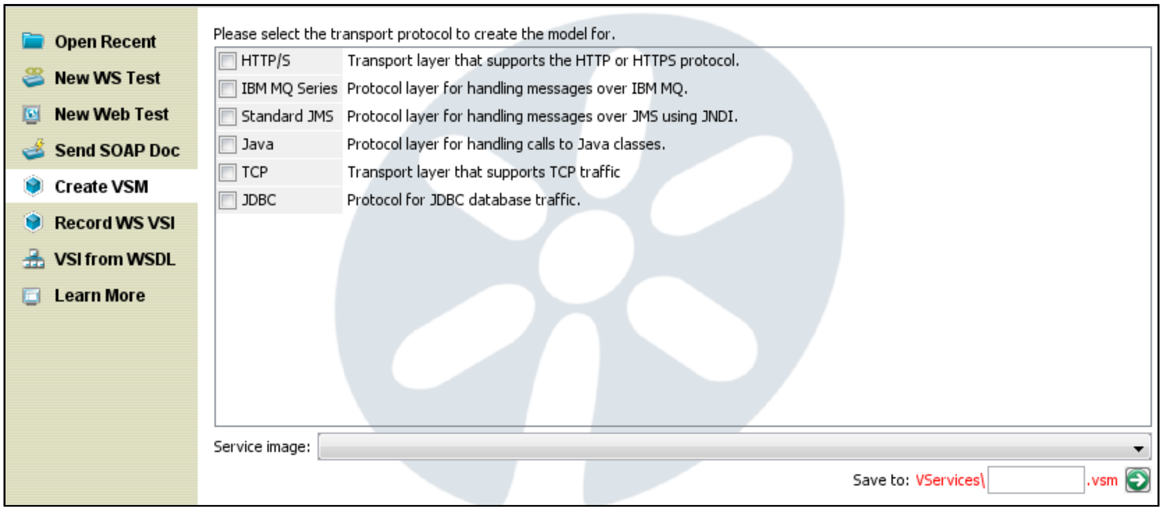
SOAPAction:

Save to: Tests\ .tst

1. Enter the SOAP server URL and SOAP Action.
2. Enter the URL of the Web Service endpoint in the SOAP Server URL field.
3. In the SOAP Action field, enter the SOAP action as indicated in the <soap: operation> tag in the WSDL for the method being called. This is required for SOAP 1.1 and usually required to be left blank for SOAP 1.2.
4. Type or paste the SOAP Request into the editor.
5. Enter the name of the test in the Save to field.
6. Click the green arrow ➔ to create the test case.

Create VSM

The **Create VSM** option in the Quick Start window lets you create a Virtual Service Model and a corresponding Virtual Service Image.



Please select the transport protocol to create the model for.

□ HTTP/S	Transport layer that supports the HTTP or HTTPS protocol.
□ IBM MQ Series	Protocol layer for handling messages over IBM MQ.
□ Standard JMS	Protocol layer for handling messages over JMS using JNDI.
□ Java	Protocol layer for handling calls to Java classes.
□ TCP	Transport layer that supports TCP traffic
□ JDBC	Protocol for JDBC database traffic.

Service image:

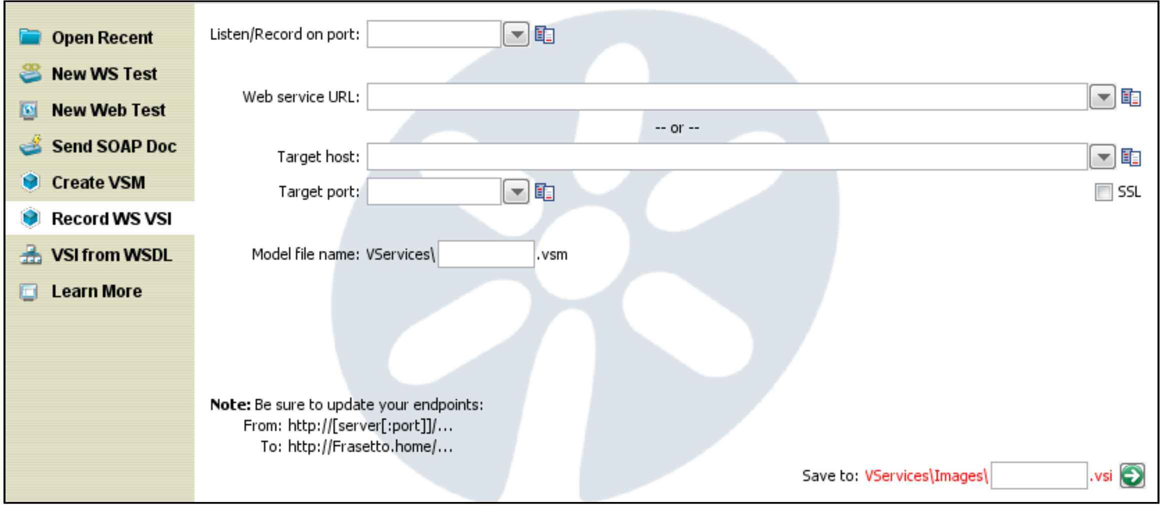
Save to: VServices\ .vsm

1. Select the transport protocol from the list of available protocols. Your selection here will determine the next sequence of windows.
2. Enter the name of the Service Image.
3. Enter the name of the VSM and select the path where you would like to save it within the project folder. When you select the path, it will be seen in the Path field.
4. Click the green arrow to go to the next screen(s).

For more detailed information about the screens and options for VSM and VSI creation, see "Creating Service Images."

Record WS VSI

The **Record WS VSI** option in the Quick Start window lets you create a web services Virtual Service Image. The parameters needed are displayed in the right pane.



Listen/Record on port:

Web service URL:

-- or --

Target host:

Target port: □ SSL

Model file name: VServices\ .vsm

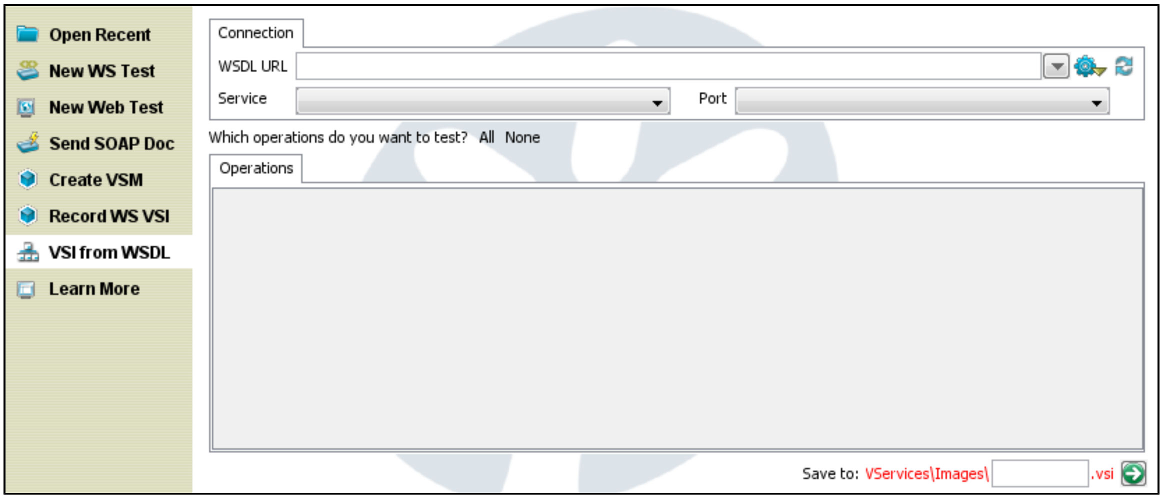
Note: Be sure to update your endpoints:
From: http://[server[:port]]/...
To: http://Frasetto.home/...

Save to: VServices\Images\ .vsi

1. Enter a port on which to listen and record.
2. Enter the URL of the web service to record.
3. Enter the target host and port to record.
4. Use the SSL check box to determine whether to use SSL with this service image.
5. Enter the name of the VSM and SI, and click the green arrow to create the virtual service.

VSI from WSDL

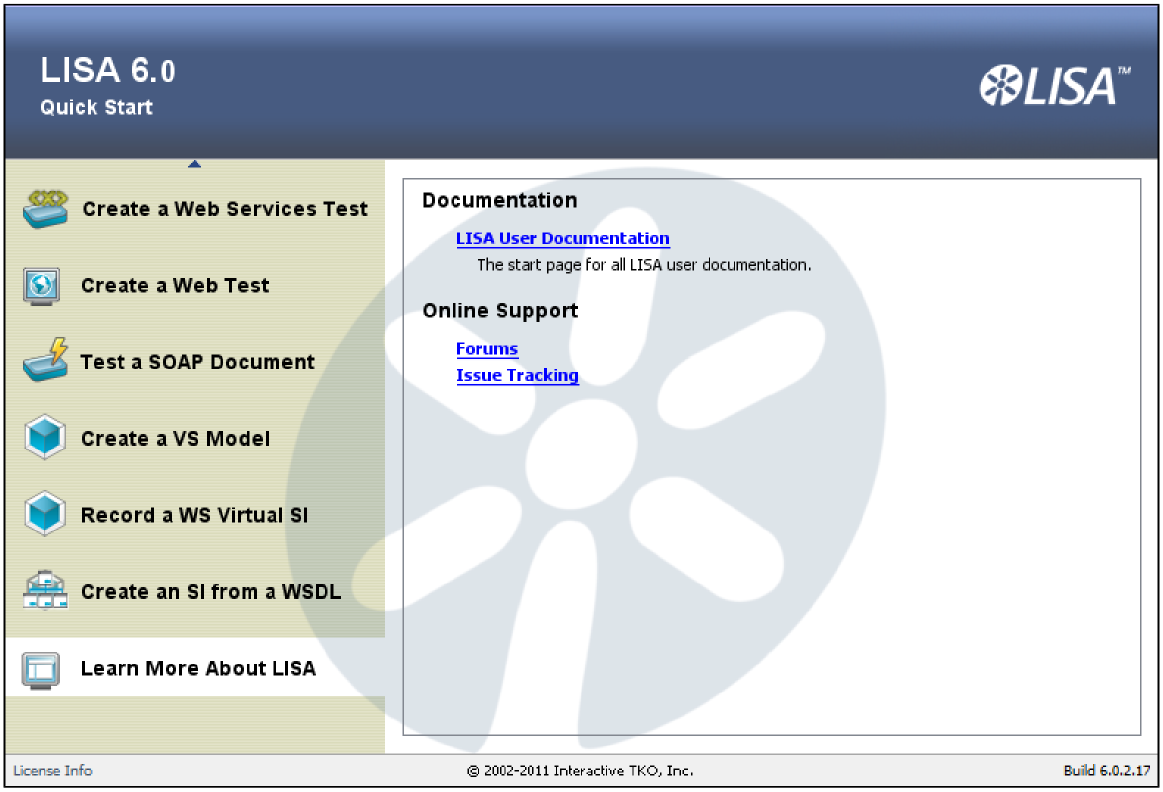
The **VSI from WSDL** option in the Quick Start window lets you create a Virtual Service Image from a WSDL. The properties needed are displayed in the right pane.



1. Enter the WSDL URL, Service and Port related details. If the WSDL URL you enter does not contain LISA properties in its path, Service and Port will be populated automatically.
2. Check the operations that you want to test from **All** or **None** or select each item individually.
3. Enter the name of the service image and select the path where you would like to save it within the project folder.
4. Click the green arrow ➔ to create the service image.

Learn More

The **Learn More** option in the Quick Start window displays a link to LISA documentation and LISA online support information.



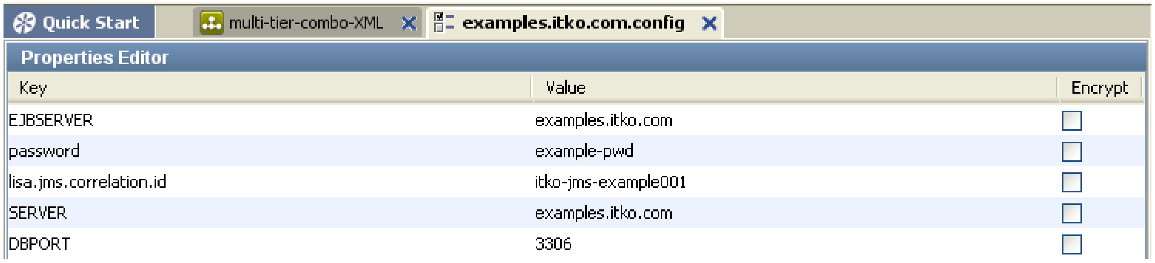
The start page for all LISA user documentation.

All available user documentation for LISA is accessible from this menu, and ITKO online support forums and issue tracking.

Tables

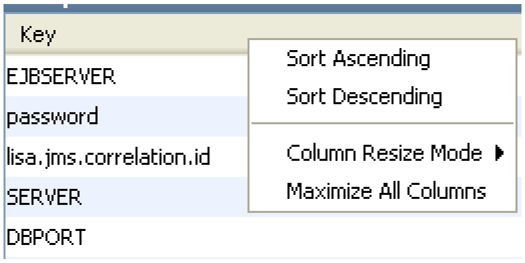
Tables in LISA Workstation have a consistent look and feel, and you can use the same features to customize the displays.

They all have the same banding.



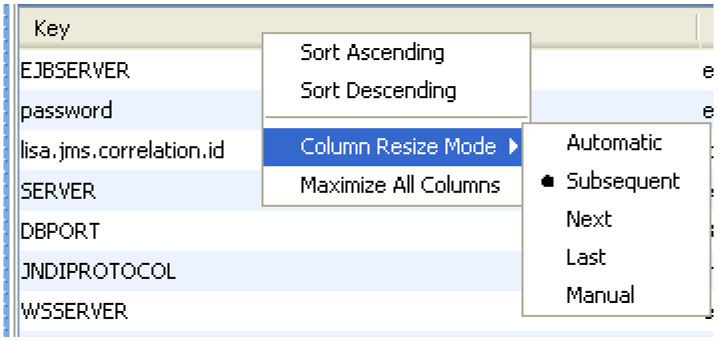
Key	Value	Encrypt
EJBSERVER	examples.itko.com	□
password	example-pwd	□
lisa.jms.correlation.id	itko-jms-example001	□
SERVER	examples.itko.com	□
DBPORT	3306	□

They all have the ability to change their column sizes and sorting (if sortable) through a right-click menu.



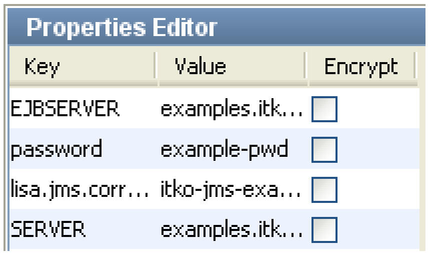
Key
EJBSERVER
password
lisa.jms.correlation.id
SERVER
DBPORT

Selecting Maximizing All Columns changes the Column Resize Mode to Manual.



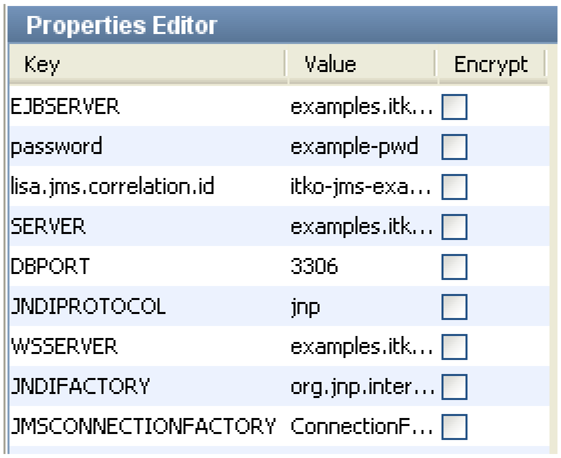
Key
EJBSERVER
password
lisa.jms.correlation.id
SERVER
DBPORT
JNDIPROTOCOL
WSSERVER

You can also double-click on the column resizers (the column header between columns) to maximize the current column (or selected columns).



Key	Value	Encrypt
EJBSERVER	examples.itk...	□
password	example-pwd	□
lisa.jms.corr...	itko-jms-exa...	□
SERVER	examples.itk...	□

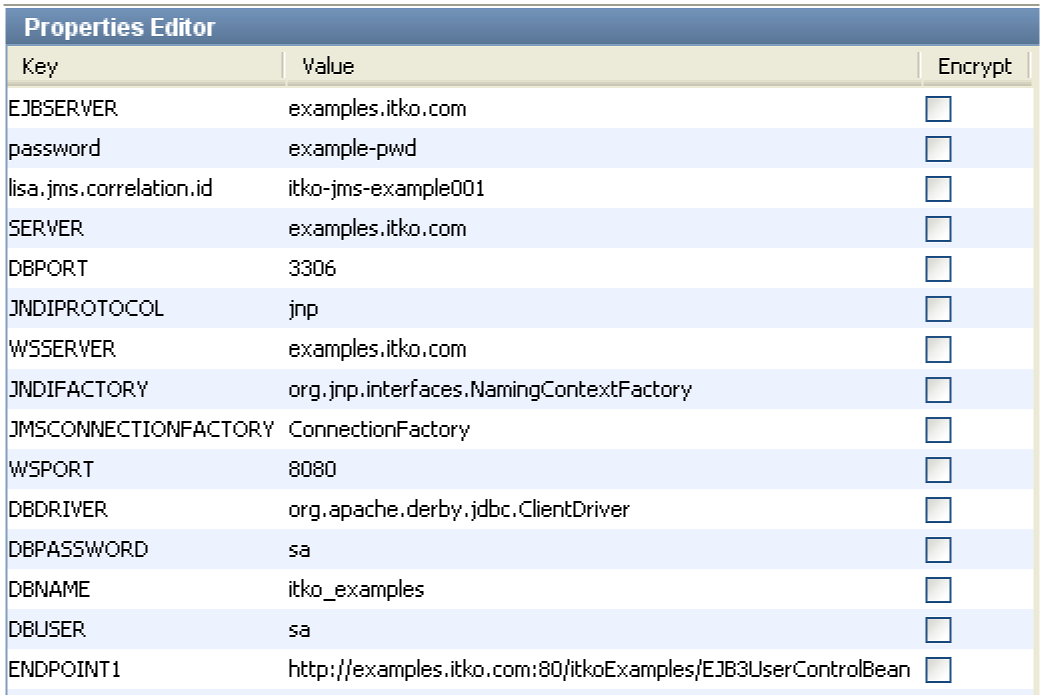
This shows the results after double-clicking the Key column resizer.



Properties Editor

Key	Value	Encrypt
EJBSERVER	examples.itk...	□
password	example-pwd	□
lisa.jms.correlation.id	itko-jms-exa...	□
SERVER	examples.itk...	□
DBPORT	3306	□
JNDIPROTOCOL	jnp	□
WSSERVER	examples.itk...	□
JNDIFACTORY	org.jnp.inter...	□
JMSCONNECTIONFACTORY	ConnectionF...	□

This shows the results after double-clicking the Key column resizer.

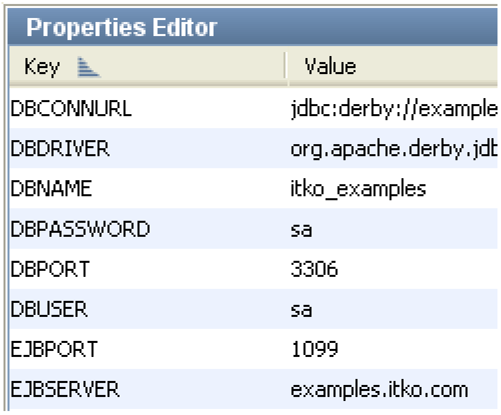


Properties Editor

Key	Value	Encrypt
EJBSERVER	examples.itko.com	□
password	example-pwd	□
lisa.jms.correlation.id	itko-jms-example001	□
SERVER	examples.itko.com	□
DBPORT	3306	□
JNDIPROTOCOL	jnp	□
WSSERVER	examples.itko.com	□
JNDIFACTORY	org.jnp.interfaces.NamingContextFactory	□
JMSCONNECTIONFACTORY	ConnectionFactory	□
WSPORT	8080	□
DBDRIVER	org.apache.derby.jdbc.ClientDriver	□
DBPASSWORD	sa	□
DBNAME	itko_examples	□
DBUSER	sa	□
ENDPOINT1	http://examples.itko.com:80/itkoExamples/EJB3UserControlBean	□

You can also click on the column header to toggle the column sort state (for sortable tables). All tables show sort indicators.

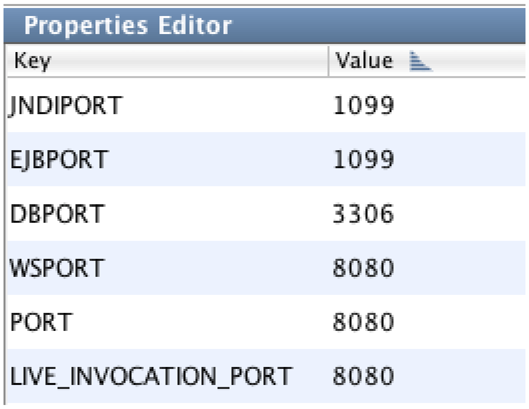
Sort Key column ascending:



Properties Editor

Key	Value
DBCONNURL	jdbc:derby://example
DBDRIVER	org.apache.derby.jdb
DBNAME	itko_examples
DBPASSWORD	sa
DBPORT	3306
DBUSER	sa
EJBPORT	1099
EJBSERVER	examples.itko.com

Sort Value column ascending:

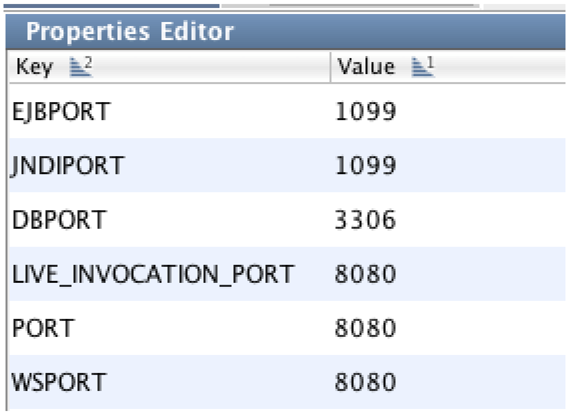


Properties Editor

Key	Value
JNDIPORT	1099
EJBPORT	1099
DBPORT	3306
WSPORT	8080
PORT	8080
LIVE_INVOCATION_PORT	8080

You may also multi-sort by holding down the Cntl key on Windows or command key on Mac. A number will show up when multiple columns are sorted to indicate sort order.

(Value Ascending, Key Ascending)

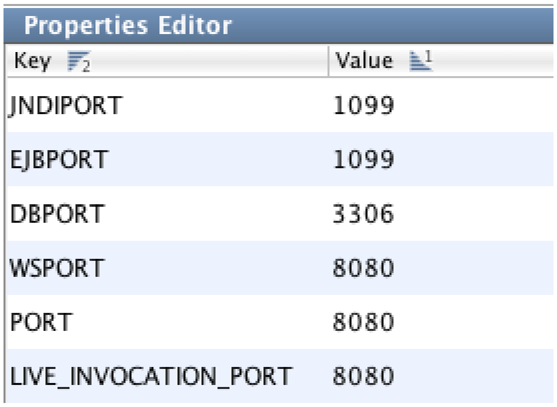


Properties Editor

Key 2	Value 1
EJBPORT	1099
JNDIPORT	1099
DBPORT	3306
LIVE_INVOCATION_PORT	8080
PORT	8080
WSPORT	8080

While holding down the control/command key it will toggle between ascending and descending (skipping the reset state), but you can control/command right-click to bring up the menu if you want to reset the sort order.

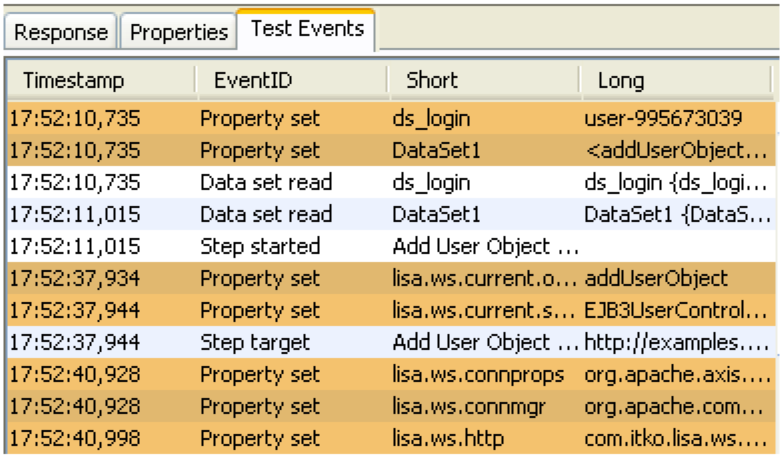
(Value Ascending, Key Descending)



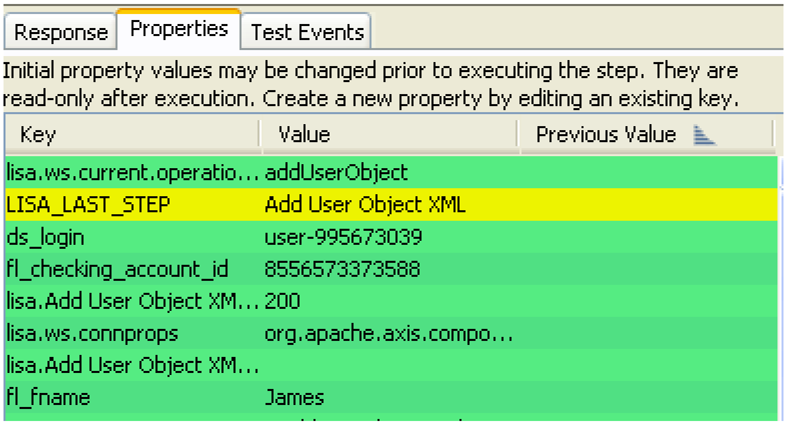
Properties Editor

Key 2	Value 1
JNDIPORT	1099
EJBPORT	1099
DBPORT	3306
WSPORT	8080
PORT	8080
LIVE_INVOCATION_PORT	8080

All tables are banded, even ones where the rows are colored.



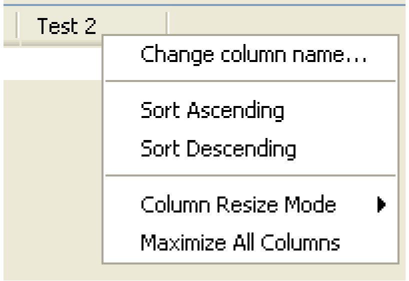
Timestamp	EventID	Short	Long
17:52:10,735	Property set	ds_login	user-995673039
17:52:10,735	Property set	DataSet1	<addUserObject…
17:52:10,735	Data set read	ds_login	ds_login {ds_logi…
17:52:11,015	Data set read	DataSet1	DataSet1 {DataS…
17:52:11,015	Step started	Add User Object …	
17:52:37,934	Property set	lisa.ws.current.o…	addUserObject
17:52:37,944	Property set	lisa.ws.current.s…	EJB3UserControl…
17:52:37,944	Step target	Add User Object …	http://examples.…
17:52:40,928	Property set	lisa.ws.connprops	org.apache.axis.…
17:52:40,928	Property set	lisa.ws.connmgr	org.apache.com…
17:52:40,998	Property set	lisa.ws.http	com.itko.lisa.ws.…



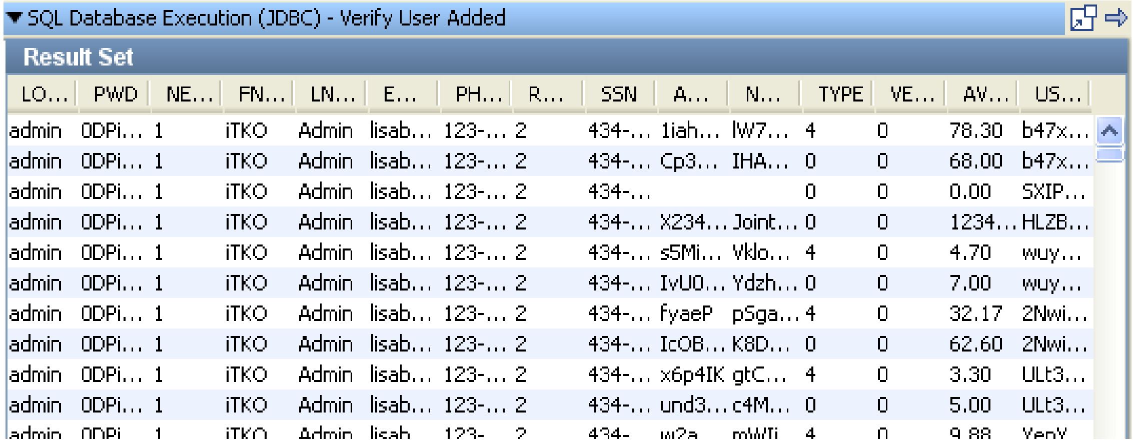
Initial property values may be changed prior to executing the step. They are read-only after execution. Create a new property by editing an existing key.

Key	Value	Previous Value
lisa.ws.current.operatio…	addUserObject	
LISA_LAST_STEP	Add User Object XML	
ds_login	user-995673039	
fl_checking_account_id	8556573373588	
lisa.Add User Object XM…	200	
lisa.ws.connprops	org.apache.axis.compo…	
lisa.Add User Object XM…		
fl_fname	James	

Some tables add additional actions in their right-click menu. The Data Sheet Dataset editor is shown, with a selection for changing the column name.



The JDBC result set is shown in the following image.

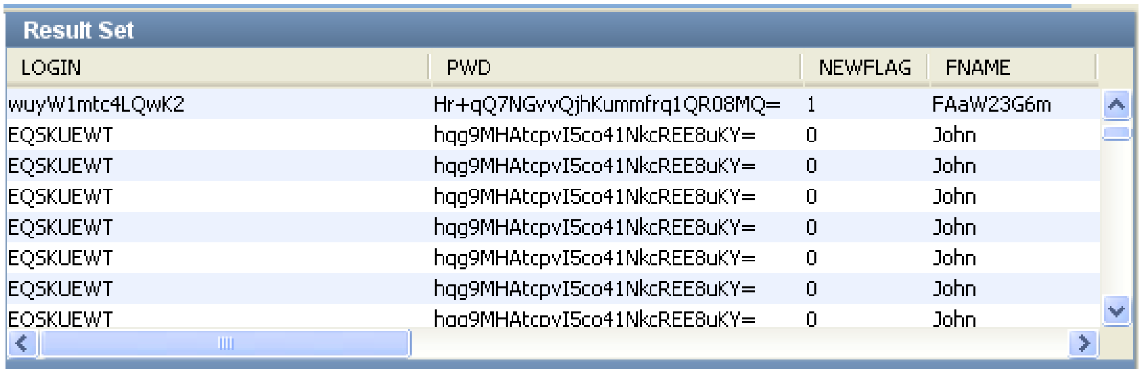


SQL Database Execution (JDBC) - Verify User Added

Result Set

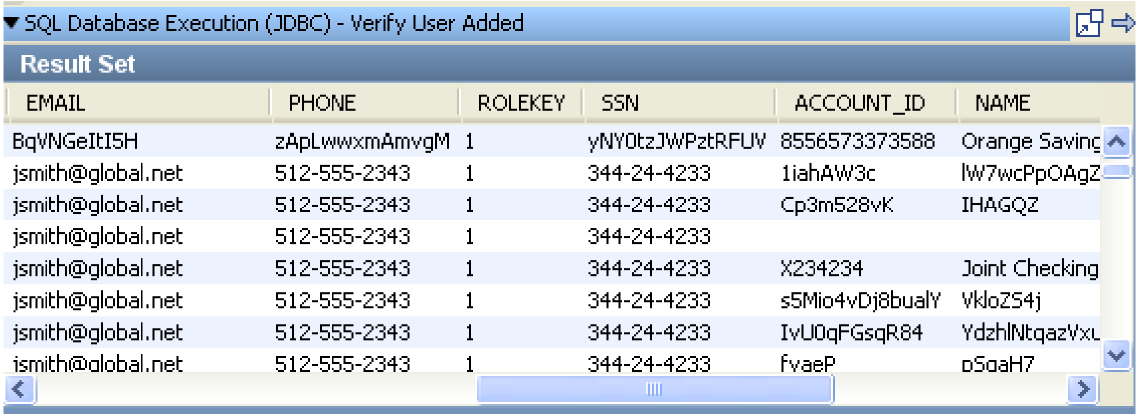
LO...	PWD	NE...	FN...	LN...	E...	PH...	R...	SSN	A...	N...	TYPE	VE...	AV...	US...
admin	ODPi...	1	iTKO	Admin	lisab...	123-...	2	434-...	1iah...	lW7...	4	0	78.30	b47x...
admin	ODPi...	1	iTKO	Admin	lisab...	123-...	2	434-...	Cp3...	IHA...	0	0	68.00	b47x...
admin	ODPi...	1	iTKO	Admin	lisab...	123-...	2	434-...			0	0	0.00	SXIP...
admin	ODPi...	1	iTKO	Admin	lisab...	123-...	2	434-...	X234...	Joint...	0	0	1234...	HLZB...
admin	ODPi...	1	iTKO	Admin	lisab...	123-...	2	434-...	s5Mi...	Vklo...	4	0	4.70	wuy...
admin	ODPi...	1	iTKO	Admin	lisab...	123-...	2	434-...	IvU0...	Ydzh...	0	0	7.00	wuy...
admin	ODPi...	1	iTKO	Admin	lisab...	123-...	2	434-...	fyaeP	pSga...	4	0	32.17	2Nwi...
admin	ODPi...	1	iTKO	Admin	lisab...	123-...	2	434-...	IcOB...	K8D...	0	0	62.60	2Nwi...
admin	ODPi...	1	iTKO	Admin	lisab...	123-...	2	434-...	x6p4IK	gtC...	4	0	3.30	ULt3...
admin	ODPi...	1	iTKO	Admin	lisab...	123-...	2	434-...	und3...	c4M...	0	0	5.00	ULt3...
admin	ODPi	1	iTKO	Admin	lisab	123-	2	434-	w2a	mWTi	4	0	9.88	YenY

You can use the Maximize option so the columns are all displayed, and scroll bars are available.



Result Set

LOGIN	PWD	NEWFLAG	FNAME
wuyW1mtc4LQwK2	Hr+qQ7NGvvQjhKummfrq1QR08MQ=	1	FAaW23G6m
EQSKUEWT	hqg9MHAtcpvI5co41NkcREE8uKY=	0	John
EQSKUEWT	hqg9MHAtcpvI5co41NkcREE8uKY=	0	John
EQSKUEWT	hqg9MHAtcpvI5co41NkcREE8uKY=	0	John
EQSKUEWT	hqg9MHAtcpvI5co41NkcREE8uKY=	0	John
EQSKUEWT	hqg9MHAtcpvI5co41NkcREE8uKY=	0	John
EQSKUEWT	hqg9MHAtcpvI5co41NkcREE8uKY=	0	John
EQSKUEWT	hqg9MHAtcpvI5co41NkcREE8uKY=	0	John

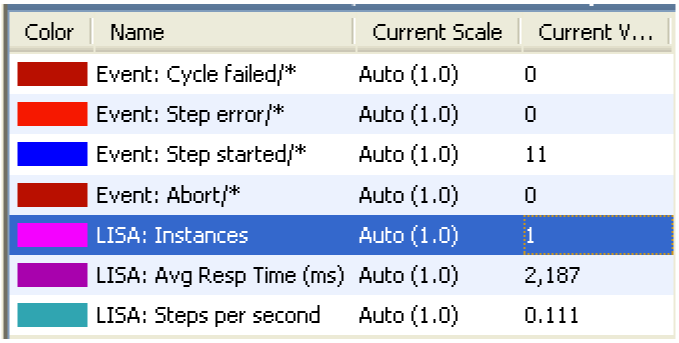


SQL Database Execution (JDBC) - Verify User Added

Result Set

EMAIL	PHONE	ROLEKEY	SSN	ACCOUNT_ID	NAME
BqVNGeItI5H	zApLwwxmAmvgM	1	yNY0tzJWPztRFUV	8556573373588	Orange Saving
jsmith@global.net	512-555-2343	1	344-24-4233	1iahAW3c	lW7wcPpOAgZ
jsmith@global.net	512-555-2343	1	344-24-4233	Cp3m528vK	IHAGQZ
jsmith@global.net	512-555-2343	1	344-24-4233		
jsmith@global.net	512-555-2343	1	344-24-4233	X234234	Joint Checking
jsmith@global.net	512-555-2343	1	344-24-4233	s5Mio4vDj8bualY	VkloZS4j
jsmith@global.net	512-555-2343	1	344-24-4233	IvU0qFGsqR84	YdzhlNtqazVxu
jsmith@global.net	512-555-2343	1	344-24-4233	fyaeP	pSgaH7

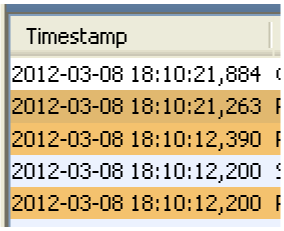
Color object are rendered differently in tables:



Color	Name	Current Scale	Current V...
🟥	Event: Cycle failed/*	Auto (1.0)	0
🟥	Event: Step error/*	Auto (1.0)	0
🟦	Event: Step started/*	Auto (1.0)	11
🟥	Event: Abort/*	Auto (1.0)	0
🟪	LISA: Instances	Auto (1.0)	1
🟪	LISA: Avg Resp Time (ms)	Auto (1.0)	2,187
🟦	LISA: Steps per second	Auto (1.0)	0.111

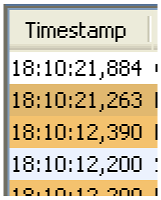
Date objects have a renderer that support smaller formats when the column shrinks.

Fullsize



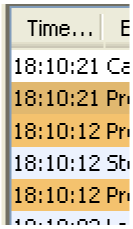
Timestamp
2012-03-08 18:10:21,884
2012-03-08 18:10:21,263
2012-03-08 18:10:12,390
2012-03-08 18:10:12,200
2012-03-08 18:10:12,200

Midsize (no date)



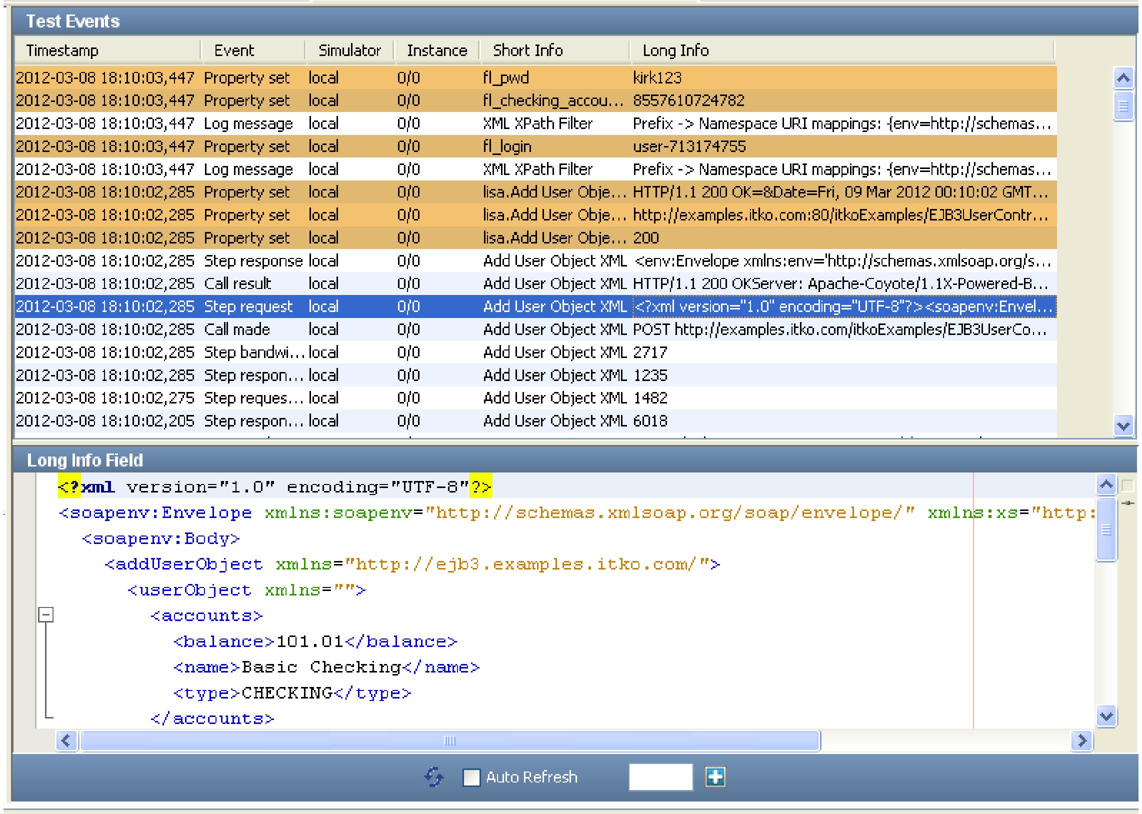
Timestamp
18:10:21,884
18:10:21,263
18:10:12,390
18:10:12,200

Compact (no ms)



Time…	E
18:10:21	Ca
18:10:21	Pr
18:10:12	Pr
18:10:12	St
18:10:12	Pr

Event Tables have a Long Info Field panel and they resize initially to maximize the first four columns and average the last two.



Timestamp	Event	Simulator	Instance	Short Info	Long Info
2012-03-08 18:10:03,447	Property set	local	0/0	fl_pwd	kirk123
2012-03-08 18:10:03,447	Property set	local	0/0	fl_checking_accou…	8557610724782
2012-03-08 18:10:03,447	Log message	local	0/0	XML XPath Filter	Prefix -> Namespace URI mappings: {env=http://schemas…
2012-03-08 18:10:03,447	Property set	local	0/0	fl_login	user-713174755
2012-03-08 18:10:03,447	Log message	local	0/0	XML XPath Filter	Prefix -> Namespace URI mappings: {env=http://schemas…
2012-03-08 18:10:02,285	Property set	local	0/0	lisa.Add User Obje…	HTTP/1.1 200 OK=&Date=Fri, 09 Mar 2012 00:10:02 GMT…
2012-03-08 18:10:02,285	Property set	local	0/0	lisa.Add User Obje…	http://examples.itko.com:80/itkoExamples/EJB3UserContr…
2012-03-08 18:10:02,285	Property set	local	0/0	lisa.Add User Obje…	200
2012-03-08 18:10:02,285	Step response	local	0/0	Add User Object XML	<env:Envelope xmlns:env='http://schemas.xmlsoap.org/s…
2012-03-08 18:10:02,285	Call result	local	0/0	Add User Object XML	HTTP/1.1 200 OKServer: Apache-Coyote/1.1X-Powered-B…
2012-03-08 18:10:02,285	Step request	local	0/0	Add User Object XML	<?xml version="1.0" encoding="UTF-8"?><soapenv:Envel…
2012-03-08 18:10:02,285	Call made	local	0/0	Add User Object XML	POST http://examples.itko.com/itkoExamples/EJB3UserCo…
2012-03-08 18:10:02,285	Step bandwi…	local	0/0	Add User Object XML	2717
2012-03-08 18:10:02,285	Step respon…	local	0/0	Add User Object XML	1235
2012-03-08 18:10:02,275	Step reques…	local	0/0	Add User Object XML	1482
2012-03-08 18:10:02,205	Step respon…	local	0/0	Add User Object XML	6018

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xs="http:
  <soapenv:Body>
    <addUserObject xmlns="http://ejb3.examples.itko.com/">
      <userObject xmlns="">
        <accounts>
          <balance>101.01</balance>
          <name>Basic Checking</name>
          <type>CHECKING</type>
        </accounts>
```

Tray Panels

LISA Workstation has tray panels for certain features, such as the Output Log Message step.

As of release 6.0.6, you can control whether the opening and closing of tray panels use animation. By default, animation is disabled to help improve performance for users who access LISA Workstation through a remote desktop.

To enable the animation, add the following line to the **site.properties** or **local.properties** file:

```
lisa.ui.tray.animation=true
```

LISA Console

The web-based LISA Console provides access to the following consoles and dashboards:

- Continuous Validation Service
- LISA Pathfinder
- Server Console
- Reporting Dashboard

Continuous Validation Service

The Continuous Validation Service (CVS) lets you schedule tests and test suites to run on a regular basis over an extended time period.

For more information, see "Continuous Validation Service (CVS)."

LISA Pathfinder

Pathfinder lets you probe into the system under test, to examine the components behind the initial request or method call.

For more information, see the *Pathfinder Guide*.

Server Console

The Server Console enables you to manage labs and to configure role-based access control. The Server Console is also where you access the VSE Dashboard.

For more information, see "Cloud DevTest Labs," "Access Control (ACL)," and "VSE Dashboard."

Reporting Dashboard

A report viewer displays event and metric information, and information derived from data that was captured during the running of tests. You can use a staging document to set the events and metrics that you want to capture for reporting purposes.

For more information, see "Reports."

Opening the LISA Console

You can open the LISA Console from the main menu of LISA Workstation or from a web browser.

The home page of the LISA Console lets you access the Pathfinder Console, the Reporting Dashboard, the Continuous Validation Service, and the Server Console. The lower right area of the home page displays the version number.

To open the LISA Console from the LISA Workstation main menu

1. From the main menu, select View > LISA Portal.
   If access control (ACL) is enabled, then the LISA Console Login dialog appears.
2. Enter your user name and password and click Login.
   The LISA Console appears.

To open the LISA Console from a web browser

1. Ensure that the registry is running.
2. Enter **http://localhost:1505/** in a web browser. If the registry is located on a remote computer, then replace **localhost** with the name or IP address of the computer.
   If access control (ACL) is enabled, then the LISA Console Login dialog appears.
3. Enter your user name and password and click Login.
   The LISA Console appears.

Web Server Timeouts

By default, the web server used by the LISA Console will wait 90 seconds for a process to run on the server. If a process takes longer than 90 seconds, then the connection will be aborted and the client application or browser should handle this appropriately.

You can change the default timeout value by adding the **lisa.webserver.socket.timeout** property to the **local.properties** file. The value is in milliseconds. For example:

```
lisa.webserver.socket.timeout=120000
```

Command-Line Utilities

The following command-line utilities are included in the **LISA_HOME/bin** directory.

CVS Manager

CVS Manager lets you add or remove monitors to the CVS Dashboard through a command-line option. For more information, see CVS Manager.

Make Mar

Make Mar lets you show the contents of MAR info files (standalone or in an archive), or to create model archive files from MAR info files. For more information, see Make Mar.

Service Image Manager

Service Image Manager is used to import transactions into a service image (new or existing), and to combine two or more service images together. For more information, see Service Image Manager.

Service Manager

Service Manager is used to check the status of, reset, or stop a running LISA server process. For more information, see Service Manager.

Test Runner

Test Runner is a "headless" version of LISA Workstation with the same functionality but no user interface. For more information, see Test Runner.

VSE Manager

VSE Manager is used for managing virtual service environments. For more information, see VSE Manager Commands.

Tutorials

This section contains a series of tutorials that illustrate various aspects of LISA. The tutorials are sequential and should be completed in order.

The first few tutorials walk you through using LISA Workstation to build simple test cases. You become familiar with basic concepts such as projects, properties, data sets, filters, and assertions.

The subsequent tutorials illustrate deeper knowledge about how to set up test steps to interact with and test several common technologies, including Java objects (POJOs), web pages, Enterprise JavaBeans (EJBs), web services, and databases. You also learn how to stage a quick test.

Several of these tutorials use the LISA Demo Server as the system under test. You can use the Demo Server installed with LISA (locally on your workstation), or you can reference a similarly configured demo server at http://examples.itko.com/itko-examples/.

For more information about installing the LISA Demo Server, see Downloading the Installer and Installing the Demo Server.

The following tutorials are available.

Tutorial 1 - Projects, Test Cases, and Properties

In this tutorial, you will learn how to create a project and a test case. You will also look at the use of properties to understand the various places from which they can originate.

LISA Concepts Discussed

In this tutorial, you do the following:

- Create and save a new project
- Create and save a new test case
- Create properties
- Add simple test steps
- Use the Interactive Test Run utility

Prerequisites

- LISA Workstation is installed and LISA license credentials are entered.
- You have reviewed Basic Concepts

Steps

Step 1 - Start LISA Workstation

To start LISA Workstation:

1. Ensure that the registry is running. If your computer has an installation of LISA Server, then you can start the registry by clicking **Start Menu > All Programs > LISA > Registry** and waiting until the LISA Registry Ready message appears. If your computer has an

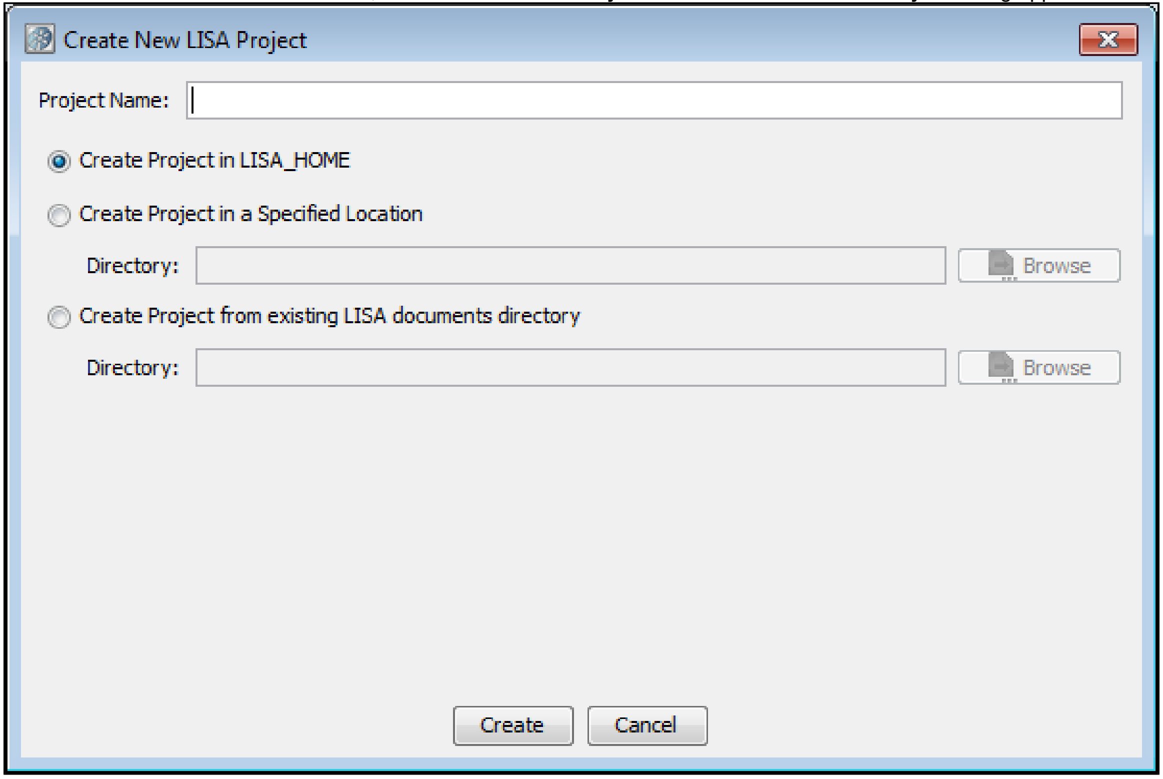
installation of LISA Workstation, you will need to use a registry that is running on another computer.
2. Click **Start Menu > All Programs > LISA > LISAWorkstation**.
3. When the **Set LISA Registry** dialog appears, select a registry and click OK.

Step 2 - Create a Project

The project that you create will hold all the test case example files that are required for the tutorials.

To create a project:

1. From the LISA Workstation main menu, select **File > New > Project**. The **Create New LISA Project** dialog appears.



● Create Project in LISA_HOME
○ Create Project in a Specified Location
○ Create Project from existing LISA documents directory

2. In the Project Name field, type **My Tutorials**.
3. Accept the default setting to create the project in the **LISA_HOME** directory.
4. Click Create. The **My Tutorials** project is created.

Step 3 - Create a Test Case

A test case is a specification of how to test a business component in the system under test.

To create a test case:

1. In the Project Panel, right-click the **Tests** folder and select **Create New Test Case**.
2. Set the file name to **tutorial1**.
3. Click Save. LISA Workstation opens a new tab labeled **tutorial1**. The green arrow in the model editor represents the start of the test case.

Step 4 - Add a Property to the Project Configuration

In this step, you set a global property in the project configuration. You will access this property later in the tutorial.

The default configuration is named **project.config**, and is created automatically for a new project. The **project.config** file is located in the **Configs** folder in the Project Panel. You can add the properties to the **project.config** file and, if required, can also create a new configuration file.

To add a property to the project configuration:

1. In the Project Panel, double-click **project.config** in the **My Tutorials** > **Configs** folder. The properties editor for **project.config** opens.
2. Click the Add icon ⊞ at the bottom of the properties editor to add a new row.
3. In the **Key** field, type **config_prop**.
4. In the **Value** field, type **42**.

5. From the main toolbar, click the Save Save icon.

Step 5 - Add a Test Step

A test case includes one or more test steps. In this procedure, you add an Output Log Message test step to write text to the log file.

To add a test step:

1. Click the **tutorial1** tab.
2. Click the Add Step button ➕, select **Utilities**, and select **Output Log Message**. **Step1** is added to the model editor.
3. Right-click **Step1** and select **Rename**. Change the name to **Log1**.
4. Make sure that **Log1** is still selected. In the right pane, click the arrow next to **Output Log Message**. The Output Log Message tray opens.

Step 6 - Add a Log Message

With the log editor open, you add a log message that includes various properties.

The properties in the log message originate from several sources:

- The **LISA_HOME** property is automatically set.
- The **java.version** property is a system property.
- You added the **config_prop** property to the project configuration in Step 4.
- You create a new property named **Log1_step_prop** in the log message itself.

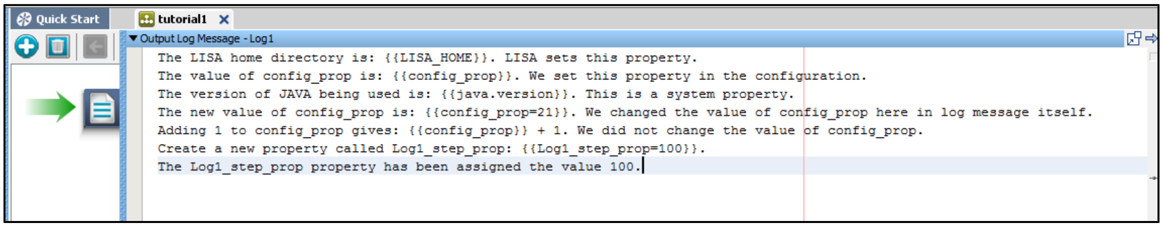
The syntax for a property is {{`property_name`}}.

To add a log message:

1. In the log editor, delete the placeholder text.
2. Copy and paste the following text into the log editor:

```
The LISA home directory is: {{LISA_HOME}}. LISA sets this property.
The value of config_prop is: {{config_prop}}. We set this property in the configuration.
The version of Java being used is: {{java.version}}. This is a system property.
The new value of config_prop is: {{config_prop=21}}. We changed the value of config_prop here in log message itself.
Adding 1 to config_prop gives: {{config_prop}} + 1. We did not change the value of config_prop.
Create a new property named Log1_step_prop: {{Log1_step_prop=100}}.
The Log1_step_prop property has been assigned the value 100.
```

The log editor should look like the following image.



```
The LISA home directory is: {{LISA_HOME}}. LISA sets this property.
The value of config_prop is: {{config_prop}}. We set this property in the configuration.
The version of JAVA being used is: {{java.version}}. This is a system property.
The new value of config_prop is: {{config_prop=21}}. We changed the value of config_prop here in log message itself.
Adding 1 to config_prop gives: {{config_prop}} + 1. We did not change the value of config_prop.
Create a new property called Log1_step_prop: {{Log1_step_prop=100}}.
The Log1_step_prop property has been assigned the value 100.
```

Step 7 - Add a Second Log Message

The second test step in the test case will write a different message to the log file.

To add a second log message:

1. Click the Add Step button ➕, select **Utilities**, and select **Output Log Message**. **Step1** is added to the model editor.
2. Right-click **Step1** and select **Rename**. Change the name to **Log2**.
3. Make sure that **Log2** is still selected. In the right pane, click the arrow next to **Output Log Message**. The Output Log Message tray opens.
4. In the log editor, delete the placeholder text.
5. Copy and paste the following text into the log editor:

```
The current value of config_prop is: {{config_prop}}.
The current value of Log1_step_prop: {{Log1_step_prop}}.
```

⚠ The log message does not change the values of **config_prop** or **Log1_step_prop**.

6. From the main application toolbar, click the Save icon, or select **File > Save**.

Step 8 - Run the Log1 Step

The Interactive Test Run (ITR) utility enables you to walk through and verify a test case.

To run the Log1 step:

1. From the toolbar, click the Start a new ITR icon . The ITR opens. The ITR contains an Execution History pane on the left and a set of tabs on the right.
2. In the Execution History pane, click the Execute Next Step icon.
   The **Log1** step is run. The **Response** tab displays the response from the **Log1** step. The properties have been replaced by actual values.

Step 9 - Observe Property Values

The ITR also enables you to observe how the properties are created and modified.

To observe property values:

1. Click the **Properties** tab in the ITR.
   The **Properties** tab displays the value of each property before and after the execution of the **Log1** step.
   A value that was created by the step is highlighted in green. A value that was modified in the step is highlighted in yellow. Notice that the value of **config_prop** was changed from 42 to 21.
2. Compare these values with the response in Step 8.

Step 10 - Run the Log2 Step

In this procedure, you use the ITR to run the second step in the test case.

To run the Log2 step:

1. In the ITR, click the Execute Next Step icon to run the **Log2** step.
2. Click the **Response** tab to view the response. Even though you set **config_prop** to 42 in the **project.config** file, you changed the value to 21 in the **Log1** step, and the value did not change in the **Log2** step. The value of the **Log1_step_prop** property also carried over from the **Log1** step to the **Log2** step.
3. Click the **Properties** tab to view the current and previous property values.
4. When you are done, close the **tutorial1** and **project.config** tabs.

Review

In this tutorial, you took a first look at properties. You saw that properties are denoted by using a special syntax, {{`property_name`}}. You can set properties by using a variation of this syntax, {{`property_name=value`}}. After you set a property, use or modify it in subsequent steps in a test case.

In this tutorial, you did the following:

- Learned how to create and save a test case
- Learned how to add a simple test step (Output Log Message)
- Used a configuration to store properties
- Saw a brief glimpse of the Interactive Test Run utility

Tutorial 2 - Data Sets

In this tutorial, you will learn how to create and use a simple data set. You will also learn how to provide this data set's data to a test case.

LISA Concepts Discussed

In this tutorial, you do the following:

- Create a simple data set
- Use the data set in a variety of ways
- Iterate through a series of test steps using a data set

Prerequisites

- You have completed Tutorial 1 - Projects, Test Cases, and Properties.
- LISA Workstation is open.

Steps

Step 1 - Create a Data Set

You will use a comma-delimited text file as the data set. This option is just one of several options available to create a data set. After you create the text file, you import it into the **My Tutorials** project.

To create a data set:

1. In a text editor such as Notepad, create a text file. Copy and paste the following properties and values into the text editor. Do not use spaces in the text file.

   ```
   month,day,year
   3,2,1956
   4,7,2007
   1,3,2010
   5,8,{{yearglobal}}
   8,10,2004
   12,11,{{yearglobal}}
   10,12,2007
   3,5,2011
   ```

   The first row specifies the names of the properties to which this data is assigned (month, day, year). The remaining rows specify the data that is read and used in the test case. Two of the rows include a property named **yearglobal**.

2. Save the file as **dates.txt**.
3. In the Project panel, right-click the Data folder in the **My Tutorials** project and select Import Files.
4. Navigate to the folder where you saved the **dates.txt** file and select the file name.
5. Click Open. The **dates.txt** file now appears in the Data folder.

Step 2 - Create a Test Case

You will add a new test case to the **My Tutorials** project.

To create a test case:

1. In the Project panel, right-click the Tests folder and select Create New Test Case.
2. Set the file name to **tutorial2**.
3. Click Save.

Step 3 - Add a Property to the Project Configuration

The **dates.txt** file includes a property named **yearglobal**. In this procedure, you add the **yearglobal** property to the project configuration.

To add a property to the project configuration:

1. In the Project panel, double-click **project.config**.
2. Click the Add icon ⊞ to add a new row.
3. In the **Key** field, enter **yearglobal**.
4. In the **Value** field, enter **1999**.
5. Click the Save Save icon.

Step 4 - Add a Test Step for Output Log Message

Use a test step, the **Output Log Message** step, to write text out to the log.

To add a test step for Output Log Message:

1. Click the **tutorial2** tab.
2. Click the Add Step ⊞ icon. The **Add step** menu is displayed.
3. Select **Utilities** and select **Output Log Message**. **Step1** is added to the model editor.
4. Right-click **Step1** and select Rename. Change the name to **DSstep1**.
5. In the right pane, click the arrow next to **Output Log Message**. The Output Log Message tray opens.
6. Delete the placeholder text.
7. Enter the following log message:

```
Date is: {{month}}/{{day}}/{{year}}
```

> ⚠ The curly brackets are important. If you do not include them, the test case will not run correctly.

8. Click anywhere in the model editor to close the Output Log Message tray.
9. Click the Save icon Save .

Step 5 - Create Another Output Log Message Step

Create another test step similar to the **DSstep1** test step.

To create another output log message step:

1. Click the Add Step ⊞ icon. The **Add step** menu is displayed.
2. **Select Utilities** and select **Output Log Message**. **Step1** is added to the model editor.
3. Right-click **Step1** and select Rename. Change the name to **DSstep2**.
4. In the right pane, click the arrow next to **Output Log Message**. The Output Log Message tray opens.
5. Delete the placeholder text.
6. Enter the following log message:

```
Date is: {{month}}/{{day}}/{{year}}
```

7. Click anywhere in the model editor to close the Output Log Message tray.
8. Click the Save icon Save .

Step 6 - Execute the Test

Use the Interactive Test Run (ITR) utility to execute the test and see what happens.

To execute the test:

1. From the toolbar, click Start ITR. The ITR opens.
2. In the Execution History pane, click the Automatically execute test icon.
3. When the test is complete, click OK.
4. In the Execution History pane, click DSstep1 and DSstep2. Notice that the month, day, and year properties have not been replaced with actual values. This result is expected, because you have not added the data set to the test case.

Step 7 - Add the Data Set

You now add the **dates.txt** data set to the **DSstep1** test step.

To add the data set:

1. In the model editor, select **DSstep1**.
2. In the right pane, double-click on the **Data Sets** step element tab.

3. Click the Add icon + below the Data Sets element.
4. From the **Common DataSets** list, select **Read rows from a delimited data file**. The data set is added to the test step. The data set editor opens in the right pane.
5. In the data set editor, set the name to **DatesDS**.
6. Click the File Location browse button ..., and then navigate to and select the **dates.txt** file in the **LISA_HOME/Projects/My Tutorials/Data** directory.
7. Click the Test and Keep button.
8. If the test is successful, the **Data Set Editor** window returns a "Test successful" message.
9. Click OK.
10. From the toolbar, click the Start ITR button and then select the **Start new ITR** option.
11. In the Execution History pane, click the Automatically execute test icon.
12. When the test is complete, click OK.
13. In the **Execution History** pane, click **DSstep1** and **DSstep2**. The first row of data in the data set is displayed in the **Response** tab. Both step responses display the same date because we read only from the data set in **DSstep1**.

Step 8 - Change the Data Set Behavior

You now modify the data set so that it loops through the test step until all the rows in the data set have been read.

To change the data set behavior:

1. In the model editor, select the **DSstep1** test step.
2. In the step element pane of **DSstep1**, click the arrow next to **DatesDS** under the Data Sets element. The data set editor opens.
3. In the **At end of data** field, select the **Execute** option.
4. Click the drop-down arrow on the **Execute** field and select **End the Test** from the list of choices that appear. This setting will cause the test to end when all the data rows have been read.
5. Click the Test and Keep button.
6. Click OK to close the test successful message.
7. In the model editor, select the **DSstep2** test step.
8. In the **Step Information** element tab, set the **Next** drop-down list to **DSstep1**. This setting will cause the two test steps to loop. The arrows in the model editor show the order of execution: **DSstep1**, followed by **DSstep2**, followed by **DatesDS**.
9. From the toolbar, click the Start ITR button and then select the **Start new ITR** option.
10. In the ITR, click the Automatically execute test icon.
    The test case runs in a loop until there are no more data rows in the data set.
11. When the test is complete, click OK.
12. Click the Save icon Save .

Review

In this tutorial, you did the following:

- Created a comma-delimited data set.
- Used the data set for running a simple test case.
- Learned how a test step accesses the data in the data set.

Tutorial 3 - Filters and Assertions

In this tutorial, you will modify the test case created in Tutorial 2 to include a filter and an assertion.

For an introduction to filters and assertions, see *Basic Concepts*.

LISA Concepts Discussed

In this tutorial, you do the following:

- Save an existing test case with a new name.
- Add an assertion to a test step.
- Add a filter to a test step.

Prerequisites

- You have completed Tutorial 2 - Data Sets.
- LISA Workstation is open.

Steps

Step 1 - Create a New Test Case from an Existing Test Case

In this step, you open **tutorial2.tst** and save it as **tutorial3.tst**.

To create a new test case from an existing test case:

1. Open the **tutorial2.tst** test case in the **My Tutorials** project.
2. From the menu bar, select **File > Save As**.
3. In the **File name** field, enter **tutorial3**.
4. Click Save. The **tutorial3** test case is created and saved under the **My Tutorials** project.

Step 2 - Change Action of Test Step

Change the Next Steps action of both test steps so that **DSstep1** is the next step, and only the first step reads from the data set.

To change the action of the test step:

1. In the model editor, select **DSstep1**.
2. In the Step Information element tab, change the **Next** step to **DSstep1**. With this action, the output goes back to the same step **DSStep1**. For the time being, alert icons appear next to **DSStep2**.
3. In the model editor, double-click **DSstep2** and change the Output Log Message as follows:

   ```
   Date contains 1999. It is: {{month}}/{{day}}/{{year}}.
   ```

   ⚠ The curly brackets are very important. If they are not included, the test case will not run correctly.

4. Click the Save icon.

Step 3 - Add an Assertion

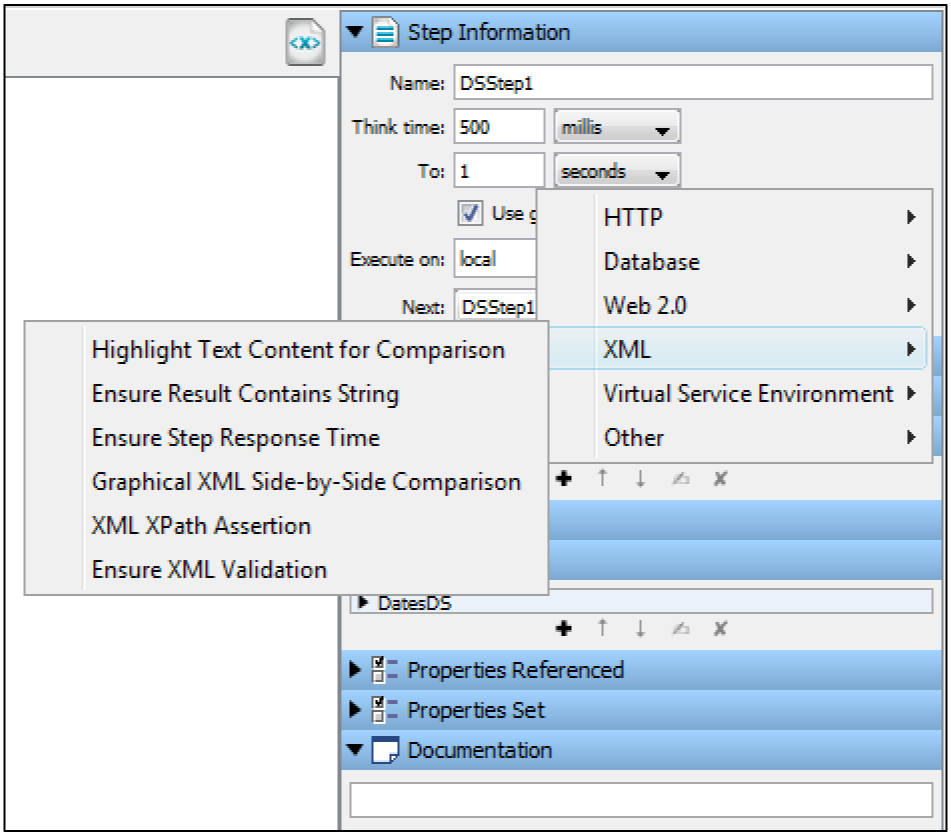
You can add various types of assertions to a test case. In this procedure, you will add an XML assertion named **Ensure Result Contains String**.

The assertion logic is as follows:

- If the response contains the string **1999**, then the **DSstep2** step will be run next.
- If the response does not contain the string **1999**, then the **DSstep1** step will be run next.

To add an assertion:

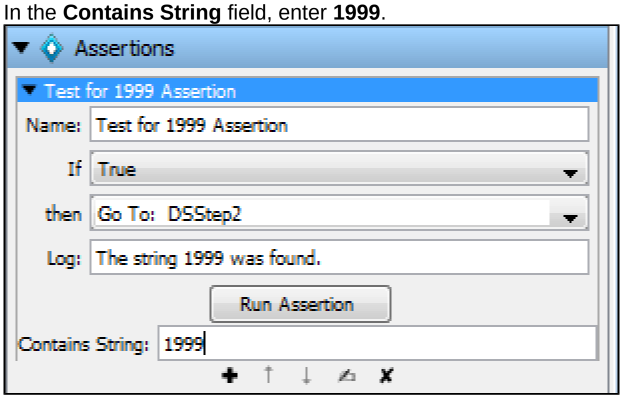
1. In the model editor, select **DSstep1**.
2. Open the **Assertions** element tab.
3. Click the Add ✚ icon.
4. From the **XML** submenu, select **Ensure Result Contains String**.


Step Information
Use g
HTTP
Database
Web 2.0
XML
Virtual Service Environment
Other
Highlight Text Content for Comparison
Ensure Result Contains String
Ensure Step Response Time
Graphical XML Side-by-Side Comparison
XML XPath Assertion
Ensure XML Validation
DatesDS
Properties Referenced
Properties Set
Documentation

Name: DSStep1
Think time: 500 millis
To: 1 seconds
Execute on: local
Next: DSStep1

The new assertion applied to DSstep1 is added to the **Assertions** tab. The assertion editor opens.

5. In the assertion editor, do the following:
   - In the **Name** field, enter **Test for 1999 Assertion**.
   - In the **If** list, select **True**.
   - In the **then** list, select **Go To: DSstep2**.
   - In the **Log** field, enter **The string 1999 was found**.
   - In the **Contains String** field, enter **1999**.



Name: Test for 1999 Assertion
If True
then Go To: DSStep2
Log: The string 1999 was found.
Contains String: 1999

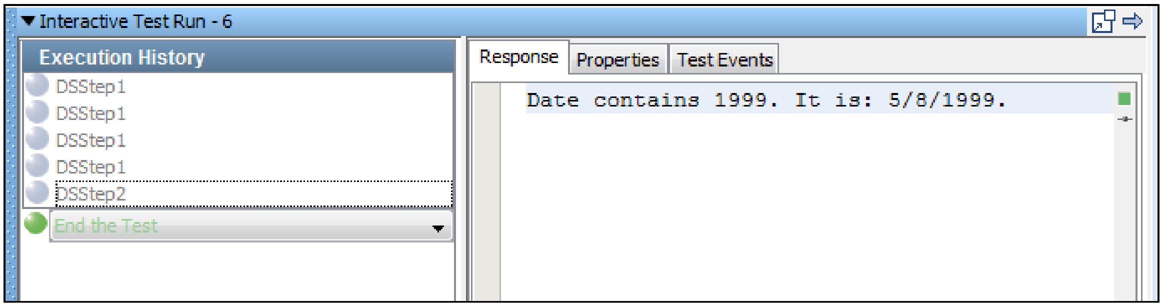
6. Click the Save icon.

Step 4 - Test the Assertion

You can use the Interactive Test Run (ITR) utility to check whether the assertion works as expected.

To test the assertion:

1. Start a new ITR session.
2. In the Execution History pane, click the Automatically execute test icon.
3. When the test is complete, click OK.
4. Review the **Response** tab. Notice that when **DSstep1** encounters a date in which the year is 1999, the **DSstep2** test step is executed next.



Interactive Test Run - 6

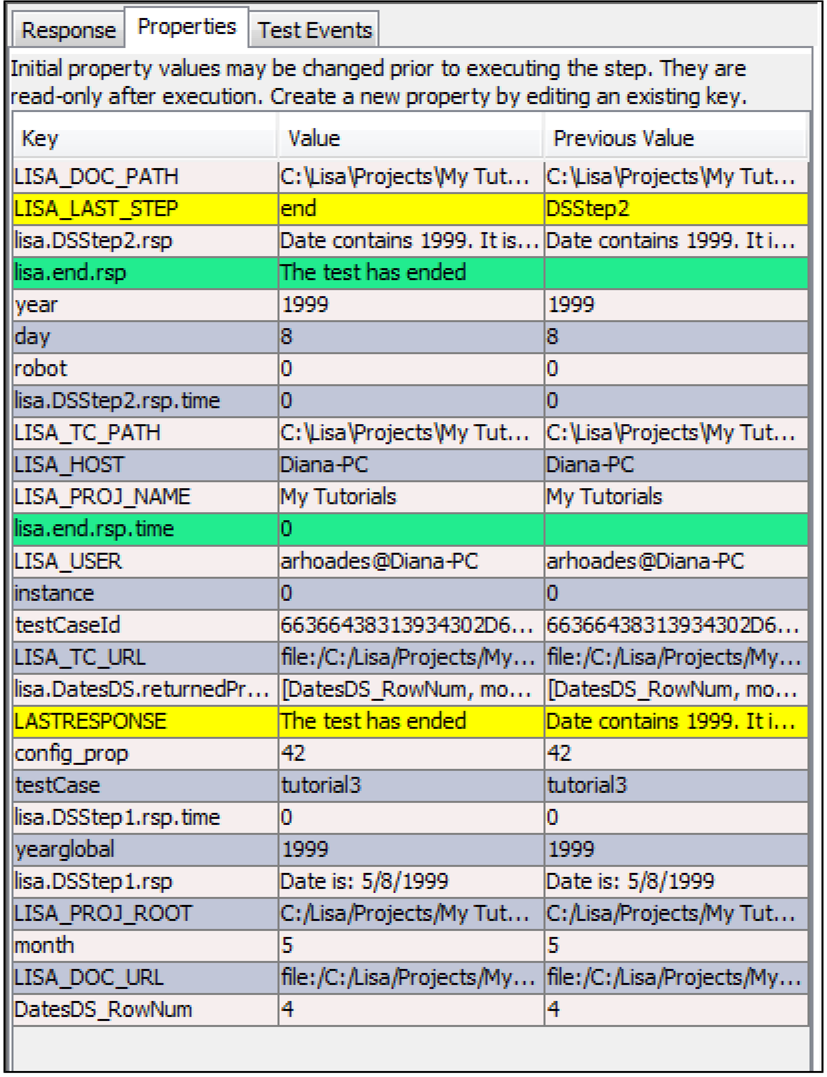
Execution History

- DSStep1
- DSStep1
- DSStep1
- DSStep1
- DSStep2
- End the Test

Response | Properties | Test Events

```
Date contains 1999. It is: 5/8/1999.
```

5. Click the **Properties** tab and review the behavior of the properties.

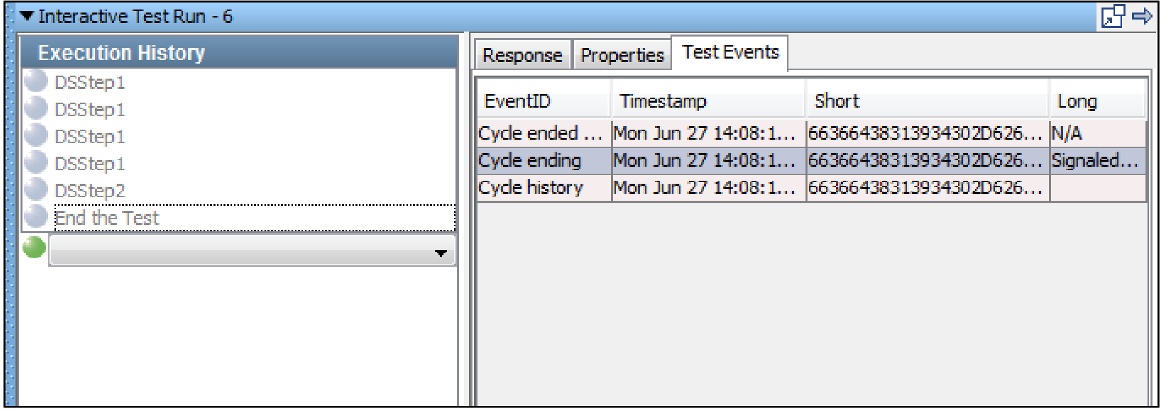


Response | Properties | Test Events

Initial property values may be changed prior to executing the step. They are read-only after execution. Create a new property by editing an existing key.

Key	Value	Previous Value
LISA_DOC_PATH	C:\Lisa\Projects\My Tut…	C:\Lisa\Projects\My Tut…
LISA_LAST_STEP	end	DSStep2
lisa.DSStep2.rsp	Date contains 1999. It is…	Date contains 1999. It i…
lisa.end.rsp	The test has ended	
year	1999	1999
day	8	8
robot	0	0
lisa.DSStep2.rsp.time	0	0
LISA_TC_PATH	C:\Lisa\Projects\My Tut…	C:\Lisa\Projects\My Tut…
LISA_HOST	Diana-PC	Diana-PC
LISA_PROJ_NAME	My Tutorials	My Tutorials
lisa.end.rsp.time	0	
LISA_USER	arhoades@Diana-PC	arhoades@Diana-PC
instance	0	0
testCaseId	66366438313934302D6…	66366438313934302D6…
LISA_TC_URL	file:/C:/Lisa/Projects/My…	file:/C:/Lisa/Projects/My…
lisa.DatesDS.returnedPr…	[DatesDS_RowNum, mo…	[DatesDS_RowNum, mo…
LASTRESPONSE	The test has ended	Date contains 1999. It i…
config_prop	42	42
testCase	tutorial3	tutorial3
lisa.DSStep1.rsp.time	0	0
yearglobal	1999	1999
lisa.DSStep1.rsp	Date is: 5/8/1999	Date is: 5/8/1999
LISA_PROJ_ROOT	C:/Lisa/Projects/My Tut…	C:/Lisa/Projects/My Tut…
month	5	5
LISA_DOC_URL	file:/C:/Lisa/Projects/My…	file:/C:/Lisa/Projects/My…
DatesDS_RowNum	4	4

6. Click the **Test Events** tab and review the events that were generated.



Interactive Test Run - 6

Execution History

- DSStep1
- DSStep1
- DSStep1
- DSStep1
- DSStep2
- End the Test

Response | Properties | Test Events

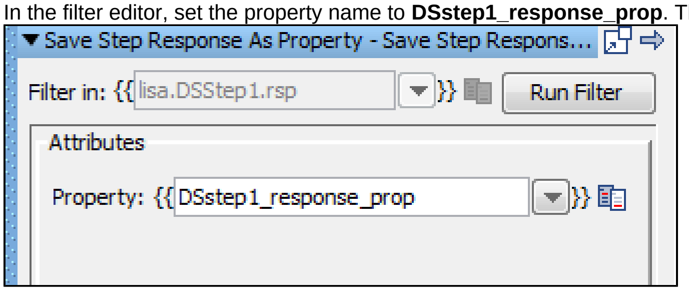
EventID	Timestamp	Short	Long
Cycle ended …	Mon Jun 27 14:08:1…	66366438313934302D626…	N/A
Cycle ending	Mon Jun 27 14:08:1…	66366438313934302D626…	Signaled…
Cycle history	Mon Jun 27 14:08:1…	66366438313934302D626…	

Step 5 - Add a Filter

You can add various types of filters to a test case. In this procedure, you add a utility filter named **Store Step Response**. This type of filter lets you save the step response as a property.

To add a filter:

1. In the model editor, select **DSstep1**.
2. Open the **Filters** element tab.
3. Click the Add icon ✚ .
4. From the **Utility Filters** submenu, select **Store Step Response**. The filter editor opens.
5. In the filter editor, set the property name to **DSstep1_response_prop**. This property is where the step response will be stored.



Filter in: {{lisa.DSStep1.rsp}}

Property: {{DSstep1_response_prop}}

6. In the model editor, double-click **DSstep2**and add the following to the end of the output log message:

```
The value of DSstep1_response_prop is: {{DSstep1_response_prop}}.
```

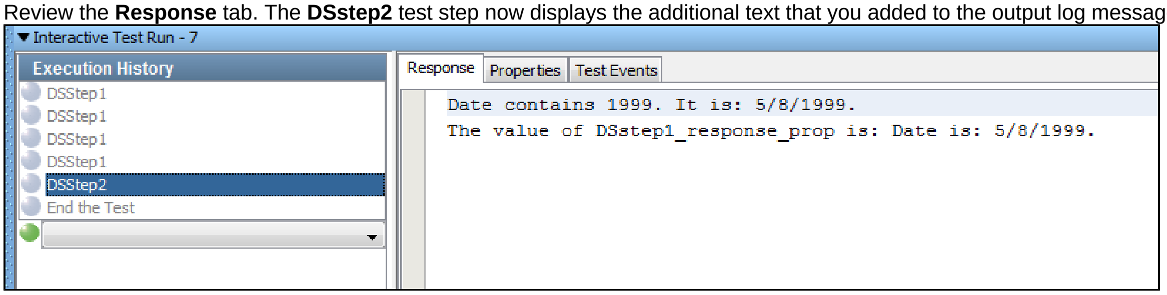
7. Click the Save icon.

Step 6 - Test the Filter

You can use the Interactive Test Run (ITR) utility to check whether the filter works as expected.

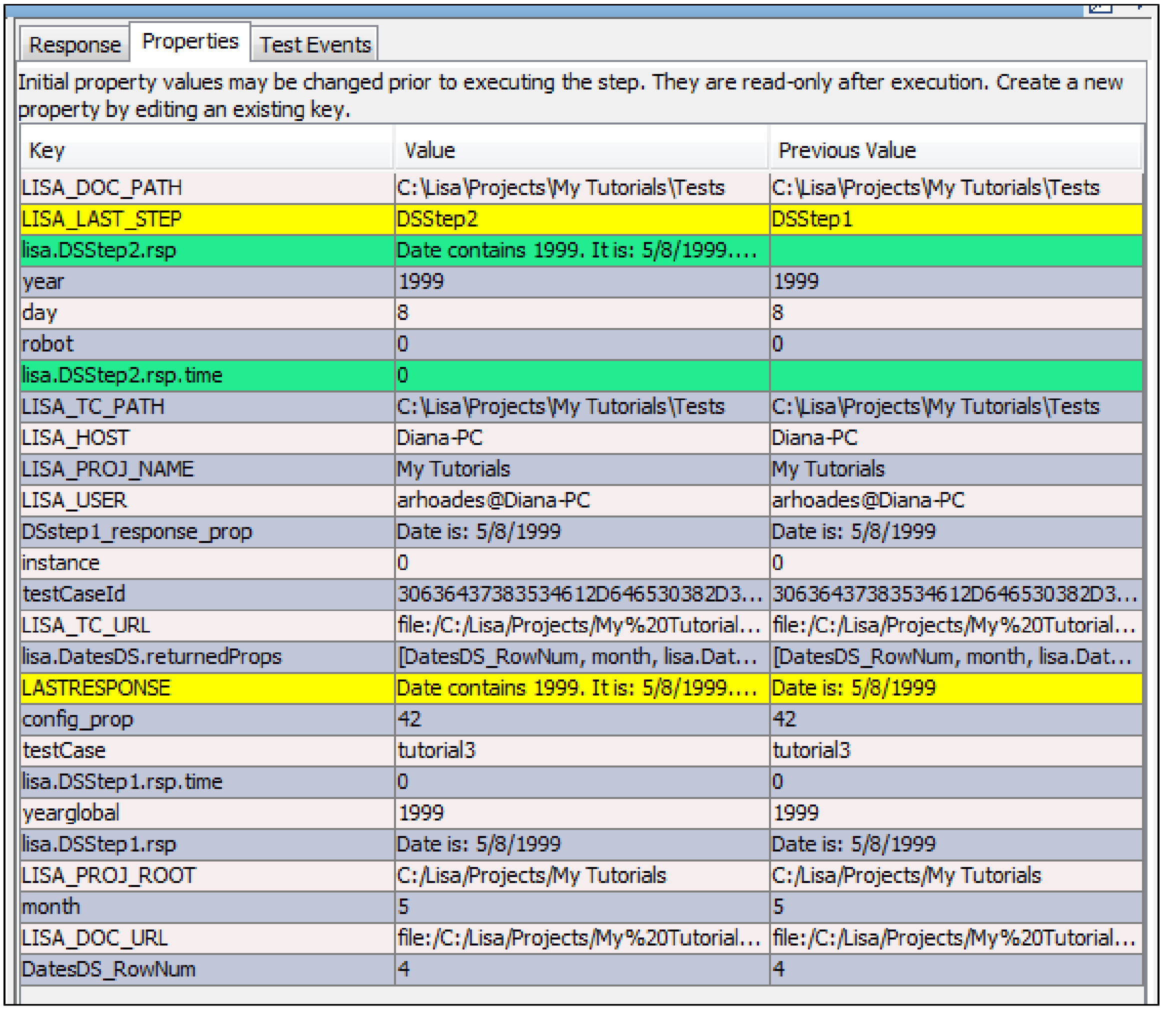
To test the filter:

1. Start a new ITR session.
2. In the Execution History pane, click the Automatically execute test icon.
3. When the test is complete, click OK.
4. Review the **Response** tab. The **DSstep2** test step now displays the additional text that you added to the output log message.



```
Date contains 1999. It is: 5/8/1999.
The value of DSstep1_response_prop is: Date is: 5/8/1999.
```

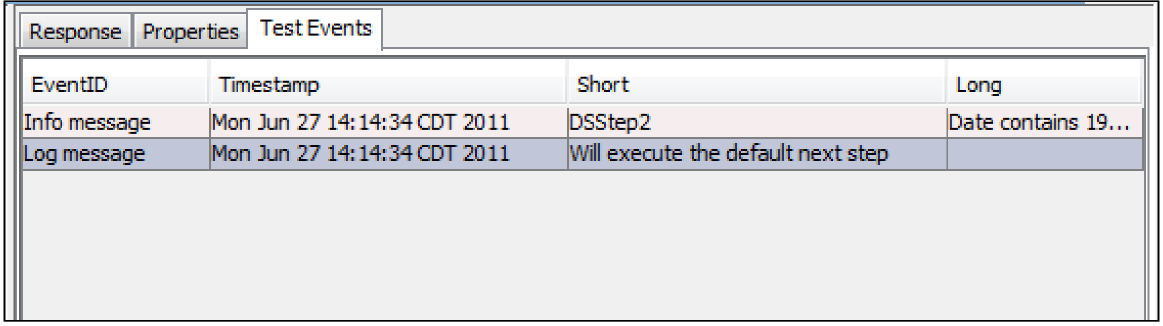
5. Click the **Properties** tab and observe where the **DSstep1_response_prop** property is created and modified.



Initial property values may be changed prior to executing the step. They are read-only after execution. Create a new property by editing an existing key.

Key	Value	Previous Value
LISA_DOC_PATH	C:\Lisa\Projects\My Tutorials\Tests	C:\Lisa\Projects\My Tutorials\Tests
LISA_LAST_STEP	DSStep2	DSStep1
lisa.DSStep2.rsp	Date contains 1999. It is: 5/8/1999....	
year	1999	1999
day	8	8
robot	0	0
lisa.DSStep2.rsp.time	0	
LISA_TC_PATH	C:\Lisa\Projects\My Tutorials\Tests	C:\Lisa\Projects\My Tutorials\Tests
LISA_HOST	Diana-PC	Diana-PC
LISA_PROJ_NAME	My Tutorials	My Tutorials
LISA_USER	arhoades@Diana-PC	arhoades@Diana-PC
DSstep1_response_prop	Date is: 5/8/1999	Date is: 5/8/1999
instance	0	0
testCaseId	3063643738353461 2D646530382D3...	3063643738353461 2D646530382D3...
LISA_TC_URL	file:/C:/Lisa/Projects/My%20Tutorial...	file:/C:/Lisa/Projects/My%20Tutorial...
lisa.DatesDS.returnedProps	[DatesDS_RowNum, month, lisa.Dat...	[DatesDS_RowNum, month, lisa.Dat...
LASTRESPONSE	Date contains 1999. It is: 5/8/1999....	Date is: 5/8/1999
config_prop	42	42
testCase	tutorial3	tutorial3
lisa.DSStep1.rsp.time	0	0
yearglobal	1999	1999
lisa.DSStep1.rsp	Date is: 5/8/1999	Date is: 5/8/1999
LISA_PROJ_ROOT	C:/Lisa/Projects/My Tutorials	C:/Lisa/Projects/My Tutorials
month	5	5
LISA_DOC_URL	file:/C:/Lisa/Projects/My%20Tutorial...	file:/C:/Lisa/Projects/My%20Tutorial...
DatesDS_RowNum	4	4

6. Click the **Test Events** tab and review the events that were generated.



EventID	Timestamp	Short	Long
Info message	Mon Jun 27 14:14:34 CDT 2011	DSStep2	Date contains 19...
Log message	Mon Jun 27 14:14:34 CDT 2011	Will execute the default next step	

Review

In this tutorial, you did the following:

- Took a first look at LISA filters and assertions.
- Opened and modified an existing test case.
- Learned how to add a simple assertion.
- Learned how to add a simple filter.
- Used the Interactive Test Run utility to check if the assertion and filter worked as expected.

More Information

LISA provides filters and assertions to cover most of the situations that you will encounter in your test case development. If there is not an appropriate filter, LISA provides a mechanism, through the LISA Software Developer's Kit, for custom filters and assertions to be developed. See the *LISA Developers Guide* for more information.

Tutorial 4 - Manipulating Java Objects (POJOs)

In this tutorial, you will create and manipulate a simple Java object and use the **java.util.Date** class to create a date object.

First, you construct the object and look at how to call methods on the object. Then you incorporate the object into a simple LISA model editor.

LISA Concepts Discussed

In this tutorial, you do the following:

- Use the Dynamic Java Execution test step
- Use the Complex Object Editor for simple objects
- Use inline filters and save the results into a property

Prerequisites

- You have completed Tutorial 3 - Filters and Assertions.
- LISA Workstation is open.

Steps

Step 1 - Create a New Test Case

To create a new test case:

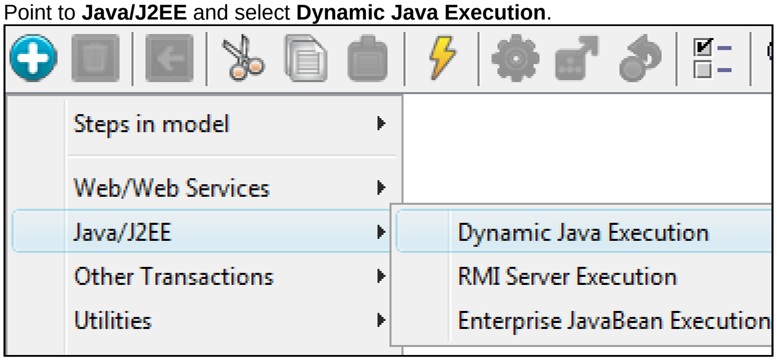
- Within the **My Tutorials** project, create a new test case called **tutorial4**.

Step 2 - Create a Dynamic Java Execution Test Step

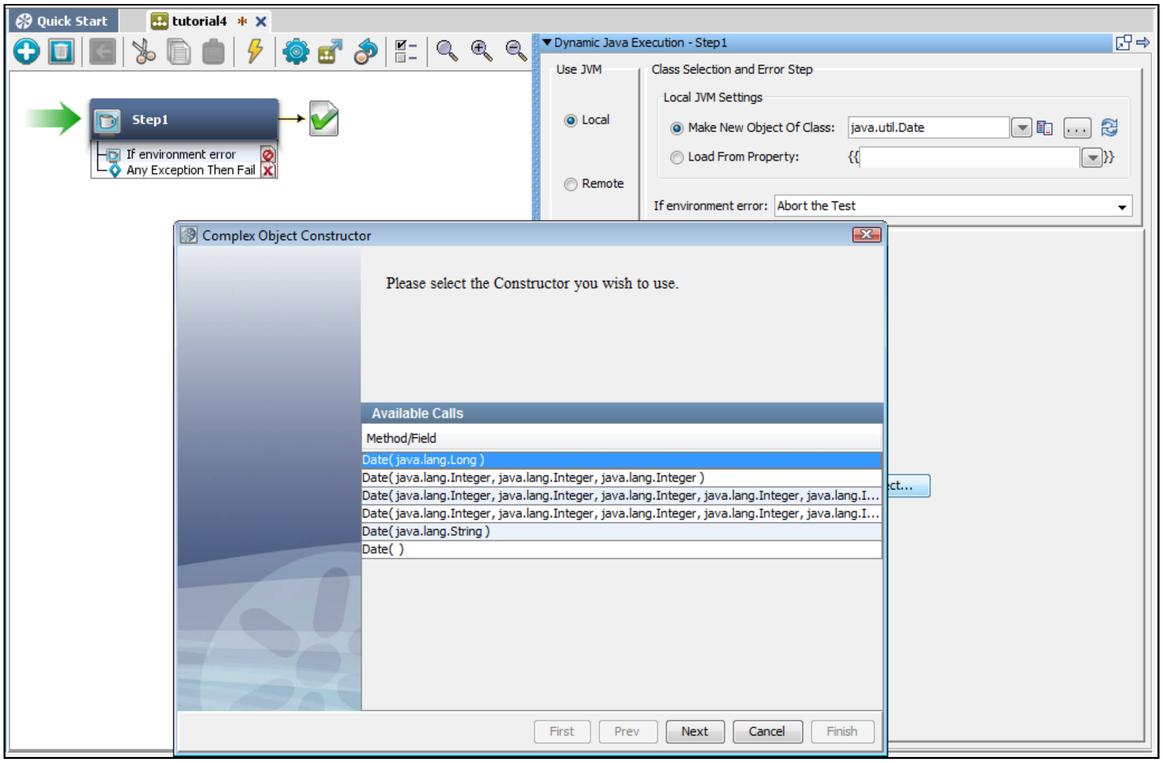
The Dynamic Java Execution test step lets you create a Java object from a class in the LISA classpath. In the following procedure, you use the **java.util.Date** class.

To create a Dynamic Java Execution test step:

1. Click the Add Step icon. The **Add step** menu is displayed.
2. Point to **Java/J2EE** and select **Dynamic Java Execution**.



The Dynamic Java Execution editor opens.



● Local

● Make New Object Of Class: java.util.Date

○ Load From Property: {{ }}

If environment error: Abort the Test

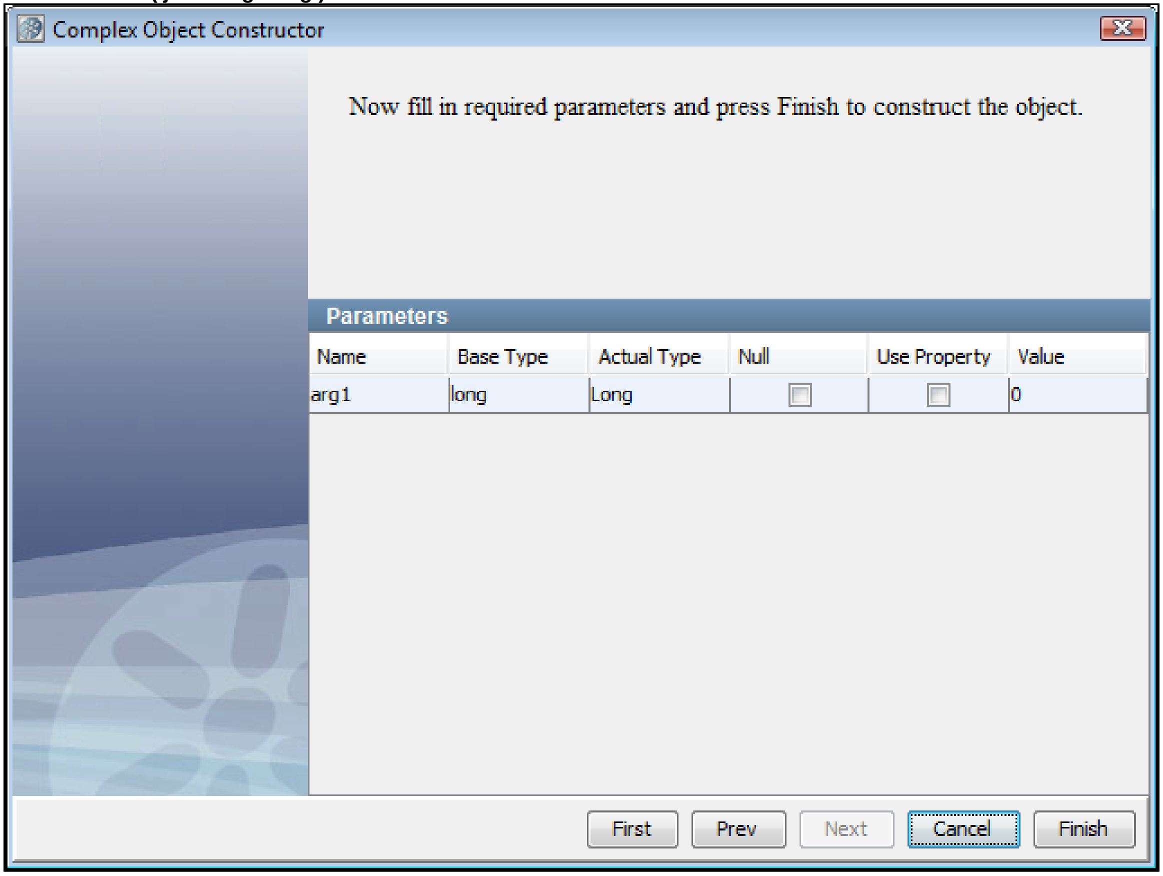
Complex Object Constructor

Please select the Constructor you wish to use.

Available Calls

Method/Field
Date(java.lang.Long)
Date(java.lang.Integer, java.lang.Integer, java.lang.Integer)
Date(java.lang.Integer, java.lang.Integer, java.lang.Integer, java.lang.Integer, java.lang.I…
Date(java.lang.Integer, java.lang.Integer, java.lang.Integer, java.lang.Integer, java.lang.I…
Date(java.lang.String)
Date( )

3. In the Local JVM Settings area, ensure that **Make New Object of Class** is selected.
4. In the field to the right of **Make New Object of Class**, type **java.util.Date**.
5. Click Construct/Load Object. The **Complex Object Constructor** wizard appears. The first step shows the available constructors.
6. Select the **Date(java.lang.Long)** constructor.



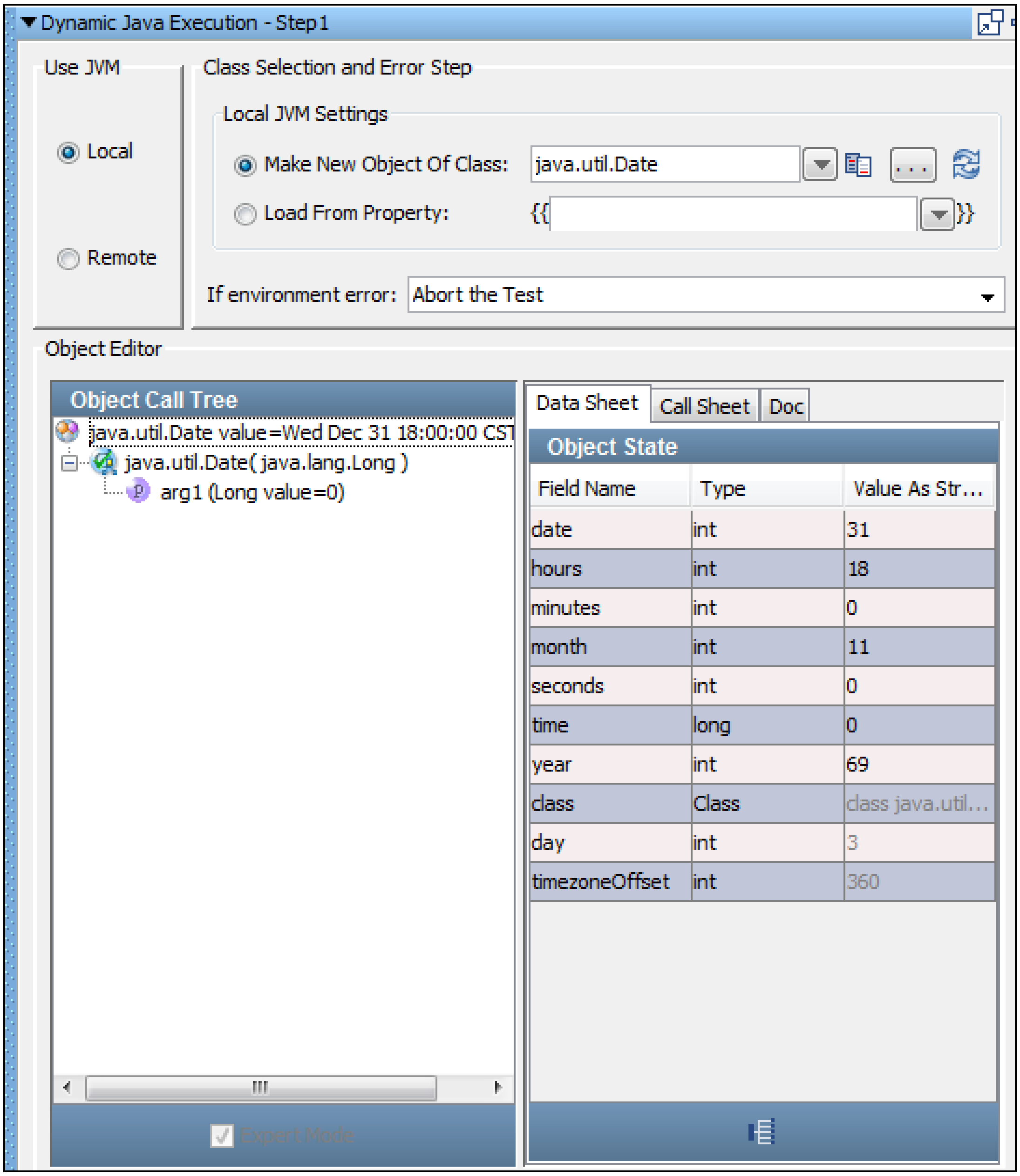
Complex Object Constructor

Now fill in required parameters and press Finish to construct the object.

Parameters

Name	Base Type	Actual Type	Null	Use Property	Value
arg1	long	Long	□	□	0

7. Click Next and Finish. The Complex Object Editor opens.



Use JVM

● Local

○ Remote

Class Selection and Error Step

Local JVM Settings

● Make New Object Of Class: java.util.Date

○ Load From Property: {{ }}

If environment error: Abort the Test

Object Editor

Object Call Tree

- java.util.Date value=Wed Dec 31 18:00:00 CST
  - java.util.Date(java.lang.Long)
    - arg1 (Long value=0)

☑ Expert Mode

Object State

Field Name	Type	Value As Str...
date	int	31
hours	int	18
minutes	int	0
month	int	11
seconds	int	0
time	long	0
year	int	69
class	Class	class java.util...
day	int	3
timezoneOffset	int	360

Now you have a Java object to manipulate in the Complex Object Editor.

Step 3 - Make a Call on the Java Object

The Complex Object Editor is divided into two panels. The left panel contains the **Object Call Tree**, which keeps track of method invocations and their input parameters and return values. The following icons are used to identify the branches in the object call tree:

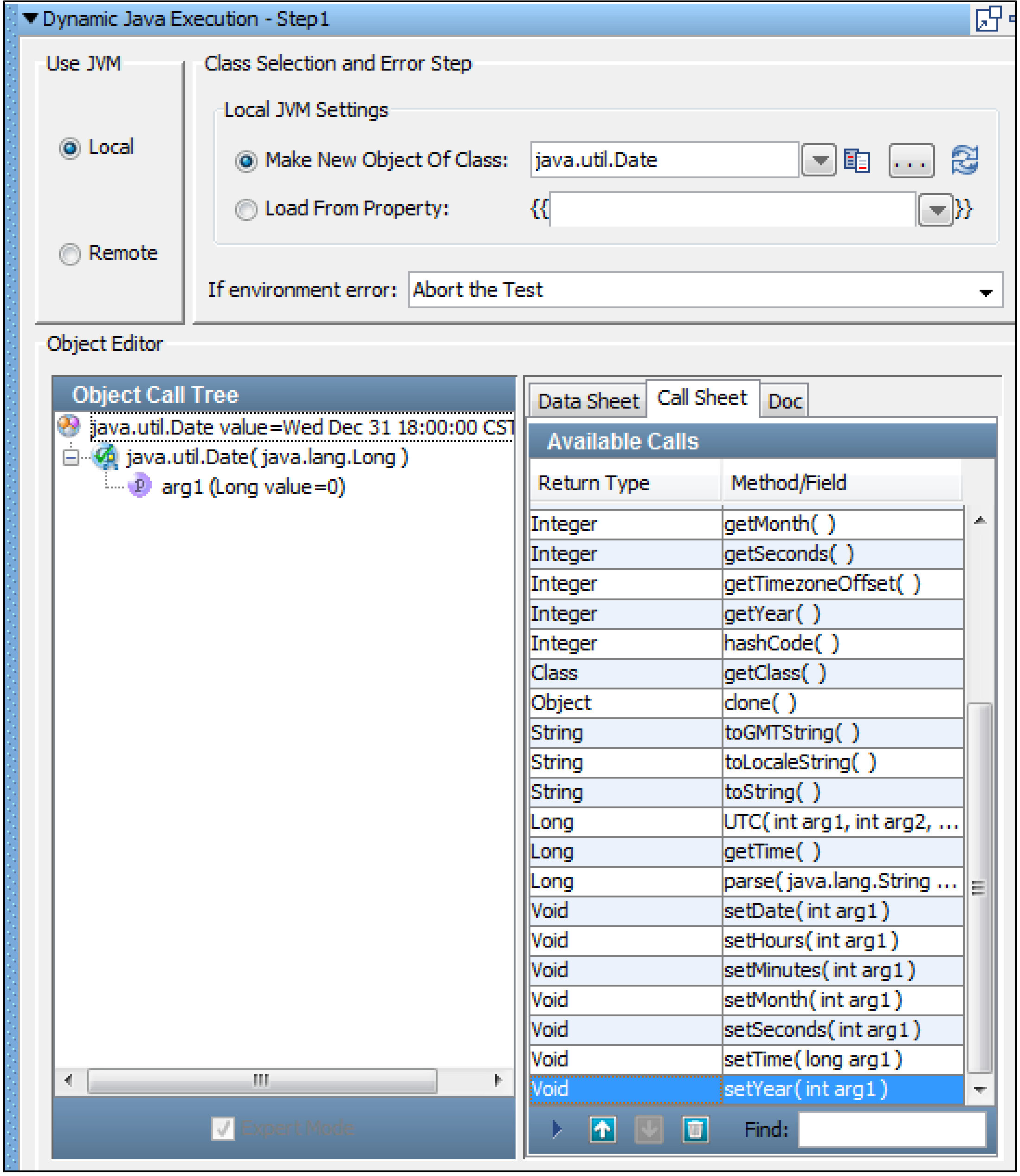
Icon	Description
	The type (class) of the object currently loaded, followed by response from calling 'toString' method of object.
	The Constructor that was called. This is shown if multiple constructors exist.
	A method call that has not been executed.

✓	A method call that has been executed.
p	The input parameters (type and current value) for the enclosing method.
r	The return value (current value if call has been executed) for the enclosing method.

The contents of the right panel vary depending on what is selected in the left panel.

To make a call on the Java object:

1. In right panel of the Complex Object Editor, click the **Call Sheet** tab.
   The Call Sheet tab shows the available methods that you can call.
2. Double-click the **setYear()** method. Alternately, you can select the **setYear()** method and click the Invoke Method ▶ icon.



Dynamic Java Execution - Step1

Use JVM

● Local

○ Remote

Class Selection and Error Step

Local JVM Settings

● Make New Object Of Class: java.util.Date

○ Load From Property: {{ }}

If environment error: Abort the Test

Object Editor

Object Call Tree

```
java.util.Date value=Wed Dec 31 18:00:00 CST
  java.util.Date( java.lang.Long )
    arg1 (Long value=0)
```

☑ Expert Mode

Data Sheet | Call Sheet | Doc

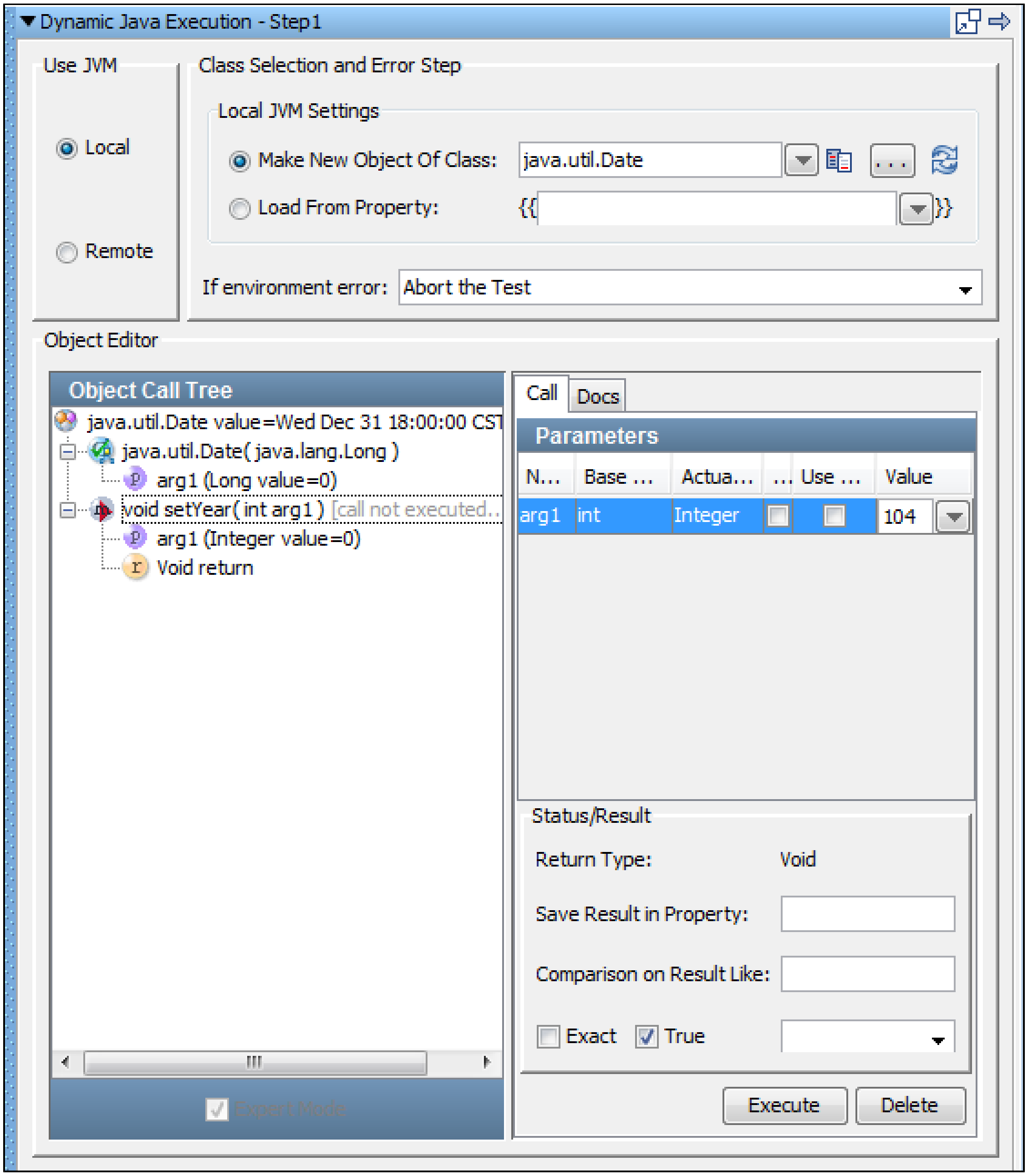
Available Calls

Return Type	Method/Field
Integer	getMonth( )
Integer	getSeconds( )
Integer	getTimezoneOffset( )
Integer	getYear( )
Integer	hashCode( )
Class	getClass( )
Object	clone( )
String	toGMTString( )
String	toLocaleString( )
String	toString( )
Long	UTC(int arg1, int arg2, ...
Long	getTime( )
Long	parse(java.lang.String ...
Void	setDate(int arg1)
Void	setHours(int arg1)
Void	setMinutes(int arg1)
Void	setMonth(int arg1)
Void	setSeconds(int arg1)
Void	setTime(long arg1)
Void	setYear(int arg1)

Find:

The **setYear()** method is added to the Object Call Tree. The right panel now displays the Call tab and Docs tab.

3. The **Call** tab lists the argument information. In the **Value** field for **arg1**, enter **104**.
4. Click Execute.



▼ Dynamic Java Execution - Step1

Use JVM: ● Local ○ Remote

Class Selection and Error Step

Local JVM Settings

● Make New Object Of Class: java.util.Date

○ Load From Property: {{ }}

If environment error: Abort the Test

Object Editor

Object Call Tree

```
java.util.Date value=Wed Dec 31 18:00:00 CST
⊟ java.util.Date( java.lang.Long )
    arg1 (Long value=0)
⊟ void setYear( int arg1 ) [call not executed..
    arg1 (Integer value=0)
    Void return
```

Call | Docs

Parameters

N...	Base ...	Actua...	...	Use ...	Value
arg1	int	Integer	□	□	104

Status/Result

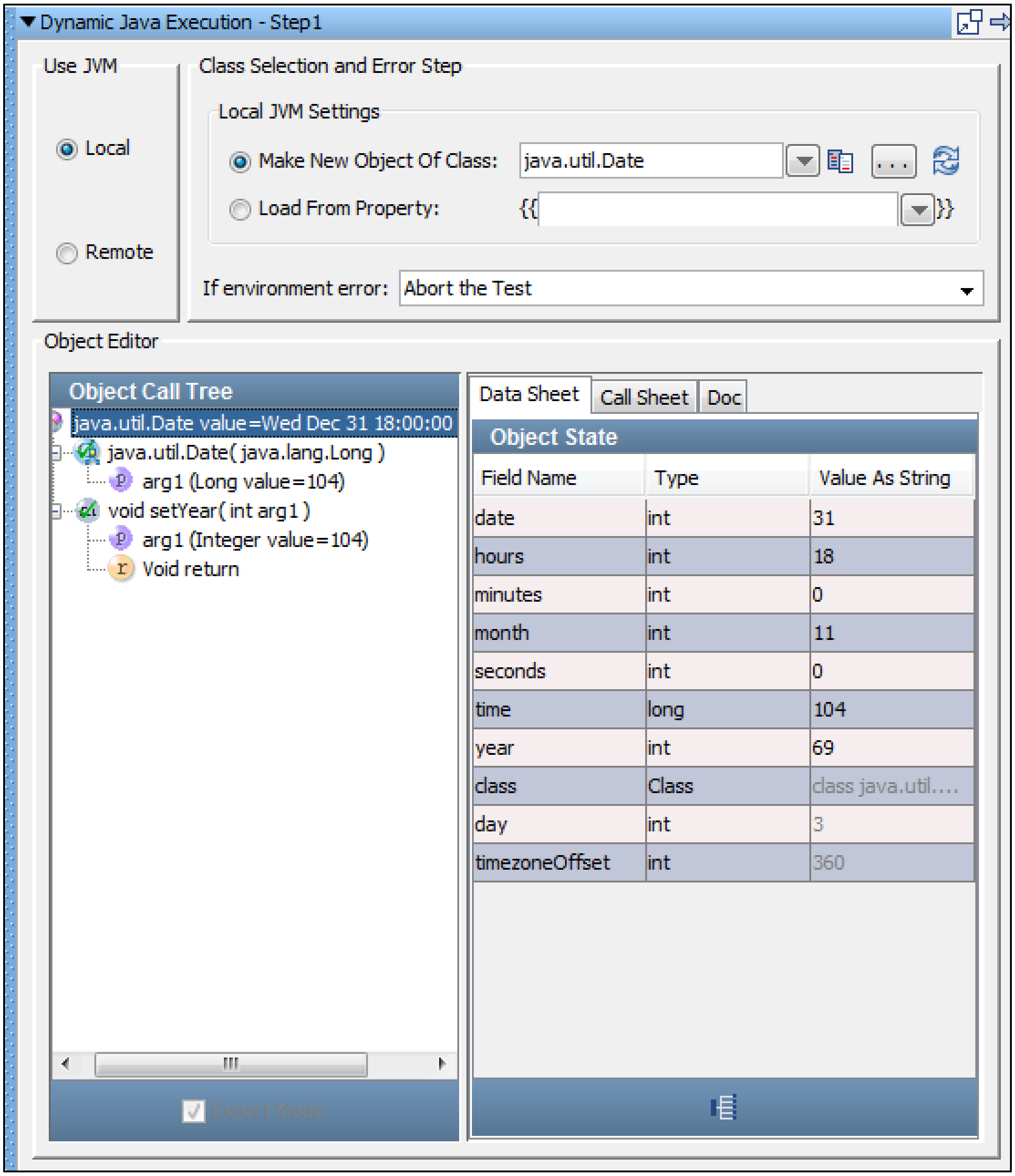
Return Type: Void

Save Result in Property:

Comparison on Result Like:

□ Exact ☑ True

5. In the Object Call Tree, select the **java.util.Date** object.
6. In the **Data Sheet** tab, verify that the **year** field is now set to **104**.



Dynamic Java Execution - Step1

Use JVM

● Local

○ Remote

Class Selection and Error Step

Local JVM Settings

● Make New Object Of Class: java.util.Date

○ Load From Property: {{ }}

If environment error: Abort the Test

Object Editor

Object Call Tree

```
java.util.Date value=Wed Dec 31 18:00:00
  java.util.Date( java.lang.Long )
    arg1 (Long value=104)
  void setYear( int arg1 )
    arg1 (Integer value=104)
    Void return
```

☑ Expert Mode

Data Sheet | Call Sheet | Doc

Object State

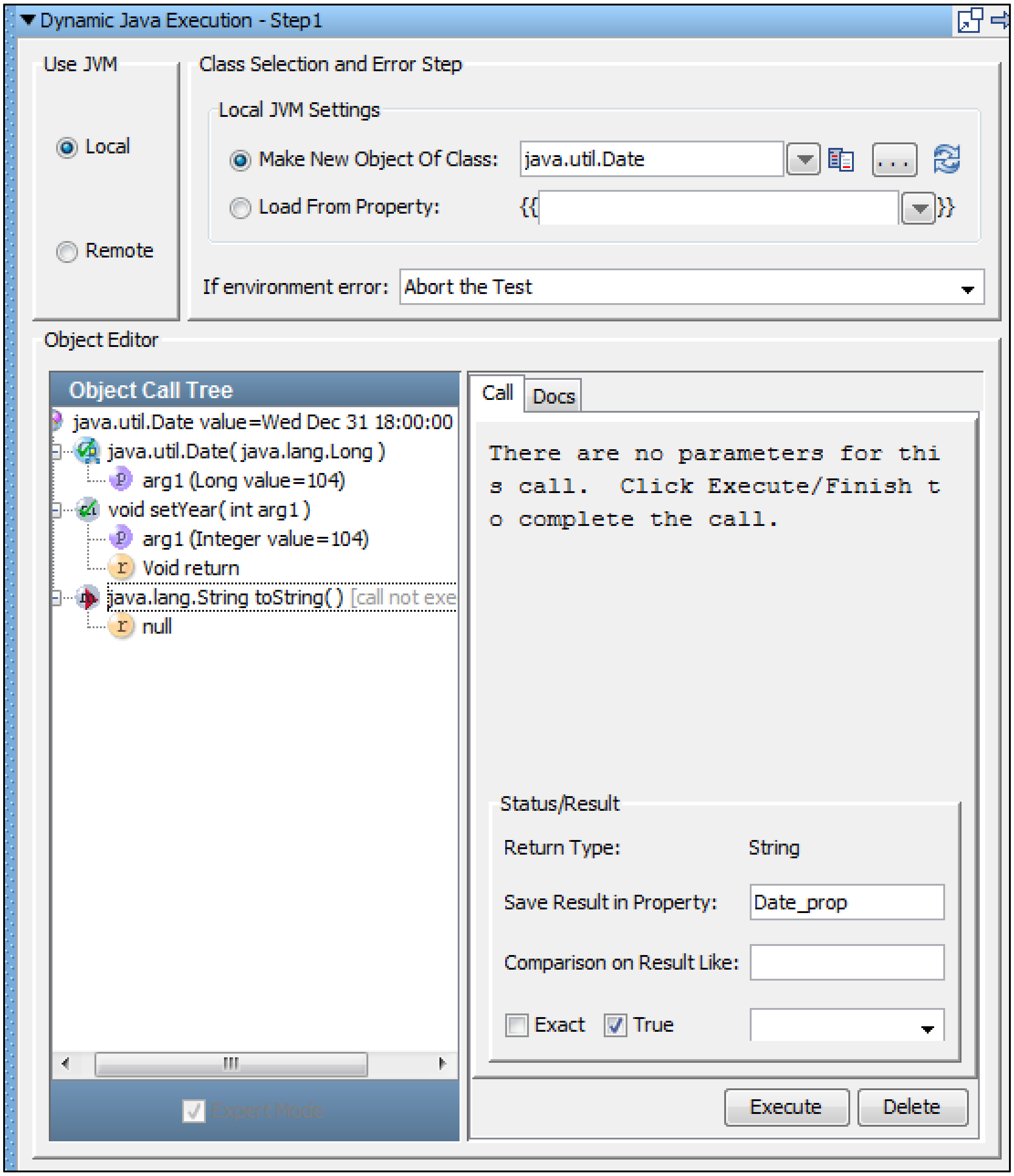
Field Name	Type	Value As String
date	int	31
hours	int	18
minutes	int	0
month	int	11
seconds	int	0
time	long	104
year	int	69
class	Class	class java.util....
day	int	3
timezoneOffset	int	360

Step 4 - Add an Inline Filter

You can add an inline filter from within the Complex Object Editor. Inline filters (and assertions) do not result in a filter being added to the test step in the element tree. Inline filter management is always done in the Complex Object Editor.

To add an inline filter:

1. With the **java.util.Date** object selected, click the **Call Sheet** tab.
2. Invoke the **toString()** method to retrieve the date to be placed in a property.
   The right panel now displays the Call tab and Docs tab.
3. In the **Status/Result** area of the Call tab, in the **Save Result in Property** field, add an inline filter by entering **Date_prop** as the property name.


Dynamic Java Execution - Step1

Use JVM

● Local

○ Remote

Class Selection and Error Step

Local JVM Settings

● Make New Object Of Class: java.util.Date

○ Load From Property: {{ }}

If environment error: Abort the Test

Object Editor

Object Call Tree

```
java.util.Date value=Wed Dec 31 18:00:00
  java.util.Date( java.lang.Long )
    arg1 (Long value=104)
  void setYear( int arg1 )
    arg1 (Integer value=104)
    Void return
  java.lang.String toString( ) [call not exe
    null
```

☑ Expert Mode

Call | Docs

```
There are no parameters for thi
s call.  Click Execute/Finish t
o complete the call.
```

Status/Result

Return Type: String

Save Result in Property: Date_prop

Comparison on Result Like:

☐ Exact ☑ True

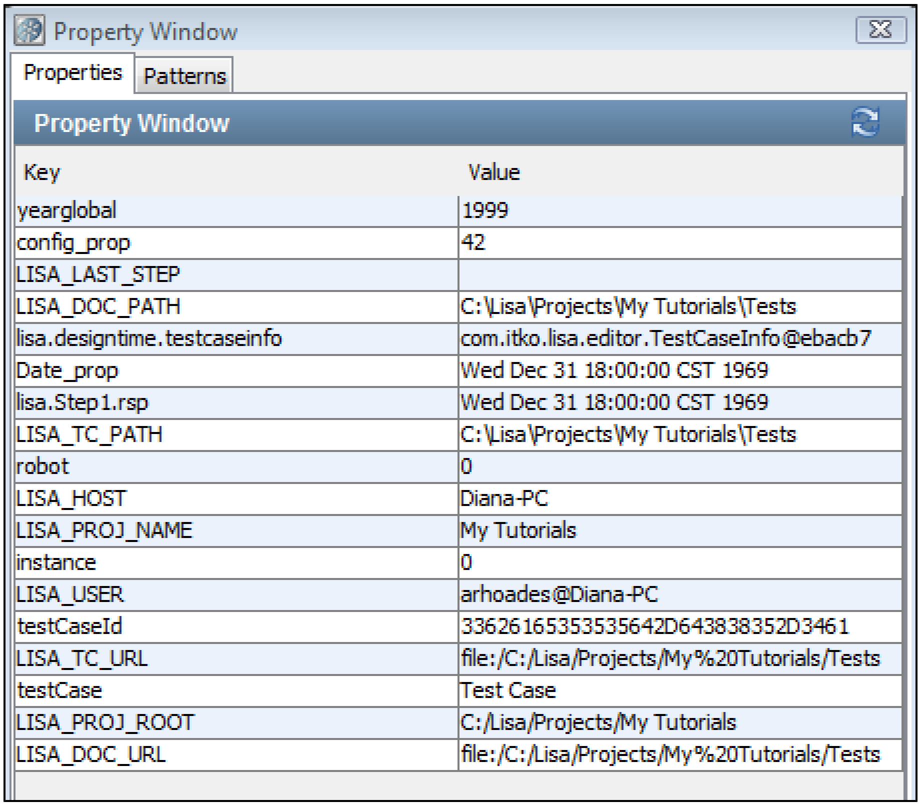
4. Click Execute.
5. Click the Save icon.

Step 5 - Verify the Property Created

You can display the Property Window to verify that the **Date_prop** property was created.

To verify the property created:

1. Click the Show model properties icon on the test case toolbar. Alternately, you can choose **Help > Property Window** from the main menu.
2. Locate the **Date_prop** property.



Key	Value
yearglobal	1999
config_prop	42
LISA_LAST_STEP	
LISA_DOC_PATH	C:\Lisa\Projects\My Tutorials\Tests
lisa.designtime.testcaseinfo	com.itko.lisa.editor.TestCaseInfo@ebacb7
Date_prop	Wed Dec 31 18:00:00 CST 1969
lisa.Step1.rsp	Wed Dec 31 18:00:00 CST 1969
LISA_TC_PATH	C:\Lisa\Projects\My Tutorials\Tests
robot	0
LISA_HOST	Diana-PC
LISA_PROJ_NAME	My Tutorials
instance	0
LISA_USER	arhoades@Diana-PC
testCaseId	336261653535642D643838352D3461
LISA_TC_URL	file:/C:/Lisa/Projects/My%20Tutorials/Tests
testCase	Test Case
LISA_PROJ_ROOT	C:/Lisa/Projects/My Tutorials
LISA_DOC_URL	file:/C:/Lisa/Projects/My%20Tutorials/Tests

3. Click Close.

Review

In this tutorial, you did the following:

- Created a test step to manipulate a Java object of **java.util.Date** type.
- Used the Complex Object Editor to manipulate the Java object.
- Learned how to add inline filters to objects and save results into a property.

Tutorial 5 - Running a Demo Server Web Application

In this tutorial, you will step through a simple web application that accompanies LISA.

The LISA Bank application is a simple front-end that is connected to a database table containing financial account information. The application business logic consists of Enterprise JavaBeans and Web Services. From the web application, you can view the profile of the user, create an account, add addresses, and so on.

The goal of this tutorial is for you to become familiar with the application. This application is used in subsequent tutorials as the system under test.

LISA Concepts Discussed

No new concepts are introduced.

Prerequisites

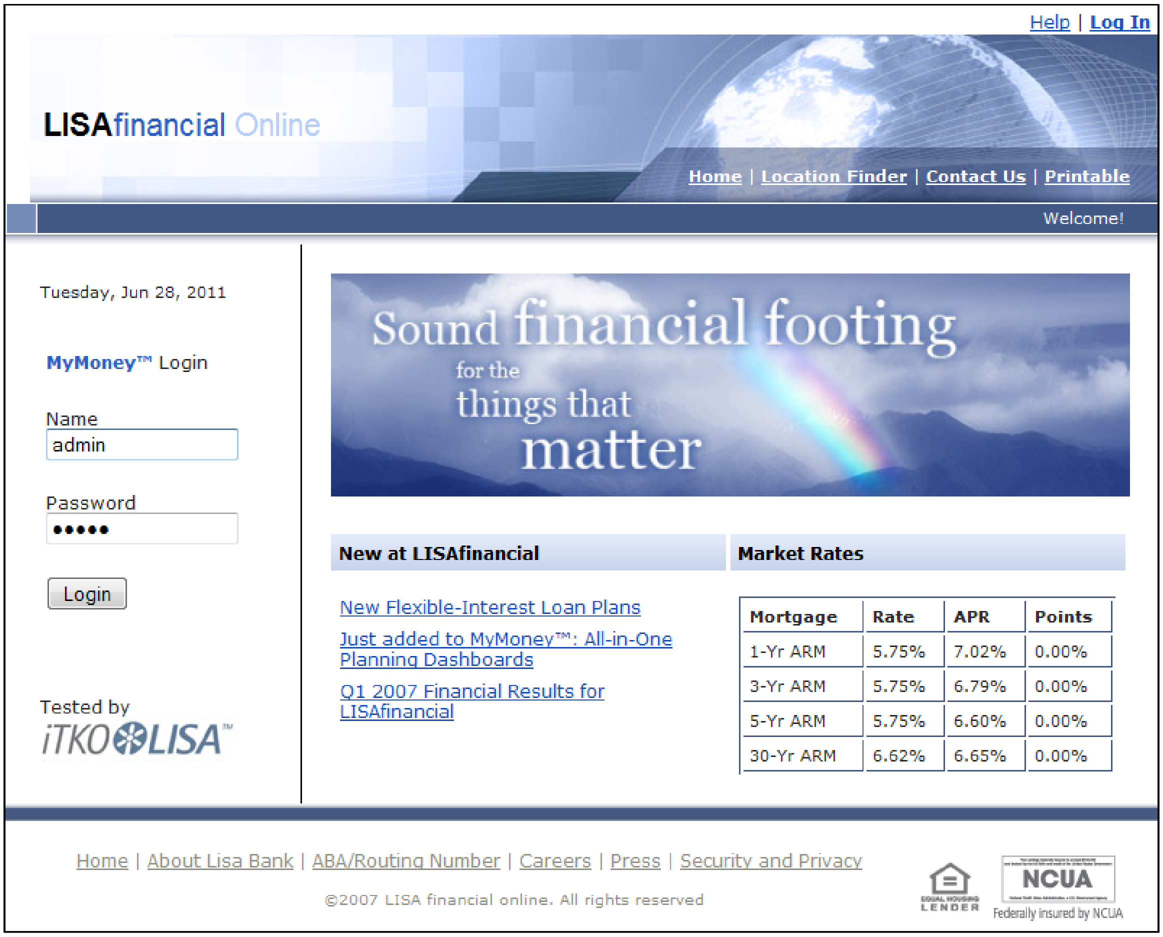
- The LISA demo server is running.

Steps

Step 1 - Launch the Web Application

To launch the web application:

1. Open a new browser window.
2. Enter the following URL. Replace **localhost** in the path with your computer's IP address.
   http://localhost:8080/lisabank/
   The login page appears.


Help | Log In
LISAfinancial Online
Home | Location Finder | Contact Us | Printable
Welcome!
MyMoney™ Login
Login
Sound financial footing for the things that matter
Tested by
iTKO LISA™
Home | About Lisa Bank | ABA/Routing Number | Careers | Press | Security and Privacy
EQUAL HOUSING LENDER
NCUA
Federally insured by NCUA

Tuesday, Jun 28, 2011

Name: admin

Password: •••••

New at LISAfinancial

New Flexible-Interest Loan Plans

Just added to MyMoney™: All-in-One Planning Dashboards

Q1 2007 Financial Results for LISAfinancial

Market Rates

Mortgage	Rate	APR	Points
1-Yr ARM	5.75%	7.02%	0.00%
3-Yr ARM	5.75%	6.79%	0.00%
5-Yr ARM	5.75%	6.60%	0.00%
30-Yr ARM	6.62%	6.65%	0.00%

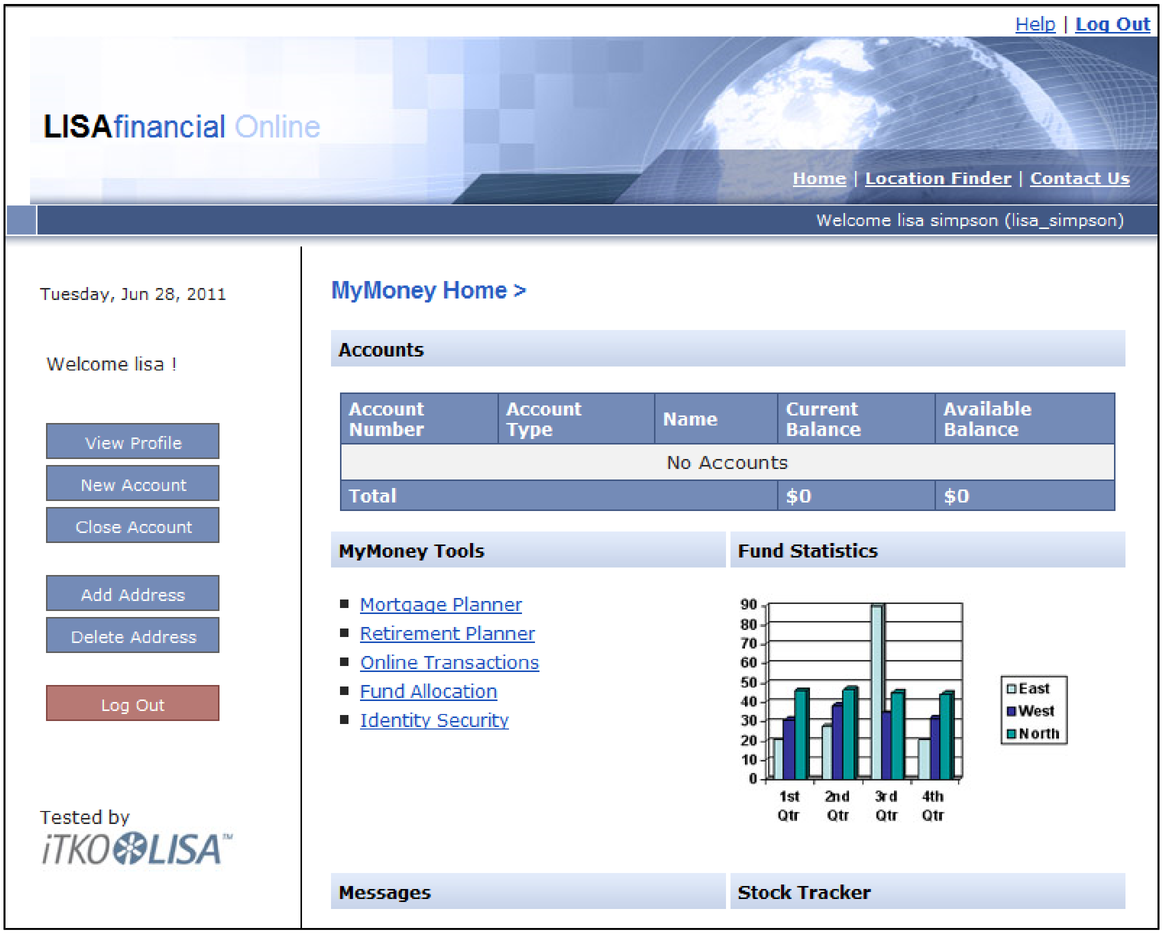
©2007 LISA financial online. All rights reserved

Step 2 - Log In to the Web Application

You will use the predefined user name **lisa_simpson**.

To log in to the web application:

1. In the **Name** field, enter **lisa_simpson**.
2. In the **Password** field, enter **golisa**.
3. Click Login. The welcome page appears. The left side contains buttons for various actions that you can perform: **View Profile, New Account, Close Account, Add Address, Delete Address**, and **Log Out**.



Welcome lisa simpson (lisa_simpson)

Tuesday, Jun 28, 2011

Welcome lisa !

MyMoney Home >

Accounts

Account Number	Account Type	Name	Current Balance	Available Balance
No Accounts				
Total			$0	$0

MyMoney Tools

- Mortgage Planner
- Retirement Planner
- Online Transactions
- Fund Allocation
- Identity Security

Fund Statistics

Messages

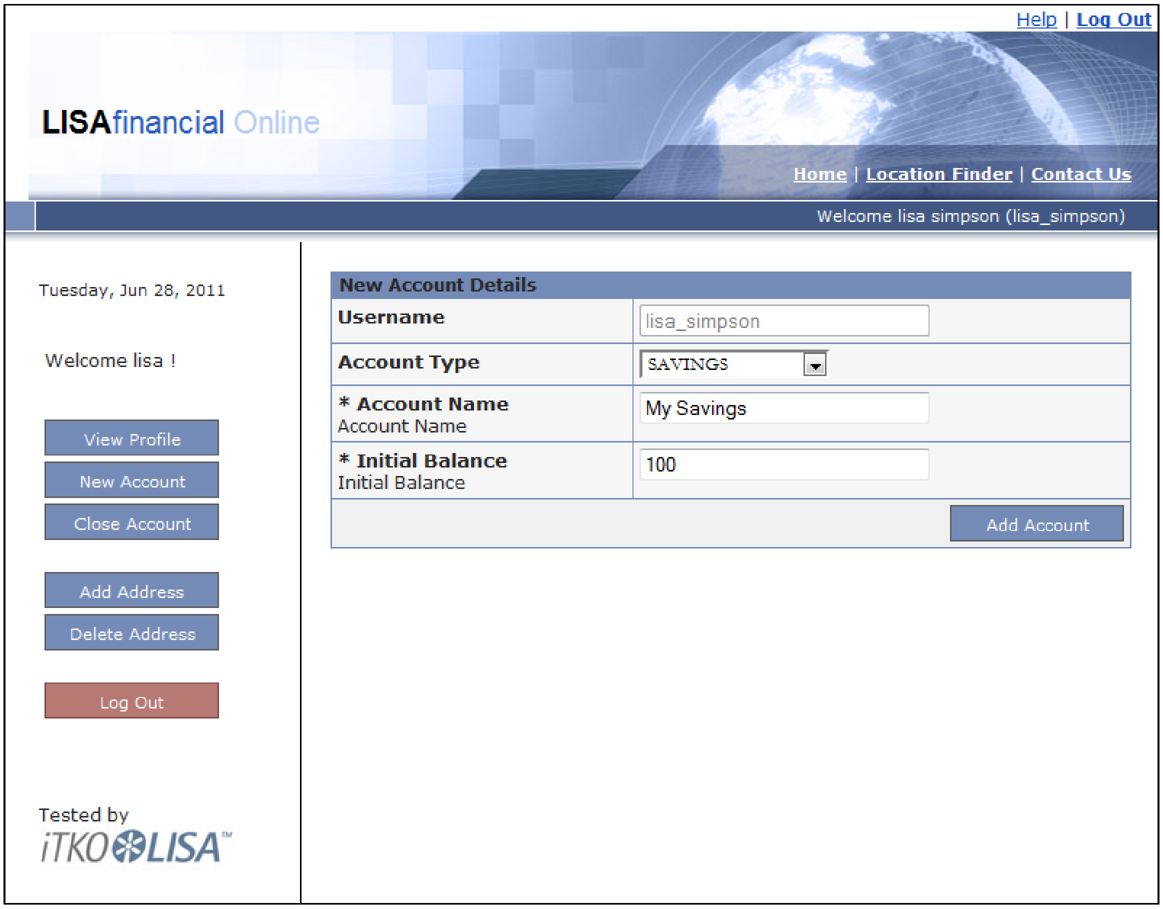
Stock Tracker

Step 3 - Create a New Account

Notice that the current user does not have any accounts.

To create a new account:

1. Click New Account.
2. From the **Account Type** list, select **SAVINGS**.
3. In the **Account Name** field, enter **My Savings**
4. In the **Initial Balance** field, enter **100.00**.
5. Click Add Account.



Welcome lisa simpson (lisa_simpson)

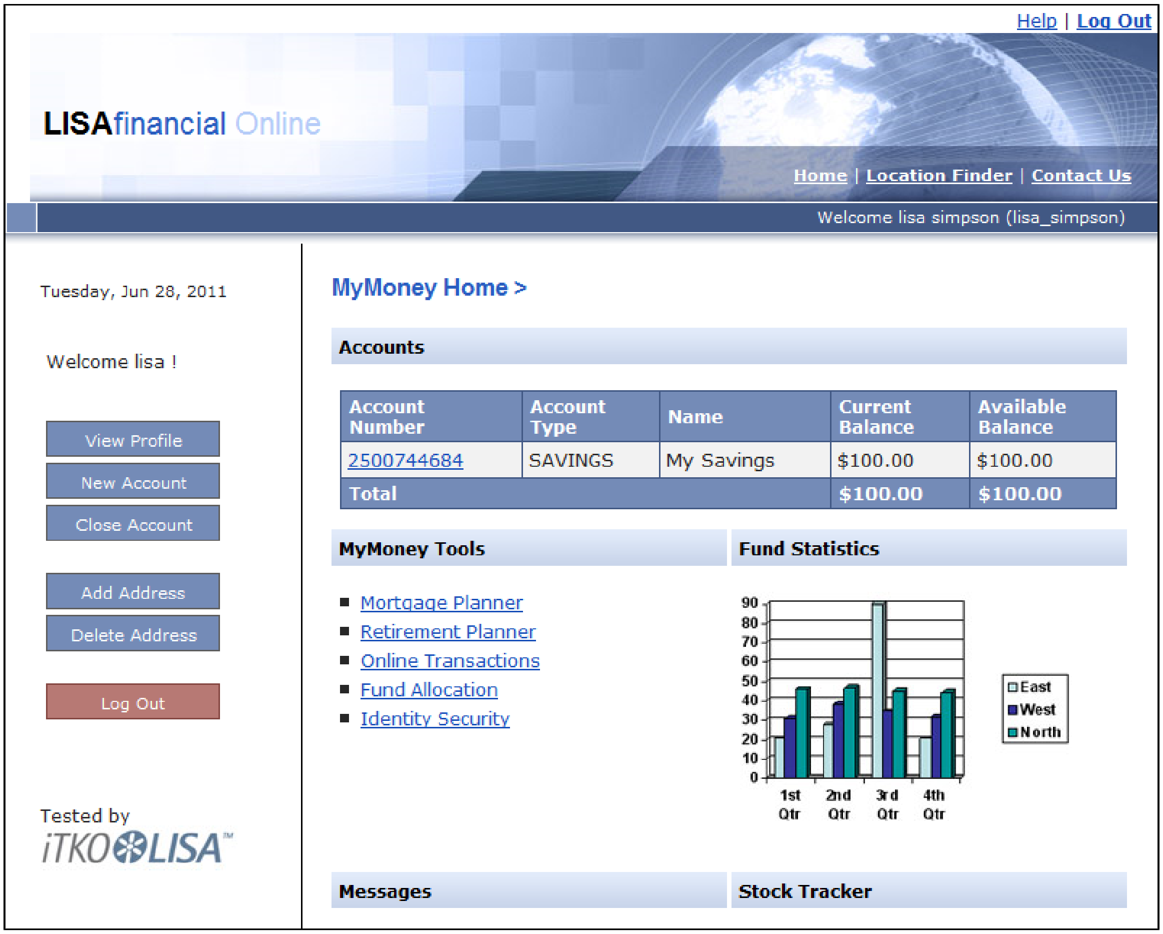
Tuesday, Jun 28, 2011

Welcome lisa !

New Account Details

Username	lisa_simpson
Account Type	SAVINGS
*** Account Name** Account Name	My Savings
*** Initial Balance** Initial Balance	100

The new savings account is added to the **Accounts** section.



Tuesday, Jun 28, 2011

Welcome lisa !

MyMoney Home >

Accounts

Account Number	Account Type	Name	Current Balance	Available Balance
2500744684	SAVINGS	My Savings	$100.00	$100.00
Total			$100.00	$100.00

MyMoney Tools

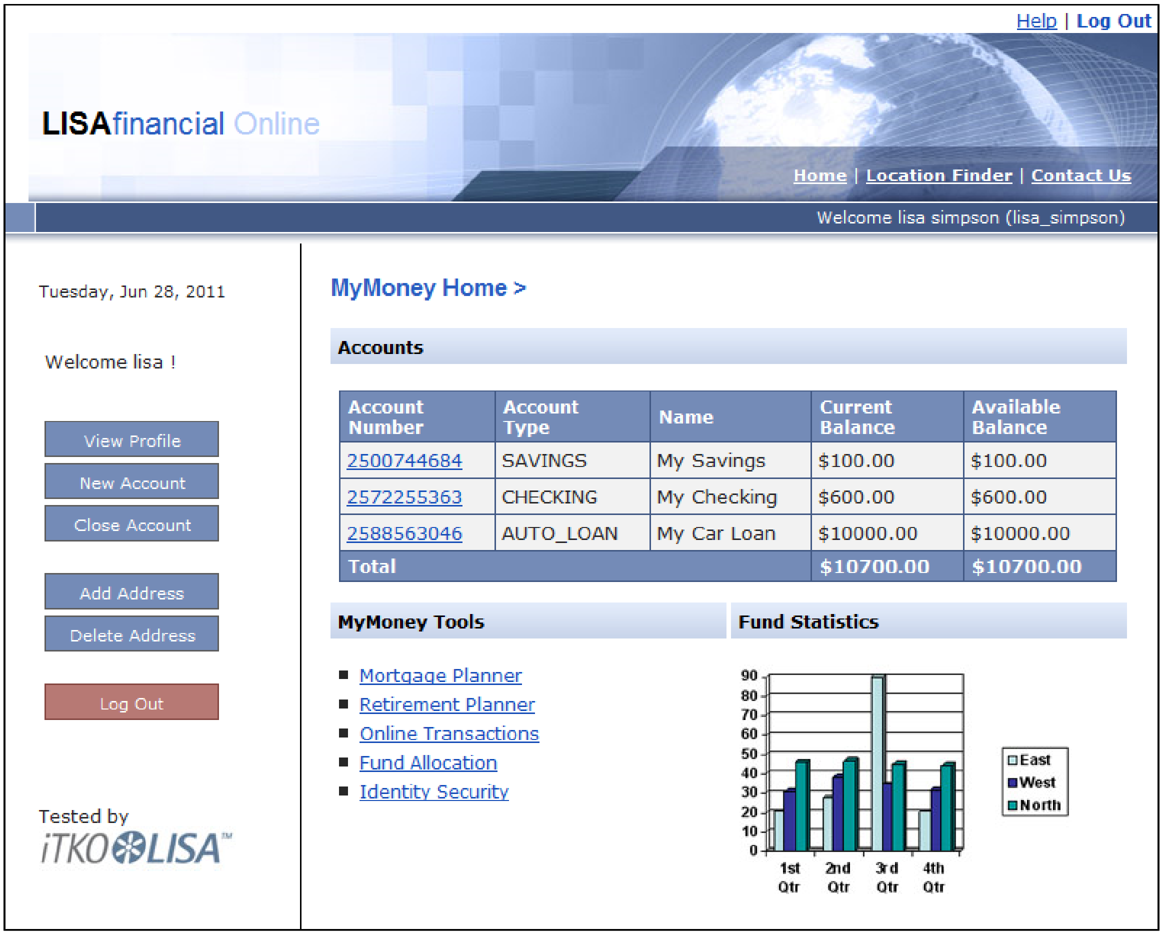
- Mortgage Planner
- Retirement Planner
- Online Transactions
- Fund Allocation
- Identity Security

Fund Statistics

Messages

Stock Tracker

6. Repeat the preceding steps to create two more accounts: **Checking** and **Auto_Loan**. Do not use commas in the **Initial Balance** field.



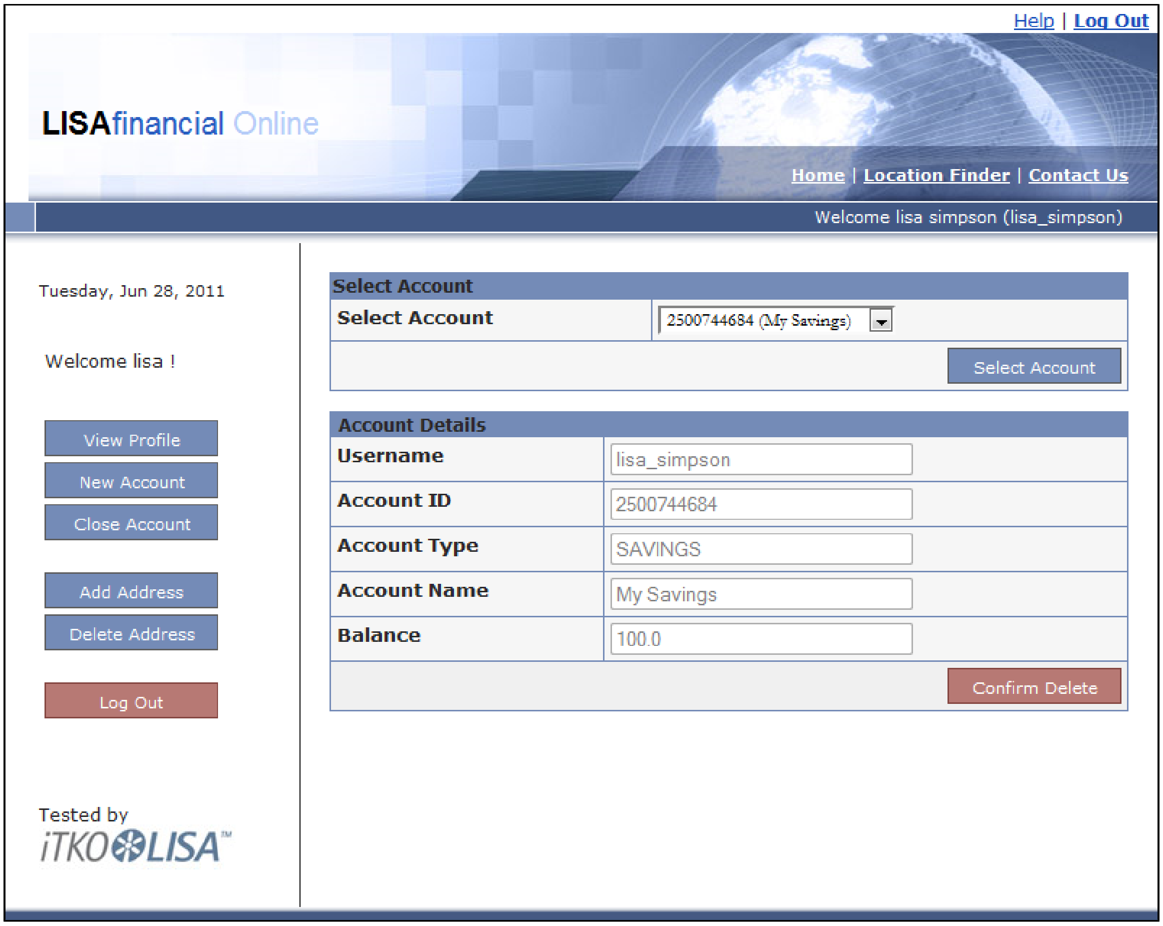
Account Number	Account Type	Name	Current Balance	Available Balance
2500744684	SAVINGS	My Savings	$100.00	$100.00
2572255363	CHECKING	My Checking	$600.00	$600.00
2588563046	AUTO_LOAN	My Car Loan	$10000.00	$10000.00
Total			$10700.00	$10700.00

Step 4 - Close an Account

The application lets you close accounts.

To close an account:

1. Click Close Account.
2. Select **My Savings** from the list and click Select Account.



Welcome lisa simpson (lisa_simpson)

Tuesday, Jun 28, 2011

Welcome lisa !

Select Account

Select Account	2500744684 (My Savings)

Account Details

Username	lisa_simpson
Account ID	2500744684
Account Type	SAVINGS
Account Name	My Savings
Balance	100.0

3. Click Confirm Delete.
   The **My Savings** account is removed from the the Accounts section.

Step 5 - Log Out of the Web Application

To log out of the web application:

- Click Log Out.

Review

In this tutorial, you did the following:

- Logged into the LISA Bank application.
- Created new accounts.
- Closed an account.

Tutorial 6 - Testing a Website

In this tutorial, you will use the LISA web recorder to record the path through a website and create test steps of HTTP/HTML Request for each HTTP request/response pair.

The HTTP/HTML Request test step enables you to make requests against a web server and receive results from within a test case. Test a simple website to verify that the pages work as expected.

LISA Concepts Discussed

In this tutorial, you do the following:

- Use the LISA web recorder to create a test case containing HTTP/HTML Request steps
- Edit and run the test case that the recorder produces
- Add a data set to the test case

Prerequisites

- You have completed Tutorial 5 - Running a Demo Server Web Application.
- LISA Workstation is open.
- You have access to the demo server (either the local demo server, or the ITKO demo server).

Tutorial Parts

Part A - Record and Run the LISABank Test Case
Part B - Running the Test Case
Part C - Modifying HTTP/HTML Request Test Steps (optional)

Part A - Record and Run the LISA Bank Test Case

Use the LISA web recorder to create a test case containing HTTP/HTML Request steps.

Step 1 - Create a New Test Case

To create a new test case:

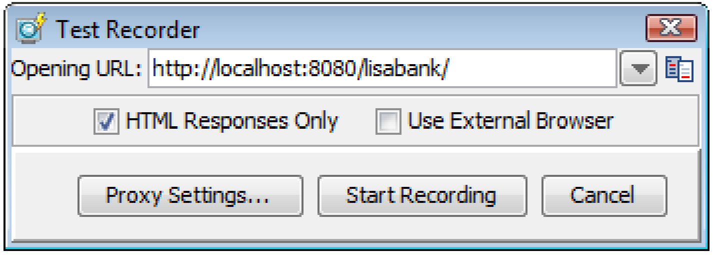
- Within the **My Tutorials** project, create a new test case named **tutorial6a** in the **Tests** subfolder.
  The model editor is opened.

Step 2 - Start the Web Recorder

There are different ways to record a website. In this tutorial, you record a website through HTTP proxy.

To start the web recorder:

1. From the main menu, select **Actions > Record Test Case for User Interface > Web Recorder (HTTP proxy)**. The **Test Recorder** dialog opens.
2. In the **Opening URL** field, enter the following URL. Replace **localhost** in the path with your machine IP address.
   http://localhost:8080/lisabank/



Test Recorder

Opening URL: http://localhost:8080/lisabank/

☑ HTML Responses Only □ Use External Browser

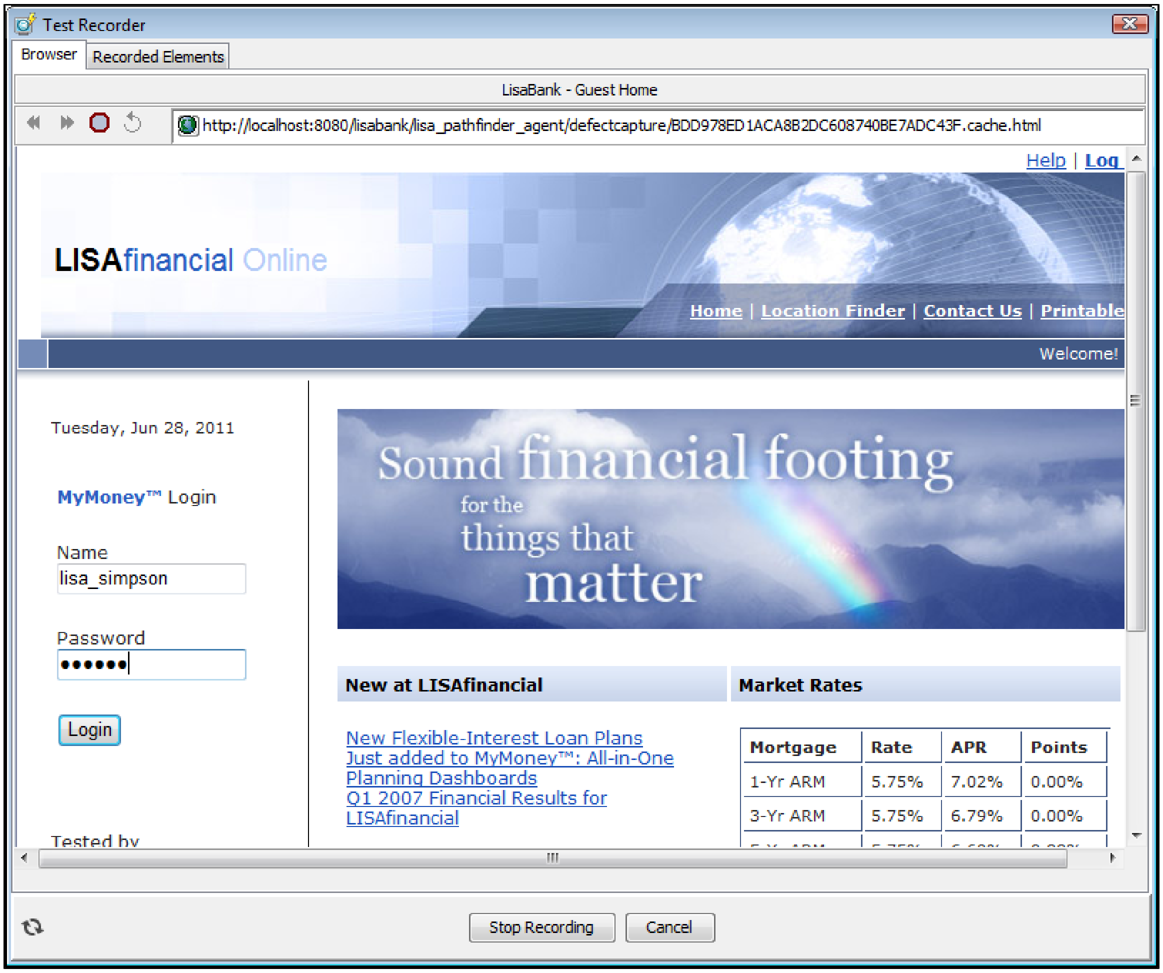
3. Click Start Recording. The Test Recorder window opens. The Test Recorder window contains two tabs: **Browser** and **Recorded Elements**. The login page of the LISA Bank application appears in the Browser tab.

Step 3 - Record the LISA Bank Application

As you perform actions within the LISA Bank application, the request and response information for each page visited are recorded.

To record the LISA Bank application

1. In the **Name** field, enter **lisa_simpson**. In the **Password** field, enter **golisa**.



LisaBank - Guest Home

http://localhost:8080/lisabank/lisa_pathfinder_agent/defectcapture/BDD978ED1ACA8B2DC608740BE7ADC43F.cache.html

Help | Log

LISAfinancial Online

Home | Location Finder | Contact Us | Printable

Welcome!

Tuesday, Jun 28, 2011

MyMoney™ Login

Name: lisa_simpson

Password: ••••••

Login

Tested by

Sound financial footing for the things that matter

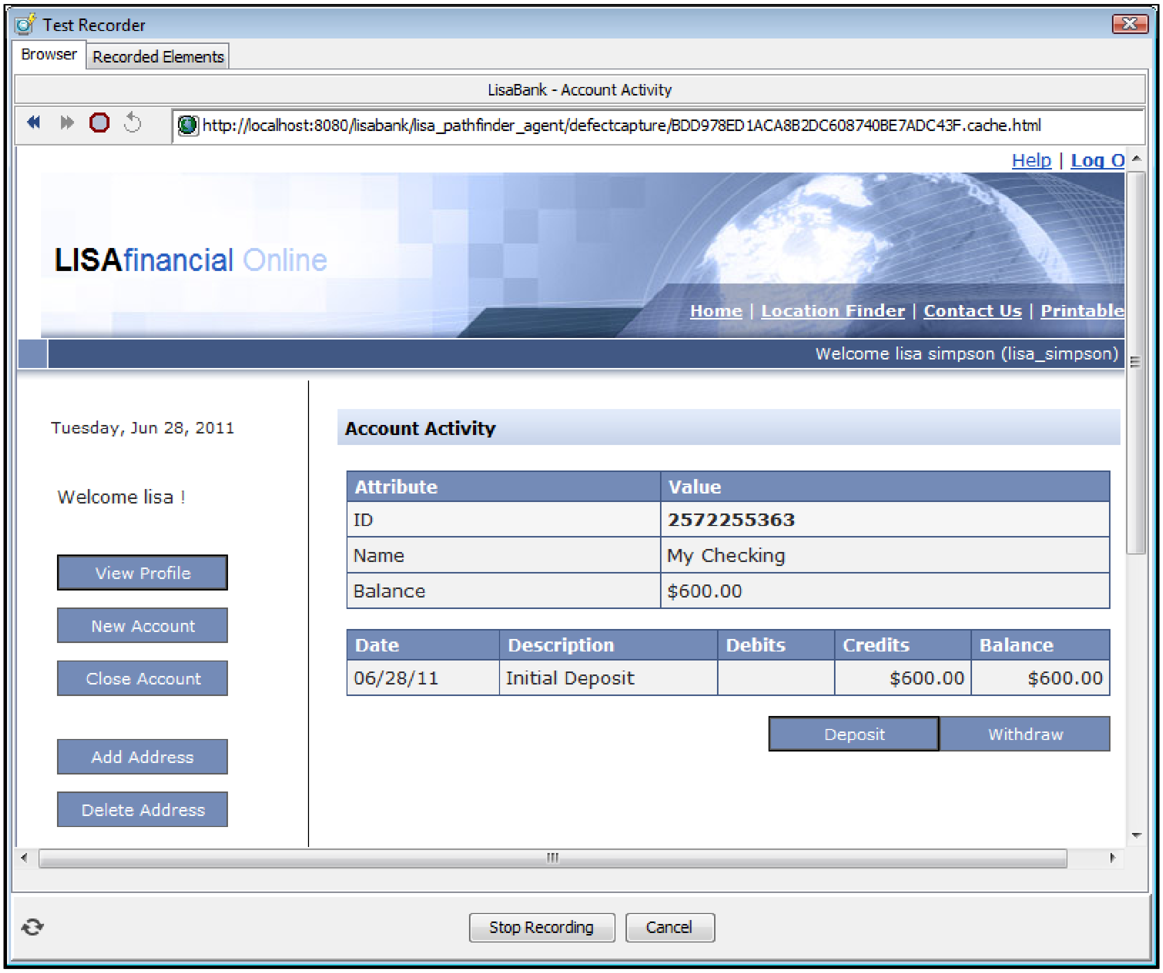
New at LISAfinancial

- New Flexible-Interest Loan Plans
- Just added to MyMoney™: All-in-One Planning Dashboards
- Q1 2007 Financial Results for LISAfinancial

Market Rates

Mortgage	Rate	APR	Points
1-Yr ARM	5.75%	7.02%	0.00%
3-Yr ARM	5.75%	6.79%	0.00%

2. Click Login.
   The welcome page appears.
3. In the Accounts section, click the account number of the checking account. The Account Activity screen appears.
4. Click Deposit.



LisaBank - Account Activity

http://localhost:8080/lisabank/lisa_pathfinder_agent/defectcapture/BDD978ED1ACA8B2DC608740BE7ADC43F.cache.html

Welcome lisa simpson (lisa_simpson)

Tuesday, Jun 28, 2011

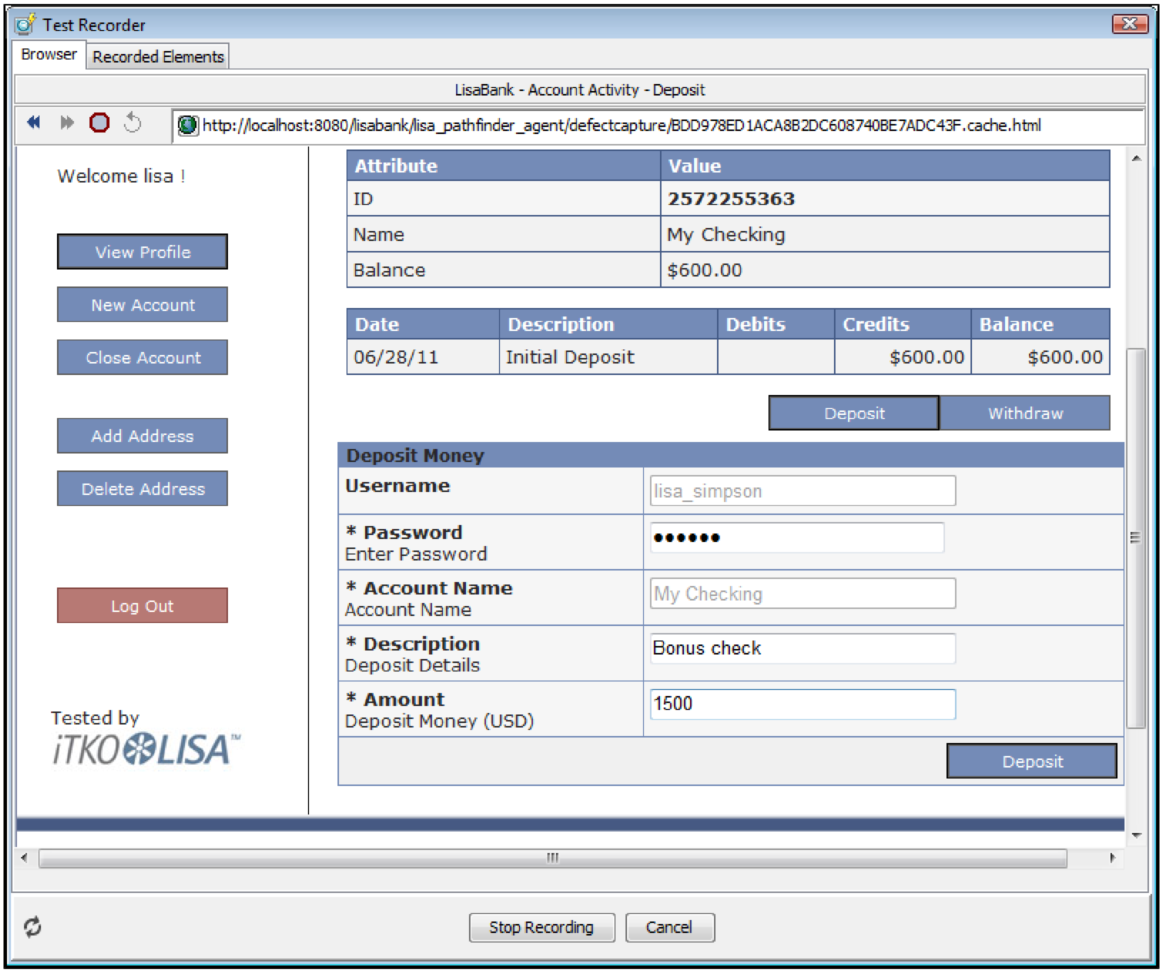
Welcome lisa !

Account Activity

Attribute	Value
ID	**2572255363**
Name	My Checking
Balance	$600.00

Date	Description	Debits	Credits	Balance
06/28/11	Initial Deposit		$600.00	$600.00

5. In the Deposit Money area, enter the password **golisa**, a description of the transaction, and the amount for this deposit.
6. Click Deposit.



LisaBank - Account Activity - Deposit

http://localhost:8080/lisabank/lisa_pathfinder_agent/defectcapture/BDD978ED1ACA8B2DC608740BE7ADC43F.cache.html

Welcome lisa !

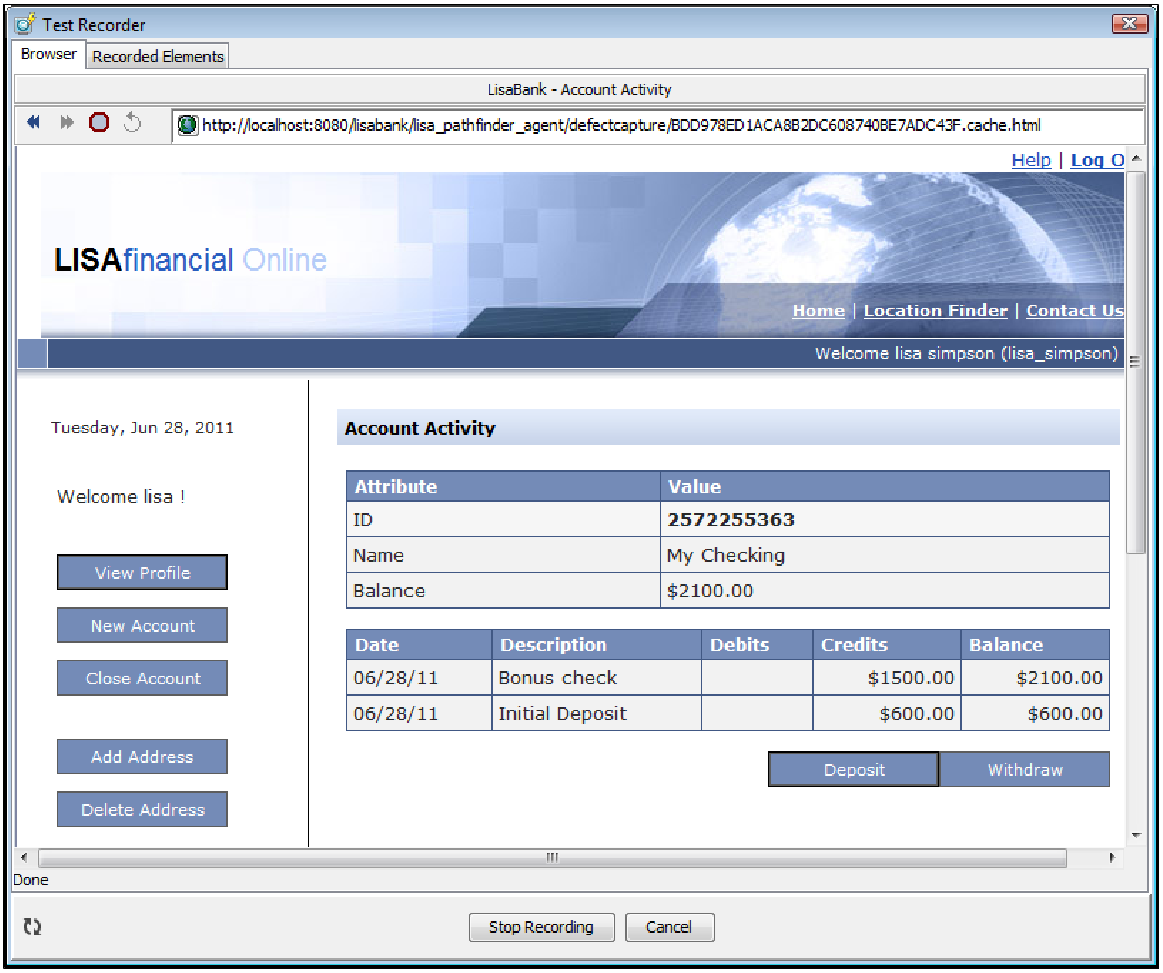
Attribute	Value
ID	**2572255363**
Name	My Checking
Balance	$600.00

Date	Description	Debits	Credits	Balance
06/28/11	Initial Deposit		$600.00	$600.00

Deposit Money

Username	lisa_simpson
*** Password** Enter Password	••••••
*** Account Name** Account Name	My Checking
*** Description** Deposit Details	Bonus check
*** Amount** Deposit Money (USD)	1500

The Account Activity screen shows the updated balance and a record of the deposit.


Test Recorder
Browser
Recorded Elements
LisaBank - Account Activity
http://localhost:8080/lisabank/lisa_pathfinder_agent/defectcapture/BDD978ED1ACA8B2DC608740BE7ADC43F.cache.html
Help | Log O
LISAfinancial Online
Home | Location Finder | Contact Us
Welcome lisa simpson (lisa_simpson)
Tuesday, Jun 28, 2011
Welcome lisa !
View Profile
New Account
Close Account
Add Address
Delete Address
Deposit
Withdraw
Done
Stop Recording
Cancel

Account Activity

Attribute	Value
ID	**2572255363**
Name	My Checking
Balance	$2100.00

Date	Description	Debits	Credits	Balance
06/28/11	Bonus check		$1500.00	$2100.00
06/28/11	Initial Deposit		$600.00	$600.00

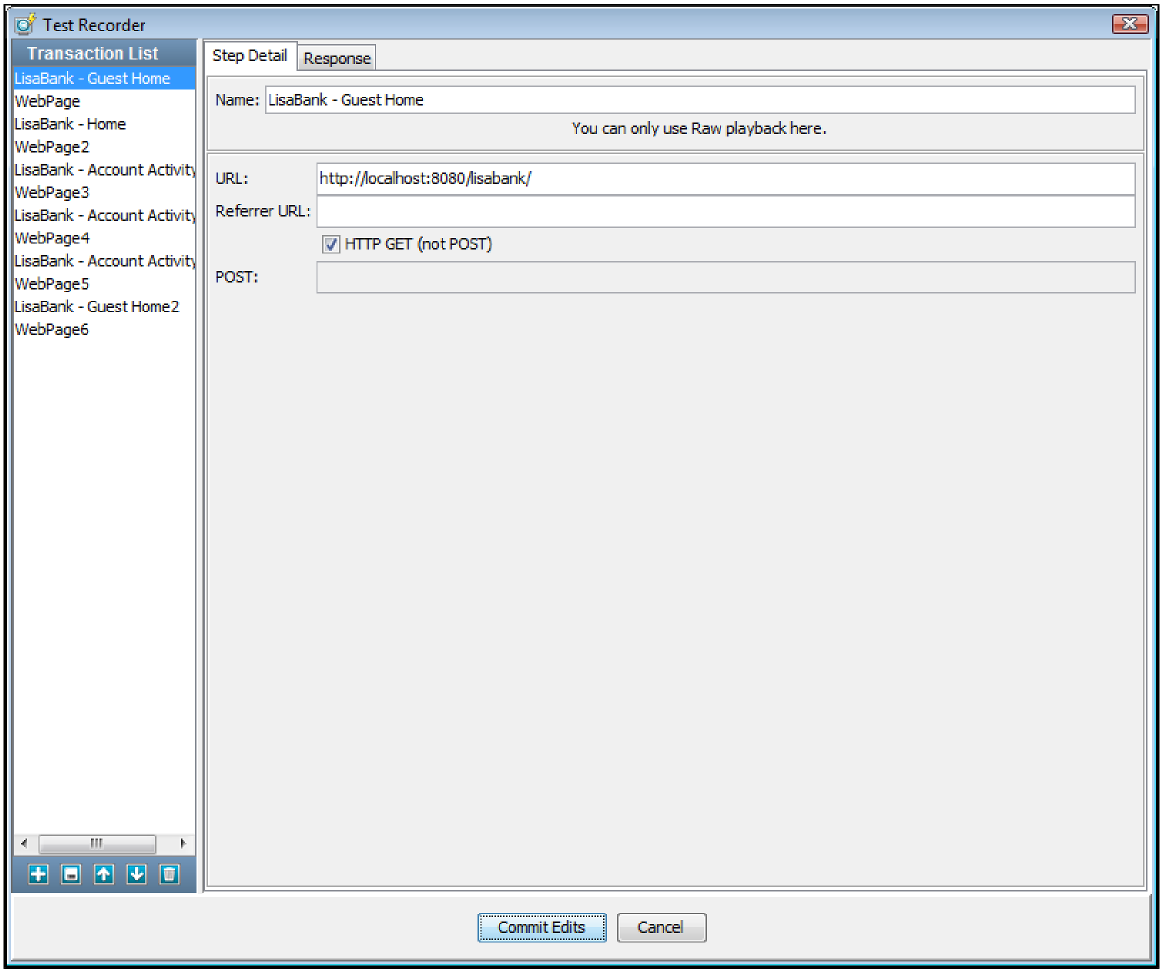
7. From the left navigation pane, click Log Out.

Step 4 - Stop the Web Recorder

After you stop recording, you can view details about the transactions.

To stop the Web recorder:

1. At the bottom of the Test Recorder window, click Stop Recording. Filters and properties are automatically created for the web page references. The form fields for the deposit are also displayed. The left pane shows a list of transactions (steps). The right pane shows the step detail and response for the selected transaction.



Name: LisaBank - Guest Home

You can only use Raw playback here.

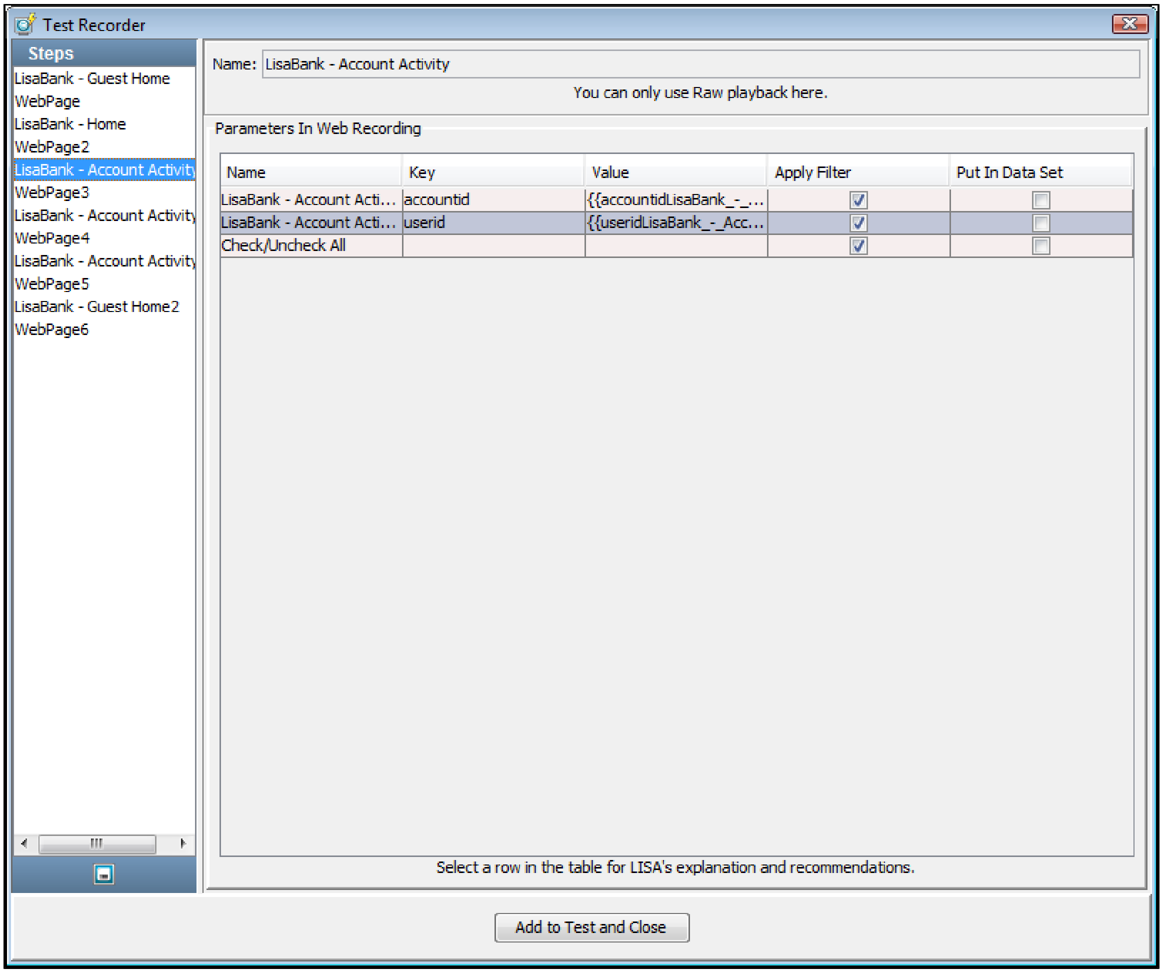
URL: http://localhost:8080/lisabank/

Referrer URL:

☑ HTTP GET (not POST)

POST:

2. Click Commit Edits. The parameters page appears.
3. Click Add to Test and Close.



Test Recorder

Steps

- LisaBank - Guest Home
- WebPage
- LisaBank - Home
- WebPage2
- LisaBank - Account Activity
- WebPage3
- LisaBank - Account Activity
- WebPage4
- LisaBank - Account Activity
- WebPage5
- LisaBank - Guest Home2
- WebPage6

Name: LisaBank - Account Activity

You can only use Raw playback here.

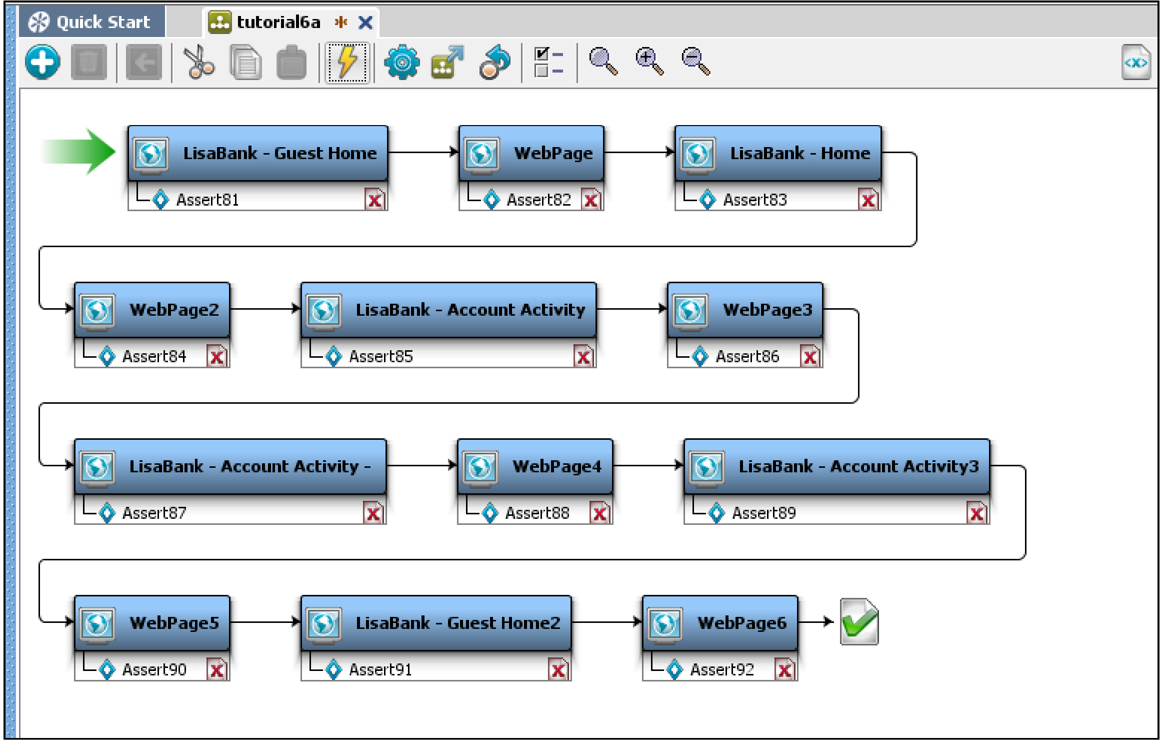
Parameters In Web Recording

Name	Key	Value	Apply Filter	Put In Data Set
LisaBank - Account Acti...	accountid	{{accountidLisaBank_-_...	☑	☐
LisaBank - Account Acti...	userid	{{useridLisaBank_-_Acc...	☑	☐
Check/Uncheck All			☑	☐

Select a row in the table for LISA's explanation and recommendations.

Add to Test and Close

The model editor is populated with a new test case, having all the transaction information from the saved recording. Each test step in the test case represents an HTTP request.



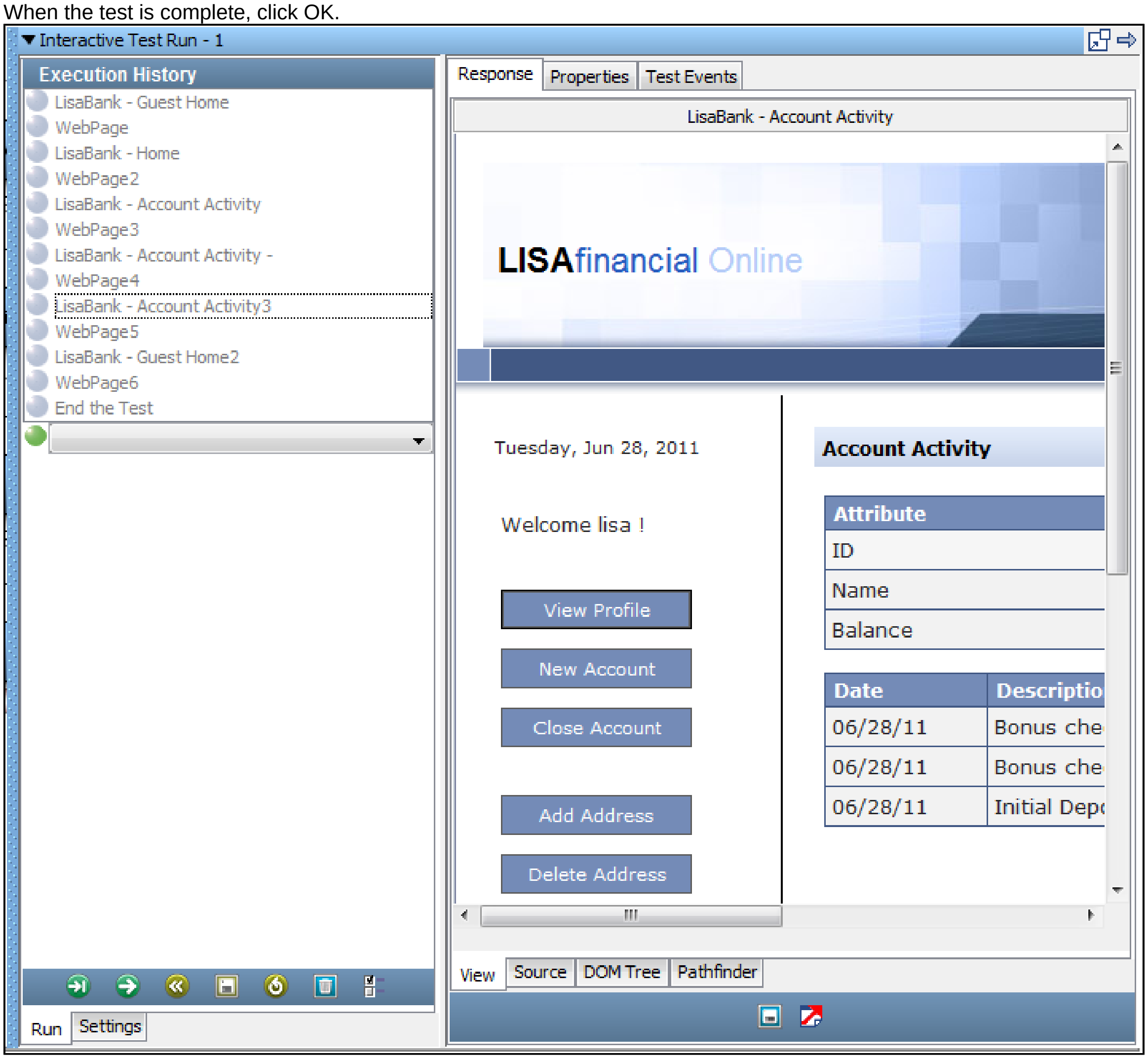
4. Save the test case.

Part B - Running the Test Case

In this section, continue from Part A - Record and Run the LISA Bank Test Case to run the saved test case in the Interactive Test Run (ITR) utility and view the results.

To run the test case,

1. Start a new ITR session.
2. In the Execution History pane, click the Automatically execute test icon.
3. When the test is complete, click OK.



The **Response > View** tab shows the rendered pages as the ITR replays the deposit into the checking account.
The **Response > Source** tab shows the HTML code for the page captured in the step.
LISA acts as the browser and sends the same HTTP requests to the Web server.

4. Close the ITR utility.

Part C - Modifying HTTP_HTML Request Test Steps (Optional)

The web recorder produces an HTTP/HTML Request test step for each request.

You can edit and modify these test steps just like the other test steps in LISA. The recorder uses the parameters that you entered during the recording as values for the Post and Get parameters in the request.

To generalize your LISA Bank test, replace these hard-coded description and deposit amount values (for example, "cash" and "1000.00") with properties from a data set. You previously worked with data sets in Tutorial 2 - Data Sets.

In the **Account Activity3** test step from the recording results, the **Host Name** and **Port** parameters are parameterized and added to the configuration. The values for **description** and **amount** are hard-coded.

The objective in this part of the tutorial is to parameterize the test case to deposit different amounts of money by using a numeric counting data set. When you subsequently run the test case, it uses deposit values different from the ones recorded.

Step 1 - Copy a Test Case

To copy a test case:

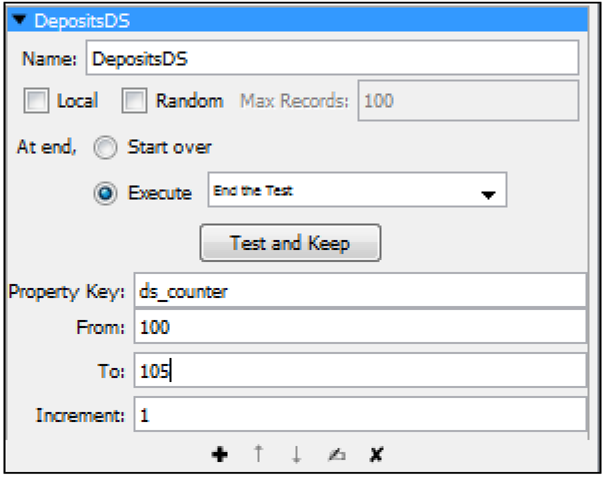
1. Ensure that the **tutorial6a** test case is open in the model editor.
2. From the menu bar, select **File > Save As**.
3. In the **File name** field, enter **tutorial6c**.
4. Click Save.

Step 2 - Add a Data Set

In the following procedure, you add a numeric counting data set. This type of data set enables you to assign a number to a property and change the number by a fixed value each time the data set is used.

To add a data set:

1. In the model editor, select the first test step.
2. In the right pane, double-click the **Data Sets** step element tab.
3. Click the Add icon ✚ below the Data Sets element.
4. From the **Common Datasets** list, select **Create a numeric counting data set**. The data set editor opens in the right pane.
5. Enter the following:
    - In the **Name** field, enter **DepositsDS**.
    - In the **At end** field, select the **Execute** option and select **End the Test** from the list.
    - In the **Property Key** field, enter **ds_counter**.
    - In the **From** field, enter **100**.
    - In the **To** field, enter **105**.
    - In the **Increment** field, enter **1**.



DepositsDS

Name: DepositsDS

□ Local □ Random Max Records: 100

At end, ○ Start over

● Execute End the Test

Test and Keep

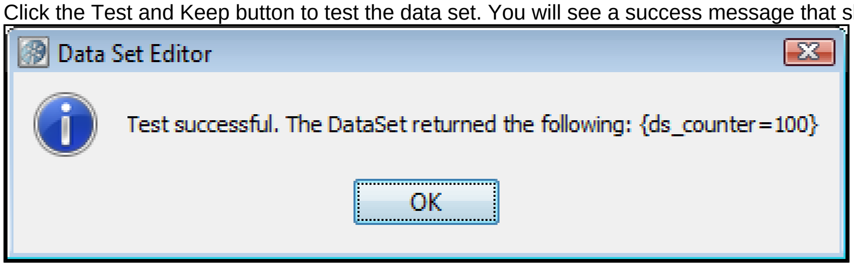
Property Key: ds_counter

From: 100

To: 105

Increment: 1

6. Click the Test and Keep button to test the data set. You will see a success message that shows the first row of data in the data set:



Data Set Editor

Test successful. The DataSet returned the following: {ds_counter=100}

OK

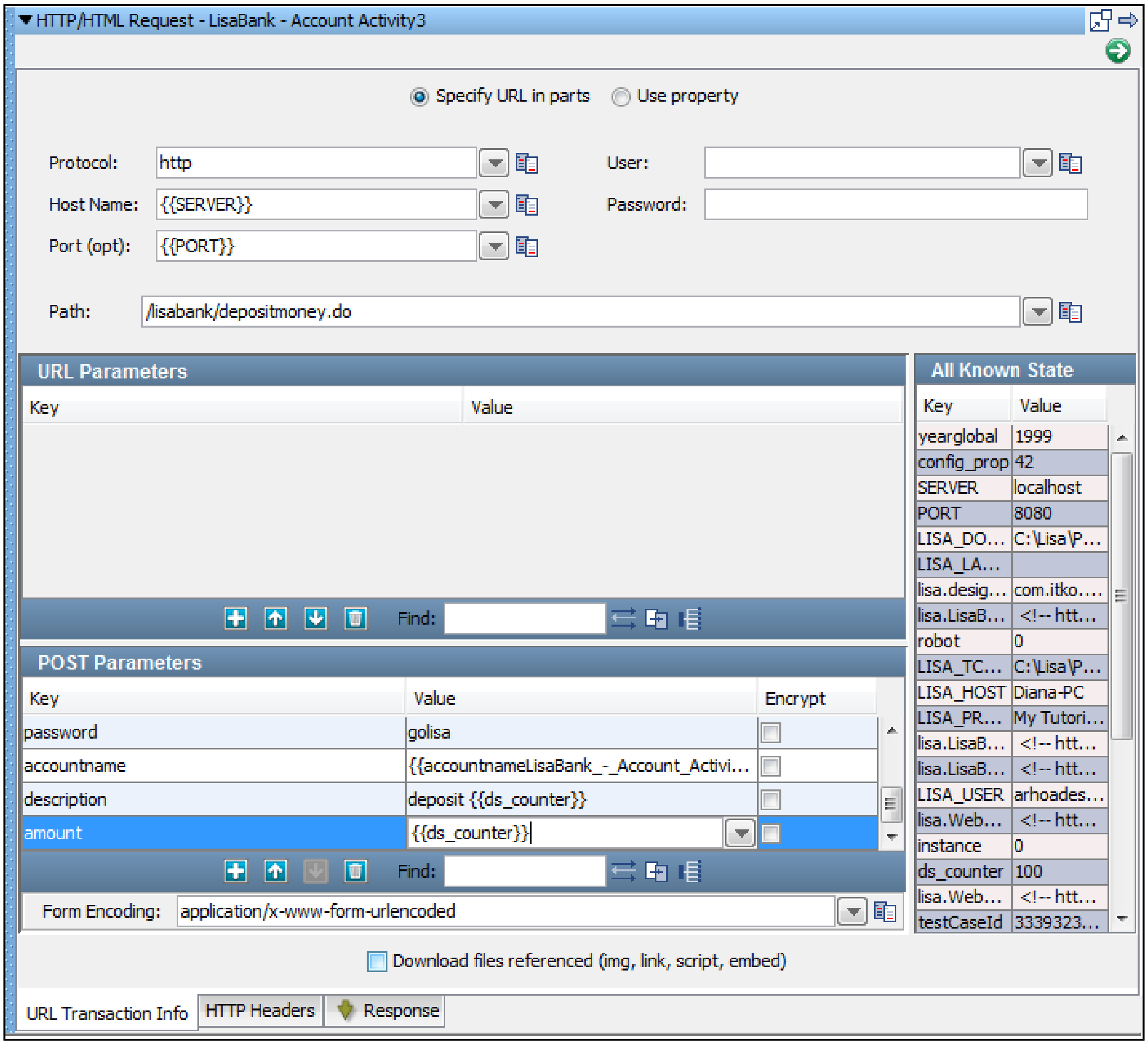
7. Click OK.

Step 3 - Modify the POST Parameters for the Recorded Deposit

You will now use the **ds_counter** property (which you created in the data set) to specify varying amounts of money for the deposit.

To modify the POST parameters for the recorded deposit:

1. In the model editor, double-click the **LISABank - Account Activity3** step.
2. In the **POST Parameters** area, change the value of the **description** key to **deposit {{`ds_counter`}}**.
3. Change the value of the **amount** key to **{{`ds_counter`}}**.



HTTP/HTML Request - LisaBank - Account Activity3

● Specify URL in parts ○ Use property

Protocol: http
Host Name: {{SERVER}}
Port (opt): {{PORT}}
User:
Password:

Path: /lisabank/depositmoney.do

URL Parameters

Key	Value

POST Parameters

Key	Value	Encrypt
password	golisa	□
accountname	{{accountnameLisaBank_-_Account_Activi…	□
description	deposit {{ds_counter}}	□
amount	{{ds_counter}}	□

Form Encoding: application/x-www-form-urlencoded

All Known State

Key	Value
yearglobal	1999
config_prop	42
SERVER	localhost
PORT	8080
LISA_DO…	C:\Lisa\P…
LISA_LA…	
lisa.desig…	com.itko.…
lisa.LisaB…	<!-- htt…
robot	0
LISA_TC…	C:\Lisa\P…
LISA_HOST	Diana-PC
LISA_PR…	My Tutori…
lisa.LisaB…	<!-- htt…
lisa.LisaB…	<!-- htt…
LISA_USER	arhoades…
lisa.Web…	<!-- htt…
instance	0
ds_counter	100
lisa.Web…	<!-- htt…
testCaseId	3339323…

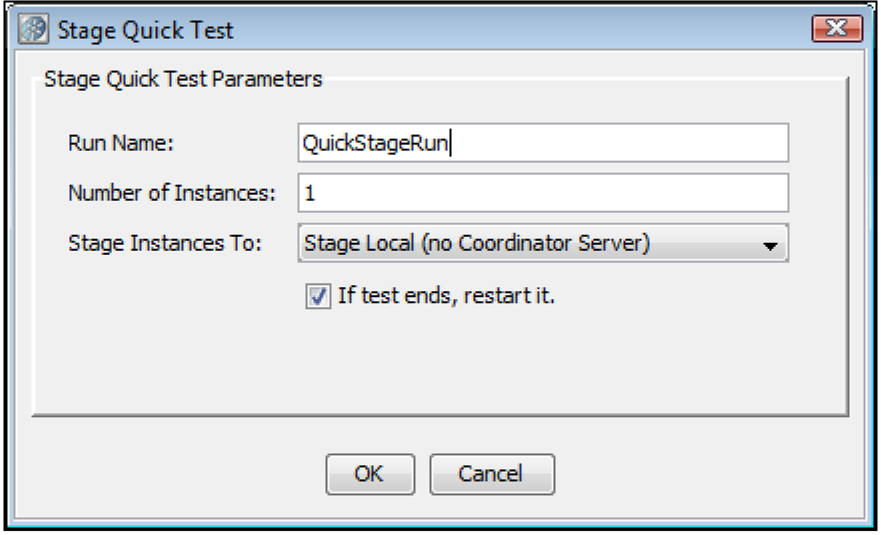
□ Download files referenced (img, link, script, embed)

4. Save the test case.

Step 4 - Stage the Test Case

To stage (or run) a Quick Test:

1. From the toolbar, click the Stage a quick test icon.
2. In the Stage Quick Test window, ensure that **If test ends, restart it** is selected.

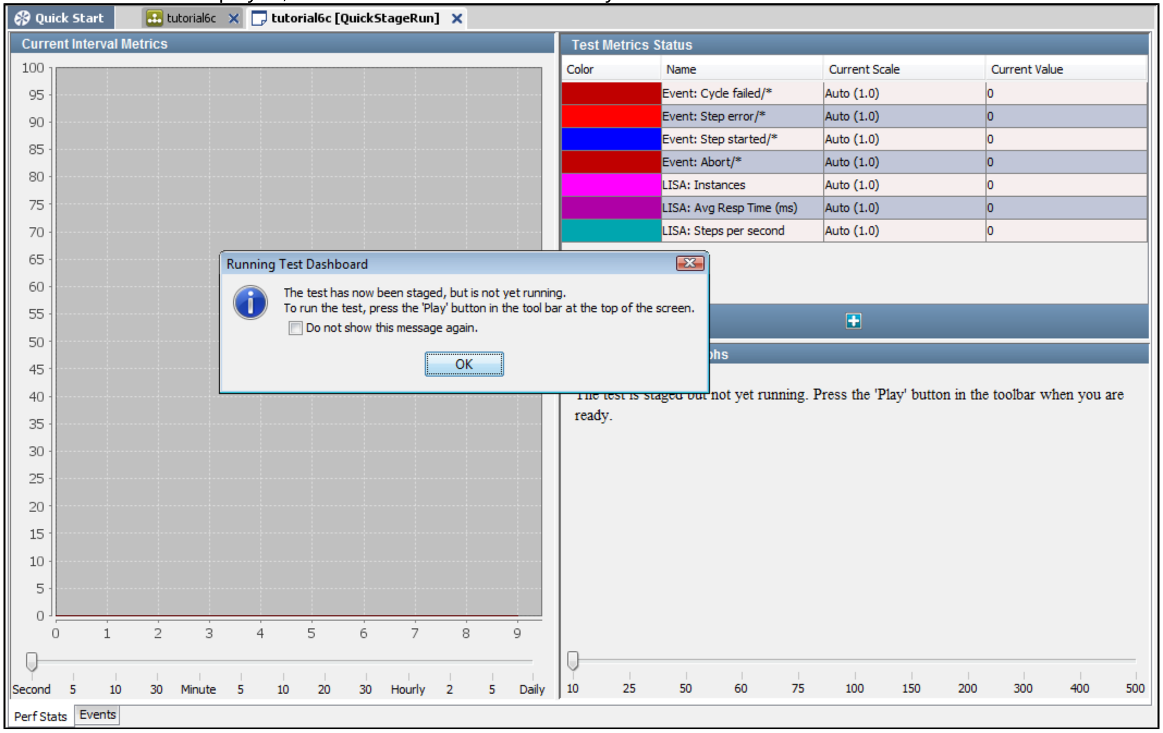


Stage Quick Test

Stage Quick Test Parameters

Run Name: QuickStageRun
Number of Instances: 1
Stage Instances To: Stage Local (no Coordinator Server)
☑ If test ends, restart it.

3. Click OK.
4. The Test Monitor is displayed, but the test has not been started yet.



Color	Name	Current Scale	Current Value
🟥	Event: Cycle failed/*	Auto (1.0)	0
🟥	Event: Step error/*	Auto (1.0)	0
🟦	Event: Step started/*	Auto (1.0)	0
🟥	Event: Abort/*	Auto (1.0)	0
🟪	LISA: Instances	Auto (1.0)	0
🟪	LISA: Avg Resp Time (ms)	Auto (1.0)	0
🟦	LISA: Steps per second	Auto (1.0)	0

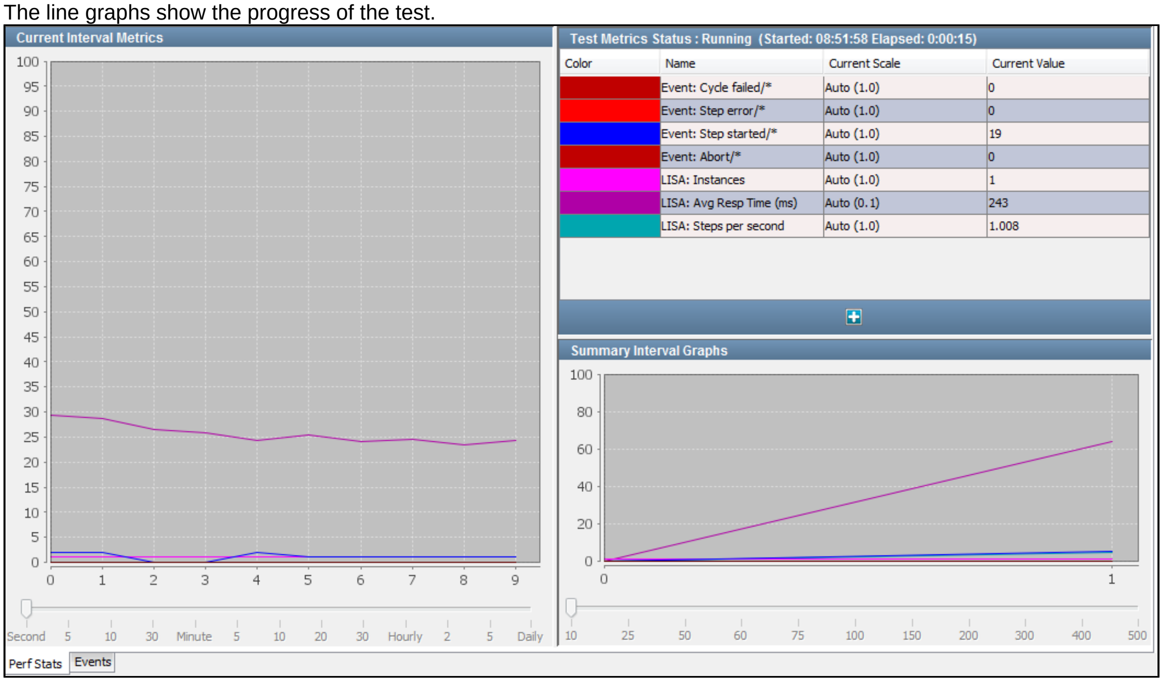
The test has now been staged, but is not yet running.
To run the test, press the 'Play' button in the tool bar at the top of the screen.

☐ Do not show this message again.

The test is staged but not yet running. Press the 'Play' button in the toolbar when you are ready.

5. Click OK.
6. From the toolbar, click the Play ➜ icon.

The line graphs show the progress of the test.



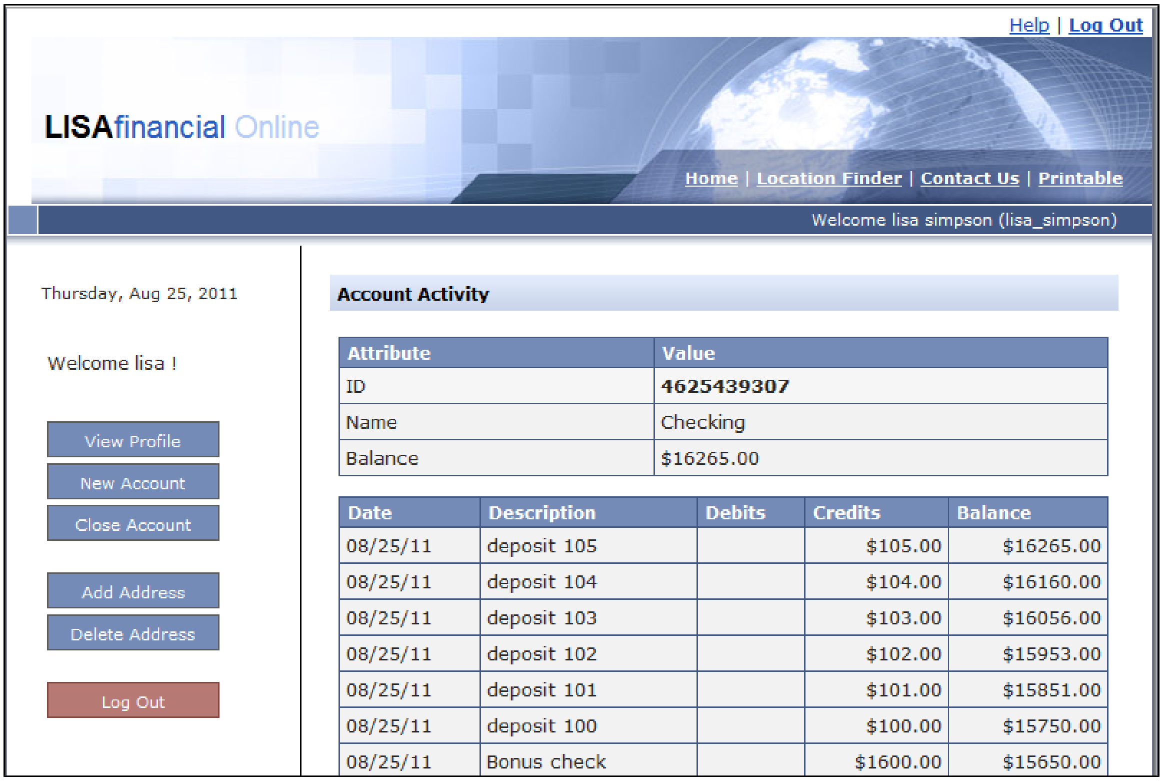
Test Metrics Status : Running (Started: 08:51:58 Elapsed: 0:00:15)

Color	Name	Current Scale	Current Value
🟥	Event: Cycle failed/*	Auto (1.0)	0
🟥	Event: Step error/*	Auto (1.0)	0
🟦	Event: Step started/*	Auto (1.0)	19
🟥	Event: Abort/*	Auto (1.0)	0
🟪	LISA: Instances	Auto (1.0)	1
🟪	LISA: Avg Resp Time (ms)	Auto (0.1)	243
🟦	LISA: Steps per second	Auto (1.0)	1.008

Step 5 - View the New Deposits in LISA Bank

To view the new deposits in LISA Bank:

1. Log in to the LISA Bank application again with the user **lisa_simpson** and password **golisa**.
2. Click the account number link for checking account to view the deposits. Notice how the deposits begin with 100 and increase by 1 until the amount 105 is reached.



Welcome lisa simpson (lisa_simpson)

Thursday, Aug 25, 2011

Welcome lisa !

Account Activity

Attribute	Value
ID	**4625439307**
Name	Checking
Balance	$16265.00

Date	Description	Debits	Credits	Balance
08/25/11	deposit 105		$105.00	$16265.00
08/25/11	deposit 104		$104.00	$16160.00
08/25/11	deposit 103		$103.00	$16056.00
08/25/11	deposit 102		$102.00	$15953.00
08/25/11	deposit 101		$101.00	$15851.00
08/25/11	deposit 100		$100.00	$15750.00
08/25/11	Bonus check		$1600.00	$15650.00

Review

In this tutorial, you used a numeric counting data set to provide input to the recorded test.

You did the following:

- Copied a test case and added a numeric counting data set.
- Modified the POST Parameters for the recorded deposit.
- Staged a quick test.

Tutorial 7 - Testing an Enterprise JavaBean (EJB)

The LISA Bank application provides a full set of EJBs to interact with an account, get the user and account information from the Java interface.

In this tutorial, you will use the **Enterprise JavaBean Execution** test step to call EJB methods within a test case and test the response with an assertion. You test a simple EJB to verify that the **addUser** and **deleteUser** methods work as expected.

> This tutorial is currently not working. When you click the Execute icon in Step 6 - Configure the EJB, an invocation exception occurs.

LISA Concepts Discussed

In this tutorial, you do the following:

- Use the Enterprise Java Bean Execution step.
- Use the Complex Object Editor with EJB objects.

Prerequisites

- You have completed Tutorial 6 - Testing a Website.
- LISA Workstation is open.
- You have access to the demo server (either the local Demo Server, or the ITKO demo server).

Steps

Step 1 - Create a New Test Case

To create a new test case,

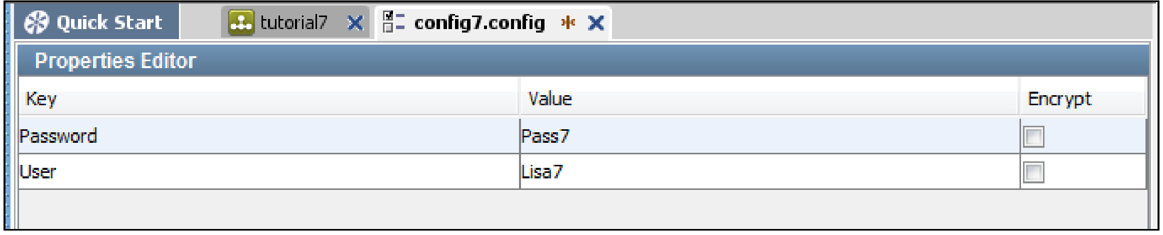
1. In the Project pane, right-click the **Tests** folder and select **Create New Test Case**.
2. Set the file name to **tutorial7**.
3. Click Save.

Step 2 - Create a New Configuration

You previously worked with configurations in Tutorial 2 - Data Sets.

To create a new configuration,

1. Open the **project.config** file.
2. If the configuration does not contain the **User** and **Password** properties, add these properties. You do not need to set the values.
3. Create a new configuration named **config7**.
4. Add the **User** property to the **config7** configuration and set the value to **Lisa7**.
5. Add the **Password** property to the **config7** configuration and set the value to **Pass7**.



Key	Value	Encrypt
Password	Pass7	□
User	Lisa7	□

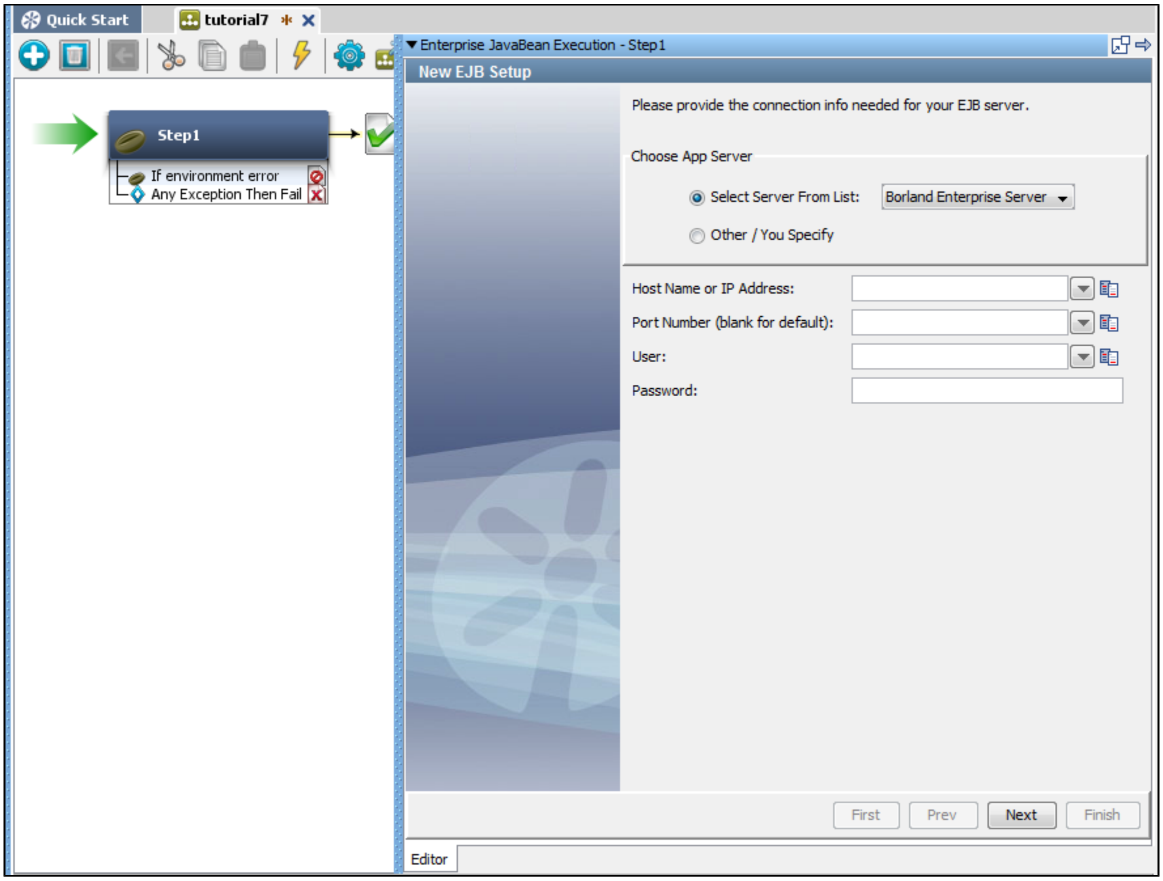
6. Click the Save icon.
7. In the Project pane, right-click the **config7** configuration and select **Make Active**. The configuration now appears in blue.

Step 3 - Add an EJB Test Step

The Enterprise JavaBean Execution test step enables you to make calls on a running EJB.

To add an EJB test step,

1. Click the **tutorial7** tab.
2. Click the Add Step ⊕ icon.
3. **Select Java/J2EE** and select **Enterprise JavaBean Execution**. The New EJB Setup wizard appears.



Please provide the connection info needed for your EJB server.

Choose App Server

● Select Server From List: Borland Enterprise Server

○ Other / You Specify

Host Name or IP Address:

Port Number (blank for default):

User:

Password:

Step 4 - Connect to the Server

The New EJB Setup wizard prompts you to specify connection information for the EJB server.

To connect to the server,

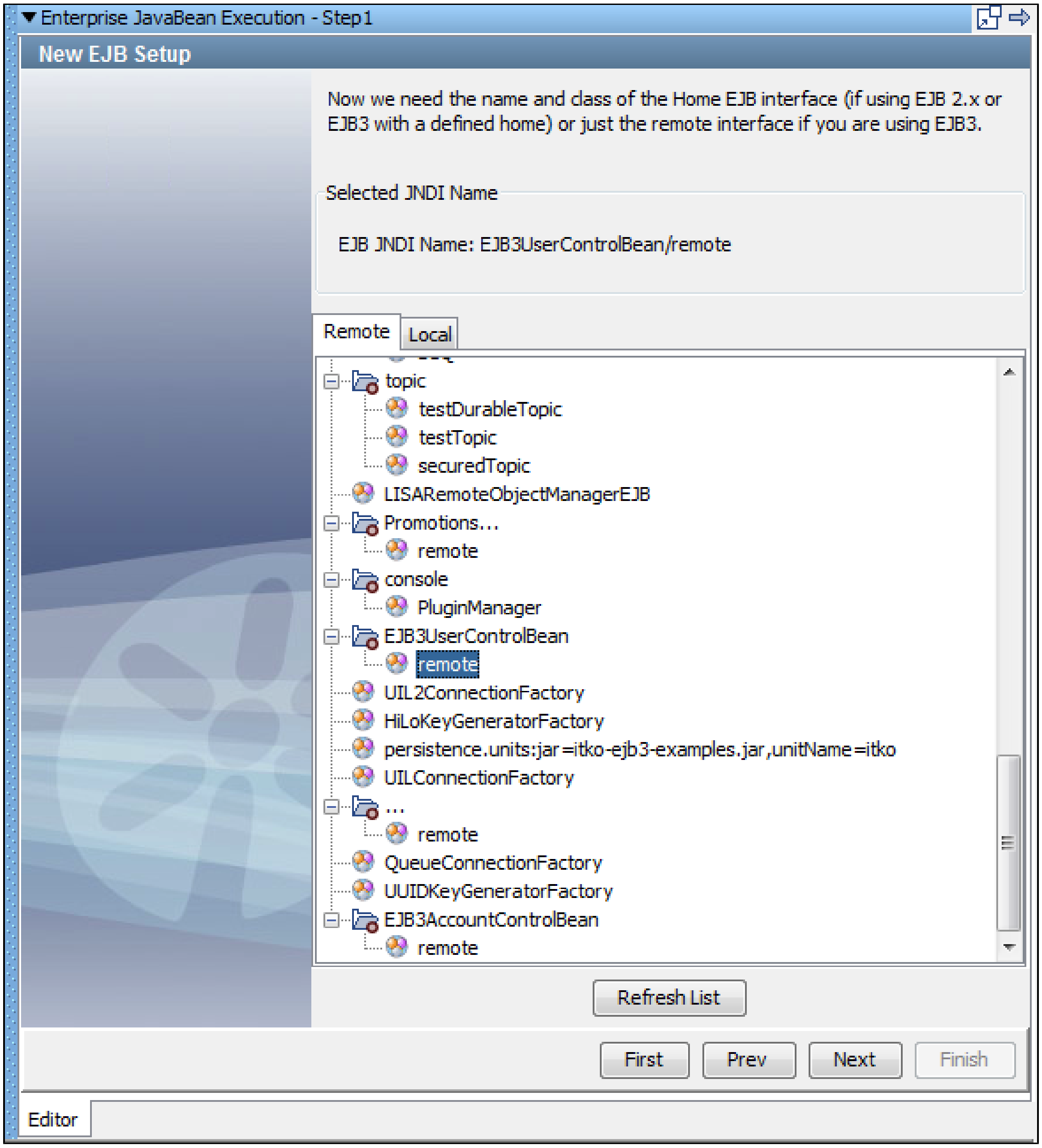
1. From the **Select Server From List** drop-down list, select **JBoss 3.2/4.0**.
2. In the **Host Name or IP Address** field, enter **localhost** if you are using the local demo server. Enter **examples.itko.com** if you are running against the iTKO demo server.
3. Click Next.
   The list of JNDI names is retrieved from the EJB server.

Step 5 - Locate the EJB Interface

The New EJB Setup wizard prompts you to specify the name of the EJB interface.

To locate the EJB interface,

1. In the **Remote** tab, select **EJB3UserControlBean/remote**.

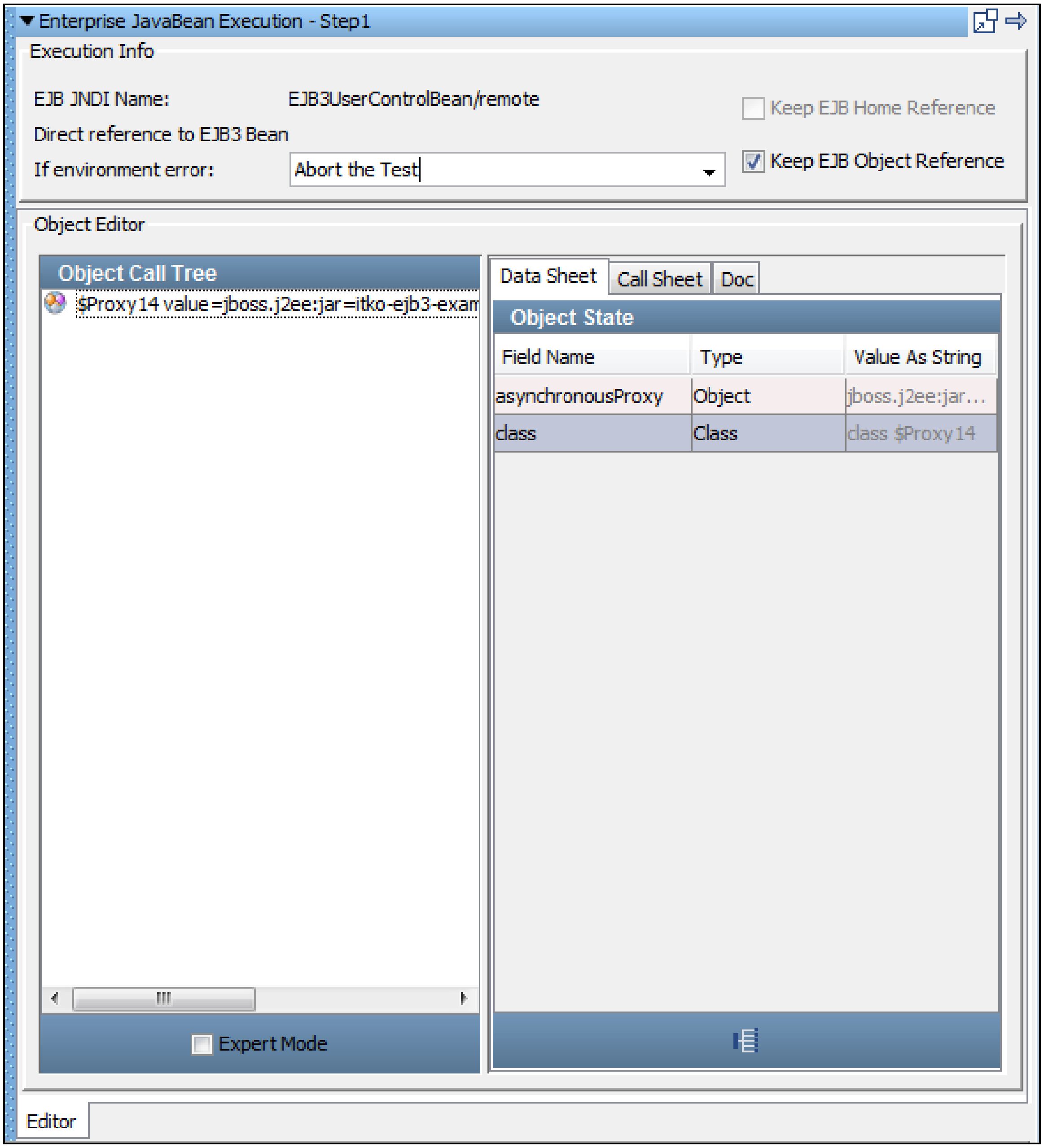


Now we need the name and class of the Home EJB interface (if using EJB 2.x or EJB3 with a defined home) or just the remote interface if you are using EJB3.

Selected JNDI Name

EJB JNDI Name: EJB3UserControlBean/remote

2. Click Next. The Complex Object Editor opens.



Enterprise JavaBean Execution - Step1

Execution Info

EJB JNDI Name: EJB3UserControlBean/remote

Direct reference to EJB3 Bean

If environment error: Abort the Test

□ Keep EJB Home Reference

☑ Keep EJB Object Reference

Object Editor

Object Call Tree

$Proxy14 value=jboss.j2ee:jar=itko-ejb3-exam

□ Expert Mode

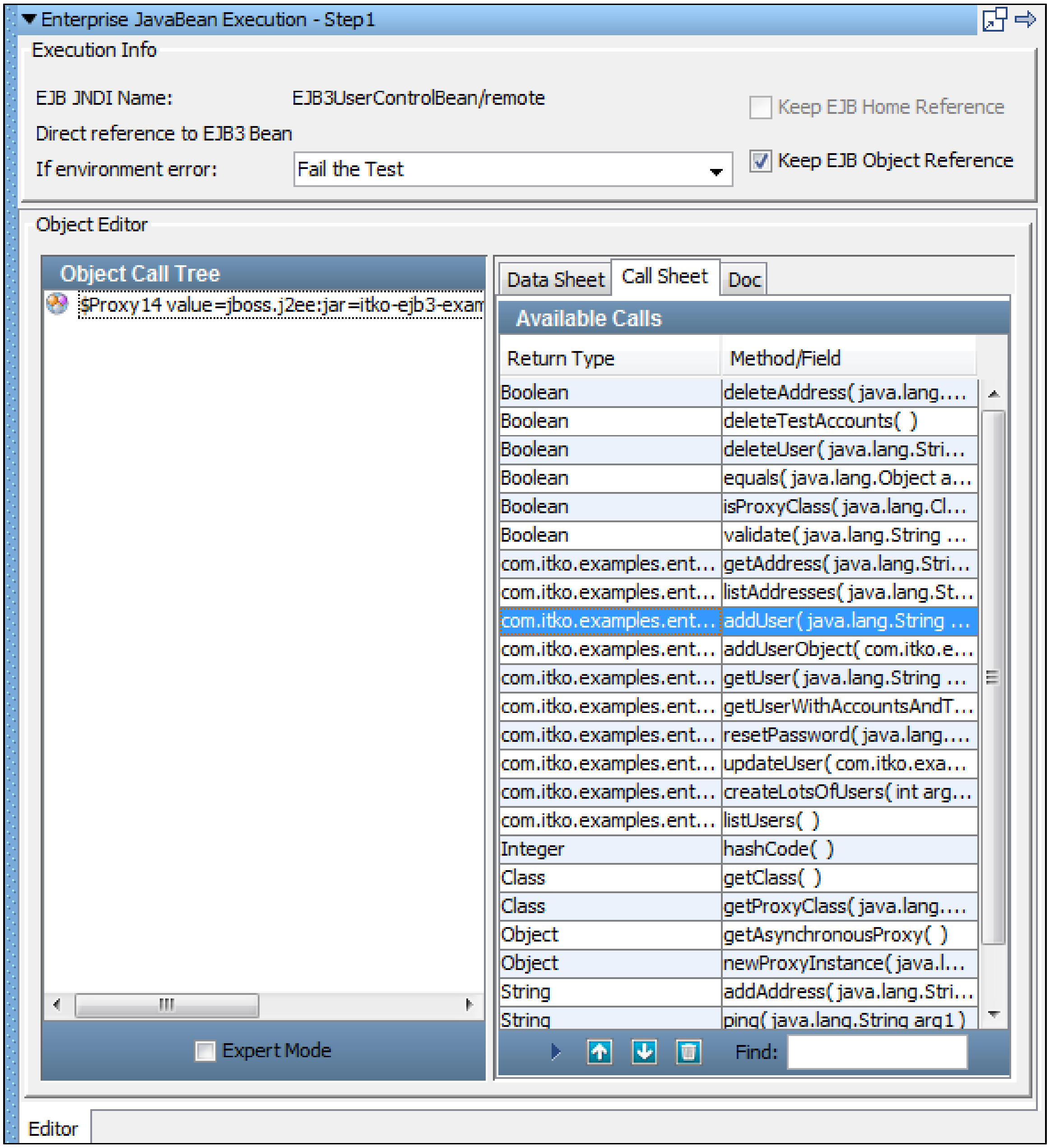
Object State

Field Name	Type	Value As String
asynchronousProxy	Object	jboss.j2ee:jar...
class	Class	class $Proxy14

Step 6 - Configure the EJB

To configure the EJB,

1. If you use the same EJB object and home object repeatedly, check the **Keep EJB Home Reference** and the **Keep EJB Object Reference** check boxes, if they are not already selected by default.
2. Set the **If environment error** field to the step to execute if an exception occurs while executing this EJB Step. Select **Fail the Test** from the list.
3. In the Object Editor area, select the **Call Sheet** tab and select the **addUser** method.
4. Click the Invoke Method ▶ icon.



Enterprise JavaBean Execution - Step1

Execution Info

EJB JNDI Name: EJB3UserControlBean/remote

Direct reference to EJB3 Bean

If environment error: Fail the Test

□ Keep EJB Home Reference

☑ Keep EJB Object Reference

Object Editor

Object Call Tree

$Proxy14 value=jboss.j2ee:jar=itko-ejb3-exam

□ Expert Mode

Data Sheet | Call Sheet | Doc

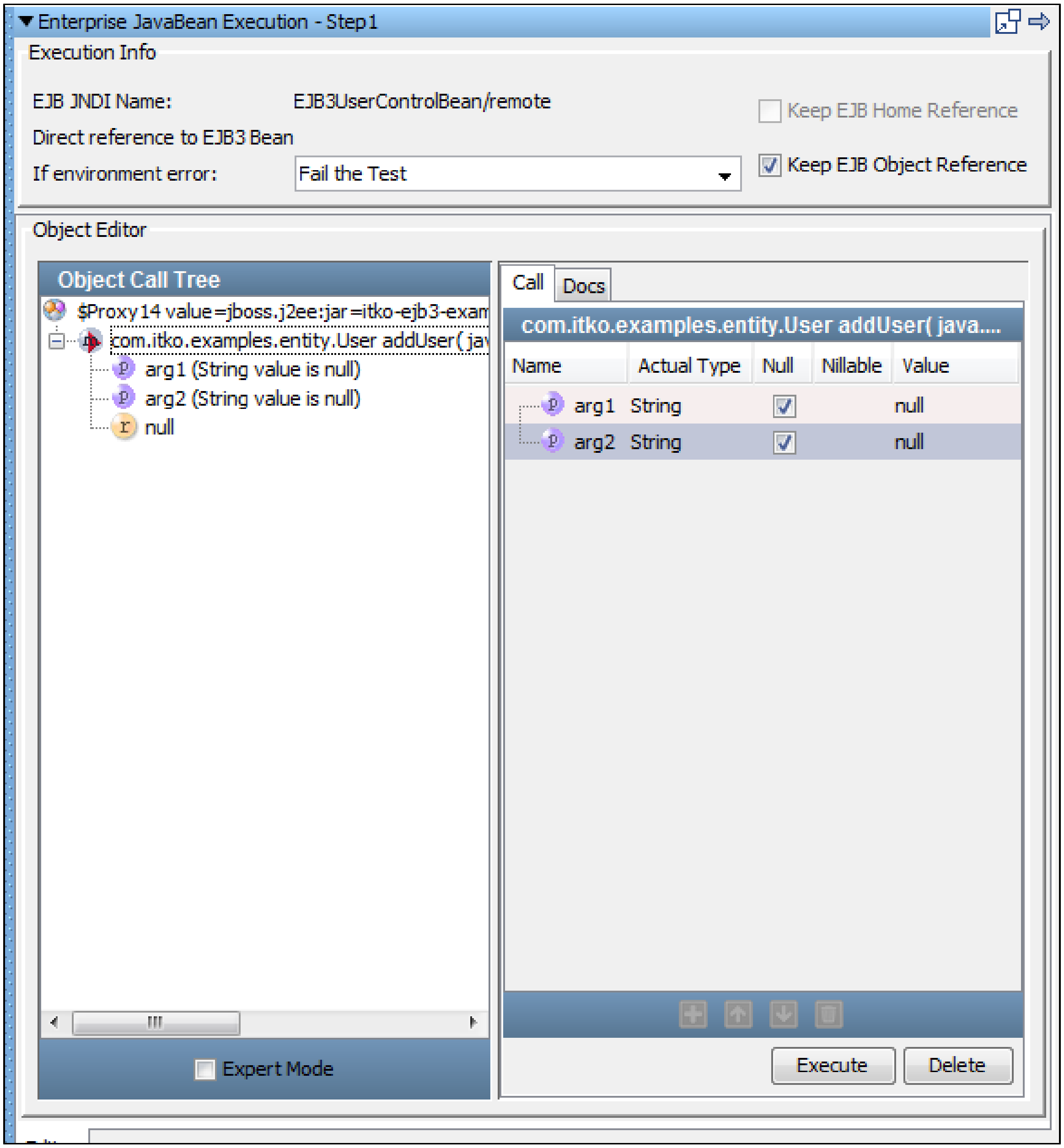
Available Calls

Return Type	Method/Field
Boolean	deleteAddress(java.lang....
Boolean	deleteTestAccounts( )
Boolean	deleteUser(java.lang.Stri...
Boolean	equals(java.lang.Object a...
Boolean	isProxyClass(java.lang.Cl...
Boolean	validate(java.lang.String ...
com.itko.examples.ent...	getAddress(java.lang.Stri...
com.itko.examples.ent...	listAddresses(java.lang.St...
com.itko.examples.ent...	addUser(java.lang.String ...
com.itko.examples.ent...	addUserObject(com.itko.e...
com.itko.examples.ent...	getUser(java.lang.String ...
com.itko.examples.ent...	getUserWithAccountsAndT...
com.itko.examples.ent...	resetPassword(java.lang....
com.itko.examples.ent...	updateUser(com.itko.exa...
com.itko.examples.ent...	createLotsOfUsers(int arg...
com.itko.examples.ent...	listUsers( )
Integer	hashCode( )
Class	getClass( )
Class	getProxyClass(java.lang....
Object	getAsynchronousProxy( )
Object	newProxyInstance(java.l...
String	addAddress(java.lang.Stri...
String	ping(java.lang.String arg1)

Find:

Editor

The Object Call Tree now displays the **addUser** method.



EJB JNDI Name: EJB3UserControlBean/remote

Direct reference to EJB3 Bean

If environment error: Fail the Test

□ Keep EJB Home Reference

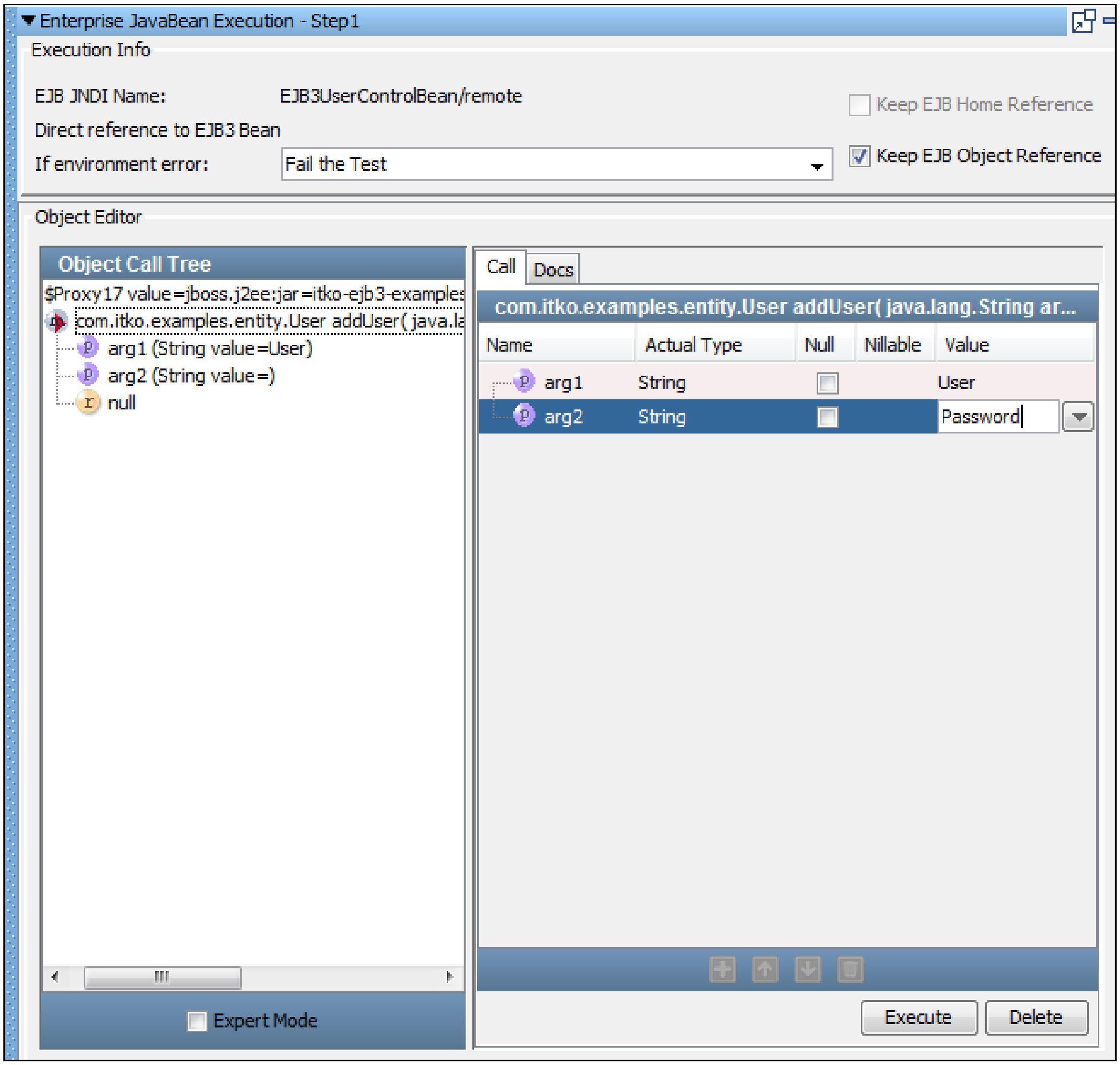
☑ Keep EJB Object Reference

$Proxy14 value=jboss.j2ee:jar=itko-ejb3-exam
com.itko.examples.entity.User addUser(jav
arg1 (String value is null)
arg2 (String value is null)
null

com.itko.examples.entity.User addUser(java....

Name	Actual Type	Null	Nillable	Value
arg1	String	☑		null
arg2	String	☑		null

5. Click the Execute icon.
   The Object Call Tree now indicates that the **addUser** method has been invoked.



Enterprise JavaBean Execution - Step1

Execution Info

EJB JNDI Name: EJB3UserControlBean/remote

Direct reference to EJB3 Bean

If environment error: Fail the Test

□ Keep EJB Home Reference

☑ Keep EJB Object Reference

Object Editor

Object Call Tree

$Proxy17 value=jboss.j2ee:jar=itko-ejb3-examples

com.itko.examples.entity.User addUser(java.la

arg1 (String value=User)

arg2 (String value=)

null

□ Expert Mode

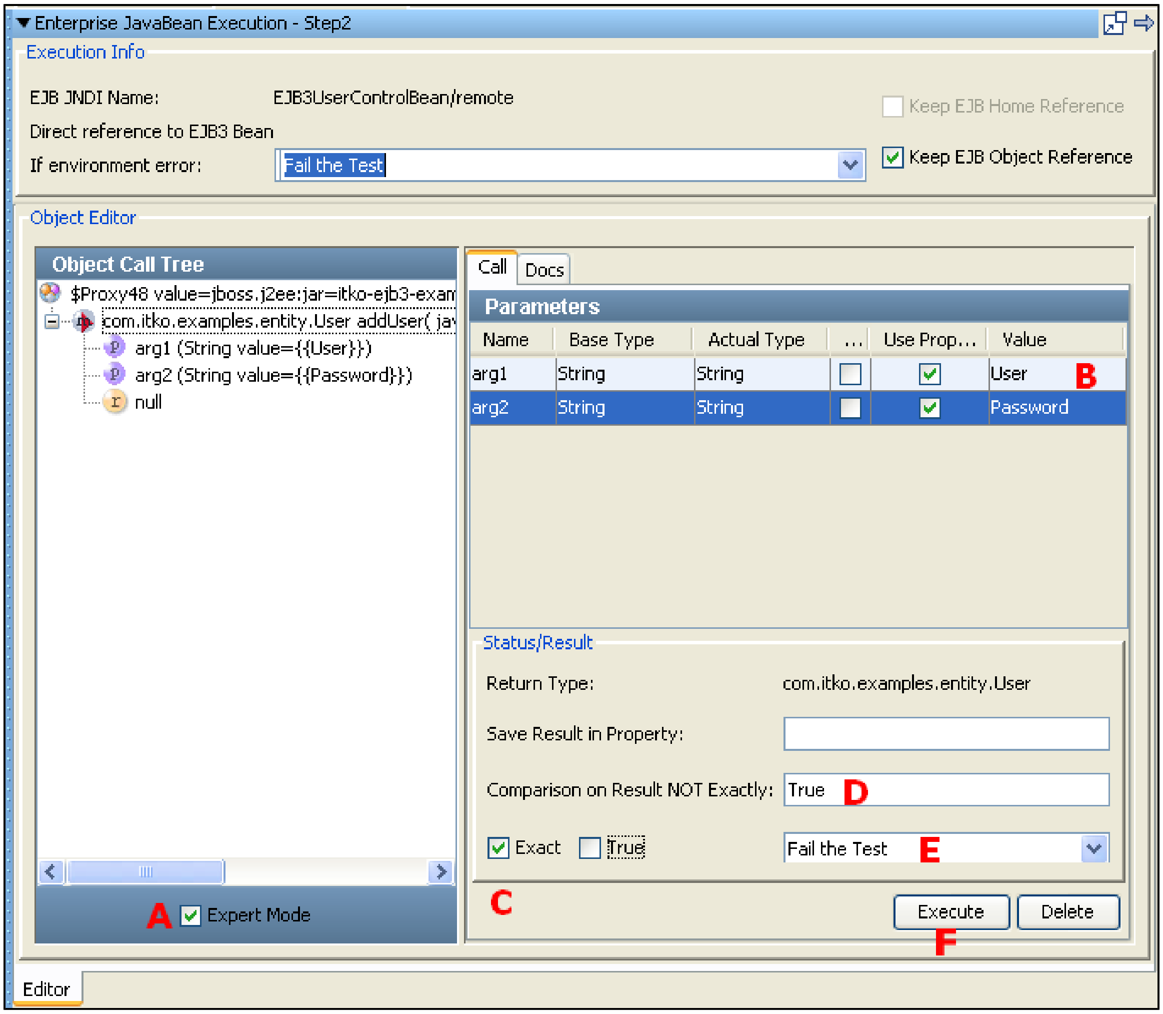
com.itko.examples.entity.User addUser(java.lang.String ar...

Name	Actual Type	Null	Nillable	Value
arg1	String	□		User
arg2	String	□		Password

Step 7 - Add an Assertion

To enter the method parameters and add an inline assertion,

1. In the Object Call Tree pane, check **Expert Mode** if it is not already enabled (see A).
2. Enter the property values for the **addUser** method parameters (arg1 and arg2). Enter the **User** and **Password** that you added to the configuration (see B).
3. In the **Status/Result** area, add the inline assertion by checking **Exact** and un-checking **True** (see C).
4. In the **Comparison on Result NOT Exactly** field, enter **True** (see D).
   (The addUser method returns a Boolean.)
5. From the Exact list, select **Fail the Test** (see E).
   (If the addUser method returns anything but true, it executes the **fail** step.)
6. Click Execute (see F).



Enterprise JavaBean Execution - Step2

Execution Info

EJB JNDI Name: EJB3UserControlBean/remote

Direct reference to EJB3 Bean

If environment error: Fail the Test

□ Keep EJB Home Reference

☑ Keep EJB Object Reference

Object Editor

Object Call Tree

- $Proxy48 value=jboss.j2ee:jar=itko-ejb3-exam
- com.itko.examples.entity.User addUser(ja
  - arg1 (String value={{User}})
  - arg2 (String value={{Password}})
  - null

☑ Expert Mode

Parameters

Name	Base Type	Actual Type	...	Use Prop...	Value
arg1	String	String	□	☑	User
arg2	String	String	□	☑	Password

Status/Result

Return Type: com.itko.examples.entity.User

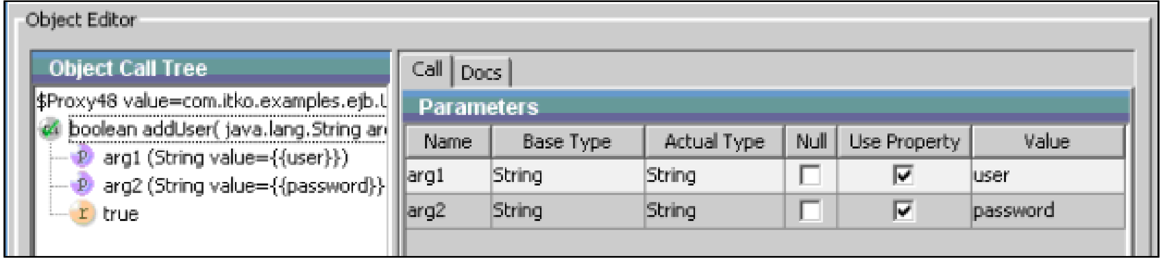
Save Result in Property:

Comparison on Result NOT Exactly: True

☑ Exact □ True

Fail the Test

The parameters to the method are displayed in the Object Call Tree, next to the input parameter p icon. The return value of this method is **true** in the Object Call Tree.



Object Editor

Object Call Tree

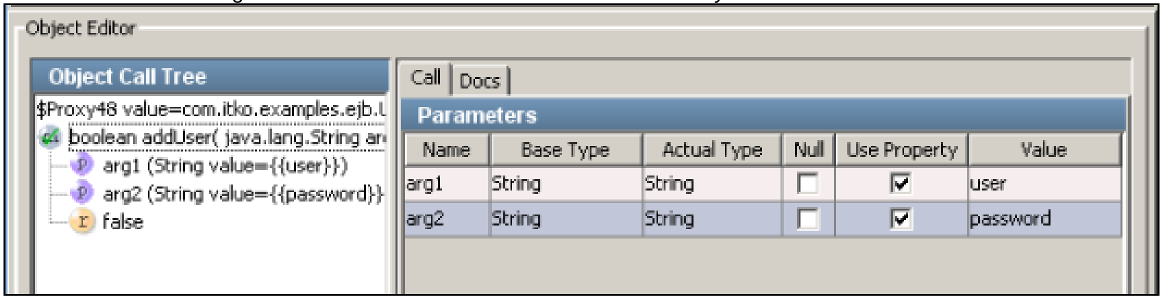
- $Proxy48 value=com.itko.examples.ejb.U
- boolean addUser(java.lang.String ar
  - arg1 (String value={{user}})
  - arg2 (String value={{password}})
  - true

Parameters

Name	Base Type	Actual Type	Null	Use Property	Value
arg1	String	String	□	☑	user
arg2	String	String	□	☑	password

7. Test the **addUser** method again by clicking Execute.

The return r changes from **true** to **false** because the user has already been added.



Object Editor

Object Call Tree

- $Proxy48 value=com.itko.examples.ejb.U
- boolean addUser(java.lang.String ar
  - arg1 (String value={{user}})
  - arg2 (String value={{password}})
  - false

Parameters

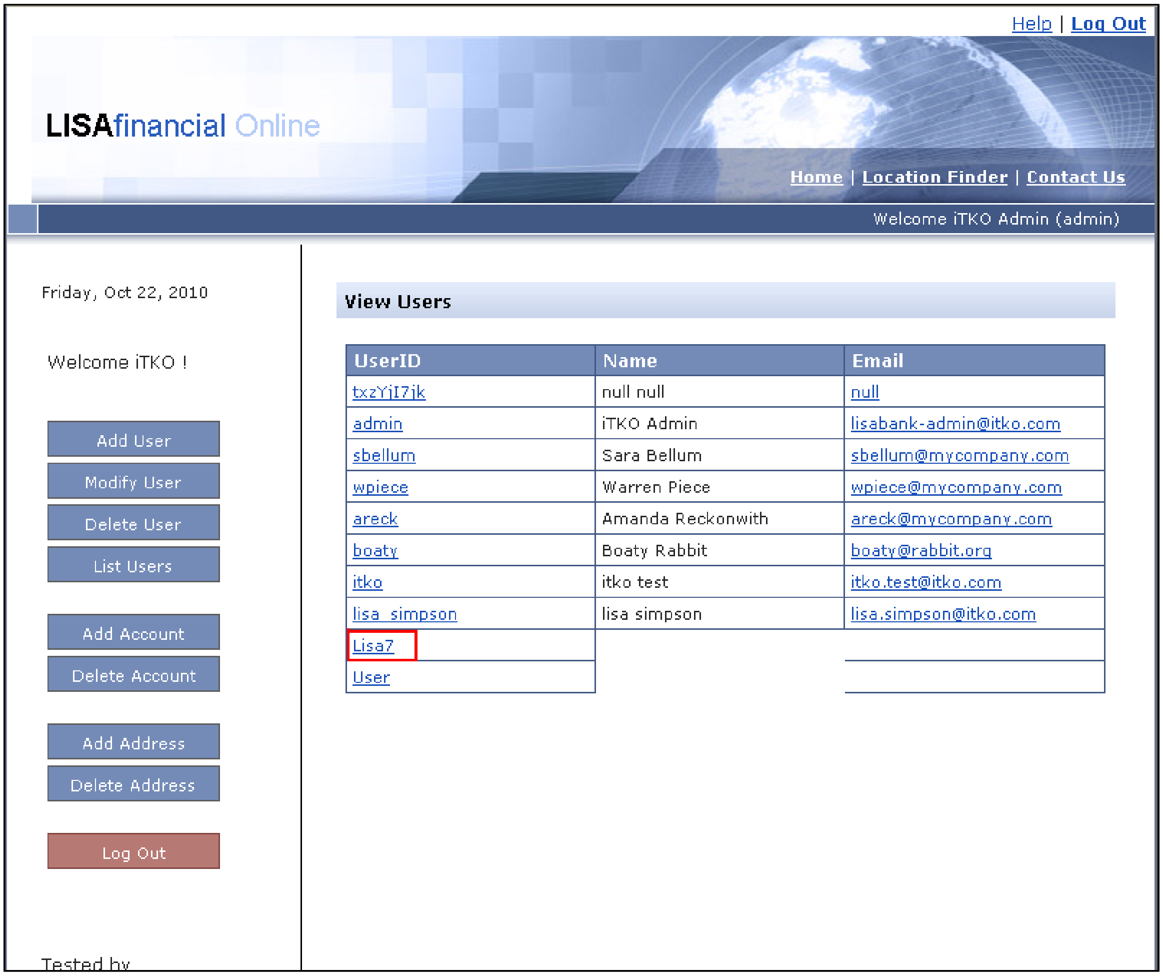
Name	Base Type	Actual Type	Null	Use Property	Value
arg1	String	String	□	☑	user
arg2	String	String	□	☑	password

Step 8 - Verify the Method Execution

From the LISA Bank application, you can verify that the user was added.

To verify the method execution,

1. Go to the LISA Bank application.
2. Login as user **admin** with the password **admin**.
3. View the list of users to confirm that **Lisa7** was added.



Friday, Oct 22, 2010

Welcome iTKO !

View Users

UserID	Name	Email
txzYjI7jk	null null	null
admin	iTKO Admin	lisabank-admin@itko.com
sbellum	Sara Bellum	sbellum@mycompany.com
wpiece	Warren Piece	wpiece@mycompany.com
areck	Amanda Reckonwith	areck@mycompany.com
boaty	Boaty Rabbit	boaty@rabbit.org
itko	itko test	itko.test@itko.com
lisa_simpson	lisa simpson	lisa.simpson@itko.com
Lisa7		
User		

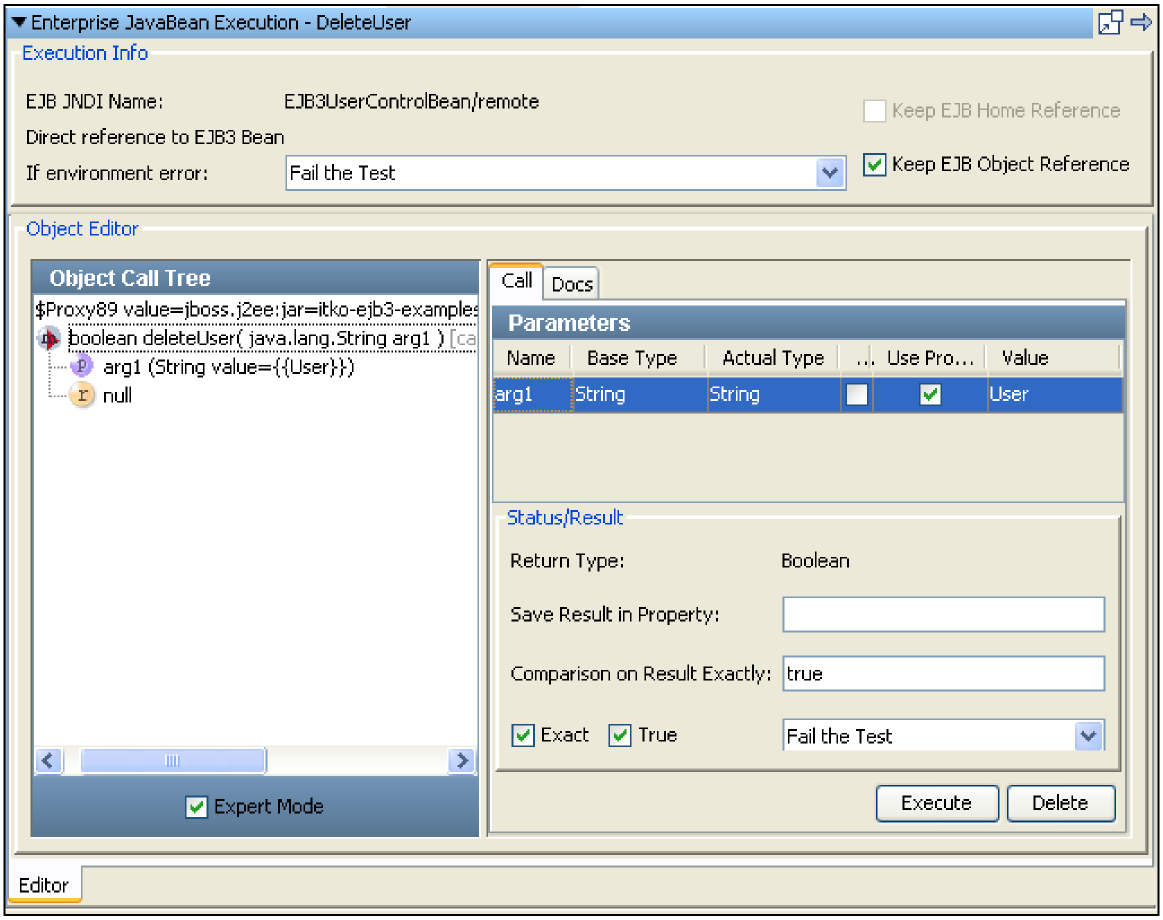
Tested by

Step 9 - Add Another EJB Test Step

Now try the preceding steps again to invoke the **deleteUser** method.

To add another EJB test step,

1. Repeat the tutorial beginning with Step 3 to add an EJB step named **DeleteUser**.
2. Use the method parameter property **User**.



▼ Enterprise JavaBean Execution - DeleteUser

Execution Info

EJB JNDI Name: EJB3UserControlBean/remote

Direct reference to EJB3 Bean

If environment error: Fail the Test

□ Keep EJB Home Reference

☑ Keep EJB Object Reference

Object Editor

Object Call Tree

$Proxy89 value=jboss.j2ee:jar=itko-ejb3-examples

boolean deleteUser(java.lang.String arg1) [ca

arg1 (String value={{User}})

null

☑ Expert Mode

Parameters

Name	Base Type	Actual Type	...	Use Pro...	Value
arg1	String	String	□	☑	User

Status/Result

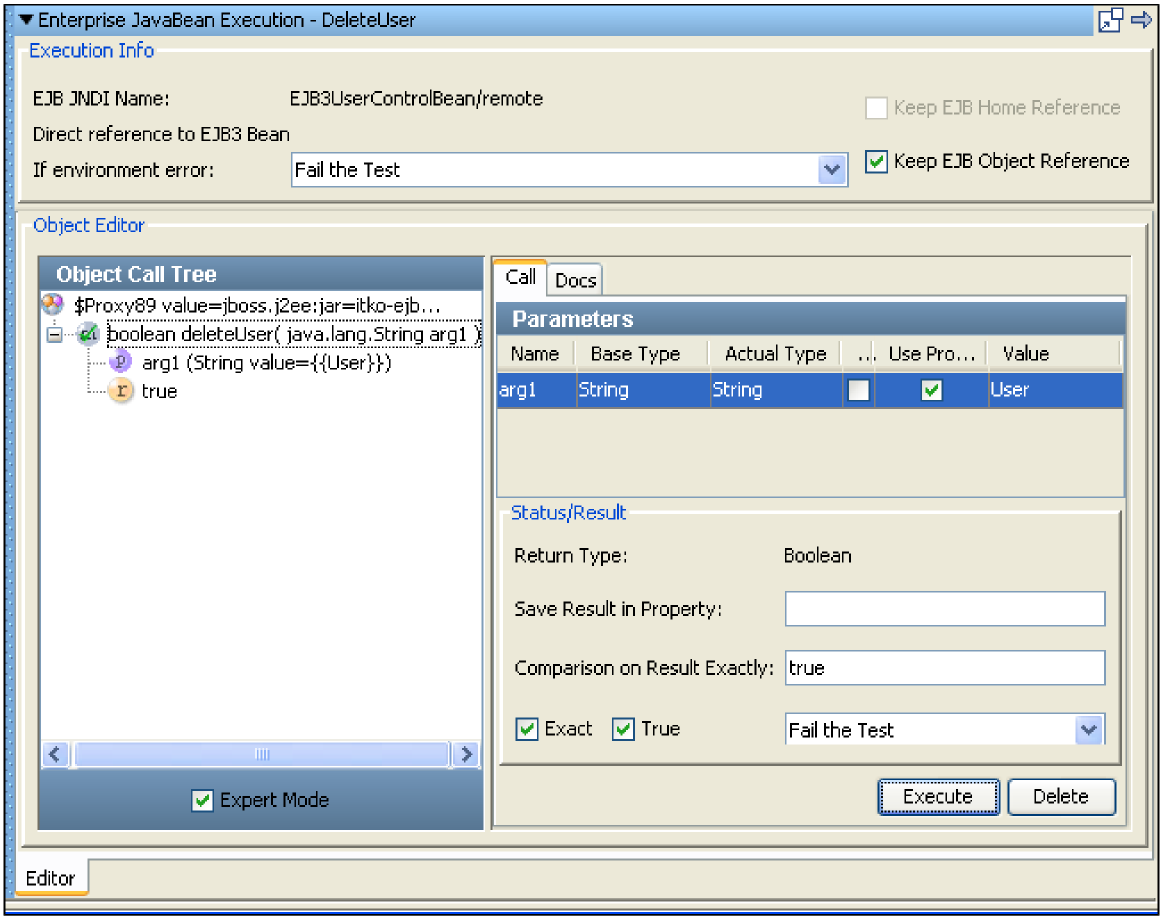
Return Type: Boolean

Save Result in Property:

Comparison on Result Exactly: true

☑ Exact ☑ True Fail the Test

3. Click Execute to execute this method and get results.



Enterprise JavaBean Execution - DeleteUser

Execution Info

EJB JNDI Name: EJB3UserControlBean/remote

Direct reference to EJB3 Bean

If environment error: Fail the Test

□ Keep EJB Home Reference

☑ Keep EJB Object Reference

Object Editor

Object Call Tree

$Proxy89 value=jboss.j2ee:jar=itko-ejb...

boolean deleteUser(java.lang.String arg1)

arg1 (String value={{User}})

true

☑ Expert Mode

Parameters

Name	Base Type	Actual Type	...	Use Pro...	Value
arg1	String	String	□	☑	User

Status/Result

Return Type: Boolean

Save Result in Property:

Comparison on Result Exactly: true

☑ Exact ☑ True Fail the Test

The return r is **true**, indicating the user has been deleted.

4. Click the Save icon.

Review

In this tutorial, you did the following:

- Created a test case consisting of two EJB test steps. The EJB object was loaded from the example application on the demo server.
- Created an EJB test step and loaded an EJB.
- Used the Complex Object Editor to manipulate EJB objects.

Tutorial 8 - Testing a Web Service

In this tutorial, you will use the Web Service Execution (XML) test step to call web service operations in a test case and test the request and response. These web service operations provide the same functionality as the equivalent method calls in the EJB used in Tutorial 7.

LISA Concepts Discussed

In this tutorial, you do the following:

- Add the Web Service Execution (XML) test step.
- Execute a web service operation.

Prerequisites

- You have completed Tutorial 5.
- LISA Workstation is open.
- You have access to the demo server (either the local Demo Server, or the ITKO demo server).

Steps

Step 1 - Create a New Test Case

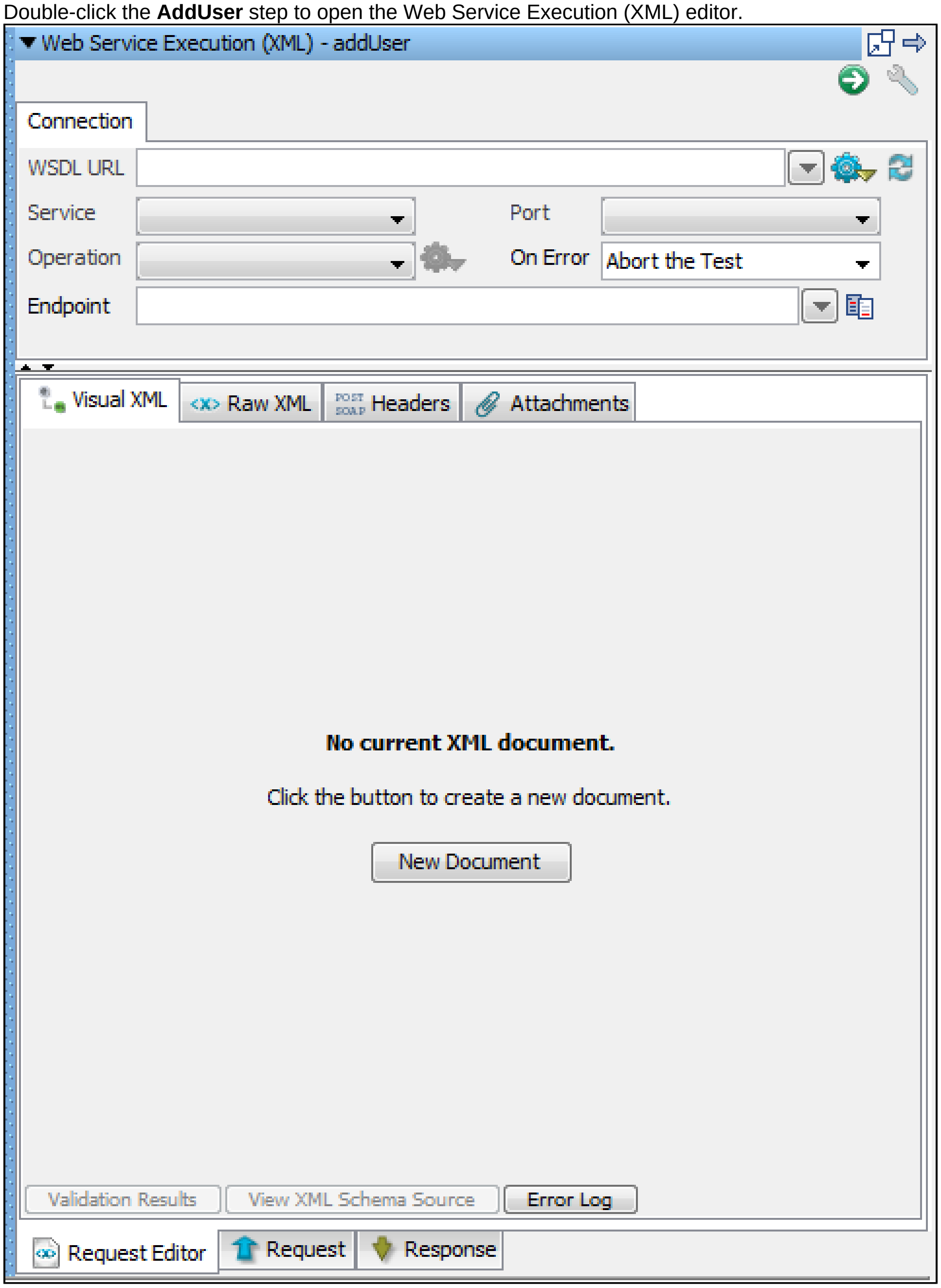
To create a new test case,

1. In the Project panel, right-click on the **Tests** folder and select **Create New Test Case**.
2. Set the file name to **tutorial8a**.
3. Click Save.
4. In the Project panel, right-click on **project.config** and select **Make active**.

Step 2 - Add a Web Service Execution (XML) Test Step

The Web Service Execution (XML) test step enables you to execute an operation on a SOAP-based web service. To add a Web Service Execution (XML) test step,

1. Click the **tutorial8a** tab.
2. Click the Add Step icon.
3. Select **Web/Web Services** and select **Web Service Execution (XML)**. **Step1** is added to the model editor.
4. Right-click **Step1** and select Rename. Change the name to **AddUser**.
5. Double-click the **AddUser** step to open the Web Service Execution (XML) editor.

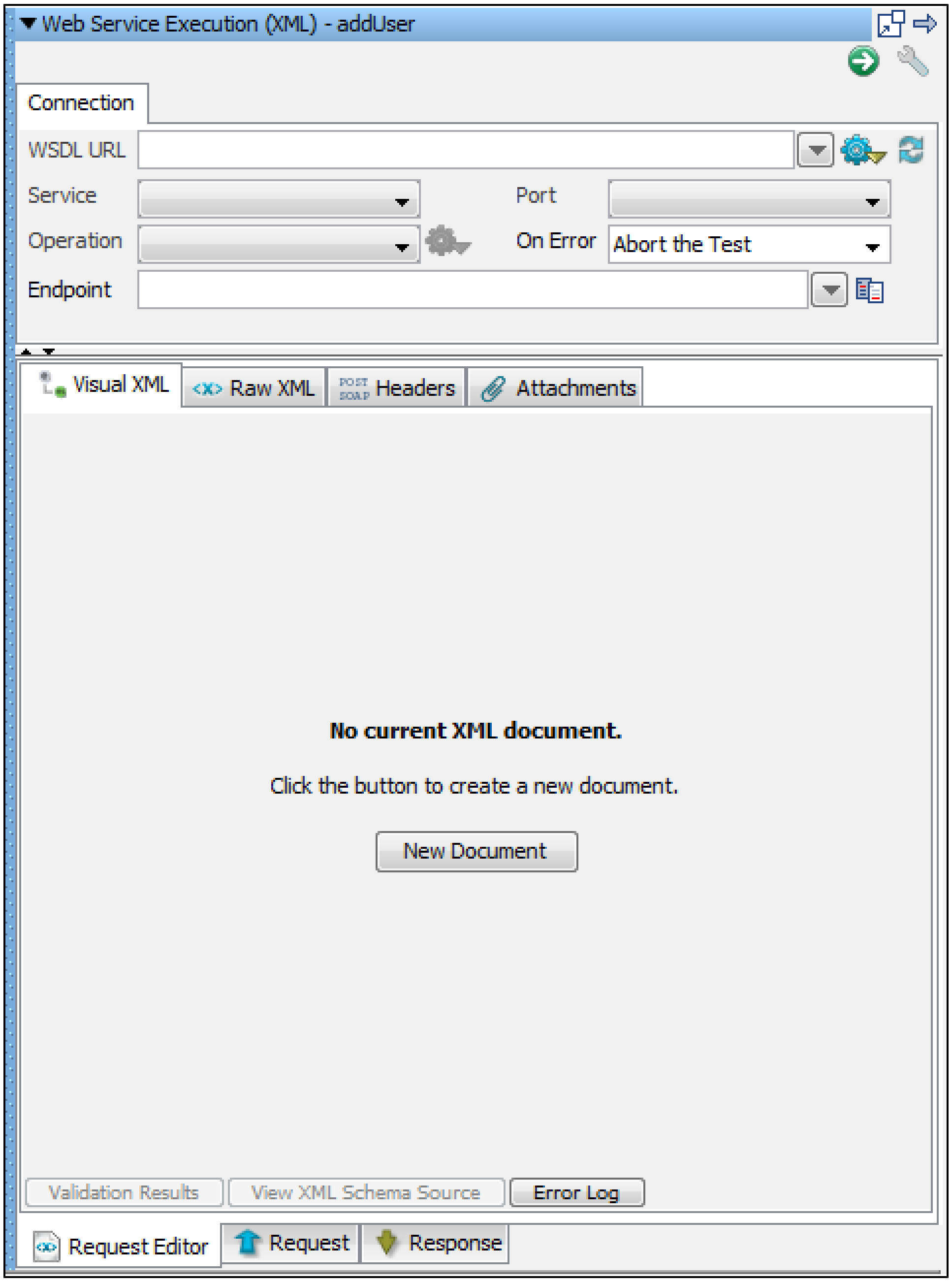


On Error: Abort the Test

No current XML document.

Click the button to create a new document.

6. Click the New Document button.



On Error: Abort the Test

No current XML document.

Click the button to create a new document.

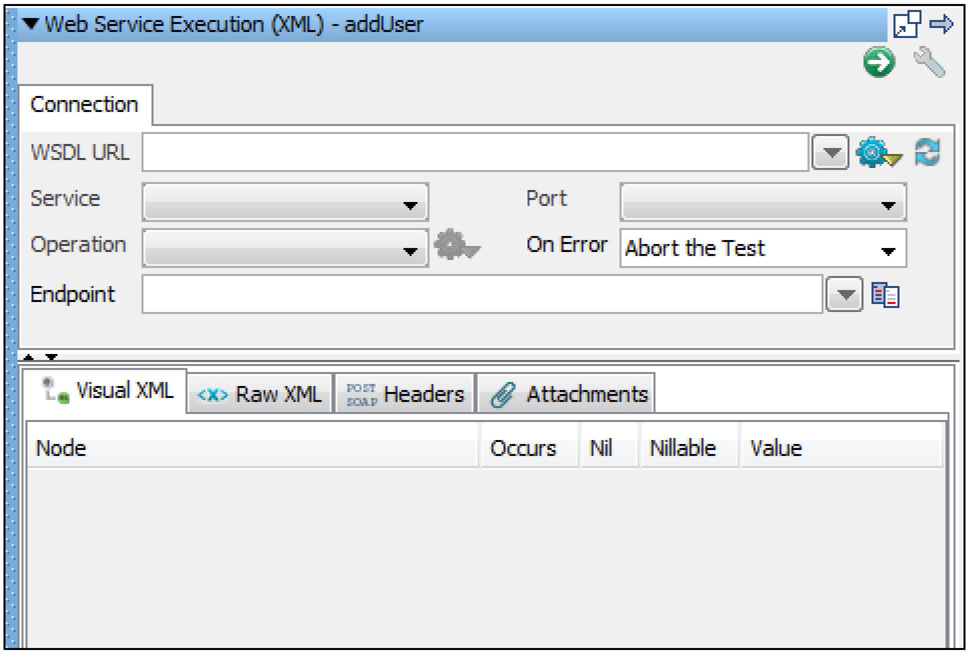
Step 3 - Create a Web Service Client

You now specify the operation to be called, and create a SOAP message to send to the operation. To create a web service client,

1. In the **WSDL URL**field, enter the following location. Notice the use of the WSSERVER and WSPORT properties to represent the server and port.

```
http://WSSERVER:WSPORT/itko-examples/services/UserControlService?wsdl
```

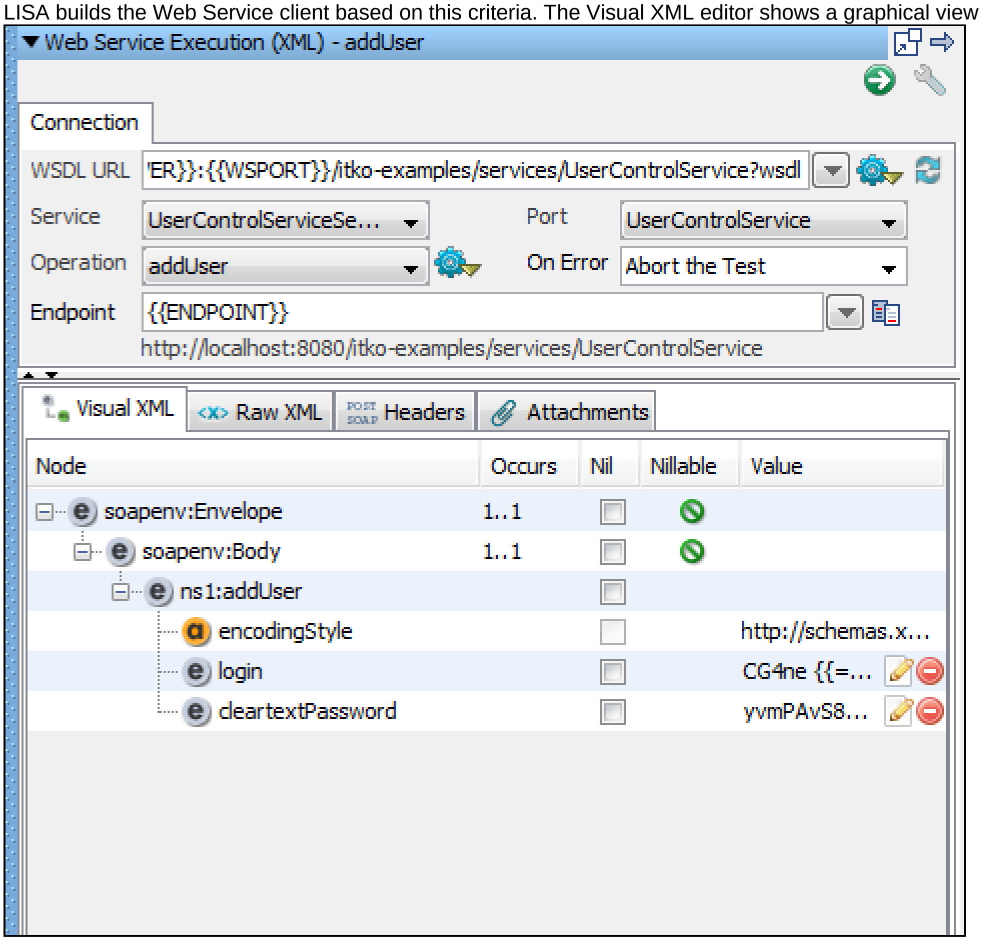
2. In the **Service** field, select **UserControlServiceService**.
3. In the **Port** field, select **UserControlService**.
4. In the **Operation** field, select the **addUser** operation.
5. In the **On Error** field, select **Abort the Test**.



WSDL URL:
Service:
Port:
Operation:
On Error: Abort the Test
Endpoint:

Node	Occurs	Nil	Nillable	Value

LISA builds the Web Service client based on this criteria. The Visual XML editor shows a graphical view of the SOAP message.



WSDL URL: 'ER}}:{{WSPORT}}/itko-examples/services/UserControlService?wsdl
Service: UserControlServiceSe...
Port: UserControlService
Operation: addUser
On Error: Abort the Test
Endpoint: {{ENDPOINT}}
http://localhost:8080/itko-examples/services/UserControlService

Node	Occurs	Nil	Nillable	Value
soapenv:Envelope	1..1	□	⊘	
soapenv:Body	1..1	□	⊘	
ns1:addUser		□		
encodingStyle		□		http://schemas.x...
login		□		CG4ne {{=...
cleartextPassword		□		yvmPAvS8...

6. Save the test case.

Step 4 - Execute the Web Service Request

To execute the web service request,

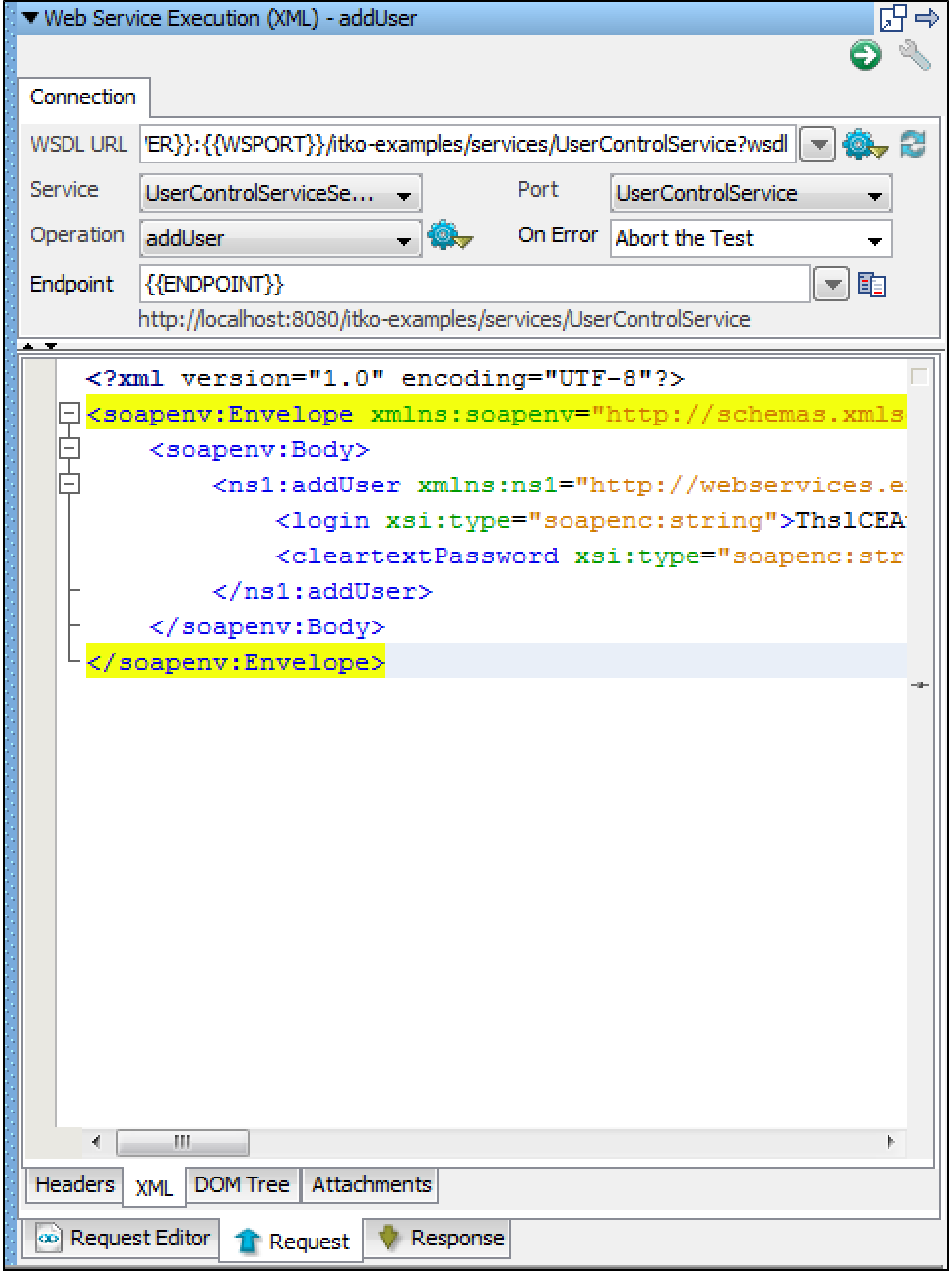
1. Click the Execute WS Request icon in the upper right corner. The test is executed.

Step 5 - View the Request and Response

The Request tab shows the resulting request data that was sent after any post processing (for example, substituting LISA properties). The Response tab shows the resulting response data that was received.

To view the request and response,

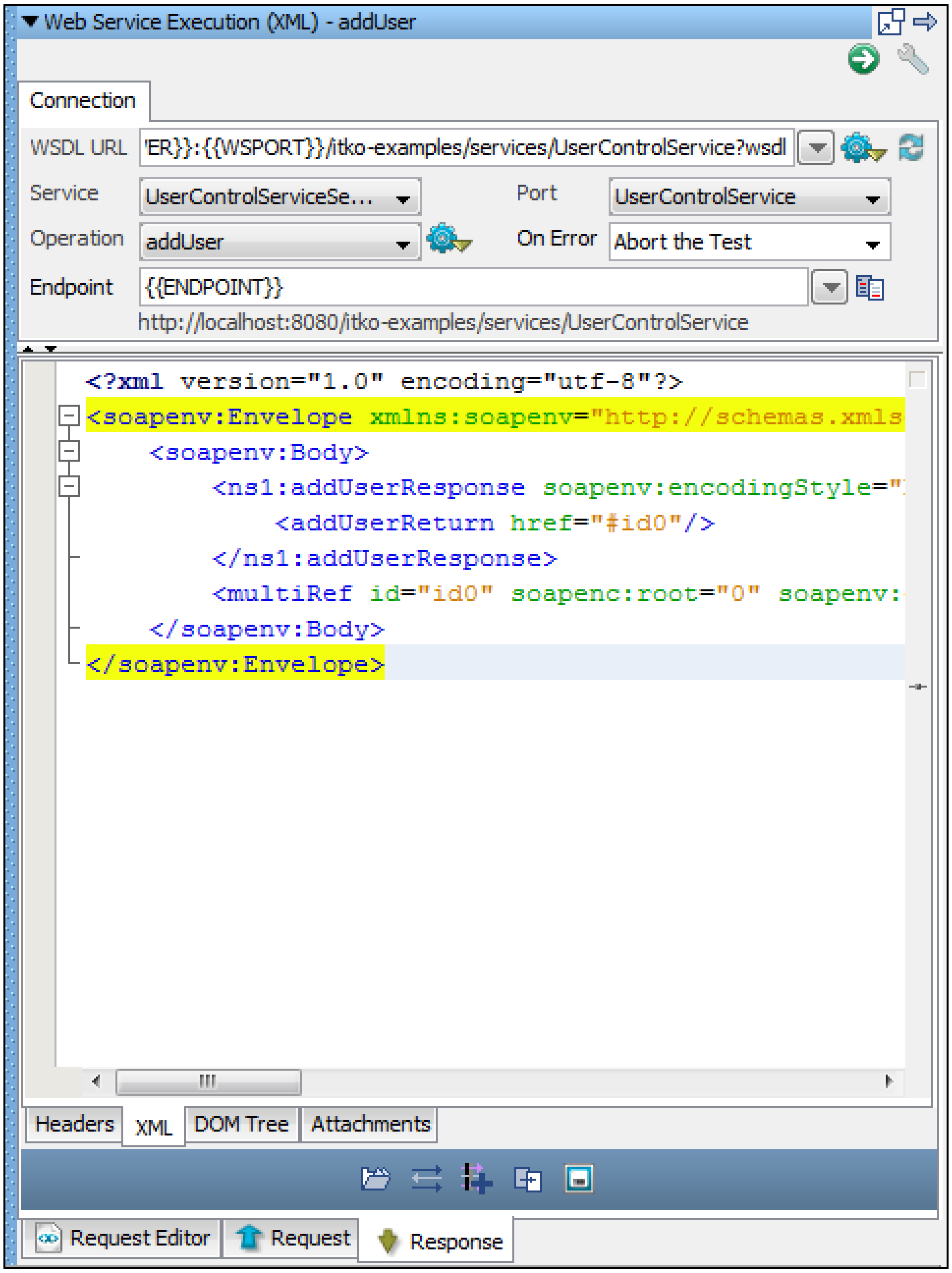
1. To view the request upon execution, click the **Request** tab.



WSDL URL 'ER}}:{{WSPORT}}/itko-examples/services/UserControlService?wsdl
Service: UserControlServiceSe...
Port: UserControlService
Operation: addUser
On Error: Abort the Test
Endpoint {{ENDPOINT}}
http://localhost:8080/itko-examples/services/UserControlService

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmls
    <soapenv:Body>
        <ns1:addUser xmlns:ns1="http://webservices.e
            <login xsi:type="soapenc:string">Ths1CEA
            <cleartextPassword xsi:type="soapenc:str
        </ns1:addUser>
    </soapenv:Body>
</soapenv:Envelope>
```

2. To view the response upon execution, click the **Response** tab.



Web Service Execution (XML) - addUser

WSDL URL 'ER}}:{{WSPORT}}/itko-examples/services/UserControlService?wsdl

Service UserControlServiceSe...

Port UserControlService

Operation addUser

On Error Abort the Test

Endpoint {{ENDPOINT}}

http://localhost:8080/itko-examples/services/UserControlService

```
<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmls
    <soapenv:Body>
        <ns1:addUserResponse soapenv:encodingStyle="
            <addUserReturn href="#id0"/>
        </ns1:addUserResponse>
        <multiRef id="id0" soapenc:root="0" soapenv:
    </soapenv:Body>
</soapenv:Envelope>
```

Review

In this tutorial, you did the following:

- Created a test case with the Web Service Execution (XML) test step.
- Executed the **addUser** operation.
- Viewed the request and response for this operation.

Tutorial 9 - Examining and Testing a Database

In this tutorial, you will examine and test a database table that is part of the web application in Tutorial 5.

You use the SQL Database Execution (JDBC) step to interact with a database within a test case and test the response with an assertion. You examine the Users table from a Derby database that is part of the application.

LISA Concepts Discussed

In this tutorial, you do the following:

- Use the SQL Database Execution (JDBC) step.
- Store application properties in a configuration.
- Add and modify an assertion.
- Add a filter.

Prerequisites

- You have completed Tutorial 5.
- LISA Workstation is open.
- You have access to the demo server (either the local Demo Server, or the ITKO demo server).

Steps

Step 1 - Create a New Test Case

To create a new test case,

1. In the Project pane, right-click on the **Tests** folder and select **Create New Test Case**.
2. Set the file name to **tutorial9**.
3. Click Save.

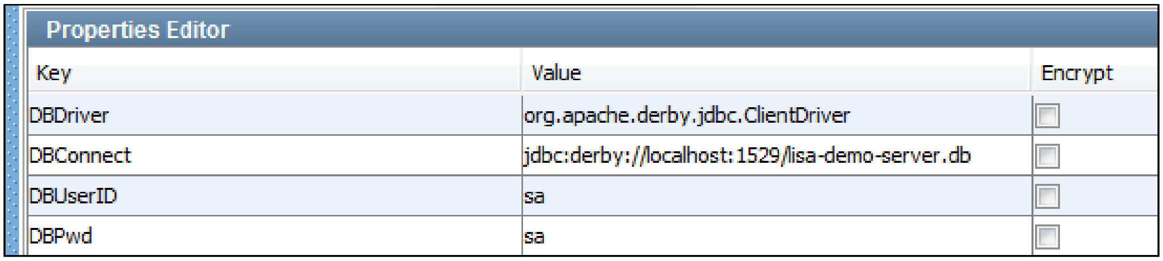
Step 2 - Add Database Properties to the Configuration

Store the properties that are needed to connect to the database in the configuration. This is a standard LISA practice that increases the portability of test cases.

To add database properties to the configuration,

1. If **project.config** is not the active configuration, then right-click **project.config** in the Project pane and select **Make Active**.
2. Open the **project.config** configuration.
3. Add the following properties.

Key	Value
DBDriver	org.apache.derby.jdbc.ClientDriver
DBConnect	jdbc:derby://localhost:1529/lisa-demo-server.db
DBUserID	sa
DBPwd	sa



Key	Value	Encrypt
DBDriver	org.apache.derby.jdbc.ClientDriver	□
DBConnect	jdbc:derby://localhost:1529/lisa-demo-server.db	□
DBUserID	sa	□
DBPwd	sa	□

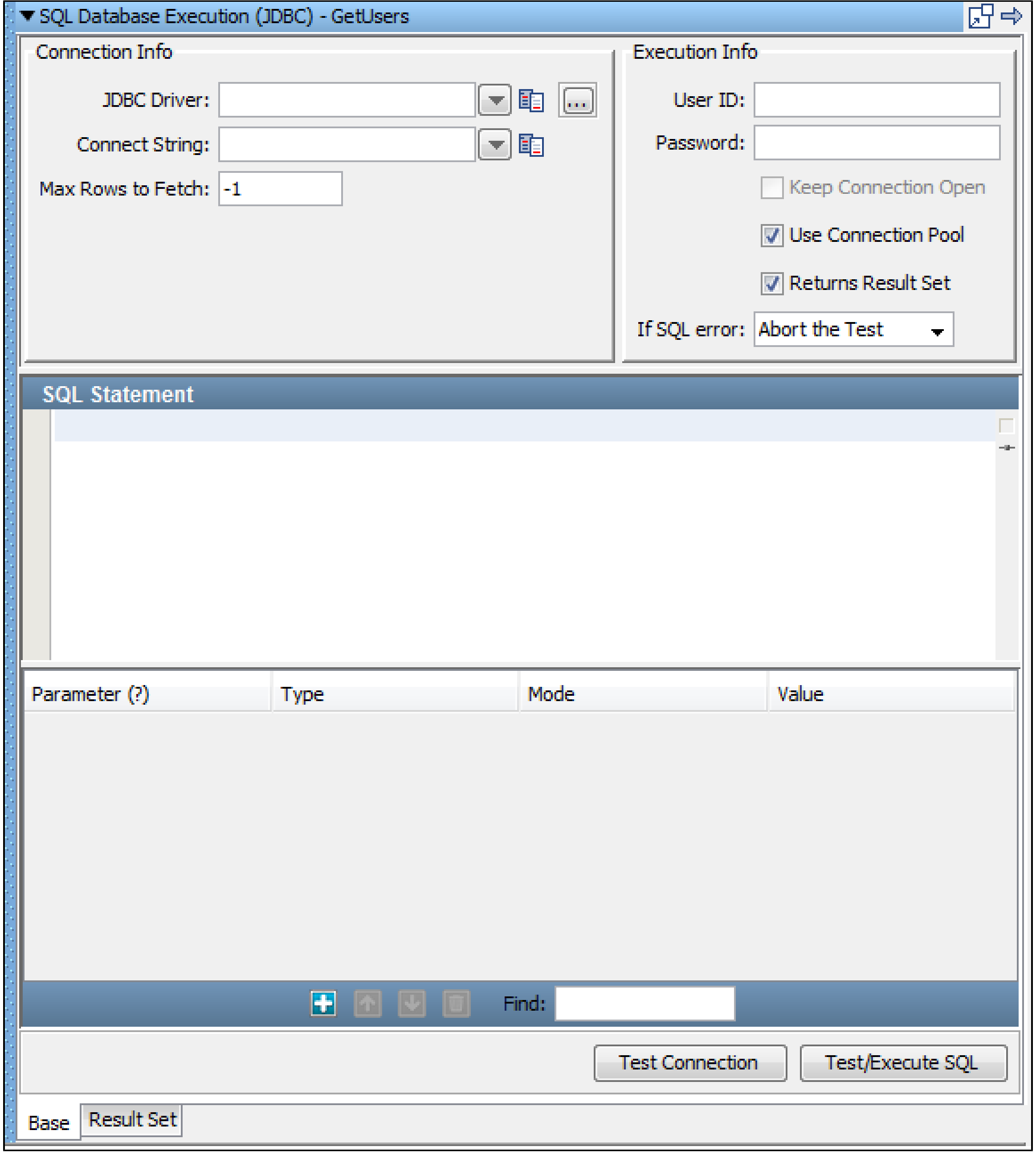
4. Click the Save icon.

Step 3 - Add a SQL Database Execution (JDBC) Test Step

The SQL Database Execution (JDBC) test step enables you to connect to a database using JDBC and make SQL queries on the database. To add a SQL Database Execution (JDBC) test step,

1. Click the **tutorial9** tab.
2. Click the Add Step ⊕ icon.
3. Select **Other Transactions** and select **SQL Database Execution (JDBC)**. **Step1** is added to the model editor.
4. Right-click **Step1** and select **Rename**. Change the name to **GetUsers**.

5. Double-click the **GetUsers** step to open the step editor.



SQL Database Execution (JDBC) - GetUsers

Max Rows to Fetch: -1

☐ Keep Connection Open

☑ Use Connection Pool

☑ Returns Result Set

If SQL error: Abort the Test

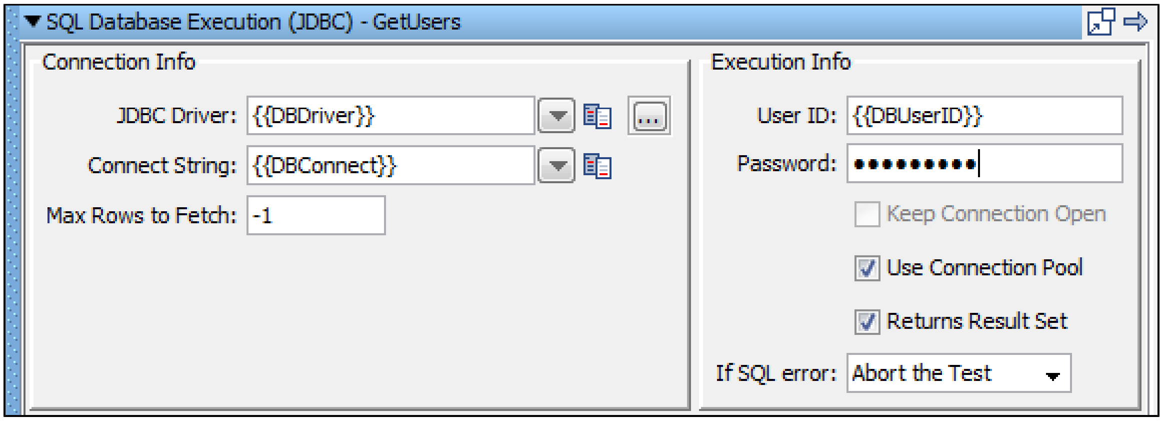
Step 4 - Connect to the Database

Use the properties that you added to the **project.config** configuration to provide connection information. To connect to the database,

1. Enter the following values in the **Connection Info** and **Execution Info**areas of the step editor. Notice that when you enter the password, the value is masked.

Field	Value
JDBC Driver	`{{DBDriver}}`
Connect String	`{{DBConnect}}`
User ID	`{{DBUserID}}`

Password	{{DBPwd}}



▼ SQL Database Execution (JDBC) - GetUsers

Connection Info

JDBC Driver: {{DBDriver}}

Connect String: {{DBConnect}}

Max Rows to Fetch: -1

Execution Info

User ID: {{DBUserID}}

Password: ●●●●●●●●●

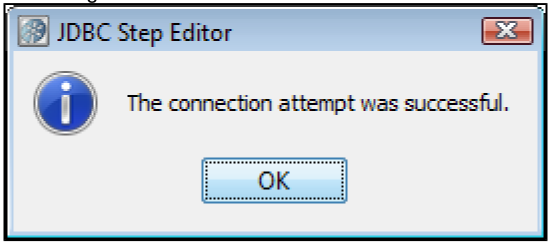
□ Keep Connection Open

☑ Use Connection Pool

☑ Returns Result Set

If SQL error: Abort the Test

2. Click the Test Connection button at the bottom of the step editor.
   A message indicates that the connection is valid.



JDBC Step Editor

The connection attempt was successful.

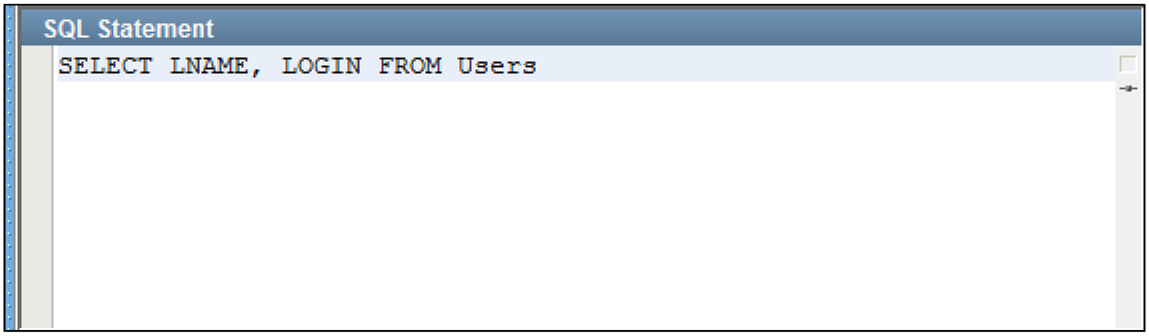
3. Click OK.

Step 5 - Execute a SQL Query

You now specify and run a SQL statement that retrieves data from the Users table. To execute a SQL query,

1. In the SQL Statement pane, enter the following statement:

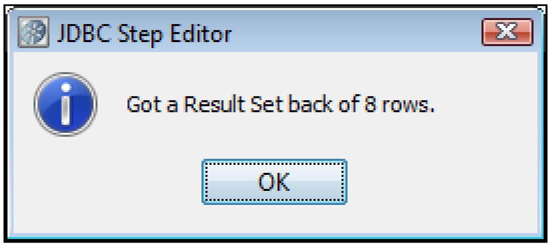
```
SELECT LNAME, LOGIN FROM Users
```



SQL Statement

```
SELECT LNAME, LOGIN FROM Users
```

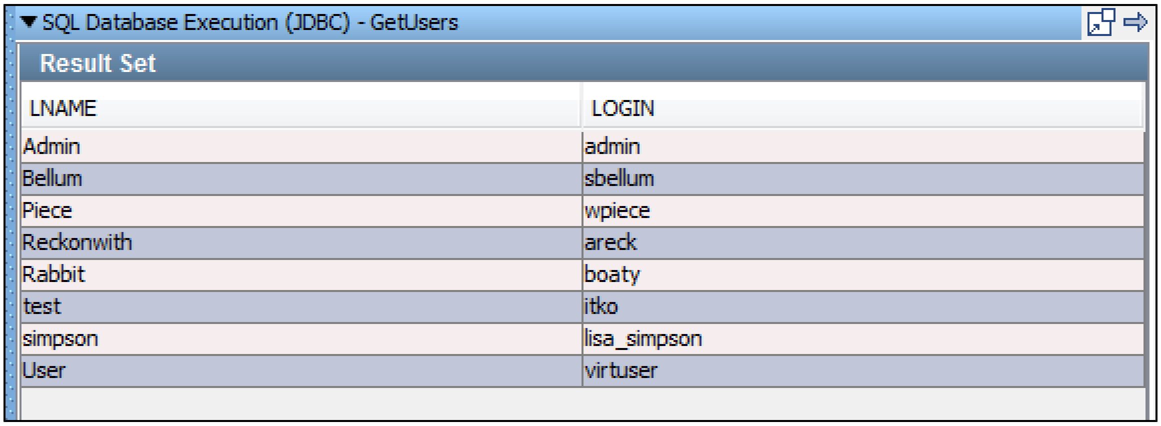
2. Click the **Test/Execute SQL** button at the bottom of the step editor. A message confirms a valid query and displays the number of rows returned.



JDBC Step Editor

Got a Result Set back of 8 rows.

3. Click OK.
   The **Result Set** tab is displayed.



SQL Database Execution (JDBC) - GetUsers

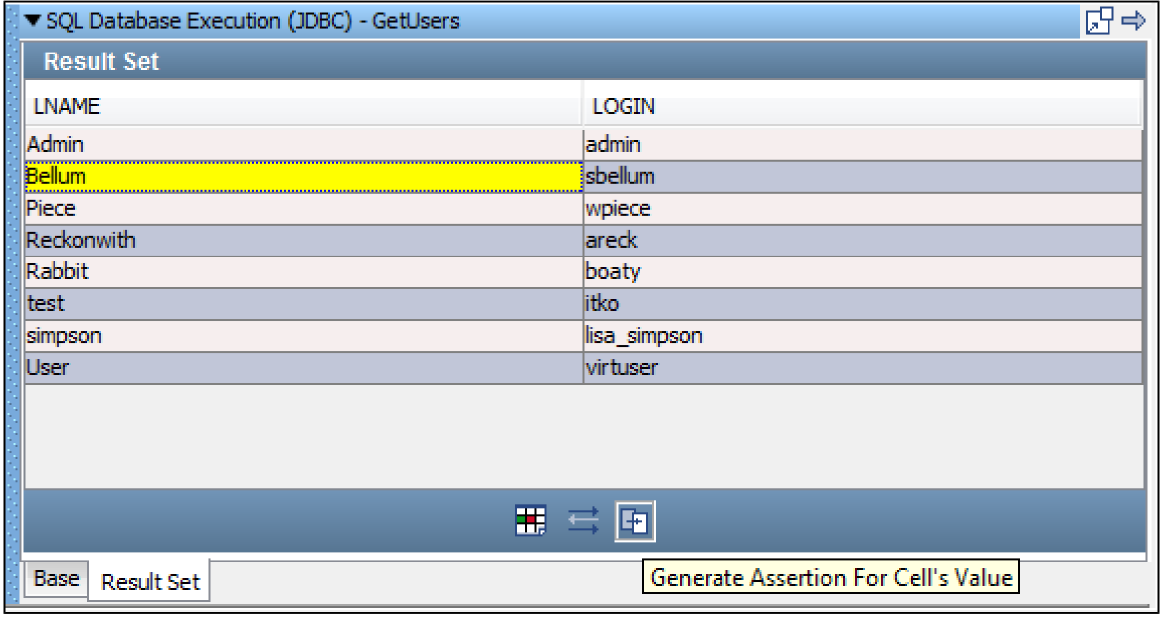
Result Set

LNAME	LOGIN
Admin	admin
Bellum	sbellum
Piece	wpiece
Reckonwith	areck
Rabbit	boaty
test	itko
simpson	lisa_simpson
User	virtuser

Step 6 - Add an Assertion

Add an assertion to test for the presence of a specific last name in the result set. To add an assertion,

1. In the **Result Set** tab, select a cell in the **LNAME** column.
2. Click the Generate Assertion for Cell's Value icon. The **Generate JDBC Result Set Value Assertion** dialog opens.

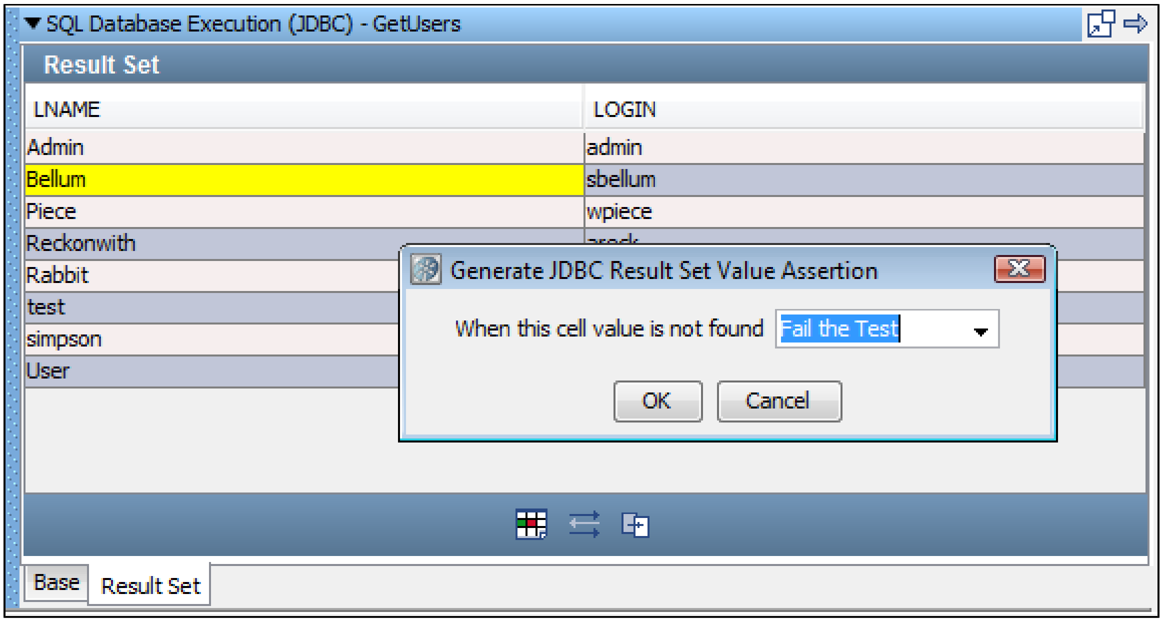


SQL Database Execution (JDBC) - GetUsers

Result Set

LNAME	LOGIN
Admin	admin
Bellum	sbellum
Piece	wpiece
Reckonwith	areck
Rabbit	boaty
test	itko
simpson	lisa_simpson
User	virtuser

3. From the drop-down list, select the **Fail the Test** option. If the last name that you selected is not found, then the test will fail.


SQL Database Execution (JDBC) - GetUsers
Result Set
Generate JDBC Result Set Value Assertion
OK
Cancel
Base
Result Set

LNAME	LOGIN
Admin	admin
Bellum	sbellum
Piece	wpiece
Reckonwith	arock
Rabbit	
test	
simpson	
User	

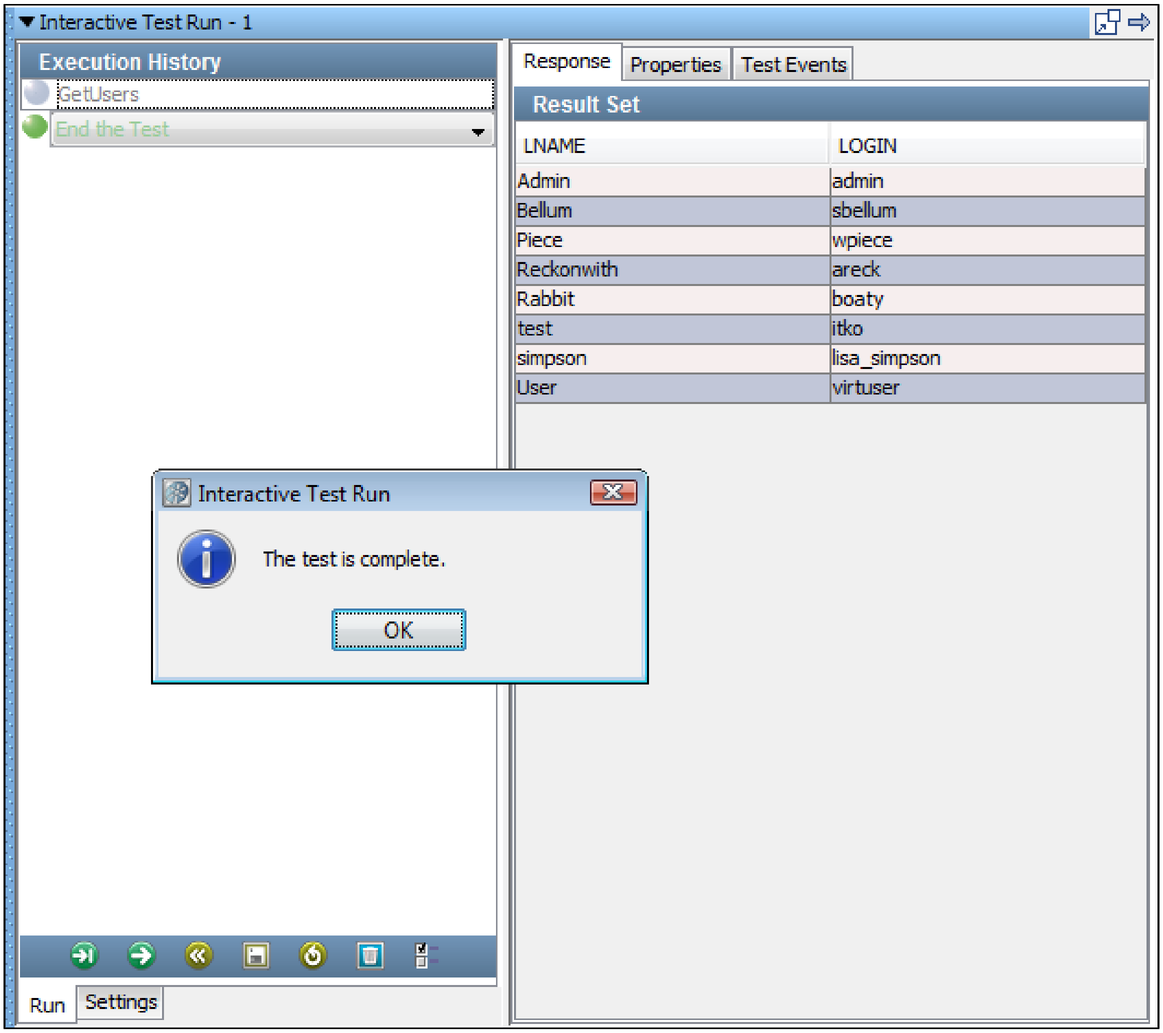
When this cell value is not found: Fail the Test

4. Click OK.
5. Click the Save icon.

Step 7 - Run the Test Case

To run the test case,

1. From the toolbar, click the Start ITR icon.
2. Click Execute Next Step. The test runs successfully. The result set is shown in the **Response** tab.



Result Set

LNAME	LOGIN
Admin	admin
Bellum	sbellum
Piece	wpiece
Reckonwith	areck
Rabbit	boaty
test	itko
simpson	lisa_simpson
User	virtuser

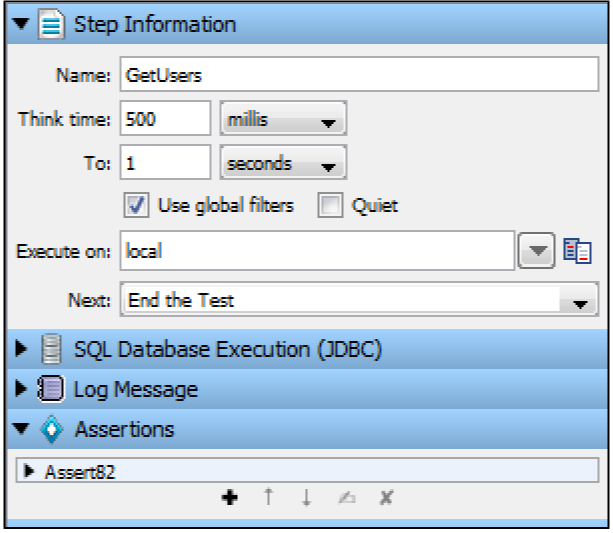
The test is complete.

3. Retract the ITR tray.

Step 8 - Change the Assertion

You now modify the assertion to cause the test to fail. To change the assertion,

1. In the model editor, click the JDBC test step.
2. Open the **Assertions** tab in the Element Tree.



Name: GetUsers

Think time: 500 millis

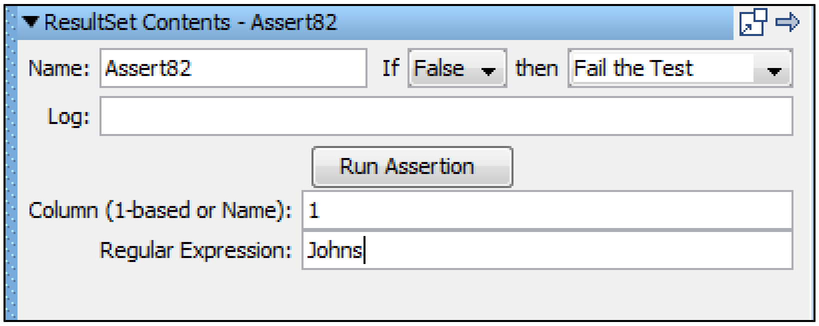
To: 1 seconds

☑ Use global filters ☐ Quiet

Execute on: local

Next: End the Test

3. Double-click the assertion that you created earlier. The assertion editor is opened. The lower portion indicates that the assertion checks the first column of the result set for the specified value.
4. Change the value of the Regular Expression field to **Johns**.


ResultSet Contents - Assert82

Name: Assert82 If False then Fail the Test

Log:

Column (1-based or Name): 1

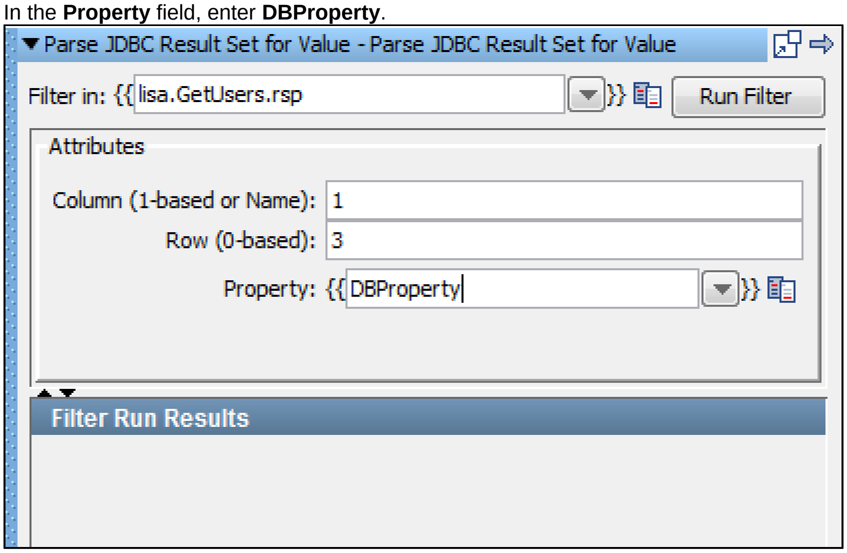
Regular Expression: Johns

5. Start a new ITR and run the test case again. The test fails.
6. Retract the ITR tray.

Step 9 - Add a Filter

Add a database filter to capture the value in the first column and fourth row of the result set. The value will be stored in a property. To add a filter,

1. In the model editor, select the JDBC test step.
2. Open the **Filters** tab in the Element Tree.
3. Click the Add ✚ icon.
4. From the **Database Filters** submenu, select **Extract Value from JDBC Result Set**. The filter editor opens.
5. In the **Column** field, enter **1**. Alternatively, you can enter the actual column name, which is **LNAME**.
6. In the **Row** field, enter **3**. This field is zero-based. Therefore, the value **3** refers to the fourth row.
7. In the **Property** field, enter **DBProperty**.


Parse JDBC Result Set for Value - Parse JDBC Result Set for Value

Filter in: {{lisa.GetUsers.rsp}}

Attributes

Column (1-based or Name): 1

Row (0-based): 3

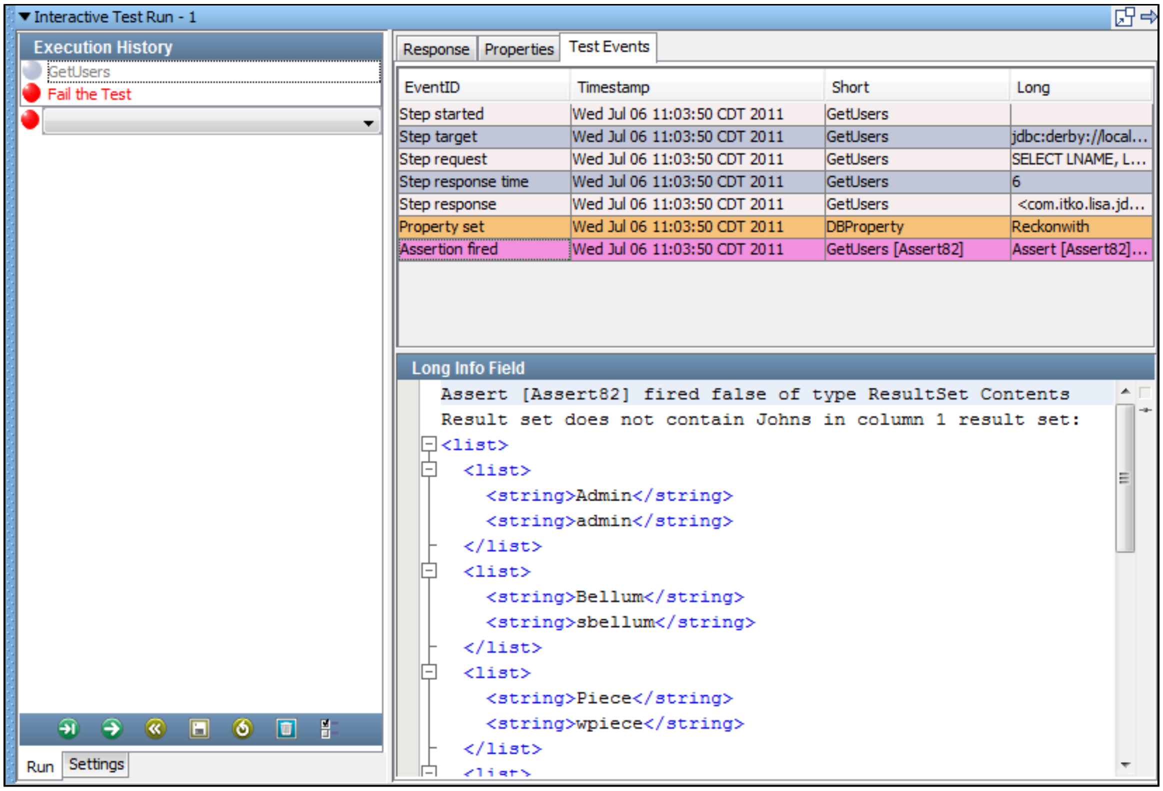
Property: {{DBProperty}}

8. Click the Save icon.

Step 10 - Test the Filter and Assertion

To test the filter and assertion,

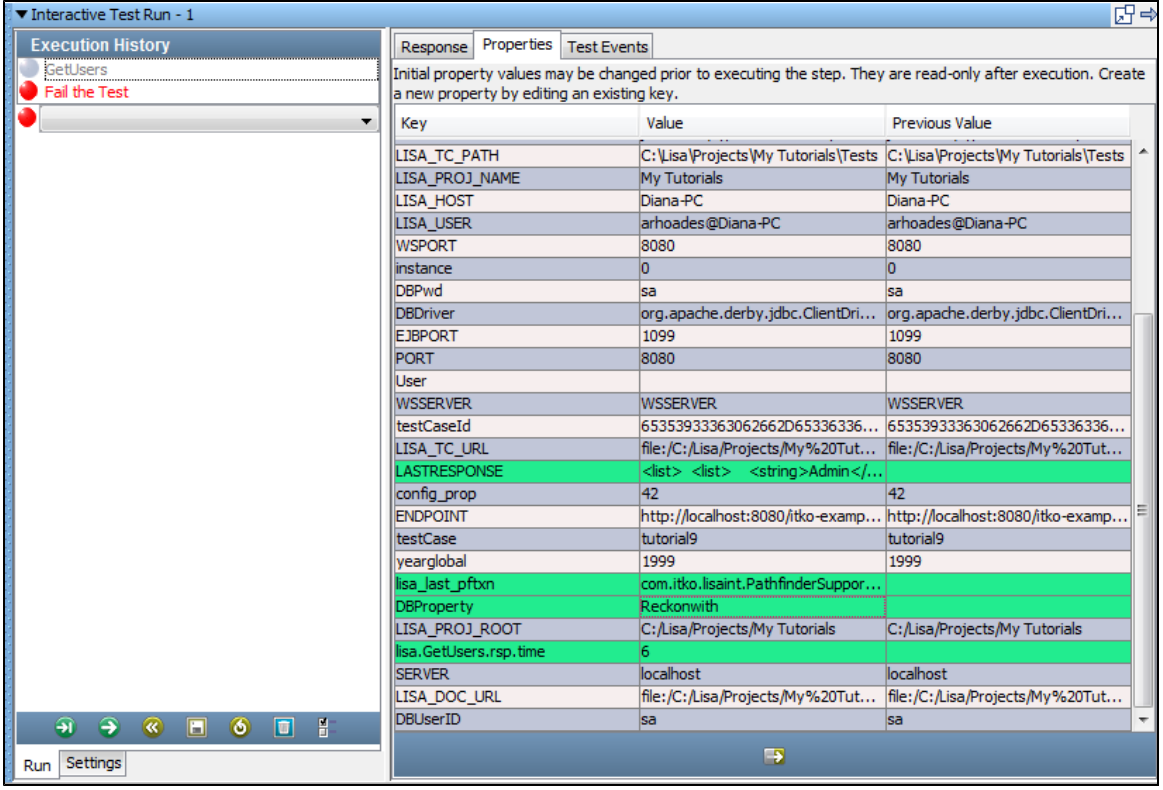
1. Start a new ITR and run the test case again.
   The test fails because **Johns** was not found in the result set.
2. Click the **Test Events** tab.
3. Click the **Property set** event. Notice that **DBProperty** was set to the value specified by the filter.
4. Click the **Assertion fired** event. The Long Info Field area indicates that the assertion fired because the first column of the result set did not contain the value **Johns**.



EventID	Timestamp	Short	Long
Step started	Wed Jul 06 11:03:50 CDT 2011	GetUsers	
Step target	Wed Jul 06 11:03:50 CDT 2011	GetUsers	jdbc:derby://local...
Step request	Wed Jul 06 11:03:50 CDT 2011	GetUsers	SELECT LNAME, L...
Step response time	Wed Jul 06 11:03:50 CDT 2011	GetUsers	6
Step response	Wed Jul 06 11:03:50 CDT 2011	GetUsers	<com.itko.lisa.jd...
Property set	Wed Jul 06 11:03:50 CDT 2011	DBProperty	Reckonwith
Assertion fired	Wed Jul 06 11:03:50 CDT 2011	GetUsers [Assert82]	Assert [Assert82]...

```
Assert [Assert82] fired false of type ResultSet Contents
Result set does not contain Johns in column 1 result set:
<list>
  <list>
    <string>Admin</string>
    <string>admin</string>
  </list>
  <list>
    <string>Bellum</string>
    <string>sbellum</string>
  </list>
  <list>
    <string>Piece</string>
    <string>wpiece</string>
  </list>
  <list>
```

5. Click the **Properties** tab.
6. Locate and review the **DBProperty** row.



Initial property values may be changed prior to executing the step. They are read-only after execution. Create a new property by editing an existing key.

Key	Value	Previous Value
LISA_TC_PATH	C:\Lisa\Projects\My Tutorials\Tests	C:\Lisa\Projects\My Tutorials\Tests
LISA_PROJ_NAME	My Tutorials	My Tutorials
LISA_HOST	Diana-PC	Diana-PC
LISA_USER	arhoades@Diana-PC	arhoades@Diana-PC
WSPORT	8080	8080
instance	0	0
DBPwd	sa	sa
DBDriver	org.apache.derby.jdbc.ClientDri...	org.apache.derby.jdbc.ClientDri...
EJBPORT	1099	1099
PORT	8080	8080
User		
WSSERVER	WSSERVER	WSSERVER
testCaseId	65353933363062662D65336336...	65353933363062662D65336336...
LISA_TC_URL	file:/C:/Lisa/Projects/My%20Tut...	file:/C:/Lisa/Projects/My%20Tut...
LASTRESPONSE	<list> <list> <string>Admin</...	
config_prop	42	42
ENDPOINT	http://localhost:8080/itko-examp...	http://localhost:8080/itko-examp...
testCase	tutorial9	tutorial9
yearglobal	1999	1999
lisa_last_pftxn	com.itko.lisaint.PathfinderSuppor...	
DBProperty	Reckonwith	
LISA_PROJ_ROOT	C:/Lisa/Projects/My Tutorials	C:/Lisa/Projects/My Tutorials
lisa.GetUsers.rsp.time	6	
SERVER	localhost	localhost
LISA_DOC_URL	file:/C:/Lisa/Projects/My%20Tut...	file:/C:/Lisa/Projects/My%20Tut...
DBUserID	sa	sa

Review

In this tutorial, you created a test case to query a database. You used the Users table from the Apache Derby database that accompanies the applications on the Demo Server. You learned how to:

- Connect to the database.
- Execute a SQL query against the database.

- Add assertions and filters.

Tutorial 10 - Staging a Quick Test

In this tutorial, you will use the quick staging option (quick test) to learn how to stage tests and read subsequent reports. You will run the quick test on the **multi-tier-combo** example that accompanies LISA. This is the simplest way to stage a test.

LISA Concepts Discussed

In this tutorial, you do the following:

- Use the multi-tier-combo test case.
- Use the quick test feature.
- Select and format reports.

Prerequisites

- You have completed Tutorials 5 through 9.
- LISA Workstation is open.
- You have access to the demo server (either the local Demo Server, or the ITKO demo server).

Steps

Step 1 - Open the Test Case

We will open a test case from the examples project.

To open the test case,

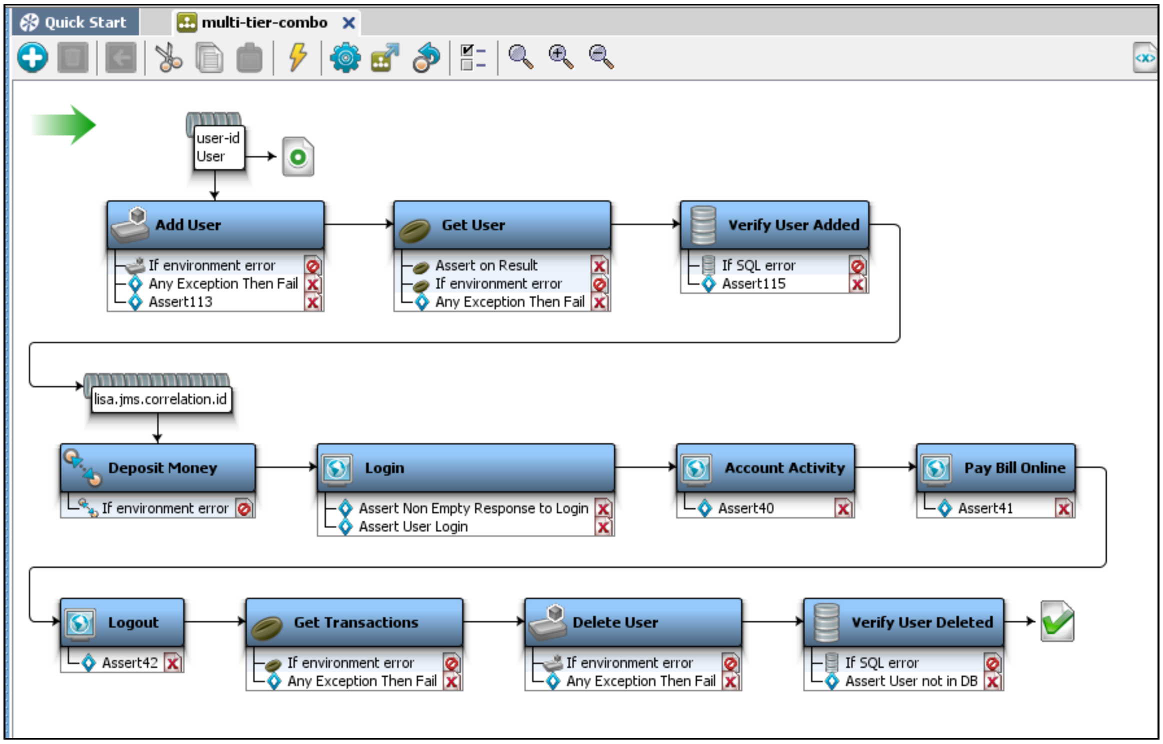
1. From the main menu, select **File > Open > Test Case > File System**.
2. Navigate to the **LISA_HOME/examples/Tests** folder.
3. Select **multi-tier-combo** and click Open.

The multi-tier-combo test case opens in the model editor.

Step 2 - Review the Test Case

Take a few minutes to review the various types of test steps in this test case. For example:

- **Add User** is a Web Service Execution (Legacy) step.
- **Get User** is an Enterprise JavaBean Execution step.
- **Verify User Added** is a SQL Database Execution (JDBC) step.
- **Deposit Money** is a JMS Messaging (JNDI) step.



You used many of these steps in tutorials 6 through 9. In this tutorial, you use all the test steps to build a more realistic test case involving several layers of the application.

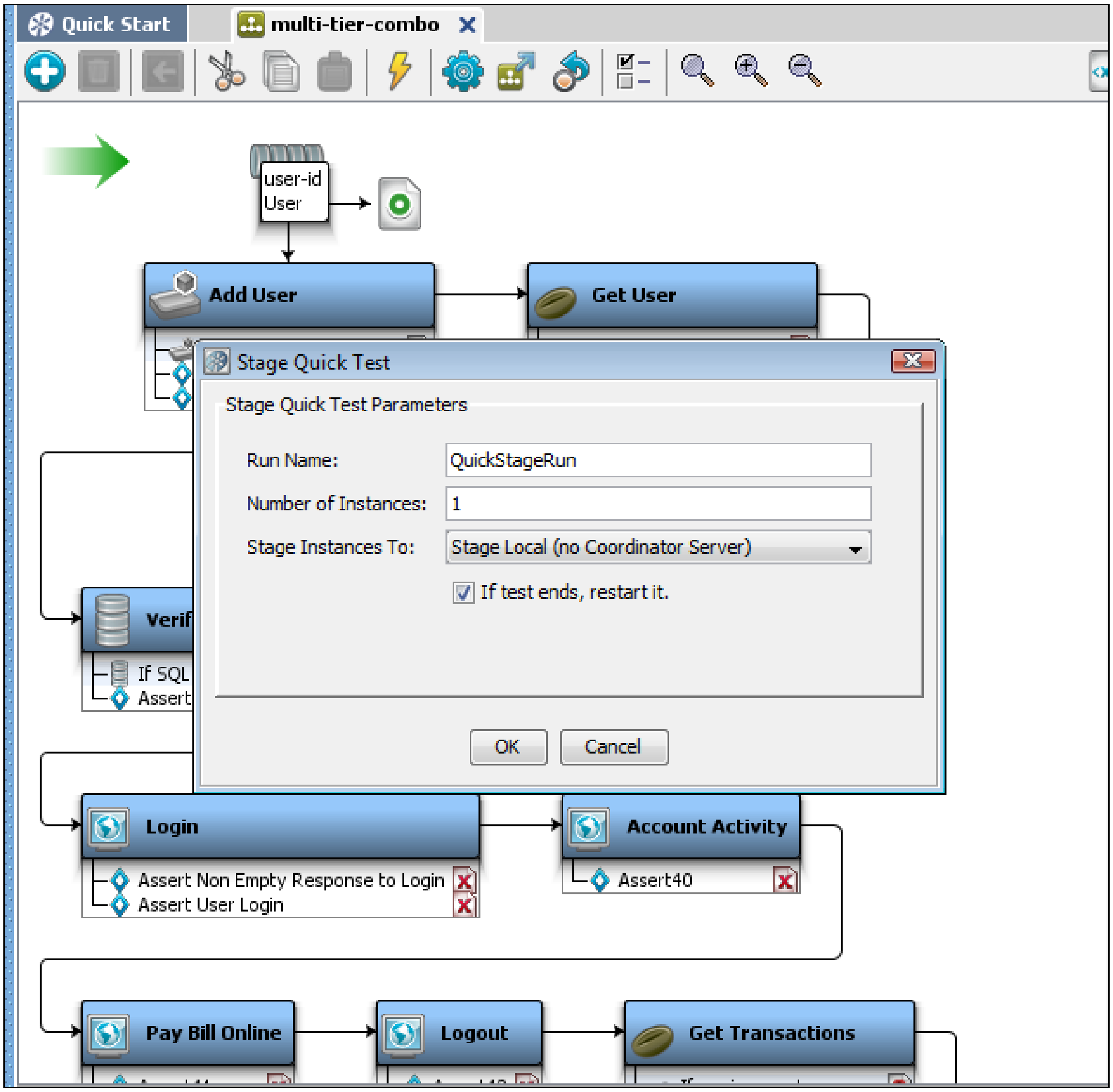
Part A - Running the Quick Test
Part B - Viewing the Generated Reports

Part A - Running a Quick Test

To run a quick test,

1. From the menu bar, click the Stage a quick test icon on the test case toolbar.

> ⚠ To stage a quick test, the example test case can be open in the model editor, or you can right-click on the test in the Project panel and enter the parameters to stage a quick test from there.



Stage Quick Test

Stage Quick Test Parameters

Run Name: QuickStageRun

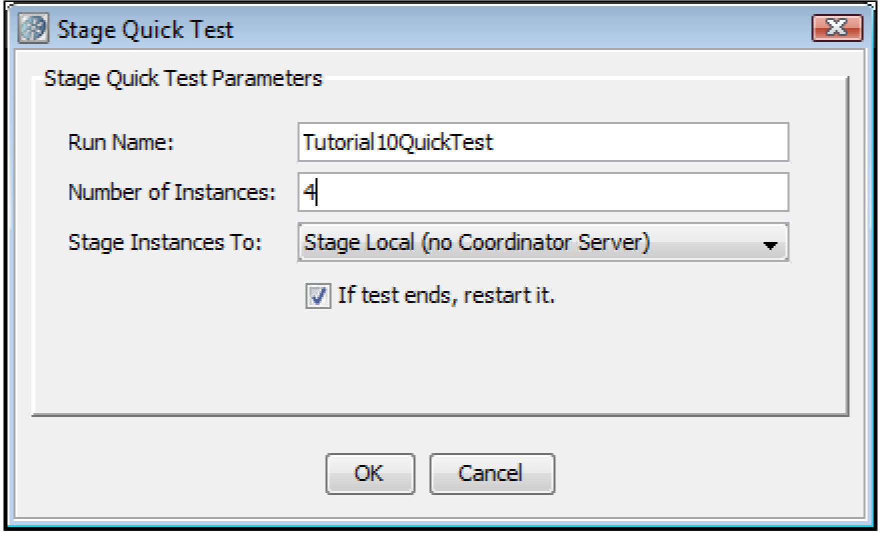
Number of Instances: 1

Stage Instances To: Stage Local (no Coordinator Server)

☑ If test ends, restart it.

OK Cancel

2. In the **Stage Quick Test**dialog, complete the required information as follows:
    - **Run Name**: Enter a unique name (**Tutorial10QuickTest**).
    - **Number of Instances**: Enter a number of users that you want to run the test concurrently (**4**).
    - **Stage Instances To**: Select the name of the Coordinator Server or stage it locally.
    - **If test ends, restart it**: Check this option to restart the test.



Stage Quick Test

Stage Quick Test Parameters

Run Name: Tutorial10QuickTest

Number of Instances: 4

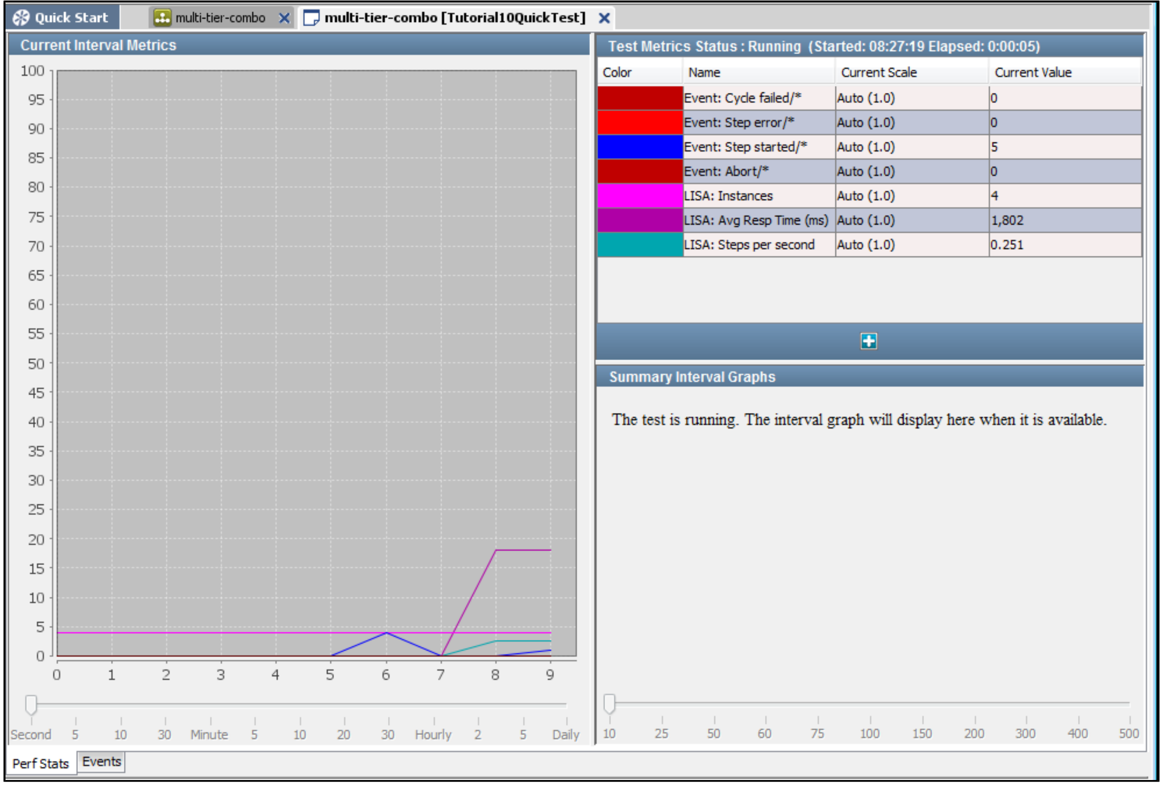
Stage Instances To: Stage Local (no Coordinator Server)

☑ If test ends, restart it.

OK Cancel

3. Click OK.

4. The Test Run window opens, but the test has not started yet.
5. From the main toolbar, click the Start button to start the test running.
   The test begins, and the graph immediately plots results.



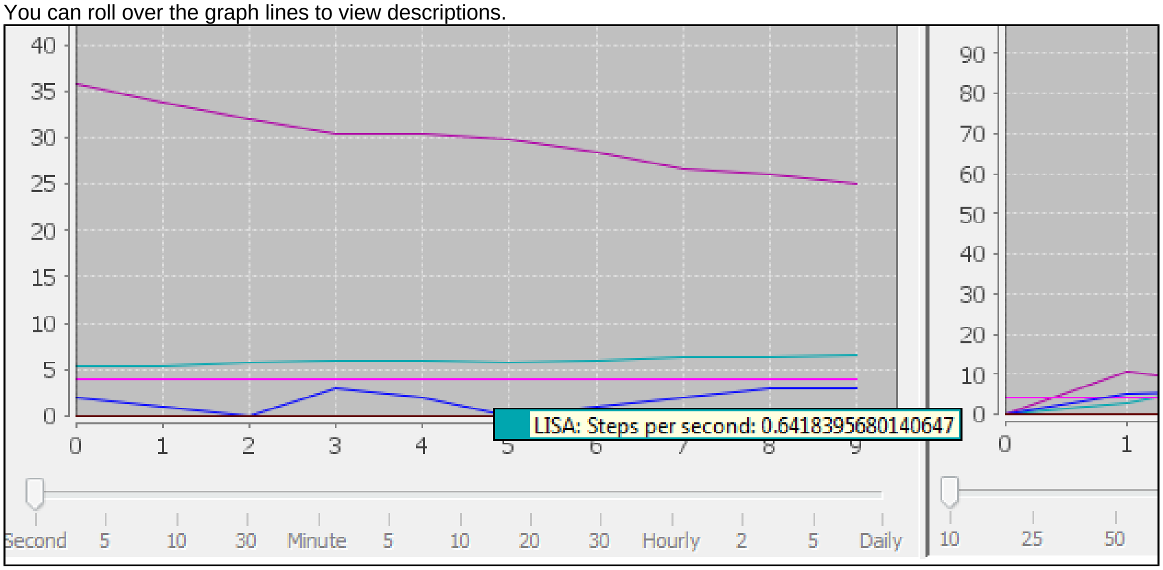
Test Metrics Status : Running (Started: 08:27:19 Elapsed: 0:00:05)

Color	Name	Current Scale	Current Value
🟥	Event: Cycle failed/*	Auto (1.0)	0
🟥	Event: Step error/*	Auto (1.0)	0
🟦	Event: Step started/*	Auto (1.0)	5
🟥	Event: Abort/*	Auto (1.0)	0
🟪	LISA: Instances	Auto (1.0)	4
🟪	LISA: Avg Resp Time (ms)	Auto (1.0)	1,802
🟦	LISA: Steps per second	Auto (1.0)	0.251

Summary Interval Graphs

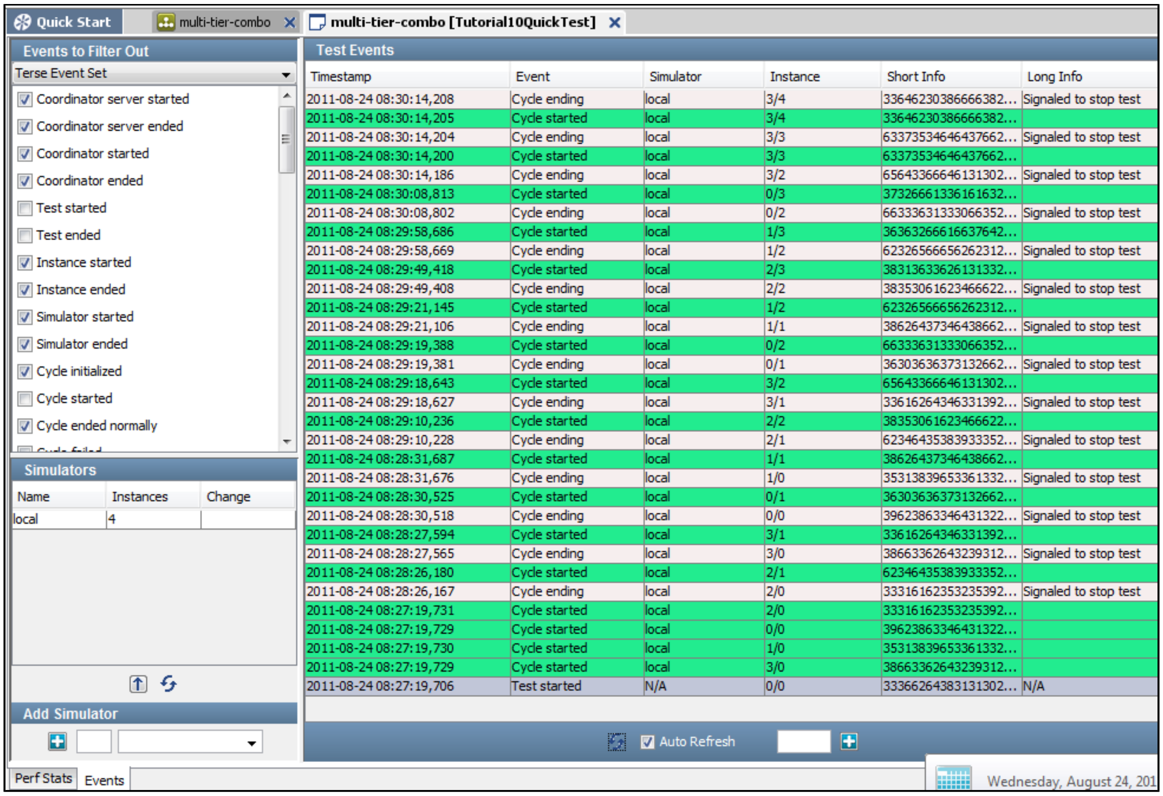
The test is running. The interval graph will display here when it is available.

6. You can roll over the graph lines to view descriptions.



LISA: Steps per second: 0.6418395680140647

   Select the **Events** tab to choose which events to display.



☑ Coordinator server started
☑ Coordinator server ended
☑ Coordinator started
☑ Coordinator ended
☐ Test started
☐ Test ended
☑ Instance started
☑ Instance ended
☑ Simulator started
☑ Simulator ended
☑ Cycle initialized
☐ Cycle started
☑ Cycle ended normally

Simulators

Name	Instances	Change
local	4	

Test Events

Timestamp	Event	Simulator	Instance	Short Info	Long Info
2011-08-24 08:30:14,208	Cycle ending	local	3/4	3364623038666382...	Signaled to stop test
2011-08-24 08:30:14,205	Cycle started	local	3/4	3364623038666382...	
2011-08-24 08:30:14,204	Cycle ending	local	3/3	63373534646437662...	Signaled to stop test
2011-08-24 08:30:14,200	Cycle started	local	3/3	63373534646437662...	
2011-08-24 08:30:14,186	Cycle ending	local	3/2	65643366646131302...	Signaled to stop test
2011-08-24 08:30:08,813	Cycle started	local	0/3	37326661336161632...	
2011-08-24 08:30:08,802	Cycle ending	local	0/2	66333631333066352...	Signaled to stop test
2011-08-24 08:29:58,686	Cycle started	local	1/3	36363266616637642...	
2011-08-24 08:29:58,669	Cycle ending	local	1/2	62326566656262312...	Signaled to stop test
2011-08-24 08:29:49,418	Cycle started	local	2/3	38313633626131332...	
2011-08-24 08:29:49,408	Cycle ending	local	2/2	38353061623466622...	Signaled to stop test
2011-08-24 08:29:21,145	Cycle started	local	1/2	62326566656262312...	
2011-08-24 08:29:21,106	Cycle ending	local	1/1	38626437346438662...	Signaled to stop test
2011-08-24 08:29:19,388	Cycle started	local	0/2	66333631333066352...	
2011-08-24 08:29:19,381	Cycle ending	local	0/1	36303636373132662...	Signaled to stop test
2011-08-24 08:29:18,643	Cycle started	local	3/2	65643366646131302...	
2011-08-24 08:29:18,627	Cycle ending	local	3/1	33616264346331392...	Signaled to stop test
2011-08-24 08:29:10,236	Cycle started	local	2/2	38353061623466622...	
2011-08-24 08:29:10,228	Cycle ending	local	2/1	62346435383933352...	Signaled to stop test
2011-08-24 08:28:31,687	Cycle started	local	1/1	38626437346438662...	
2011-08-24 08:28:31,676	Cycle ending	local	1/0	35313839653361332...	Signaled to stop test
2011-08-24 08:28:30,525	Cycle started	local	0/1	36303636373132662...	
2011-08-24 08:28:30,518	Cycle ending	local	0/0	39623863346431322...	Signaled to stop test
2011-08-24 08:28:27,594	Cycle started	local	3/1	33616264346331392...	
2011-08-24 08:28:27,565	Cycle ending	local	3/0	38663362643239312...	Signaled to stop test
2011-08-24 08:28:26,180	Cycle started	local	2/1	62346435383933352...	
2011-08-24 08:28:26,167	Cycle ending	local	2/0	33316162353235392...	Signaled to stop test
2011-08-24 08:27:19,731	Cycle started	local	2/0	33316162353235392...	
2011-08-24 08:27:19,729	Cycle started	local	0/0	39623863346431322...	
2011-08-24 08:27:19,730	Cycle started	local	1/0	35313839653361332...	
2011-08-24 08:27:19,729	Cycle started	local	3/0	38663362643239312...	
2011-08-24 08:27:19,706	Test started	N/A	0/0	33366264383131302...	N/A

☑ Auto Refresh

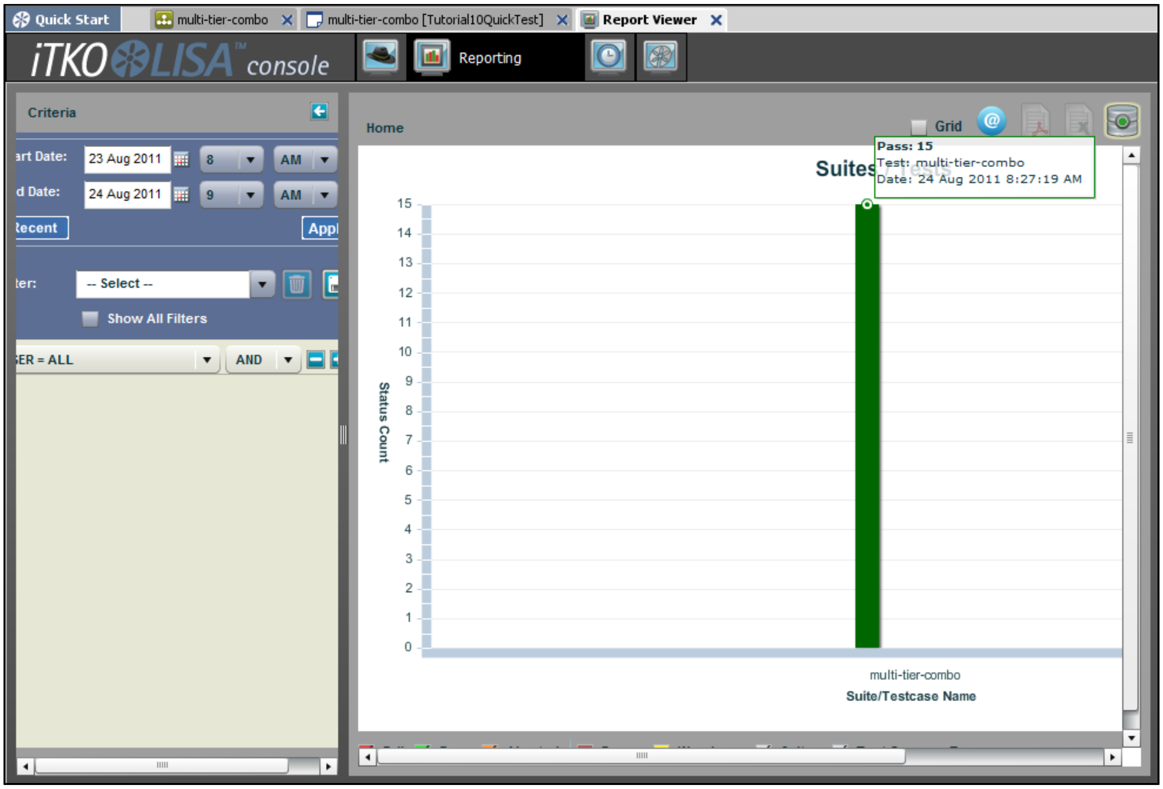
Wednesday, August 24, 201

Part B - Viewing the Generated Reports

LISA provides a report viewer to view reports.

To view the generated reports,

1. From the main menu, click **View > Reporting Console** or click the Reports Reports icon on the toolbar. The Report Viewer opens.



art Date: 23 Aug 2011 8 AM

d Date: 24 Aug 2011 9 AM

ter: -- Select --

☐ Show All Filters

ER = ALL AND

Pass: 15
Test: multi-tier-combo
Date: 24 Aug 2011 8:27:19 AM

2. Set the date criteria to open the graphs plotted within those dates.
   In the preceding image, the graph shows that all the tests in this test case passed.
   You can right-click the graph for different menus. For more information about reports, see the "Reports" section of the *User Guide*.

Review

In this tutorial, you tested the multi-tier-combo example using a quick test. You learned how to:

- Look at a test case containing several types of test steps.
- Configure and run a test in the quick test feature.
- Examine a report generated from the test run.

Building Test Cases

To build test cases, you must know about setting properties, using configurations and applying them to your project, applying filters, adding assertions, adding data sets, and adding companions. This section also introduces the Complex Object Editor.

In this section, the following topics are covered:

Anatomy of a Test Case

A test case is a specification of how to test a business component in the system under test. A test case is stored as an XML document, and contains all the information needed to test the given component or system.

A test case in LISA is a workflow with the test steps being connected by paths that represent successful and non-successful step conclusions. Assertions may accompany the step and different paths are provided based on the firing of any of the assertions.

⚠ Save your test cases regularly.

This section includes the following topics:

Test Case Quick Start

To start working with test cases

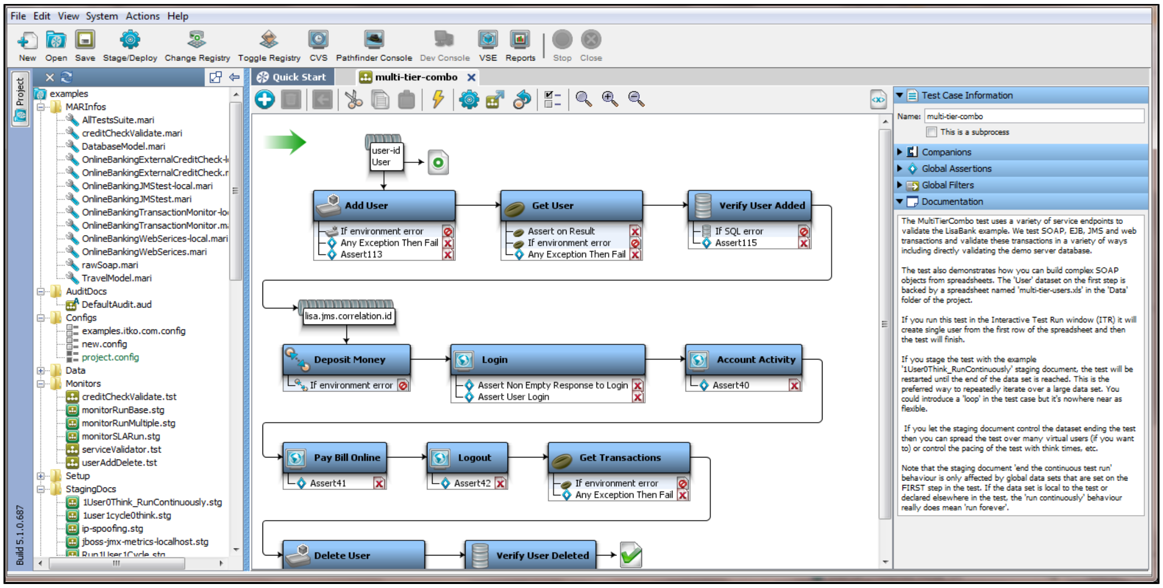
1. Start the registry. See LISA Registry.
2. Create or open a project in LISA Workstation. See Project Panel.
3. Create a test case within the project. See Creating Test Cases.

You can open an existing test case either from a valid LISA project or from outside a project by selecting File > Open > Test Case from the main menu.

The **multi-tier-combo.tst** test case is shown in the following images. For more information about the multi-tier-combo test case, see Multi-tier-combo Test Case.

LISA Workstation is divided into three main areas (from left to right):

- Project Panel
- Model Editor
- Element Panel



Name: multi-tier-combo

□ This is a subprocess

The MultiTierCombo test uses a variety of service endpoints to validate the LisaBank example. We test SOAP, EJB, JMS and web transactions and validate these transactions in a variety of ways including directly validating the demo server database.

The test also demonstrates how you can build complex SOAP objects from spreadsheets. The 'User' dataset on the first step is backed by a spreadsheet named 'multi-tier-users.xls' in the 'Data' folder of the project.

If you run this test in the Interactive Test Run window (ITR) it will create single user from the first row of the spreadsheet and then the test will finish.

If you stage the test with the example '1User0Think_RunContinuously' staging document, the test will be restarted until the end of the data set is reached. This is the preferred way to repeatedly iterate over a large data set. You could introduce a 'loop' in the test case but it's nowhere near as flexible.

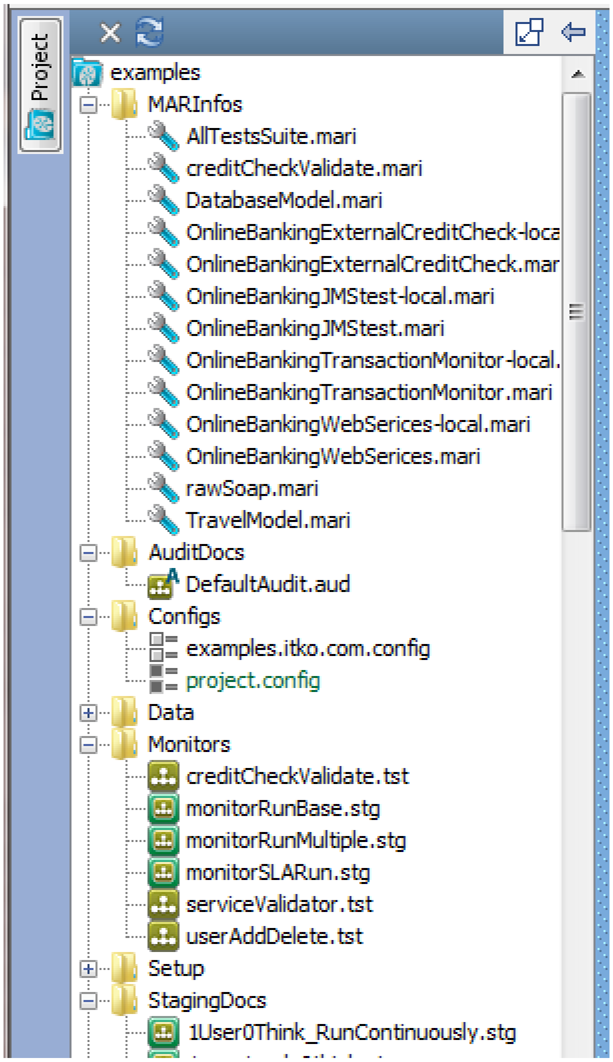
If you let the staging document control the dataset ending the test then you can spread the test over many virtual users (if you want to) or control the pacing of the test with think times, etc.

Note that the staging document 'end the continuous test run' behaviour is only affected by global data sets that are set on the FIRST step in the test. If the data set is local to the test or declared elsewhere in the test, the 'run continuously' behaviour really does mean 'run forever'.

Project Panel

The Project panel, located in the left portion of the window, is dockable. You can open or close the Project panel by using the Project button on the left.

When LISA Workstation first opens, the last project you had open will open by default.



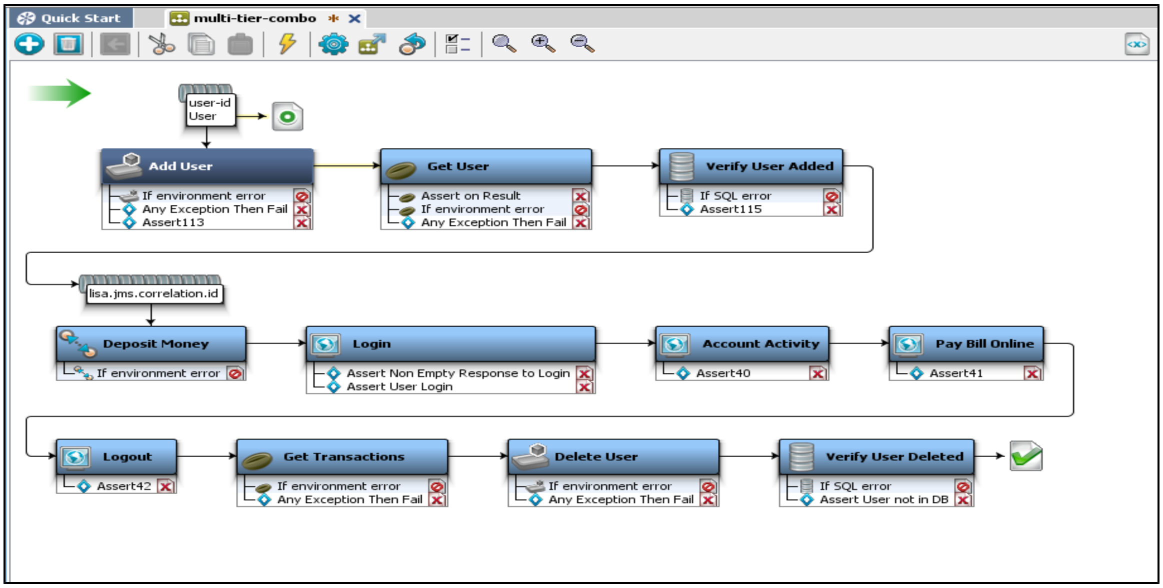
```
examples
  MARInfos
    AllTestsSuite.mari
    creditCheckValidate.mari
    DatabaseModel.mari
    OnlineBankingExternalCreditCheck-loca
    OnlineBankingExternalCreditCheck.mar
    OnlineBankingJMStest-local.mari
    OnlineBankingJMStest.mari
    OnlineBankingTransactionMonitor-local.
    OnlineBankingTransactionMonitor.mari
    OnlineBankingWebSerices-local.mari
    OnlineBankingWebSerices.mari
    rawSoap.mari
    TravelModel.mari
  AuditDocs
    DefaultAudit.aud
  Configs
    examples.itko.com.config
    project.config
  Data
  Monitors
    creditCheckValidate.tst
    monitorRunBase.stg
    monitorRunMultiple.stg
    monitorSLARun.stg
    serviceValidator.tst
    userAddDelete.tst
  Setup
  StagingDocs
    1User0Think_RunContinuously.stg
```

For more information, see Project Panel.

Model Editor

The Model Editor is where you create and view the test case workflow. The test case workflow consists of all test steps, filters, and assertions applied to a particular test case.

The Model Editor is the place to create and view test cases. This is the middle portion of the window and is displayed by a tab that is the name of the test case.



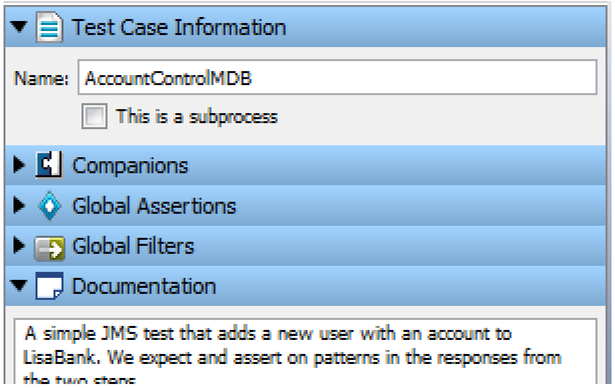
The green arrow ➡ marks the start of a test case.

The workflow in the Model Editor provides a graphical view of a test case. This is very helpful as it gives a quick visual check on the test case workflow.

In the Model Editor, each step in the test case is represented by an icon in the workflow. The icons change according to the type of test step. For example, if you have a database-related step, a database icon and the associated filters and assertions are attached to the step.

Element Panel

The Element panel contains the elements required for a test case or a test step.



▼ Test Case Information

Name: AccountControlMDB

□ This is a subprocess

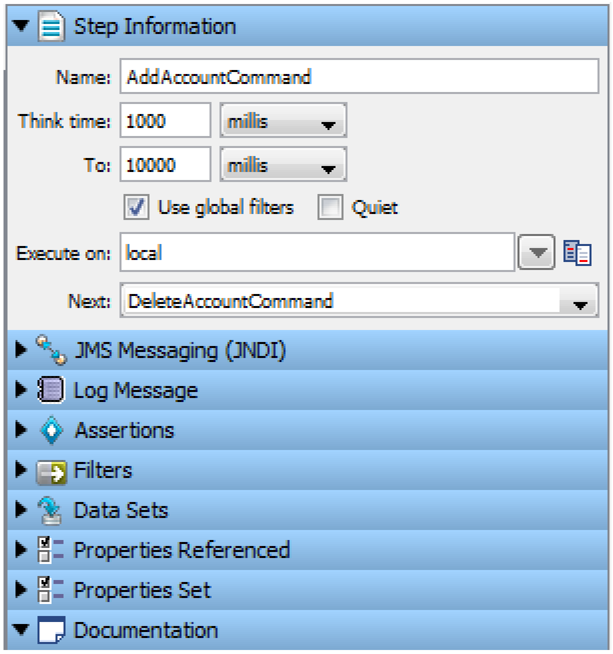
▶ Companions

▶ Global Assertions

▶ Global Filters

▼ Documentation

A simple JMS test that adds a new user with an account to LisaBank. We expect and assert on patterns in the responses from the two steps



Step Information

Name: AddAccountCommand

Think time: 1000 millis

To: 10000 millis

☑ Use global filters ☐ Quiet

Execute on: local

Next: DeleteAccountCommand

You can add or delete an element by clicking the required test case/test step element.

Some elements can be applied at the global level (to an entire test case). Some elements can be applied at a step level (only to a particular test step).

There are some sections in which you must enter information at the beginning of a test case.

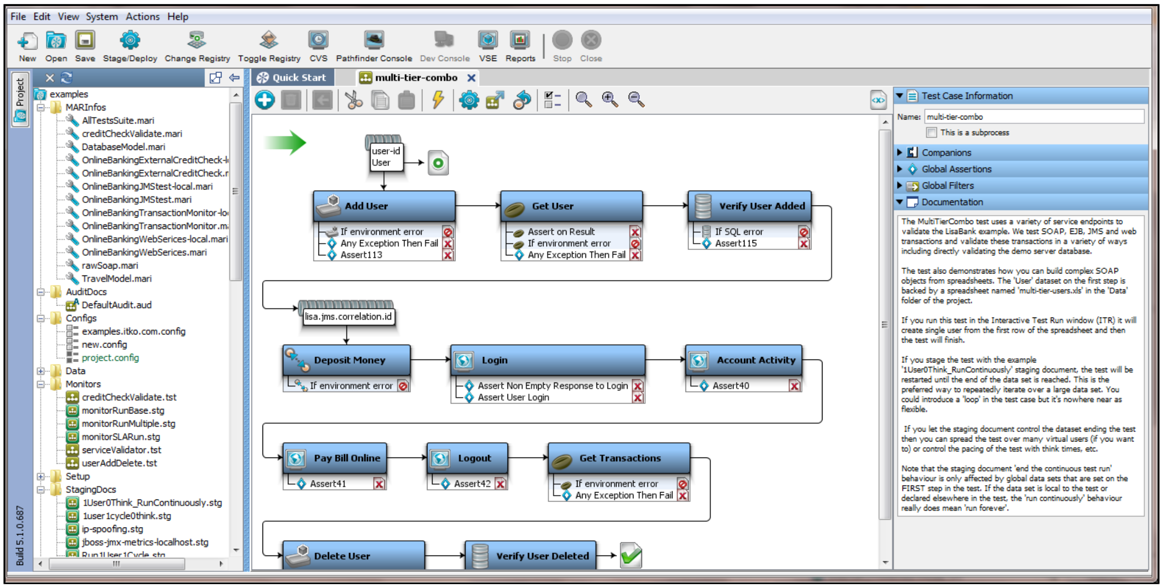
For example:

- **Test Case Information:** Enter the test case name and check if this is a subprocess.
- **Documentation:** Enter the documentation for the test case.

After you add steps to this test case, a workflow of steps begins to form and a new set of elements appears at a test step level in the Elements panel.

Multi-tier-combo Test Case

The examples project contains a test case named **multi-tier-combo**.



Name: multi-tier-combo

☐ This is a subprocess

The MultiTierCombo test uses a variety of service endpoints to validate the LisaBank example. We test SOAP, EJB, JMS and web transactions and validate these transactions in a variety of ways including directly validating the demo server database.

The test also demonstrates how you can build complex SOAP objects from spreadsheets. The 'User' dataset on the first step is backed by a spreadsheet named 'multi-tier-users.xls' in the 'Data' folder of the project.

If you run this test in the Interactive Test Run window (ITR) it will create single user from the first row of the spreadsheet and then the test will finish.

If you stage the test with the example '1User0Think_RunContinuously' staging document, the test will be restarted until the end of the data set is reached. This is the preferred way to repeatedly iterate over a large data set. You could introduce a 'loop' in the test case but it's nowhere near as flexible.

If you let the staging document control the dataset ending the test then you can spread the test over many virtual users (if you want to) or control the pacing of the test with think times, etc.

Note that the staging document 'end the continuous test run' behaviour is only affected by global data sets that are set on the FIRST step in the test. If the data set is local to the test or declared elsewhere in the test, the 'run continuously' behaviour really does mean 'run forever'.

This test case uses a variety of service endpoints to validate the LISA Bank demo application. It tests SOAP, EJB, JMS, and web transactions and validates these transactions in various ways, including directly validating the demo server database.

This test case also demonstrates how to build complex SOAP objects from spreadsheets. The User data set on the first step is backed by the **multi-tier-users.xls** spreadsheet in the Data folder of the project.

If you run this test in the Interactive Test Run (ITR) window, the test will create a single user from the first row of the spreadsheet and then will finish.

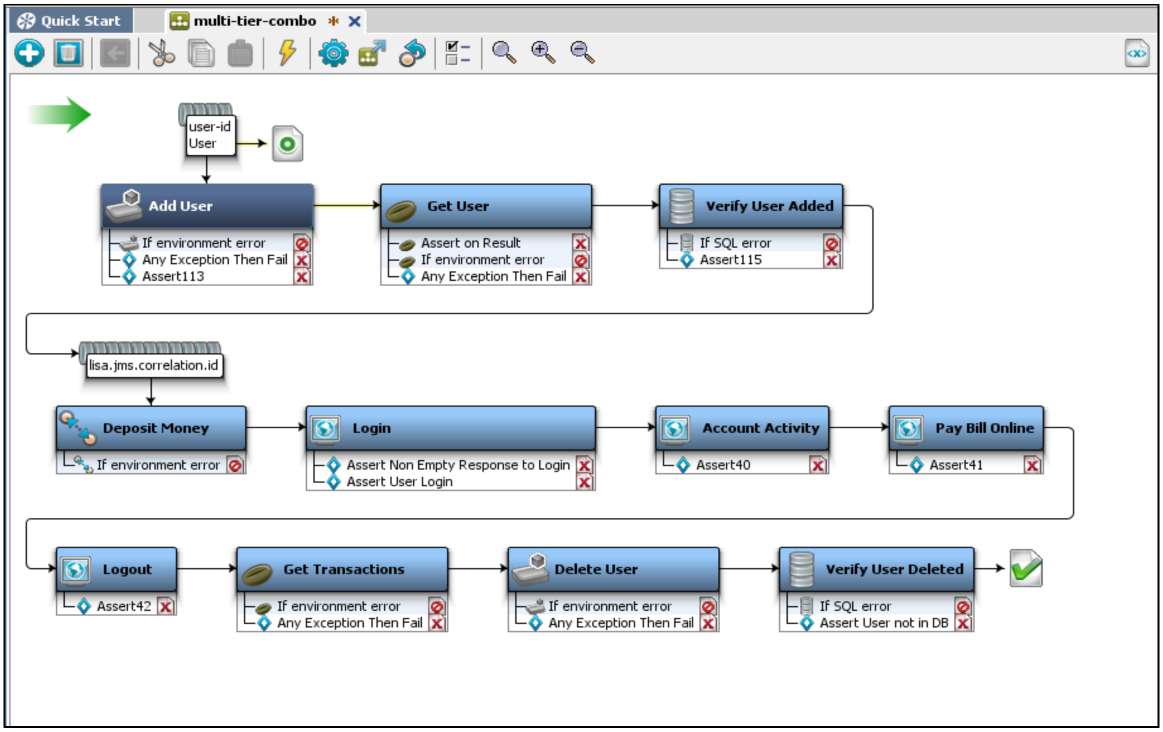
If you stage the test with the example **1User0Think_RunContinuously** staging document, the test will be restarted until the end of the data set is reached. This method is the preferred way to repeatedly iterate over a large data set. You can introduce a loop in the test case, but that is not as flexible.

If you let the staging document control the data set ending the test, then you can spread the test over many virtual users or control the pacing of the test with think times, for example.

The staging document "end the continuous test run" behavior is affected only by global data sets that are set on the first step in the test. If the data set is local to the test or declared elsewhere in the test, the "run continuously" behavior really does mean "run forever."

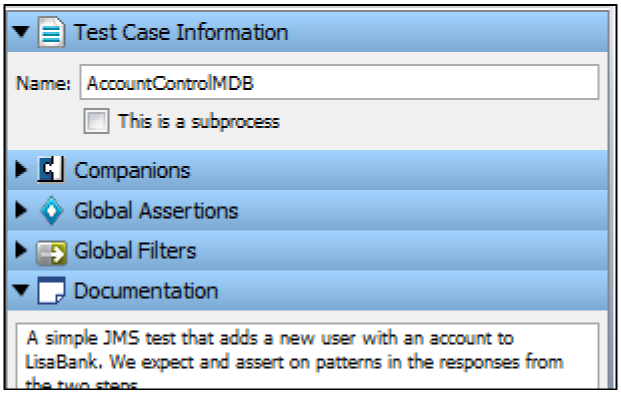
Notice the **example** project folders being opened in the Project panel and a set of test case elements in the Element panel.

Here you can see in the Model Editor section the test case information.



Elements of a Test Case

The elements of a test case help in building the test case as a whole. Following are the test case elements as they appear in the Test Case panel on the right side of LISA Workstation.



Test Case Information

Name: AccountControlMDB

□ This is a subprocess

Companions

Global Assertions

Global Filters

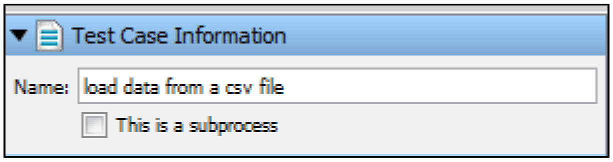
Documentation

A simple JMS test that adds a new user with an account to LisaBank. We expect and assert on patterns in the responses from the two steps

Test Case Information

The **Test Case Information** tab is where you can change the name of a test case.

This tab is also used as an entry point for creating a subprocess, or converting a test case into a subprocess. A subprocess is a test case that is designed to be called by another test case rather than to be run as a standalone test.


Test Case Information

Name: load data from a csv file

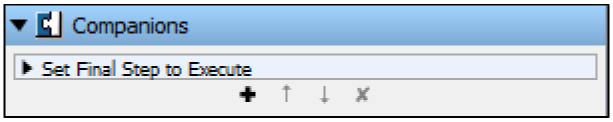
□ This is a subprocess

For more information about subprocesses, see Building Subprocesses.

Companions

A companion is an element that runs before and after every test case execution. Companions are used to configure global behavior in the test case. A restart causes the companions to run again.

Double-click the companion in the Element tree to open its editor.


Companions

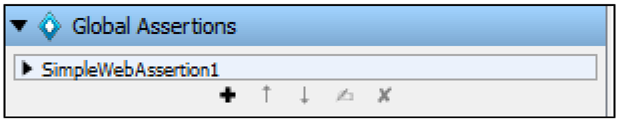
Set Final Step to Execute

For more information, see Companions.

Global Assertions

An assertion is an element that runs after a step and all its filters have run, to verify that the results from running the step match expectations. Global assertions are assertions that are applied to the entire test case.

Double-click the assertion in the Global Assertion list to open its editor.


Global Assertions

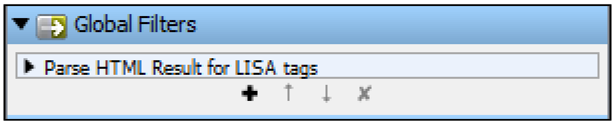
SimpleWebAssertion1

For more information, see Assertions.

Global Filters

A filter is an element that runs before and after a test step, giving you the opportunity to change the data in the result, or store values in properties. Global filters are filters that are applied to the entire test case.

Double-click the filter in the Global Filter list to open its editor.

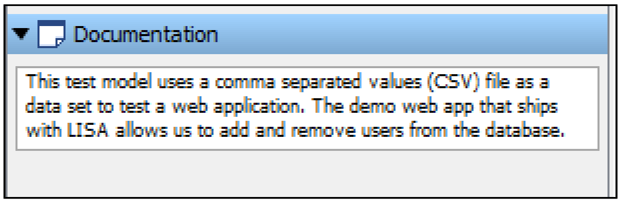

Global Filters

Parse HTML Result for LISA tags

For more information, see Filters.

Documentation

The Documentation text area lets you add documentation for your test case. This text is not used by LISA in any process, but it is a convenient place: and more importantly, a good practice to put a description of your test case, and notes for other users who will use this test case.

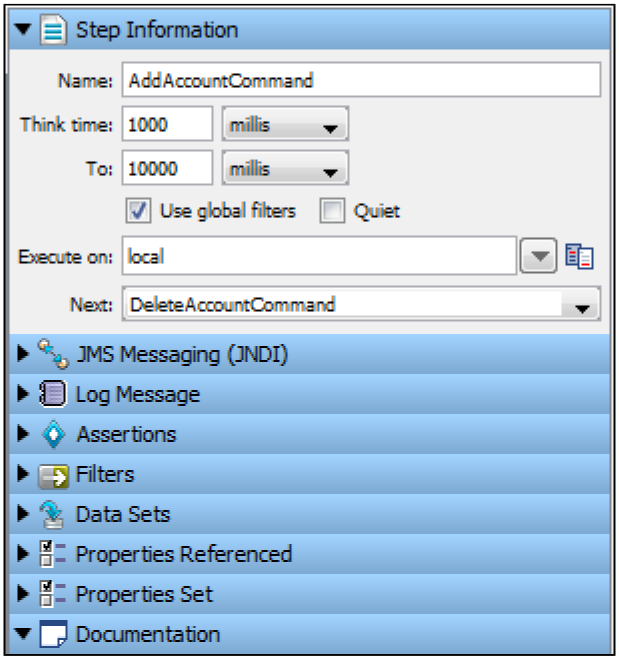

Documentation

This test model uses a comma separated values (CSV) file as a data set to test a web application. The demo web app that ships with LISA allows us to add and remove users from the database.

If the test case is used as a subprocess, the documentation will be passed to the calling step.

Elements of a Test Step

A test step is an element in the LISA workflow that performs a basic action to validate a business function in the system under test. Steps can be used to invoke portions of the system under test. These steps will typically be chained together to build workflows as test cases in the Model Editor. From within each step, you have the ability to create filters to extract data or create assertions to validate response data.



Step Information

Name: AddAccountCommand

Think time: 1000 millis

To: 10000 millis

☑ Use global filters ☐ Quiet

Execute on: local

Next: DeleteAccountCommand

JMS Messaging (JNDI)

Log Message

Assertions

Filters

Data Sets

Properties Referenced

Properties Set

Documentation

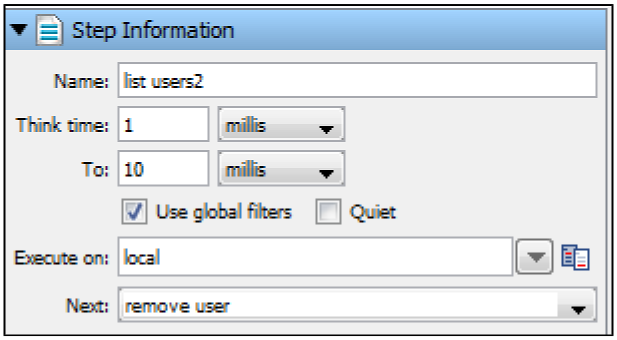
These elements are described briefly in the following sections, and discussed in detail in Building Test Steps.

- Step Information
- Log Message
- Assertions
- Filters
- Data Sets
- Properties Referenced
- Properties Set

Step Information

The step information section provides a place to document basic information about the test step.

You can enter the step name, think time, Execute on details, and Next step details. You can also specify to run the step using global filters and to run the step quietly.



Step Information

Name: list users2

Think time: 1 millis

To: 10 millis

☑ Use global filters ☐ Quiet

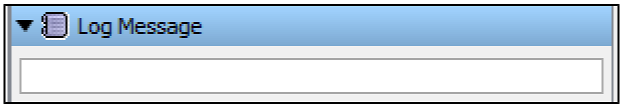
Execute on: local

Next: remove user

For more information, see Building Test Steps.

Log Message

A log message is a text field in which you can enter a message for a particular step. This message will be seen upon execution of the test step or case.

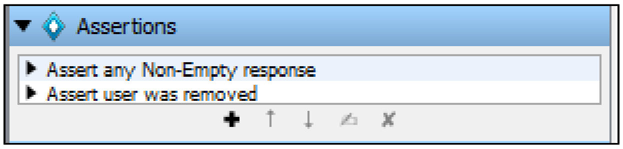


For more information, see Test Step Logger.

Assertions

An assertion is an element that runs after a step and all its filters have run, to verify that the results from running the step match expectations. The result of an assertion is Boolean - either true or false (there are no other possibilities).

The outcome determines whether the test step passes or fails, and the next step to run in the test case. That is, the assertion can dynamically alter the test case workflow by introducing conditional logic (branching) into the workflow.

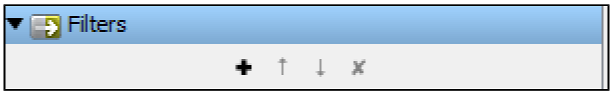


Assert any Non-Empty response
Assert user was removed

For more information, see Assertions.

Filters

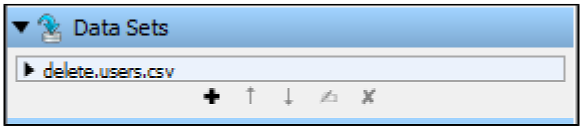
A filter is an element that runs before and after a test step, giving you the opportunity to change the data in the result, or store values in properties.



For more information, see Filters.

Data Sets

A data set is a collection of values that can be used to set properties in a test case while a test is running. This provides a mechanism to introduce external test data to a test case.



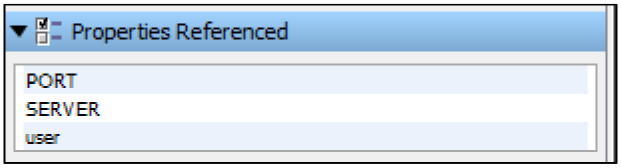
delete.users.csv

For more information, see Data Sets.

Properties Referenced

This section contains a list of properties used or referenced by the test step.

Select and right-click the property to open the extended view and get its variable value.

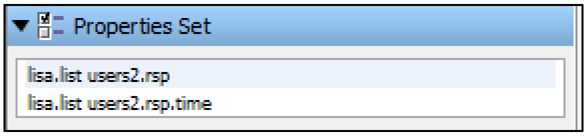


PORT
SERVER
user

For more information, see Properties.

Properties Set

This section contains a list of properties set by the test step. The Properties Referenced and Set are for a particular step and will change when another step is highlighted.



lisa.list users2.rsp
lisa.list users2.rsp.time

For more information, see Properties.

Properties

Test properties are name–value pairs, also known as key–value pairs.

The key to data independence, reusability, and portability in test cases is the ability to abstract specific data values out of the test case and replace them with variables. These variables are referred to as *properties*. Some properties are predefined and guide how LISA operates. Other properties are created by you while you are building your tests.

Properties are both ubiquitous and indispensable in test cases. A sound understanding of properties is vital to the creation of test cases. In the context of a test case, any time there is something that can change, it is appropriate to use a property. This includes values in test steps and values in configurations, for example.

Properties can be defined in several ways. After they are defined, they are available to any subsequent steps, assertions, and filters in the test case (they are global to the test case). Properties can, with few exceptions, be overridden in a test case.

Whenever a property value is set, an **EVENT_SETPROP** event is recorded that contains the property name and value.

Property values are not limited to string values. A property can hold strings, numbers, XML fragments, serialized Java objects, or the complete response from a test step. Many properties are created during a test run that are available to the subsequent test steps. For example, the **lisa.stepname.rsp** property contains the response for the **stepname** step.

In this section, the following topics are covered:

Specifying a Property

The syntax for a property is `{{property_name}}`.

When a property is identified and about to be used, `{{property_name}}` is replaced by its current value. There are times when a property is expected, and is the only choice. Other times, you are asked for the property name explicitly. In these cases, you enter the property name without the braces. When properties are embedded in a text string, you must use the brace notation. There is an additional syntax that allows property expressions to be used: `{{=expression}}` or `{{property_name=expression}}`.

A property name can contain spaces. However, using spaces is not recommended. The characters that define the property syntax (**{**,**}**, and **=**) cannot be used.

⚠ All property names that start with **lisa.** are reserved for internal use. Properties that start with **lisa** may be hidden or deleted**.**

If you reference a property that does not exist, the property is left in the braces to indicate that the property was not found, or it is invalid.

Property Expressions

LISA properties can store many different types of data. They can also evaluate and store expressions. These expressions can contain any valid Java or JavaScript expressions that can be evaluated by BeanShell. BeanShell is a Java interpreter environment. Further, these expressions could be string patterns, which give real-looking fake strings, appropriate for most given purposes.

For more information about BeanShell, see Using BeanShell in LISA or www.beanshell.org.

To use a property expression, you use the syntax **`{{=expression}}`** or **`{{key=expression}}`**. In the first case, **`{{expression}}`**, BeanShell will be used to evaluate the expression and replace **`{{expression}}`** by the result of the evaluation. For example, **`{{=Math.random()}}`** will evaluate the static Java method and replace the **`{{}}`** construct with the random number that was returned. In the latter case, using **`{{ rand=Math.random()}}`** will set a property **rand** equal to the random number that was returned, in addition to replacing the **{{}}** construct with the random number.

Existing properties can be used in a property expression. They are referenced by name only, that is, without the braces, because they are already defined as properties. If the property is not found, the property expression is returned within braces, to indicate that there is a problem in the expression.

String Patterns

String patterns are special types of property expressions that have a syntax **{{ =[:patternname:] }}.** For example, to format a first name, the string pattern property could be **{{ =[:First Name:] }}**. The property would evaluate to a fake first name that looks like a real name.

This is much better than the possibility of dealing with random strings that do not look like a real name. Of course, the string pattern functionality supports many patterns in addition to first names: last names, dates, Social Security numbers, credit card numbers, credit card expiration dates, and many more. This fake data comes with LISA, in TESTDATA table in its reportdb database.

The recommendation is that if you need a first name in your test case, you use {{ =[:First Name:] }} in a data set. The "{{=[" part is a signal to use the string pattern and it has a list of things "registered" that it knows about.

As an example, if you put the following information in a log step:

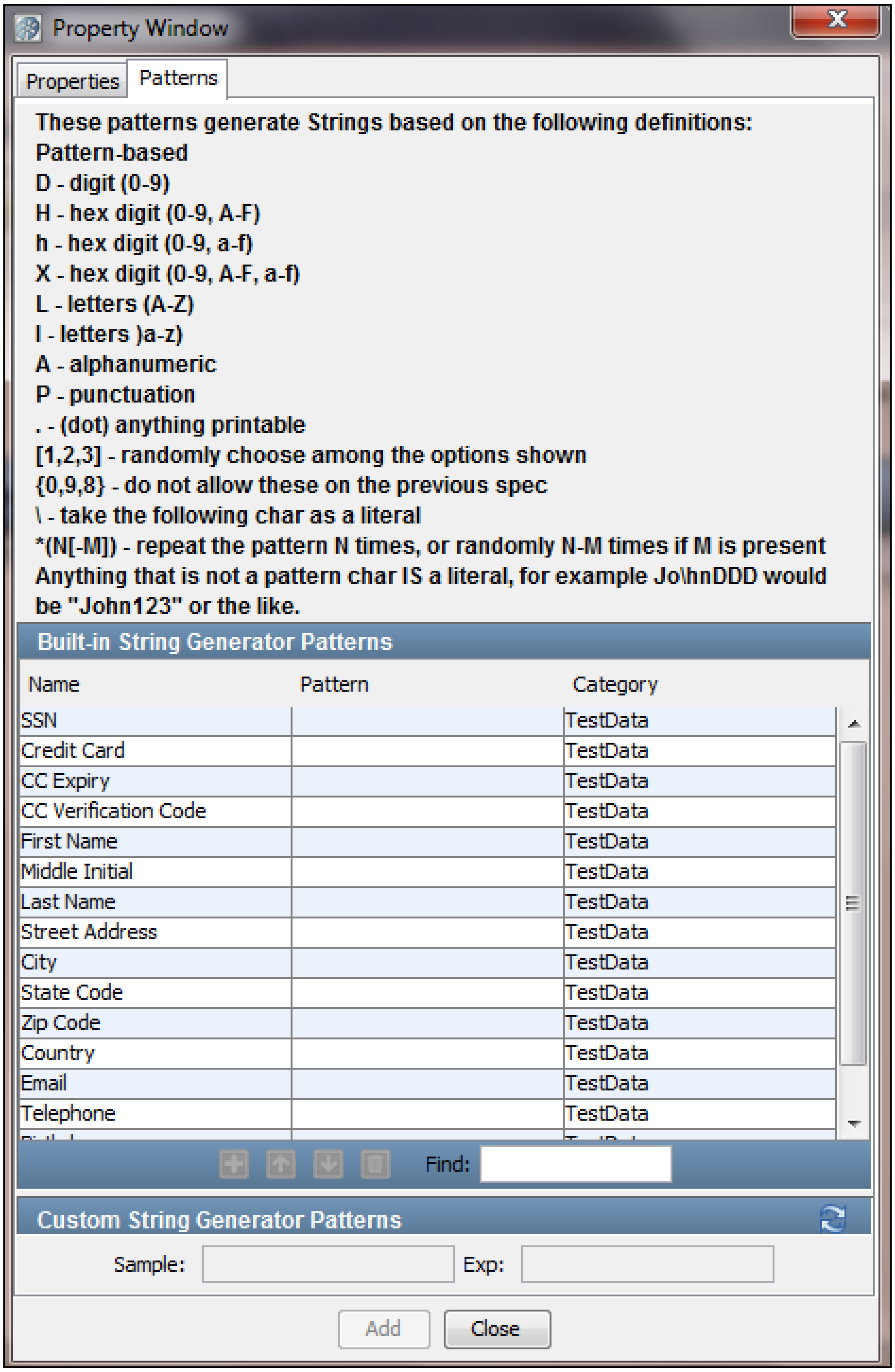
```
{{ =[:First Name:]}} {{ =[:Middle Initial:] }} {{ =[:Last Name:] }}

{{ =[:Street Address:] }}

{{ =[:City:] }}, {{ =[:State Code:] }}

{{ =[:ZIP Code:] }} {{= [:Country:] }}
SSN: {{ =[:SSN: DDD-DD-DDDD] }}

Card: {{ =[:Credit Card:] }} Expires {{= [:CC Expiry:] }}

Phone: {{ =[:Telephone:] }}

Email: {{ =[:Email:] }}
```

You get the response:

```
Marilyn Mcguire
3071 Bailey Drive
Oelwein, IA
50662 US
SSN: 483-16-8190
Card: 4716-2361-6304-6128 Expires 3/2014
Phone: 319-283-0064
Email: Marilyn.C.Mcguire@spambob.com
```

If you run the step again you will get a different set of data. It will keep track of the number of rows in the test data database and randomly select a row between 1 and N.

- Select Help > Property Window > Patterns to open the following screen, which shows all existing string patterns and shows the documentation.



These patterns generate Strings based on the following definitions:
Pattern-based
D - digit (0-9)
H - hex digit (0-9, A-F)
h - hex digit (0-9, a-f)
X - hex digit (0-9, A-F, a-f)
L - letters (A-Z)
l - letters)a-z)
A - alphanumeric
P - punctuation
. - (dot) anything printable
[1,2,3] - randomly choose among the options shown
{0,9,8} - do not allow these on the previous spec
\ - take the following char as a literal
*(N[-M]) - repeat the pattern N times, or randomly N-M times if M is present
Anything that is not a pattern char IS a literal, for example Jo\hnDDD would be "John123" or the like.

Built-in String Generator Patterns

Name	Pattern	Category
SSN		TestData
Credit Card		TestData
CC Expiry		TestData
CC Verification Code		TestData
First Name		TestData
Middle Initial		TestData
Last Name		TestData
Street Address		TestData
City		TestData
State Code		TestData
Zip Code		TestData
Country		TestData
Email		TestData
Telephone		TestData

Custom String Generator Patterns

Implementation Information

The data is stored by default in the reports database in the TESTDATA table.

When LISA Workstation is started, it checks to see if there is any data there and if there is no data, then com/itko/lisa/test/data/TestData.csv inside lisa-core.jar is read to load up the database. If reports.db gets deleted for some reason, the test data will be recreated. Reading the database is done only at startup and takes about 15 seconds.

Creating your own String Pattern

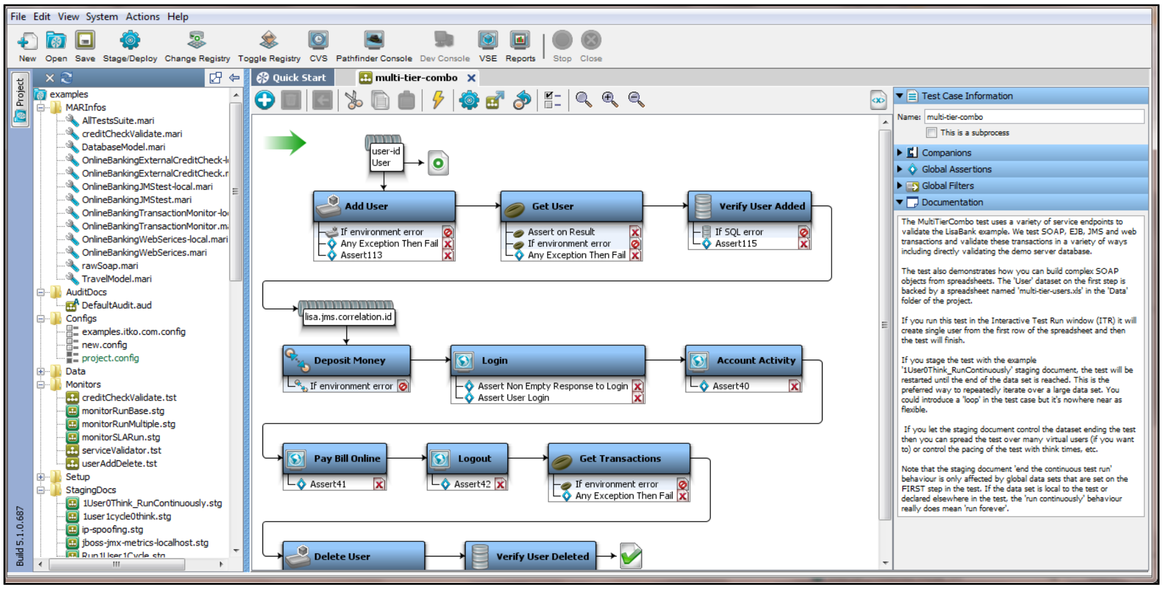
To add your own data, the only thing to be careful about is to assign the ID correctly. Start with 1 and increase with no gaps until you get to your number of rows.

The string generator code essentially does *select * from testdata where id = 'n'* after getting *n* from a random object. So, ID must be the primary key of the table to help ensure efficient lookups.

Example

A good way to get some practice with property expressions is to build a simple test case that has a single step: an Output Log Message. Because this step just writes to a log, and displays its response in the Interactive Test Run (ITR) Response Panel, you can experiment with using property expressions.

The following example uses the multi-tier-combo test case in the examples directory (multi-tier-combo.tst).



Name: multi-tier-combo

□ This is a subprocess

The MultiTierCombo test uses a variety of service endpoints to validate the LisaBank example. We test SOAP, EJB, JMS and web transactions and validate these transactions in a variety of ways including directly validating the demo server database.

The test also demonstrates how you can build complex SOAP objects from spreadsheets. The 'User' dataset on the first step is backed by a spreadsheet named 'multi-tier-users.xls' in the 'Data' folder of the project.

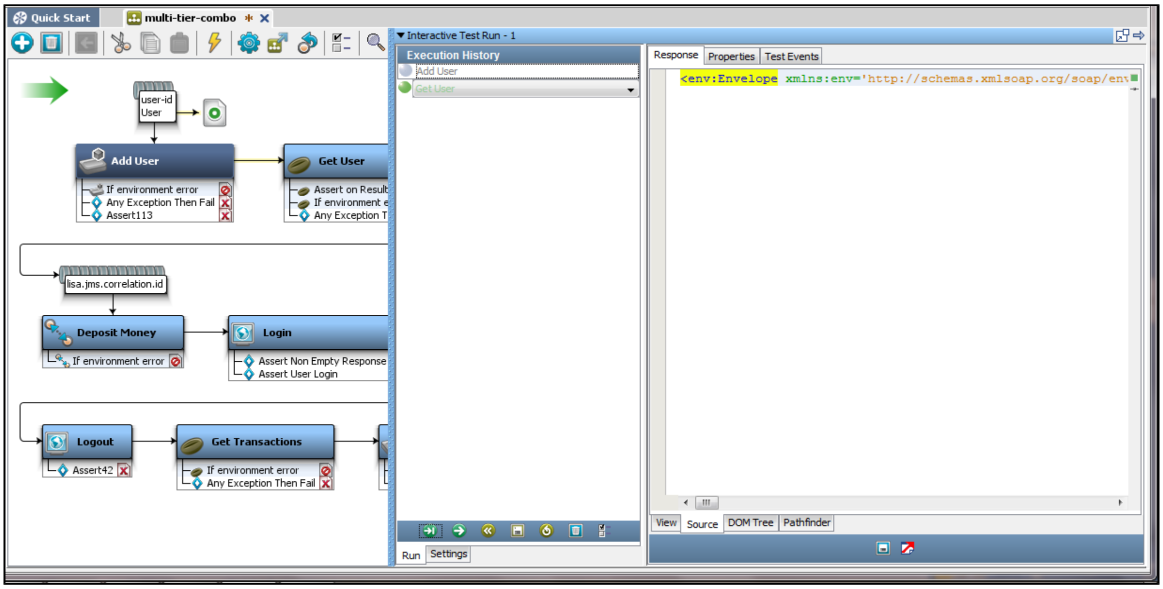
If you run this test in the Interactive Test Run window (ITR) it will create single user from the first row of the spreadsheet and then the test will finish.

If you stage the test with the example '1User0Think_RunContinuously' staging document, the test will be restarted until the end of the data set is reached. This is the preferred way to repeatedly iterate over a large data set. You could introduce a 'loop' in the test case but it's nowhere near as flexible.

If you let the staging document control the dataset ending the test then you can spread the test over many virtual users (if you want to) or control the pacing of the test with think times, etc.

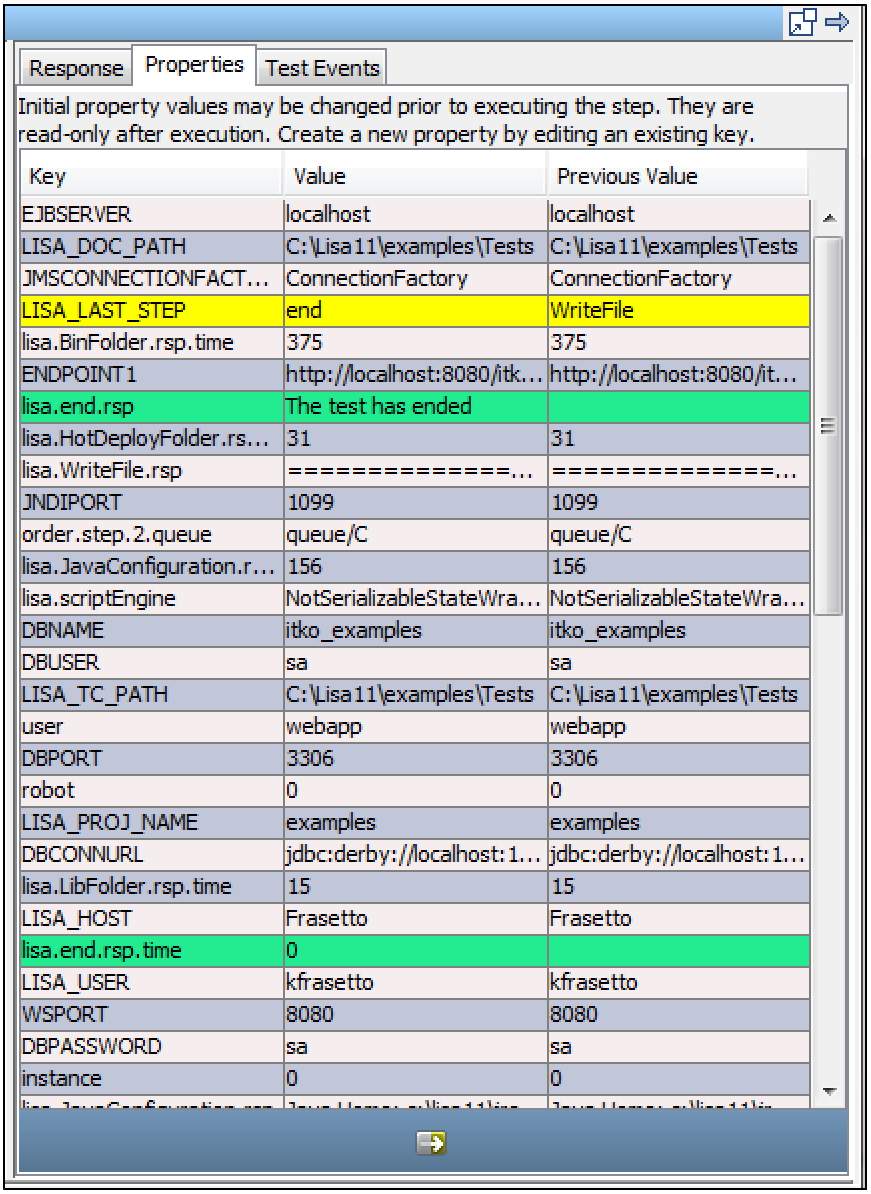
Note that the staging document 'end the continuous test run' behaviour is only affected by global data sets that are set on the FIRST step in the test. If the data set is local to the test or declared elsewhere in the test, the 'run continuously' behaviour really does mean 'run forever'.

This illustration shows our example in the LISA ITR utility.



```
<env:Envelope xmlns:env='http://schemas.xmlsoap.org/soap/env
```

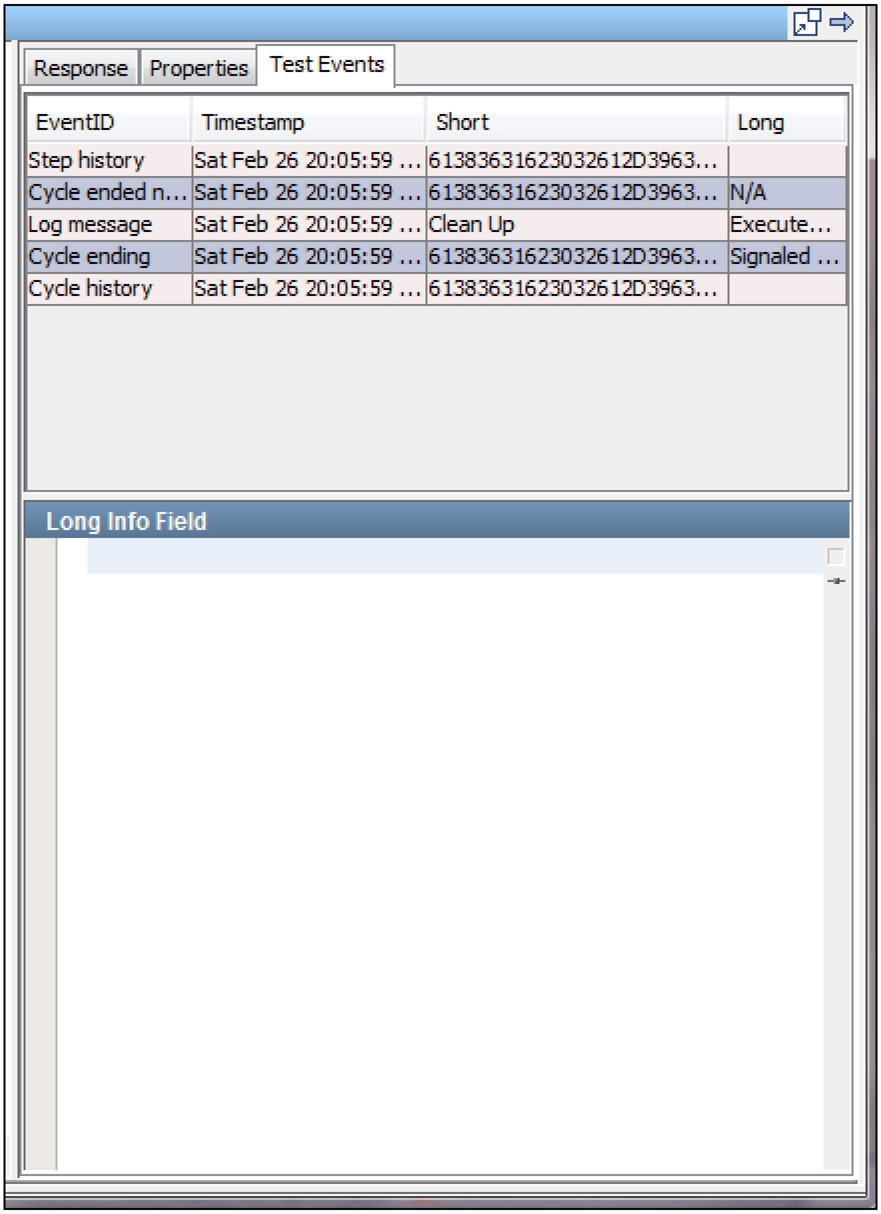
These illustrations show the Properties tab in the ITR. This is for the step Get User.



Initial property values may be changed prior to executing the step. They are read-only after execution. Create a new property by editing an existing key.

Key	Value	Previous Value
EJBSERVER	localhost	localhost
LISA_DOC_PATH	C:\Lisa11\examples\Tests	C:\Lisa11\examples\Tests
JMSCONNECTIONFACT...	ConnectionFactory	ConnectionFactory
LISA_LAST_STEP	end	WriteFile
lisa.BinFolder.rsp.time	375	375
ENDPOINT1	http://localhost:8080/itk...	http://localhost:8080/it...
lisa.end.rsp	The test has ended	
lisa.HotDeployFolder.rs...	31	31
lisa.WriteFile.rsp	==============...	==============...
JNDIPORT	1099	1099
order.step.2.queue	queue/C	queue/C
lisa.JavaConfiguration.r...	156	156
lisa.scriptEngine	NotSerializableStateWra...	NotSerializableStateWra...
DBNAME	itko_examples	itko_examples
DBUSER	sa	sa
LISA_TC_PATH	C:\Lisa11\examples\Tests	C:\Lisa11\examples\Tests
user	webapp	webapp
DBPORT	3306	3306
robot	0	0
LISA_PROJ_NAME	examples	examples
DBCONNURL	jdbc:derby://localhost:1...	jdbc:derby://localhost:1...
lisa.LibFolder.rsp.time	15	15
LISA_HOST	Frasetto	Frasetto
lisa.end.rsp.time	0	
LISA_USER	kfrasetto	kfrasetto
WSPORT	8080	8080
DBPASSWORD	sa	sa
instance	0	0
lisa.JavaConfiguration.rsp	Java Home: c:\lisa11\jre	Java Home: c:\lisa11\jr

This illustration shows the Test Events tabs in the ITR.



EventID	Timestamp	Short	Long
Step history	Sat Feb 26 20:05:59 ...	6138363162303261 2D3963...	
Cycle ended n...	Sat Feb 26 20:05:59 ...	6138363162303261 2D3963...	N/A
Log message	Sat Feb 26 20:05:59 ...	Clean Up	Execute...
Cycle ending	Sat Feb 26 20:05:59 ...	6138363162303261 2D3963...	Signaled ...
Cycle history	Sat Feb 26 20:05:59 ...	6138363162303261 2D3963...	

Look for the event Properties set in the Test Events tab.

Java developers can also take advantage of the BeanShell environment in the JavaScript step to test property expressions.

LISA Property Sources

Properties can originate from several sources that include:

- LISA
- Environmental variables
- Command line variables on startup
- Configurations
- Companions
- Test steps
- Filters
- Data sets
- String patterns

Because properties can be overridden, it is important to understand the property hierarchy, or the order in which properties are read in a test case.

The following hierarchy is used:

1. Properties loaded during the set up of a test

2. Operating system environment variables (like java.version or os.user, for example)
3. LISA property files
4. Command line attributes
5. The default configuration
6. Any alternative configuration properties (from active configuration or runtime configuration file)
7. Properties set during a test run
8. Properties in companions
9. Properties set during test execution (for example, in data sets, filters and steps). Remember that properties set here override values set earlier.

Common LISA Properties and Environment Variables

HOT_DEPLOY: Points to a project-specific **hotDeploy** directory.

LASTRESPONSE: The response to the last executed step.

LISA_HOME: Points to the LISA install directory, and is automatically set. This value includes a final slash. To reference a directory such as "examples", specify:

```
{{LISA_HOME}}examples
```

No slash is needed before the directory name.

LISA_HOST: The name of the system on which the testing environment is running.

LISA_JAVA_HOME: The Java VM that LISA will use. Use this only if you do not want to use the built-in VM in LISA. If there is no Java installed, LISA uses the bundled JRE it comes with. You also must rename the **jre** directory in the LISA install directory to something like **jre_notinuse**.

LISA_POST_CLASSPATH: Used to add information after the LISA classpath. LISA does not use the OS environment CLASSPATH variable. To add your own JARs after the LISA classpath, use LISA_POST_CLASSPATH.

LISA_PRE_CLASSPATH: Used to add information before the LISA classpath. LISA does not use the OS environment CLASSPATH variable. To add your own JARs before the LISA classpath, use LISA_PRE_CLASSPATH.

LISA_PROJ_NAME: The name of the project to which the current document belongs.

LISA_PROJ_PATH: The fully qualified path of the project directory. The value is operating system-dependent. A backslash (\) is used as the separator character on Windows. A forward slash (/) is used as the separator character on all other operating systems. The following example is based on a Windows installation:

```
C:\Program Files\LISA\examples
```

There is one limitation to using LISA_PROJ_PATH in a Custom Java step: the syntax {{LISA_PROJ_PATH}} is not supported. Because the Custom Java step invokes a Java compiler to compile the script, and Java treats backslashes as escape characters in strings, this particular string raises a compiler error. The workaround is to use LISA_PROJ_PATH as a variable. For example:

```
File f = new File ( LISA_PROJ_PATH );
```

LISA_PROJ_ROOT: The fully qualified path of the project directory. The value is operating system-independent. A forward slash (/) is used as the separator character on all operating systems, including Windows. The following example is based on a Windows installation:

```
C:/Program Files/LISA/examples
```

LISA_PROJ_URL: The URL of the project directory. For example:

```
file:/C:/Program%20Files/LISA/examples
```

LISA_TC_PATH: The fully qualified path of the directory where the test case is located.

LISA_TC_URL: The URL of the directory where the test case is located.

LISA_USER: The user that loaded the test case.

Property Files

The main property files are:

- lisa.properties file
- local.properties file
- site.properties file

> Detailed information about properties is available in these appendixes:
>
> Appendix A - LISA Property File (lisa.properties)
> Appendix B - Custom Property Files (local.properties, site.properties)

Configurations

A configuration is a named collection of properties that usually specify environment-specific values for the system under test.

By removing hard-coded environment data from the test case, you can run the same test against different environments by simply using a different configuration. Configurations are used everywhere within LISA: for example, in a test case document, test suite document, staging document, test case execution or test suite execution.

A configuration must be defined at the LISA project level. You can specify the values of these properties at the beginning of a test case.

The default configuration of any project is **project.config**. You can create additional new configurations within a project and "Make it Active" for a particular test case/suite.

If you create a new config file, you will *not* be able to add any new keys within it. To add keys within the new config, you must add them in the **project.config** file. You can then select the newly-defined keys added in the **project.config** file from the drop-down available in the new config file.

Properties are added to your configuration automatically as you develop your test.

For example, in a web service test when you enter the name of the WSDL, the server name and the port that you entered are replaced with properties such as **`WSSERVER`** and **`WSPORT`.** The values of these properties are automatically added to your default project configuration. Now you can change the location of the web service merely by editing the configuration, rather than looking for hard-coded values in several test steps.

In another example, when working with Enterprise JavaBeans (EJBs) or Java objects, you may want to switch hot deploy directories, or add extra JAR files to your class path, to use different versions of your Java code. There are standard properties for these locations: HOT_DEPLOY and MORE_JARS. These can be set in your configuration.

For information about other standard properties, see Properties.

Configurations are for storing properties related to the system under test. Avoid using them for storage of "test-like" parameters and global parameters. These can be stored in a companion.

> ⚠ Backslashes "\" are not preserved in configuration files. If you edit the config file manually and put something that has a backslash, the file will be overwritten without the backslashes.

The following topics are available in this chapter:

LISA Project Configuration

The configuration at the project level can be seen in the **Configs** section in the Project panel.

The project.config is by default the active configuration (marked in green) for any LISA project.

All newly-defined properties must be in the default project.config file. Later they can be pulled into the new configuration file.

New configurations can be applied to a test case by making them "Active".

> Configurations are defined at the project level.

Default Configuration

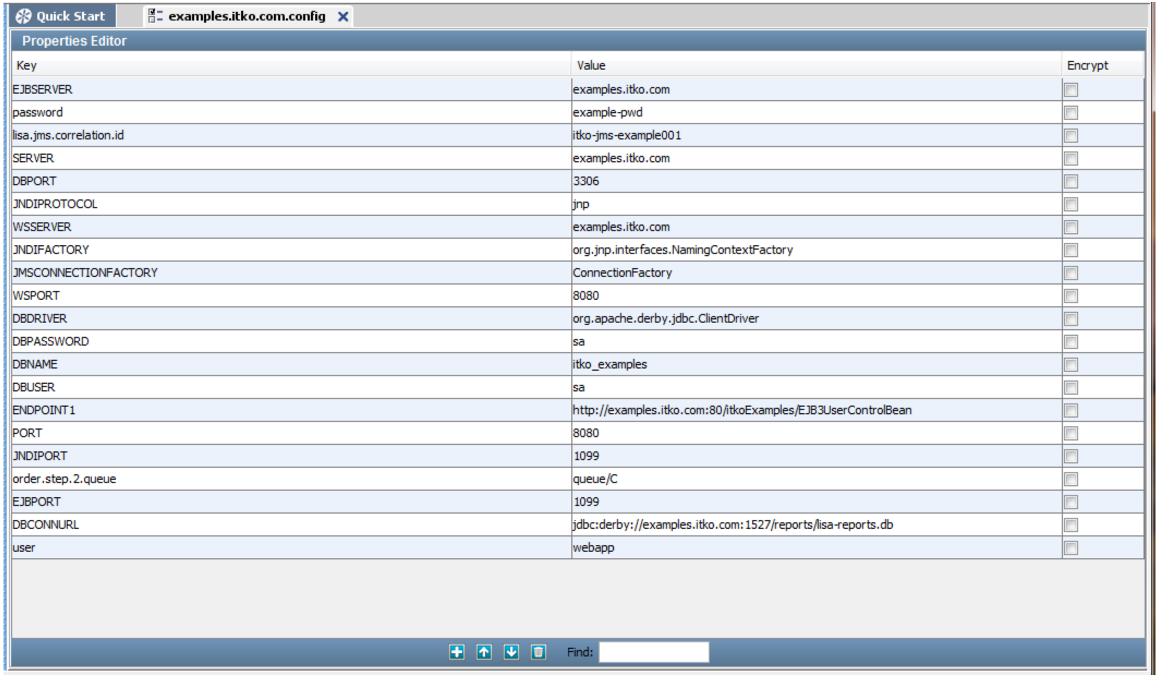
The default configuration of any project is **project.config**. It is also the *active* configuration for the project. It has the superset of all the keys defined in every other configuration.

After you open a project, the default configuration (**project.config**) is shown in green in the Configs folder.

You cannot delete or rename the default configuration.

You can change the active configuration of a project. For instructions on how to do this, see Marking a Configuration as Active.

Double-clicking **project.config** will open the configuration in the Properties Editor window. The name of the tab will show the name of the configuration.



Key	Value	Encrypt
EJBSERVER	examples.itko.com	□
password	example-pwd	□
lisa.jms.correlation.id	itko-jms-example001	□
SERVER	examples.itko.com	□
DBPORT	3306	□
JNDIPROTOCOL	jnp	□
WSSERVER	examples.itko.com	□
JNDIFACTORY	org.jnp.interfaces.NamingContextFactory	□
JMSCONNECTIONFACTORY	ConnectionFactory	□
WSPORT	8080	□
DBDRIVER	org.apache.derby.jdbc.ClientDriver	□
DBPASSWORD	sa	□
DBNAME	itko_examples	□
DBUSER	sa	□
ENDPOINT1	http://examples.itko.com:80/itkoExamples/EJB3UserControlBean	□
PORT	8080	□
JNDIPORT	1099	□
order.step.2.queue	queue/C	□
EJBPORT	1099	□
DBCONNURL	jdbc:derby://examples.itko.com:1527/reports/lisa-reports.db	□
user	webapp	□

These are standard LISA config parameters that are available in all configurations.

To add parameters in other configurations, first add them here and then select from the drop-down in the required configuration.

Adding a Configuration

A project can have many alternate configurations, but it can have only one active configuration. Any alternate configuration or the default configuration can be made active. Alternate configurations can only override default properties.

To add a configuration

1. Right-click the **Configs** folder in the Project panel.

2. Click Create New Config.
3. Enter the name of the new configuration.
4. Click OK.

Adding Key-Value Pairs

To add keys within the new configuration, you must add them in the **project.config** file. You can select the newly defined keys from the drop-down in the new configuration file. In a new configuration file, you will *not* be able to add any new keys.

When you create a configuration file, the only keys that you can add are the ones that are already defined in **project.config**, in addition to the standard config keys provided with LISA.

Marking a Configuration as Active

When you want to apply a different configuration to your project, you must make it active.

Within the Configs folder, you can mark any configuration as active.

To mark a configuration as active

1. Right-click the required config file in the Configs folder.
2. Click Make Active.

A configuration can be both the default and active.

The active configuration applies to the whole project, not a single test case.

Each test case in a single project shares the active configuration applied to the project. You cannot assign a separate configuration for each test case within a project.

The active configuration takes precedence in the Interactive Test Run (ITR) utility, but the default configuration is used in a Stage Test Case if no configuration is specified.

Editing a Configuration

In any configuration other than **project.config**, you can add properties only if they exist in **project.config**.

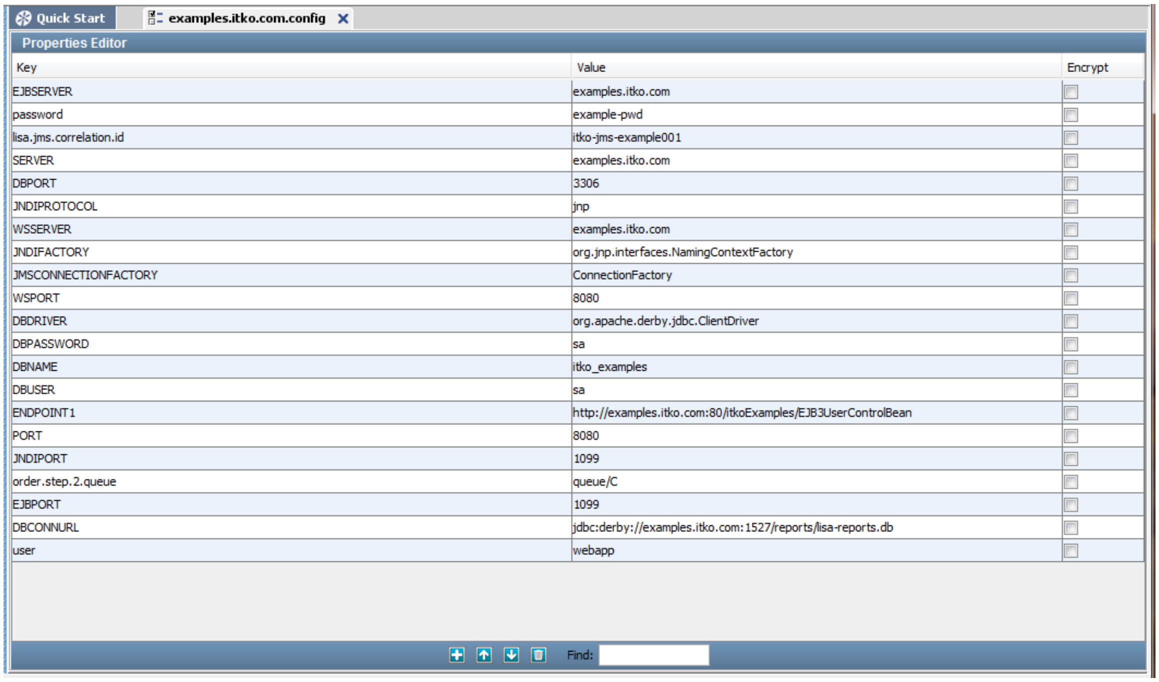
A best practice is to use properties in the path names stored in the configuration. Properties such as `LISA_HOME` or `LISA_PROJ_ROOT` will allow for portability of test cases.

A property value can contain multiple lines.

The extended view consists of a dialog for editing a property value. This view can be useful when the value is long or when the value contains multiple lines. To access the extended view, right-click the property value cell and select Launch Extended View.

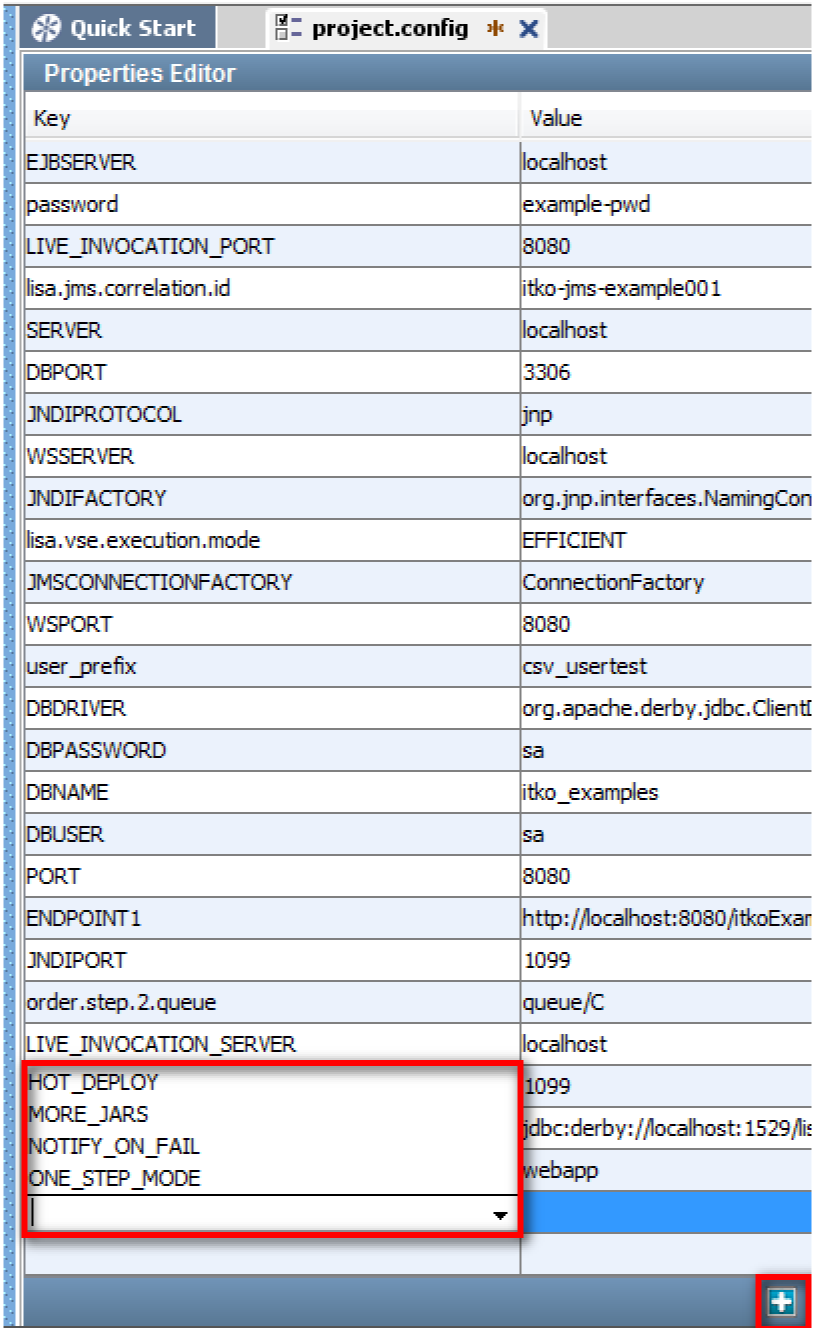
To edit a configuration

1. Double-click the configuration in the **Config** folder. The Properties Editor appears.



Key	Value	Encrypt
EJBSERVER	examples.itko.com	□
password	example-pwd	□
lisa.jms.correlation.id	itko-jms-example001	□
SERVER	examples.itko.com	□
DBPORT	3306	□
JNDIPROTOCOL	jnp	□
WSSERVER	examples.itko.com	□
JNDIFACTORY	org.jnp.interfaces.NamingContextFactory	□
JMSCONNECTIONFACTORY	ConnectionFactory	□
WSPORT	8080	□
DBDRIVER	org.apache.derby.jdbc.ClientDriver	□
DBPASSWORD	sa	□
DBNAME	itko_examples	□
DBUSER	sa	□
ENDPOINT1	http://examples.itko.com:80/itkoExamples/EJB3UserControlBean	□
PORT	8080	□
JNDIPORT	1099	□
order.step.2.queue	queue/C	□
EJBPORT	1099	□
DBCONNURL	jdbc:derby://examples.itko.com:1527/reports/lisa-reports.db	□
user	webapp	□

2. You can add a property by clicking the Add icon ⊞ at the bottom of the Properties Editor. A new line is added to the property list. Click the drop-down to select the chosen key. Common property names, such as HOT_DEPLOY and MORE_JARS, appear in the drop-down.



Key	Value
EJBSERVER	localhost
password	example-pwd
LIVE_INVOCATION_PORT	8080
lisa.jms.correlation.id	itko-jms-example001
SERVER	localhost
DBPORT	3306
JNDIPROTOCOL	jnp
WSSERVER	localhost
JNDIFACTORY	org.jnp.interfaces.NamingCon
lisa.vse.execution.mode	EFFICIENT
JMSCONNECTIONFACTORY	ConnectionFactory
WSPORT	8080
user_prefix	csv_usertest
DBDRIVER	org.apache.derby.jdbc.ClientI
DBPASSWORD	sa
DBNAME	itko_examples
DBUSER	sa
PORT	8080
ENDPOINT1	http://localhost:8080/itkoExar
JNDIPORT	1099
order.step.2.queue	queue/C
LIVE_INVOCATION_SERVER	localhost
	1099
	jdbc:derby://localhost:1529/lis
	webapp

Copying a Configuration

To copy a configuration

- Select the configuration file to be copied, right-click and select Copy to copy the selected configuration.

To paste a configuration

1. Click the Configs folder, right-click and click Paste. You will be prompted to rename the pasted config because it will appear to be a duplicate of the copied config.
2. Rename the configuration file and click OK to add the new configuration in the Config folder.

Deleting a Configuration

To delete a configuration

1. Select the configuration to be deleted and right-click.
2. Click Delete from the dialog.

You cannot delete the default configuration (project.config).

Renaming a Configuration

To rename a configuration

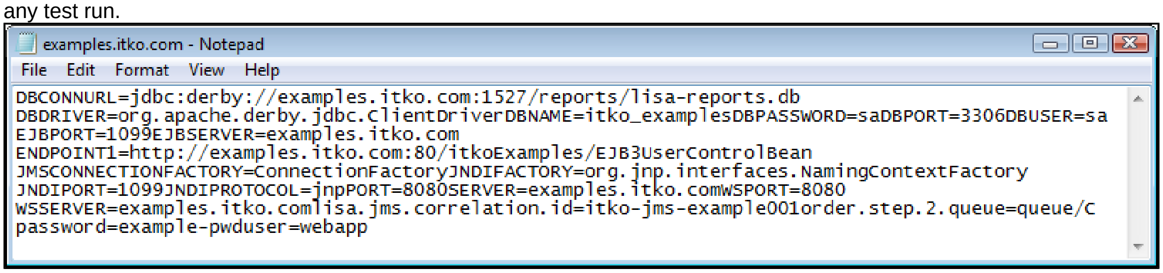
1. Select the configuration and right-click to open a menu.
2. Click Rename to rename the configuration.
3. In the window that opens, enter the new name and click OK to rename the file.

You cannot rename the default configuration (project.config) within LISA. When right-clicking on project.config, the Rename option will not appear.

Creating a New Configuration File

You may need to create a new configuration file to import into LISA.

A configuration file is a text file with a .config extension that contains the properties as key-value pairs. All configuration files are an integral part of any test run.



```
DBCONNURL=jdbc:derby://examples.itko.com:1527/reports/lisa-reports.db
DBDRIVER=org.apache.derby.jdbc.ClientDriverDBNAME=itko_examplesDBPASSWORD=saDBPORT=3306DBUSER=sa
EJBPORT=1099EJBSERVER=examples.itko.com
ENDPOINT1=http://examples.itko.com:80/itkoExamples/EJB3UserControlBean
JMSCONNECTIONFACTORY=ConnectionFactoryJNDIFACTORY=org.jnp.interfaces.NamingContextFactory
JNDIPORT=1099JNDIPROTOCOL=jnpPORT=8080SERVER=examples.itko.comWSPORT=8080
WSSERVER=examples.itko.comlisa.jms.correlation.id=itko-jms-example001order.step.2.queue=queue/C
password=example-pwduser=webapp
```

Configuration files can be created, or edited in any text editor and can be saved with a **.**config extension.

When starting a test run, you will have the opportunity to choose a configuration file of your choice. There is an example of this later in this chapter.

We recommend that you establish a naming convention for configuration files, making identification of alternate configurations easier.

These configuration files need to be imported into LISA.

Importing a Configuration File

To import a configuration file

1. Select the Configs folder and right-click to open a menu.
2. Click Import to import a configuration file.
3. On the window that opens, enter the name of the configuration file to be imported and click Open to import the file.
4. A notification message appears explaining that a config file is being imported and a successful message also appears.
5. Click OK.

You will see the new configuration file imported in the list of configurations, and any new properties will also be added to **project.config**.

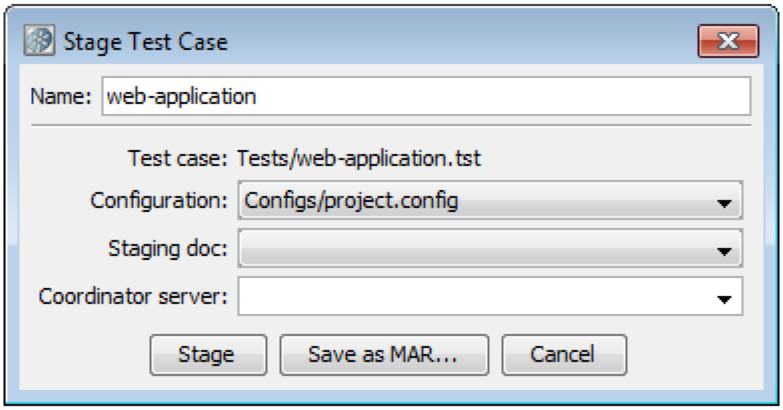
Applying a Configuration when Running a Test Case

Running a test case is one of several places where configurations are used.

To start a test case execution, you must give the test case details with the configuration to be applied.

To stage/start a test case from the main menu

1. Select Actions > Stage Test. The Stage Test Case dialog appears.



Name: web-application

Test case: Tests/web-application.tst

Configuration: Configs/project.config

Staging doc:

Coordinator server:

2. The Configuration pull-down will include all the configurations defined in the project. Assigning a configuration file here is optional. If omitted, the default configuration in the test case document will be used.
3. Select the staging document to refer to while executing the test case.
4. Select the coordinator server.
   When your selections are complete, you can either click Stage to stage the test, or Save as MAR... to create a model archive including the test case, configuration, staging document and coordinator server information you specified here.

Filters

A filter is a LISA code element that runs before and after a test step, giving you the opportunity to change the data in the result, or store values in properties.

Most filters execute after the step has run. A filter can be used to extract a value from a web page, XML and DOM responses, a Java object, a text document, and many other test step responses.

After the data has been filtered, the data can be used in an assertion, or in any subsequent test step. Filters usually operate on the response of the system under test. For example, filters are used to parse values from an HTML page, or to perform conversions on the response. Filters can also be useful in other places; for example, to save a property value to a file, or convert a property to be the "last response".

There are two basic ways to apply a filter, as a **global filter** or as a **step filter**. The available filter types are the same, but how the filters are applied differs.

- **Global filter**: A filter defined on the *test case* level is a global filter, and executes before/after every test step that is not set to ignore global filters. You can set a step to ignore global filters in the Step Information Element of the step.
- **Step filter:** A filter defined on a *test step* level is a step filter and executes before/after each execution of that test step.

You can add as many global and step filters as you need. They are executed in the order that they appear in the test case.

Filters are mainly used as property setters.

The following topics are available in this chapter.

Adding a Filter

There are several ways to add filters.

Adding a Filter Manually

To add a filter manually, select the filter type from a list and enter the parameters for the filter.

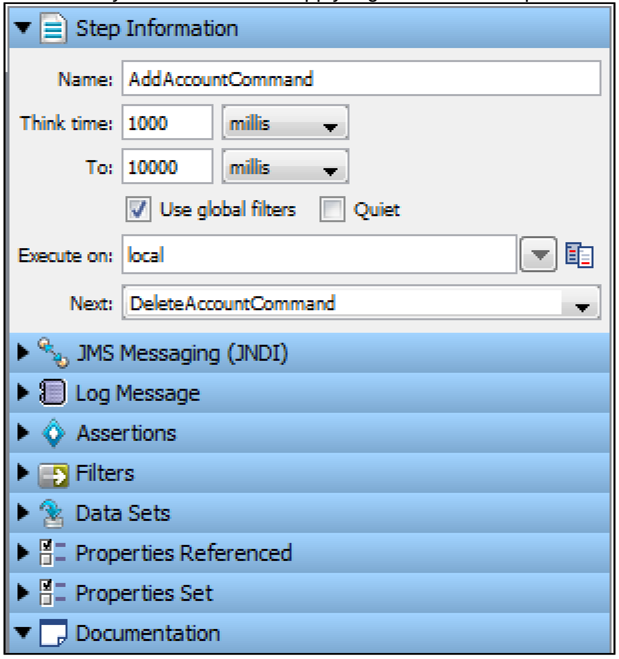
You can add two types of filters manually: global filters and step filters.

The first method lets you add a filter at the test case level (global filter). Global filters are applicable to all the steps in the test case and are automatically run for every step in the test case, unless a given step is instructed otherwise.

The second method lets you add a filter at the test step level (step filter). A filter created using this method is applicable only to that step and will execute for that step only.

Add a Global Filter

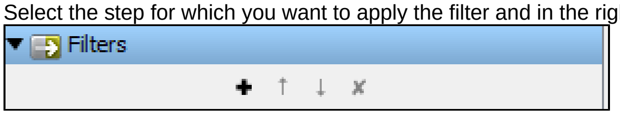
1. Open a test case and click anywhere in the editor area to open the Test Case Elements panel.
2. On the Global Filters element, click the Add + icon to add a global filter. You will be prompted to select a filter type of LISA Integration Support for Pathfinder or LISA Integration Support for webMethods Integration Server. For more information about adding each of these types of filters, go to **LISA Integration Support for Pathfinder** or **LISA Integration Support for webMethods Integration Server**.
3. When you have at least one global filter on a test case, you can see that for each step, by default the Use Global Filters check box is selected. If you do not want to apply a global filter for a particular step, clear the box.



Name: AddAccountCommand
Think time: 1000 millis
To: 10000 millis
☑ Use global filters ☐ Quiet
Execute on: local
Next: DeleteAccountCommand

Add a Step Filter

1. Select the step for which you want to apply the filter and in the right panel, click the Filter element.



2. Click the Add + icon of the filter element to get the list of available filters to choose from, or right-click the step and select Add Filter and select the appropriate filter for this step.

This will open the Filter menu listing the common filters in LISA. Each filter has its own editor and applicable parameters that need to be set.

Example

This example uses the Override Last Response Property/Convert Property Value into Last Response filter.

This filter converts a property value into the last response.

To configure the filter, enter the following parameters:

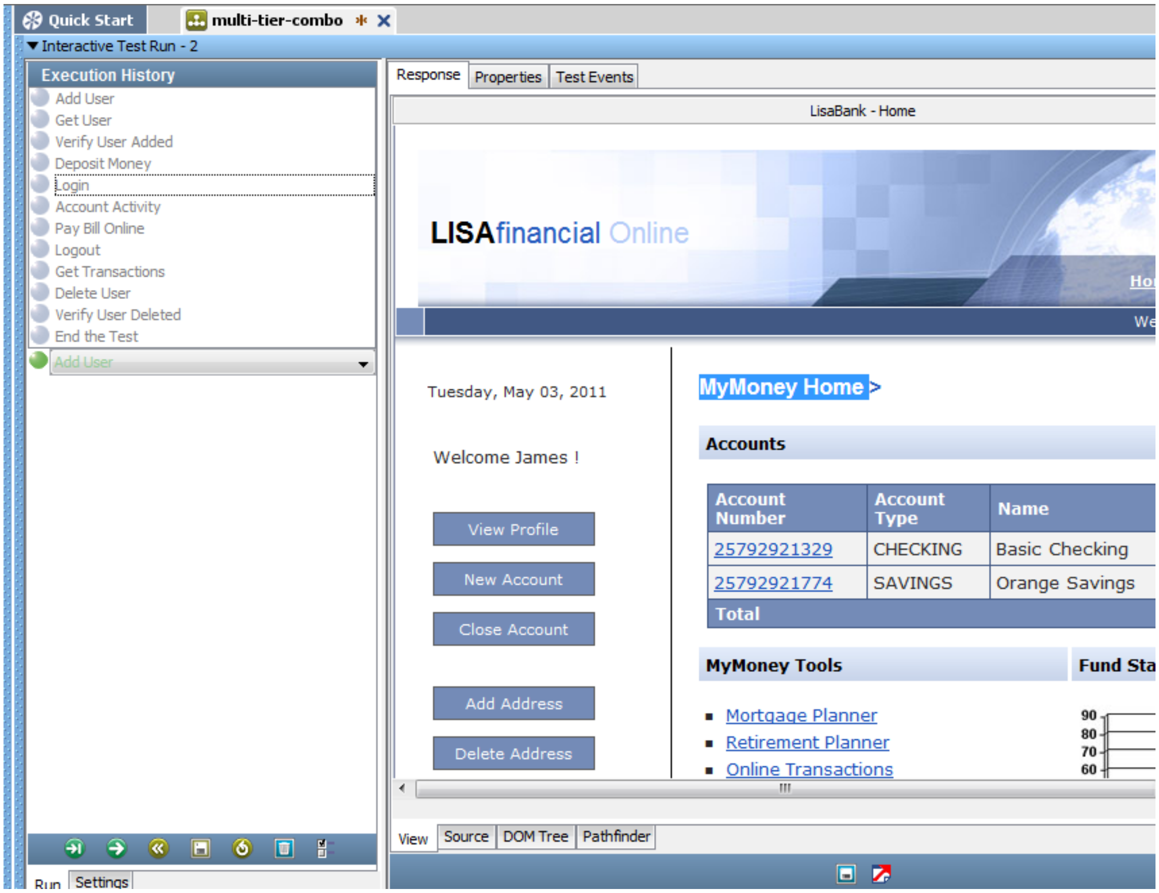
- **Filter in (Property to Convert):** The name of the property you want considered as the step's last response. The property should be in the pull-down menu. You can type the property name if you do not find it there. The filter will give an error if it is not an existing property.
- **Convert to XML:** Select this if you want the response to be converted to valid XML.
- **Run Filter:** Click to run the filter.

Adding a Filter from an HTTP Response

When you have access to the response from an HTTP-based step, you can use the response to add a filter directly.

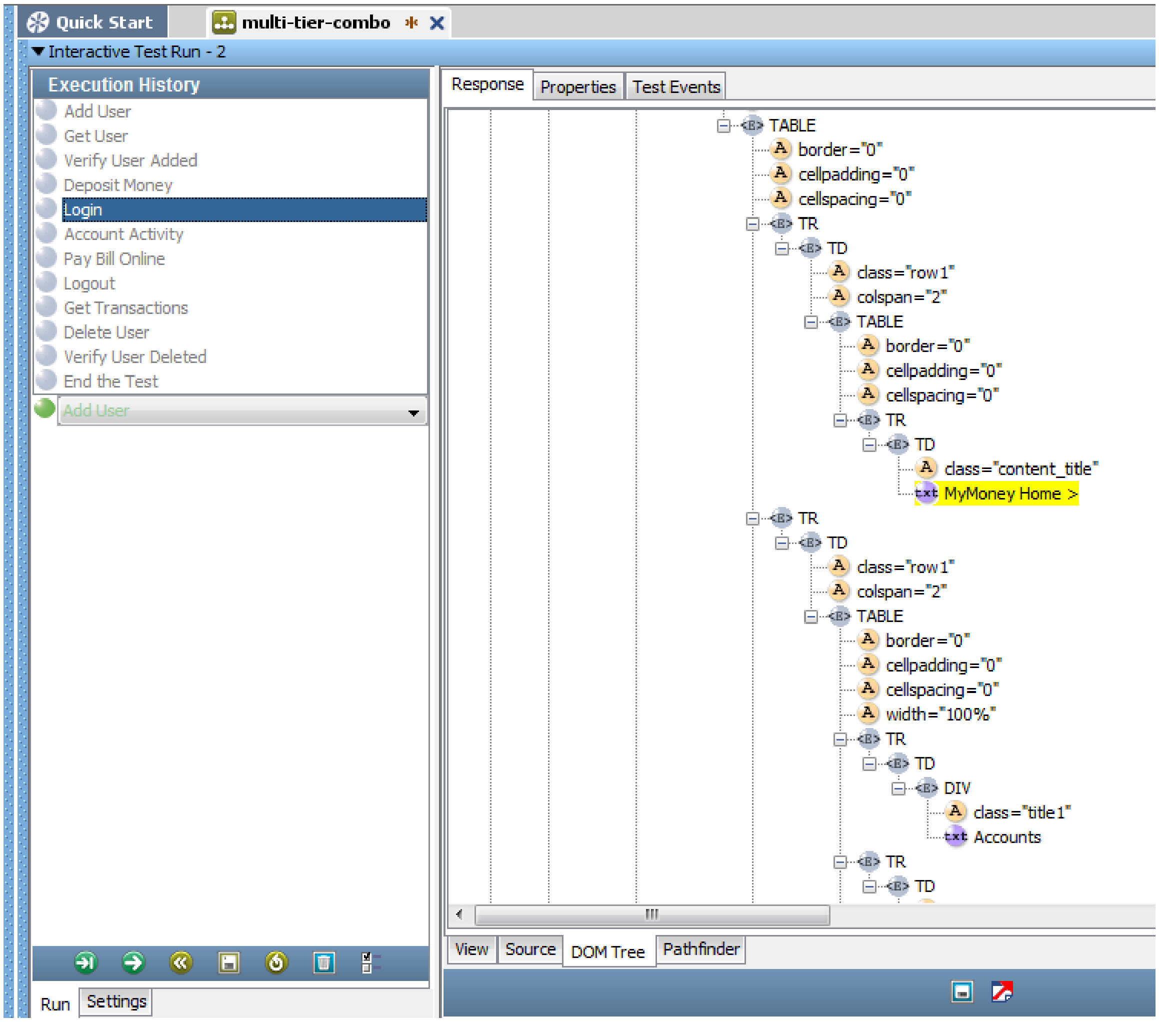
This example uses the response of the login step in the multi-tier-combo test case in the examples directory (multi-tier-combo.tst). The point of this example is to capture the text where **MyMoney Home** currently appears on the screen. (It will not always be the same text).

1. Run the multi-tier-combo test case in the ITR, then select the Login test step.



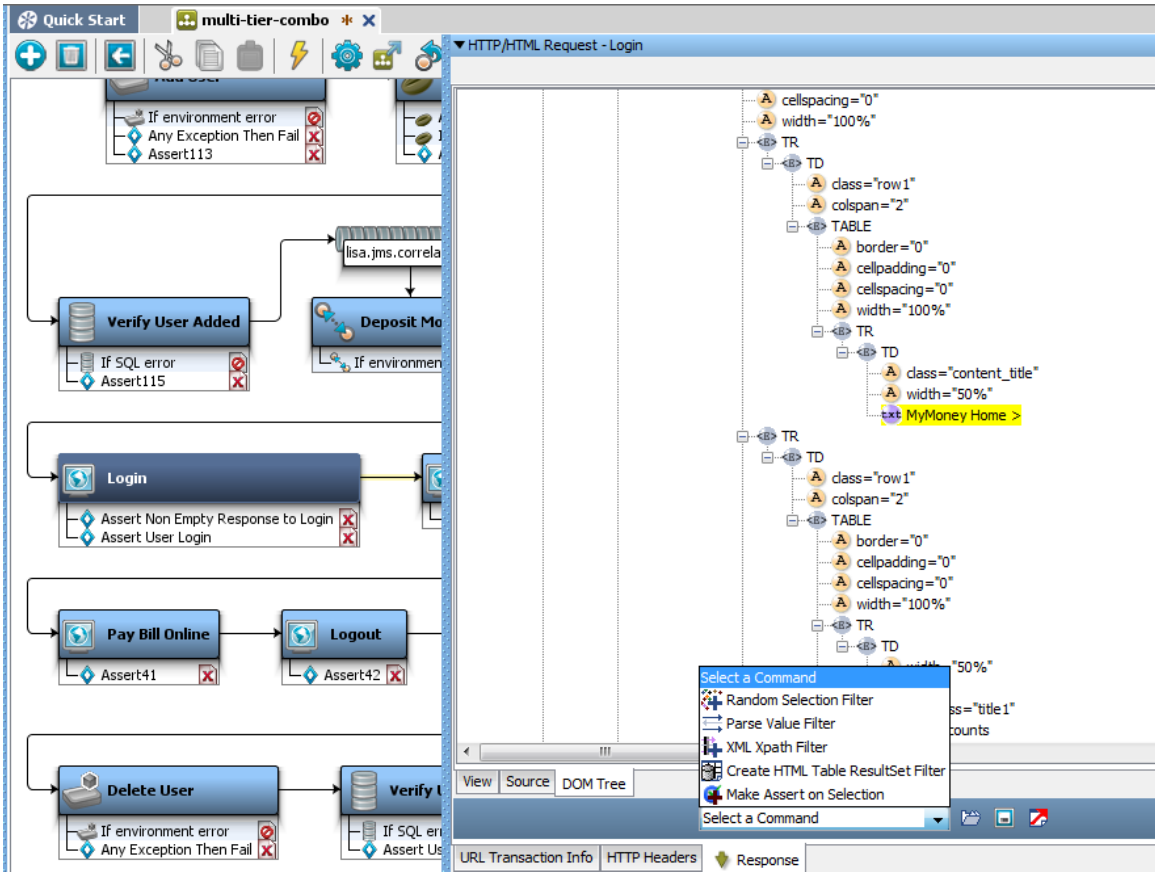
Account Number	Account Type	Name
25792921329	CHECKING	Basic Checking
25792921774	SAVINGS	Orange Savings
Total		

2. Select the text **MyMoney Home** in the View tab and click the DOM Tree tab to make sure that this text is selected in the tree view.

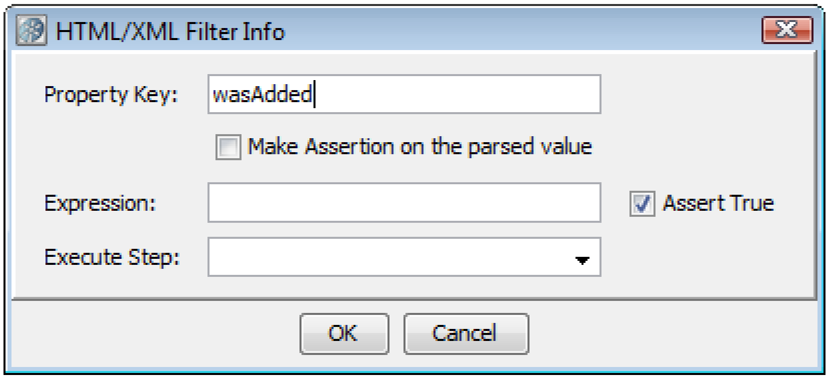


```
TABLE
  border="0"
  cellpadding="0"
  cellspacing="0"
  TR
    TD
      class="row1"
      colspan="2"
      TABLE
        border="0"
        cellpadding="0"
        cellspacing="0"
        TR
          TD
            class="content_title"
            MyMoney Home >
  TR
    TD
      class="row1"
      colspan="2"
      TABLE
        border="0"
        cellpadding="0"
        cellspacing="0"
        width="100%"
        TR
          TD
            DIV
              class="title1"
              Accounts
        TR
          TD
```

3. To apply an inline filter, double-click the login step in the model editor to open the HTTP/HTML Step Editor.



4. Move to the DOM Tree tab, find **MyMoney Home** in the DOM Tree view, and select it.
5. At the bottom of the screen, in the Select a Command box, select Parse Value Filter from the drop-down menu.
6. In the dialog window that is displayed, enter the name for the Property Key "wasAdded":


HTML/XML Filter Info

Property Key: wasAdded

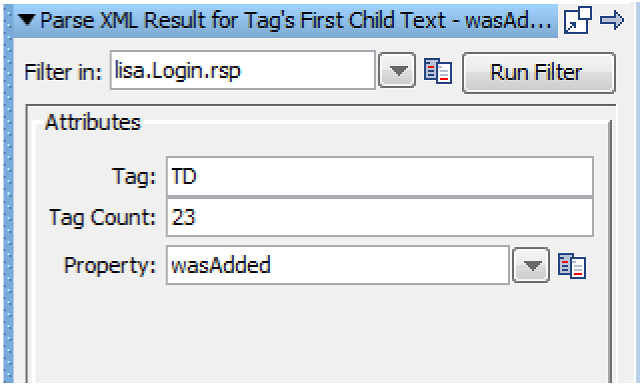
☐ Make Assertion on the parsed value

Expression:

☑ Assert True

Execute Step:

7. Click OK.


Parse XML Result for Tag's First Child Text - wasAd...

Filter in: lisa.Login.rsp

Attributes

Tag: TD

Tag Count: 23

Property: wasAdded

You can also add an assertion here. For example, you would probably want to test the value of the property "wasAdded," to see if it is in fact equal to Added user. More information is available in Adding Assertions.

The filter that was generated can be seen as a filter in the login test step.

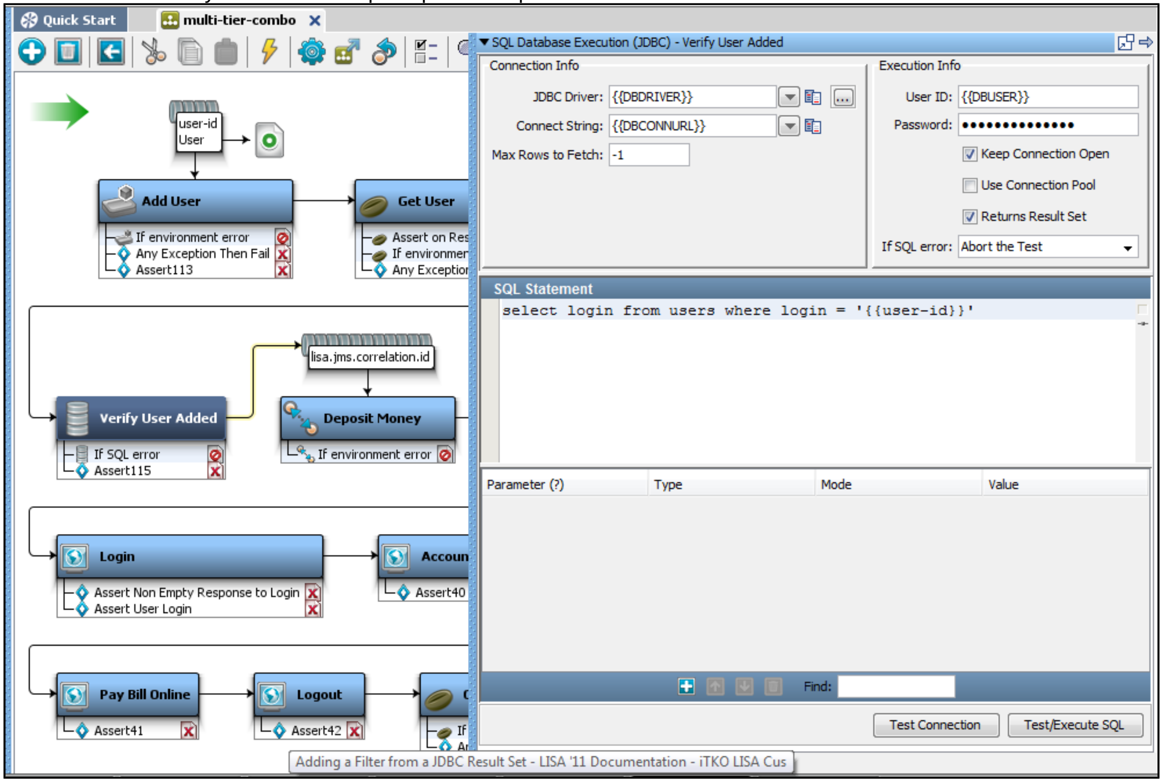
> ✓ The same filtering capabilities are available when an HTML response is displayed in the step editor.

Adding a Filter from a JDBC Result Set

When you have access to the result set response from a JDBC step, you can use the response to add a filter directly.

Here is an example of the JDBC Result Set response, using the response of the Verify User Added step in the multi-tier-combo test case in the examples directory (multi-tier-combo.tst).

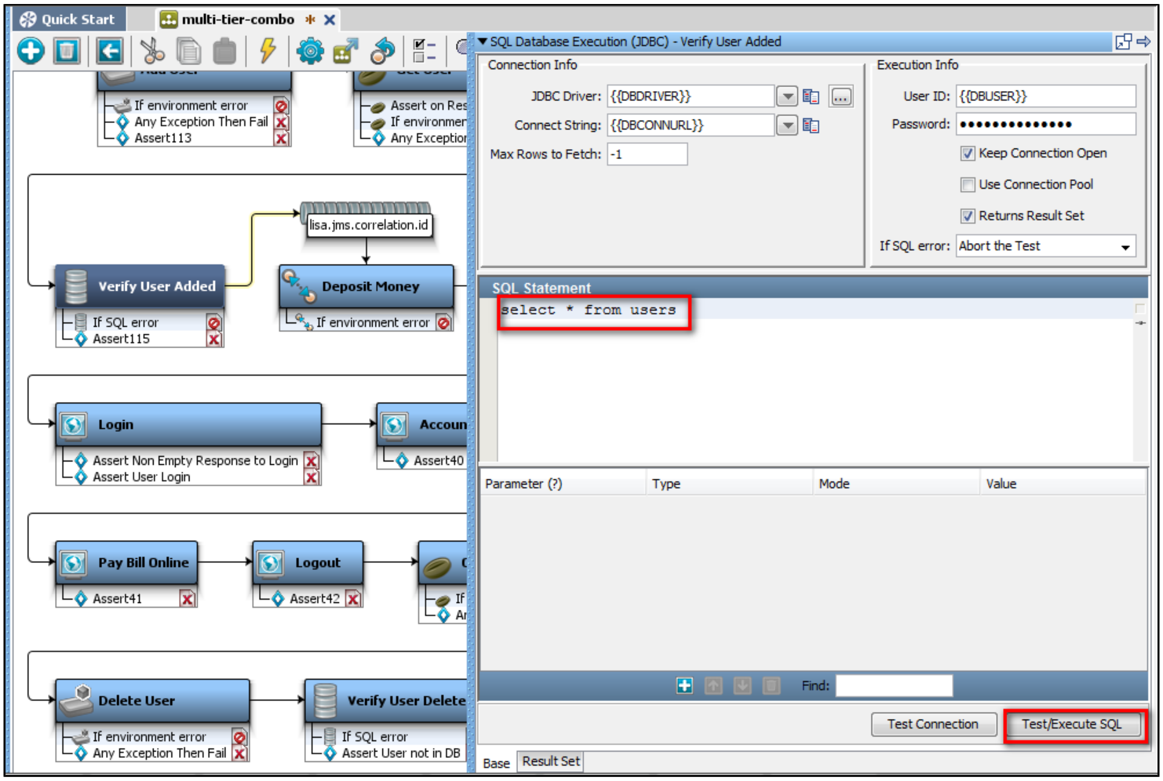
1. Double-click the Verify User Added step to open its step editor.



JDBC Driver: {{DBDRIVER}}
Connect String: {{DBCONNURL}}
Max Rows to Fetch: -1
User ID: {{DBUSER}}
Password: ••••••••••••••
☑ Keep Connection Open
☐ Use Connection Pool
☑ Returns Result Set
If SQL error: Abort the Test

```
select login from users where login = '{{user-id}}'
```

2. Edit the SQL statement to read **Select * from users** and click the Test/Execute SQL button to get values in the result set.



SQL Database Execution (JDBC) - Verify User Added

Connection Info

JDBC Driver: {{DBDRIVER}}

Connect String: {{DBCONNURL}}

Max Rows to Fetch: -1

Execution Info

User ID: {{DBUSER}}

Password: ••••••••••••••

☑ Keep Connection Open

□ Use Connection Pool

☑ Returns Result Set

If SQL error: Abort the Test

SQL Statement

```
select * from users
```

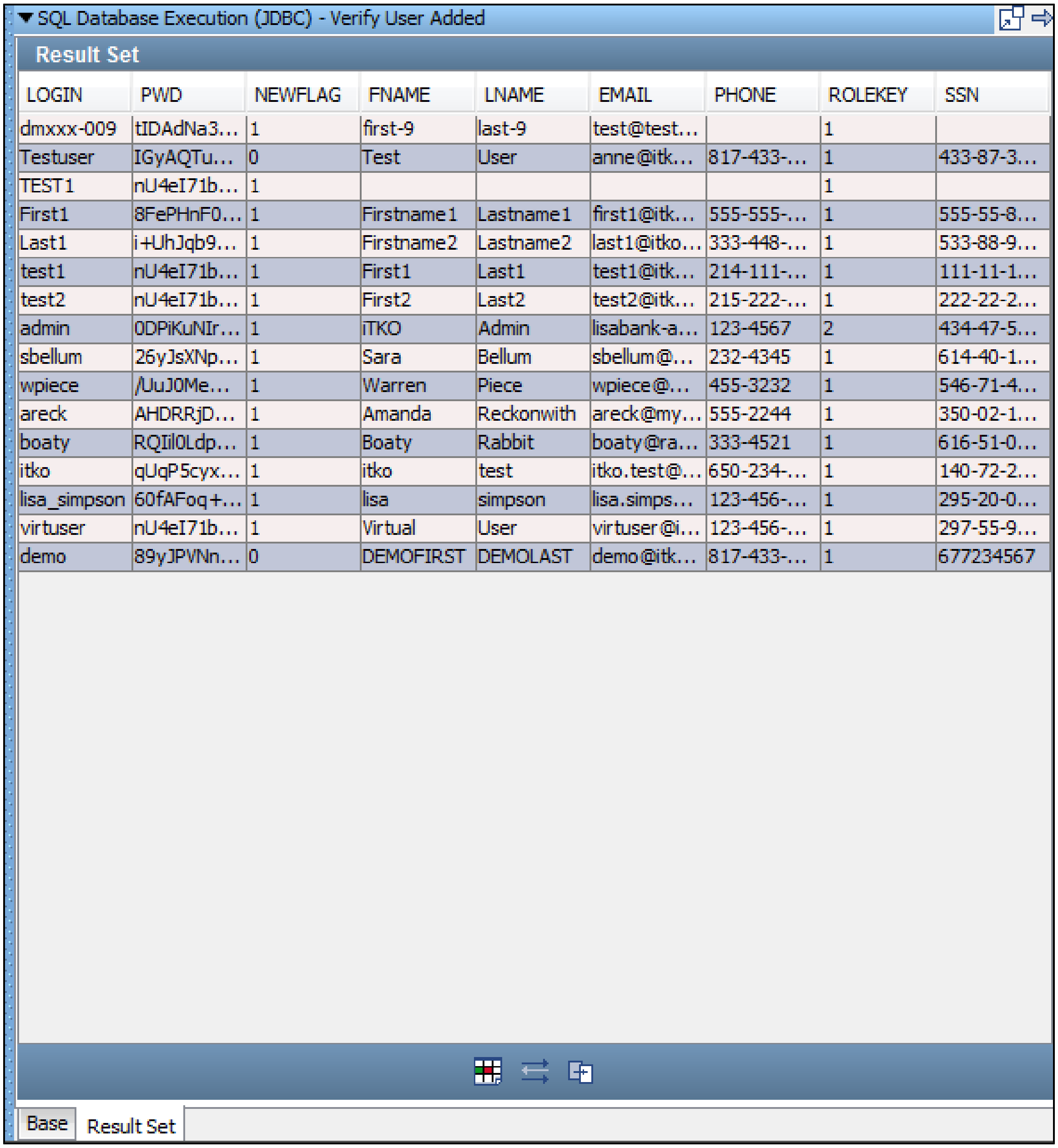
Parameter (?)	Type	Mode	Value

Find:

Test Connection | Test/Execute SQL

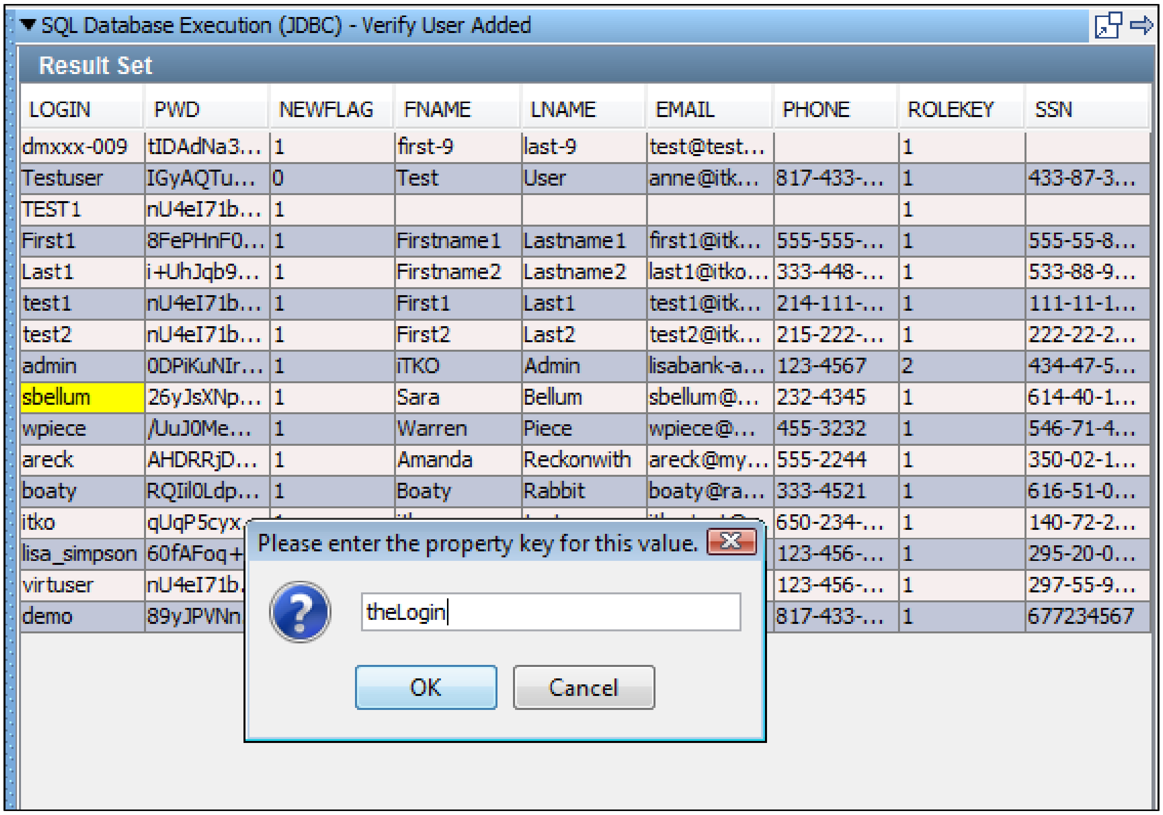
Base | Result Set

3. Click the Result Set tab and click the Test/Execute SQL button to get values in the result set.



LOGIN	PWD	NEWFLAG	FNAME	LNAME	EMAIL	PHONE	ROLEKEY	SSN
dmxxx-009	tIDAdNa3...	1	first-9	last-9	test@test...		1	
Testuser	IGyAQTu...	0	Test	User	anne@itk...	817-433-...	1	433-87-3...
TEST1	nU4eI71b...	1					1	
First1	8FePHnF0...	1	Firstname1	Lastname1	first1@itk...	555-555-...	1	555-55-8...
Last1	i+UhJqb9...	1	Firstname2	Lastname2	last1@itko...	333-448-...	1	533-88-9...
test1	nU4eI71b...	1	First1	Last1	test1@itk...	214-111-...	1	111-11-1...
test2	nU4eI71b...	1	First2	Last2	test2@itk...	215-222-...	1	222-22-2...
admin	0DPiKuNIr...	1	iTKO	Admin	lisabank-a...	123-4567	2	434-47-5...
sbellum	26yJsXNp...	1	Sara	Bellum	sbellum@...	232-4345	1	614-40-1...
wpiece	/UuJ0Me...	1	Warren	Piece	wpiece@...	455-3232	1	546-71-4...
areck	AHDRRjD...	1	Amanda	Reckonwith	areck@my...	555-2244	1	350-02-1...
boaty	RQIil0Ldp...	1	Boaty	Rabbit	boaty@ra...	333-4521	1	616-51-0...
itko	qUqP5cyx...	1	itko	test	itko.test@...	650-234-...	1	140-72-2...
lisa_simpson	60fAFoq+...	1	lisa	simpson	lisa.simps...	123-456-...	1	295-20-0...
virtuser	nU4eI71b...	1	Virtual	User	virtuser@i...	123-456-...	1	297-55-9...
demo	89yJPVNn...	0	DEMOFIRST	DEMOLAST	demo@itk...	817-433-...	1	677234567

4. Click the cell in the result set tab that represents the location of the information you want to capture (**sbellum**).
5. Click the Generate Filter for Current Col/Row Value ⇄ icon .
6. In the dialog that opens, enter the property key **theLogin**.

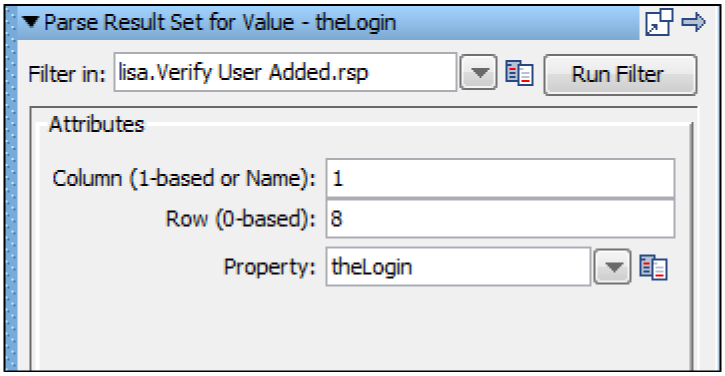


LOGIN	PWD	NEWFLAG	FNAME	LNAME	EMAIL	PHONE	ROLEKEY	SSN
dmxxx-009	tIDAdNa3...	1	first-9	last-9	test@test...		1	
Testuser	IGyAQTu...	0	Test	User	anne@itk...	817-433-...	1	433-87-3...
TEST1	nU4eI71b...	1					1	
First1	8FePHnF0...	1	Firstname1	Lastname1	first1@itk...	555-555-...	1	555-55-8...
Last1	i+UhJqb9...	1	Firstname2	Lastname2	last1@itko...	333-448-...	1	533-88-9...
test1	nU4eI71b...	1	First1	Last1	test1@itk...	214-111-...	1	111-11-1...
test2	nU4eI71b...	1	First2	Last2	test2@itk...	215-222-...	1	222-22-2...
admin	0DPiKuNIr...	1	iTKO	Admin	lisabank-a...	123-4567	2	434-47-5...
sbellum	26yJsXNp...	1	Sara	Bellum	sbellum@...	232-4345	1	614-40-1...
wpiece	/UuJ0Me...	1	Warren	Piece	wpiece@...	455-3232	1	546-71-4...
areck	AHDRRjD...	1	Amanda	Reckonwith	areck@my...	555-2244	1	350-02-1...
boaty	RQIil0Ldp...	1	Boaty	Rabbit	boaty@ra...	333-4521	1	616-51-0...
itko	qUqP5cyx					650-234-...	1	140-72-2...
lisa_simpson	60fAFoq+					123-456-...	1	295-20-0...
virtuser	nU4eI71b.					123-456-...	1	297-55-9...
demo	89yJPVNn					817-433-...	1	677234567

Please enter the property key for this value.

theLogin

7. Click OK. LISA will add a filter named Parse Result Set for Value in the list user step.
8. Click the filter editor to see the filter.



Filter in: lisa.Verify User Added.rsp

Column (1-based or Name): 1

Row (0-based): 8

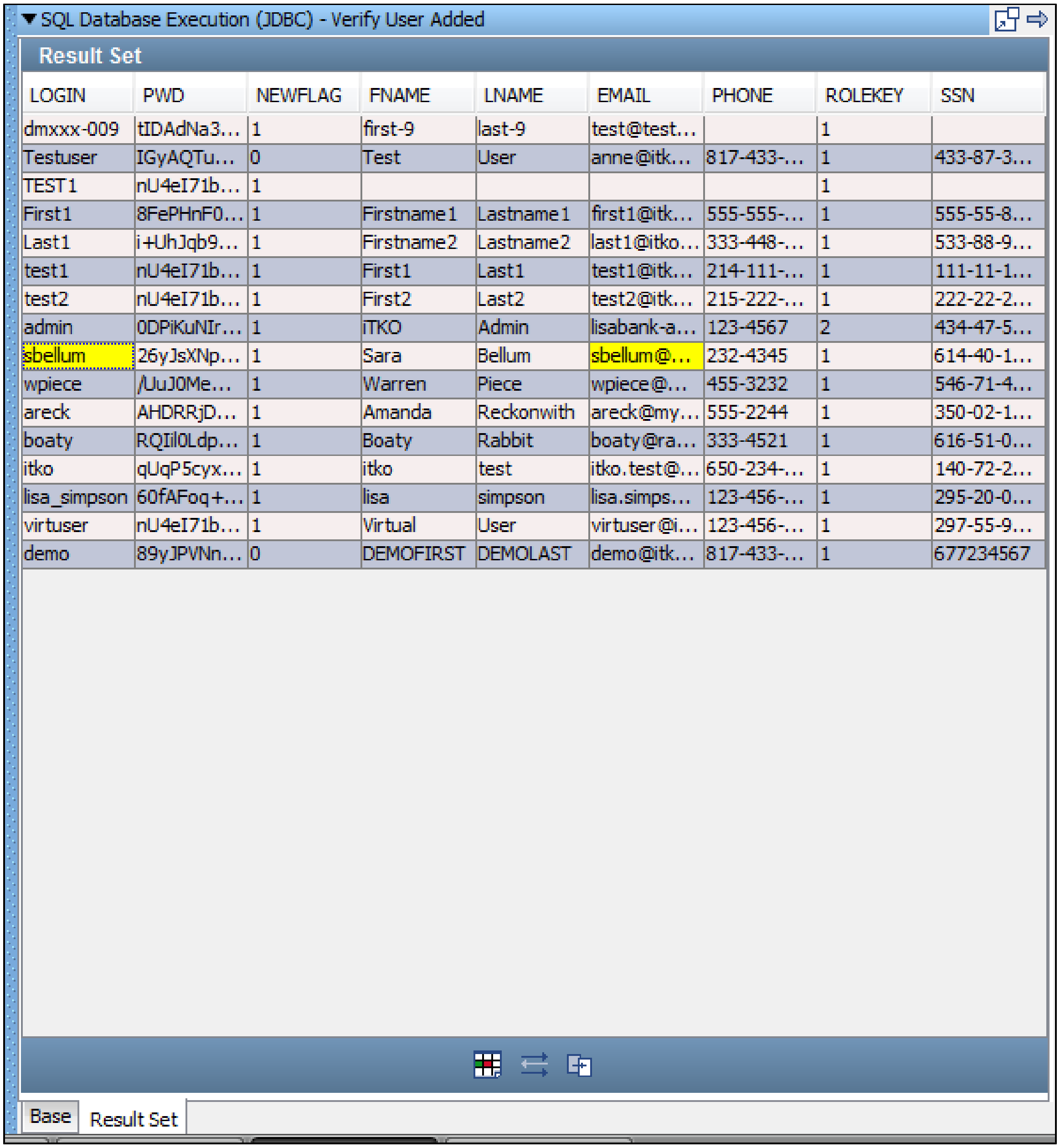
Property: theLogin

In the example, the value in the cell in the 1st column, and 2nd row, **sbellum**, will be stored in the property theLogin.

Applying a Second Filter

There is a second filter that can be applied here. You can look for a value in one column of the result set, and then capture a value from another column in the same row.

From within the result set, select the two values in two different columns from the same row, using the Ctrl key.

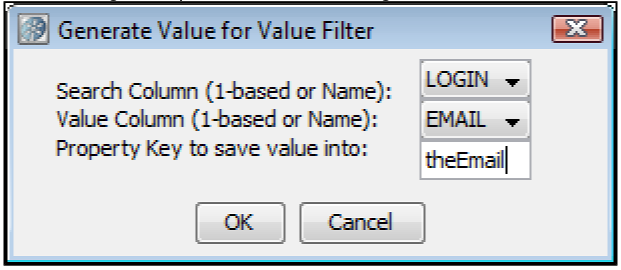


SQL Database Execution (JDBC) - Verify User Added

Result Set

LOGIN	PWD	NEWFLAG	FNAME	LNAME	EMAIL	PHONE	ROLEKEY	SSN
dmxxx-009	tIDAdNa3...	1	first-9	last-9	test@test...		1	
Testuser	IGyAQTu...	0	Test	User	anne@itk...	817-433-...	1	433-87-3...
TEST1	nU4eI71b...	1					1	
First1	8FePHnF0...	1	Firstname1	Lastname1	first1@itk...	555-555-...	1	555-55-8...
Last1	i+UhJqb9...	1	Firstname2	Lastname2	last1@itko...	333-448-...	1	533-88-9...
test1	nU4eI71b...	1	First1	Last1	test1@itk...	214-111-...	1	111-11-1...
test2	nU4eI71b...	1	First2	Last2	test2@itk...	215-222-...	1	222-22-2...
admin	0DPiKuNIr...	1	iTKO	Admin	lisabank-a...	123-4567	2	434-47-5...
sbellum	26yJsXNp...	1	Sara	Bellum	sbellum@...	232-4345	1	614-40-1...
wpiece	/UuJ0Me...	1	Warren	Piece	wpiece@...	455-3232	1	546-71-4...
areck	AHDRRjD...	1	Amanda	Reckonwith	areck@my...	555-2244	1	350-02-1...
boaty	RQIil0Ldp...	1	Boaty	Rabbit	boaty@ra...	333-4521	1	616-51-0...
itko	qUqP5cyx...	1	itko	test	itko.test@...	650-234-...	1	140-72-2...
lisa_simpson	60fAFoq+...	1	lisa	simpson	lisa.simps...	123-456-...	1	295-20-0...
virtuser	nU4eI71b...	1	Virtual	User	virtuser@i...	123-456-...	1	297-55-9...
demo	89yJPVNn...	0	DEMOFIRST	DEMOLAST	demo@itk...	817-433-...	1	677234567

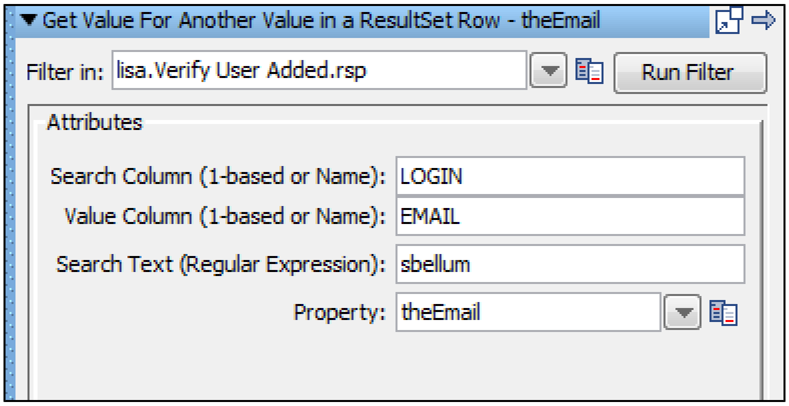
1. Select Filter for a value and then get another column value filter using the ▦ icon. Select two cells in the same row to create this filter. One will be the search column and the other will be the column whose value you want to extract.
2. In the dialog that opens, check or reassign the columns for the search and the value, then enter the property key **theEmail**.



Generate Value for Value Filter

Search Column (1-based or Name): LOGIN
Value Column (1-based or Name): EMAIL
Property Key to save value into: theEmail

3. Click OK. LISA will add a filter named **Get Value For Another Value in a ResultSet Row** in the Verify User Added step.



Get Value For Another Value in a ResultSet Row - theEmail

Filter in: lisa.Verify User Added.rsp

Attributes

Search Column (1-based or Name): LOGIN

Value Column (1-based or Name): EMAIL

Search Text (Regular Expression): sbellum

Property: theEmail

This filter looks for **sbellum** in the LOGIN column, and if found, stores the value in the EMAIL column in the same row in a property named **theEmail**.

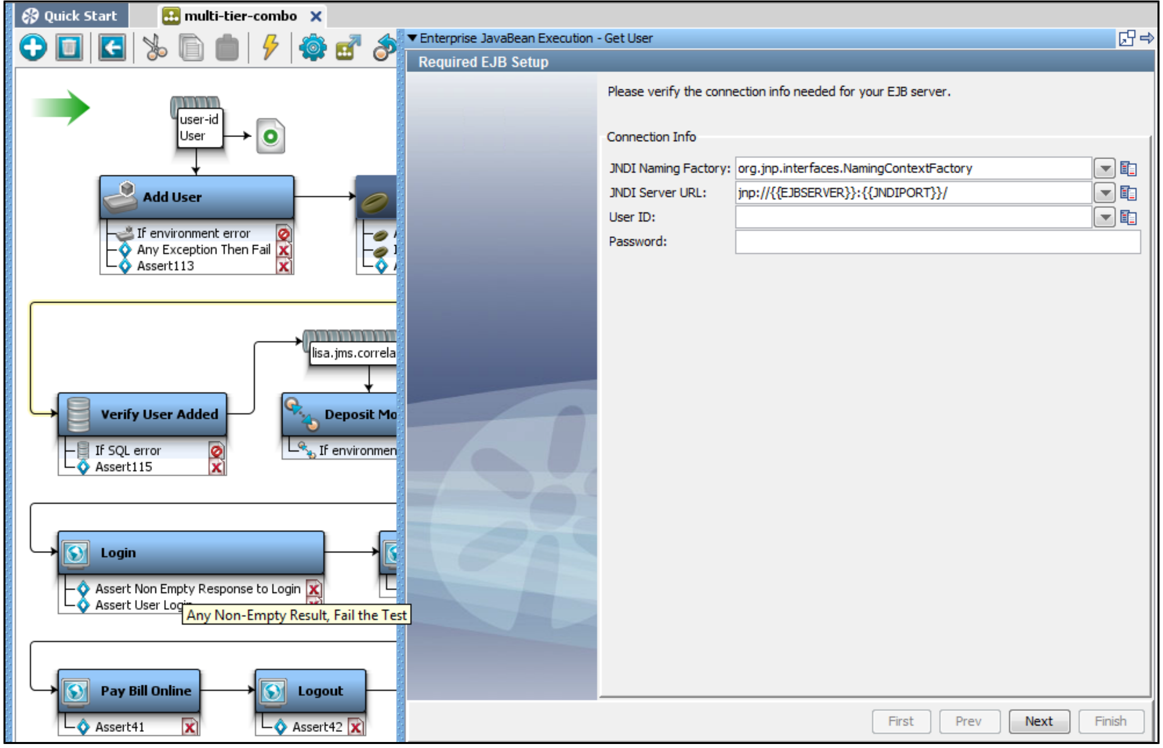
> ⚠ The same filtering capabilities are available when a JDBC result set is displayed in the Step Editor.

Adding a Filter from a Returned Java Object

When the result of your test step is a Java object, you can use the Inline Filter panel in the Complex Object Editor to filter the returned value from the method call directly. Following is an example of how to add a filter this way.

This example uses the get user step (EJB step) in the multi-tier-combo test case in the examples directory.

1. Double-click the get user step in the workflow, to open its step editor.



Enterprise JavaBean Execution - Get User

Required EJB Setup

Please verify the connection info needed for your EJB server.

Connection Info

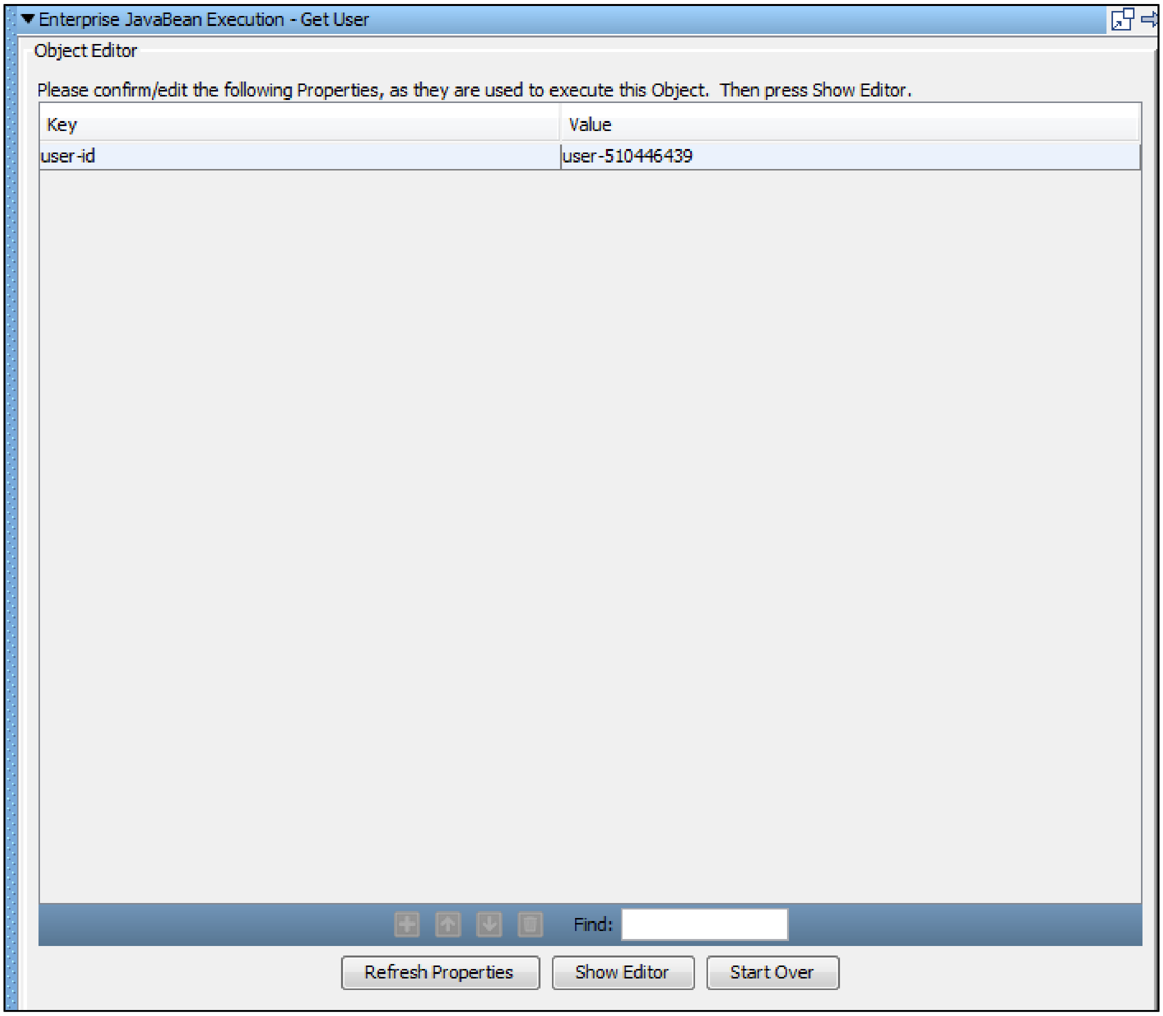
JNDI Naming Factory: org.jnp.interfaces.NamingContextFactory

JNDI Server URL: jnp://{{EJBSERVER}}:{{JNDIPORT}}/

User ID:

Password:

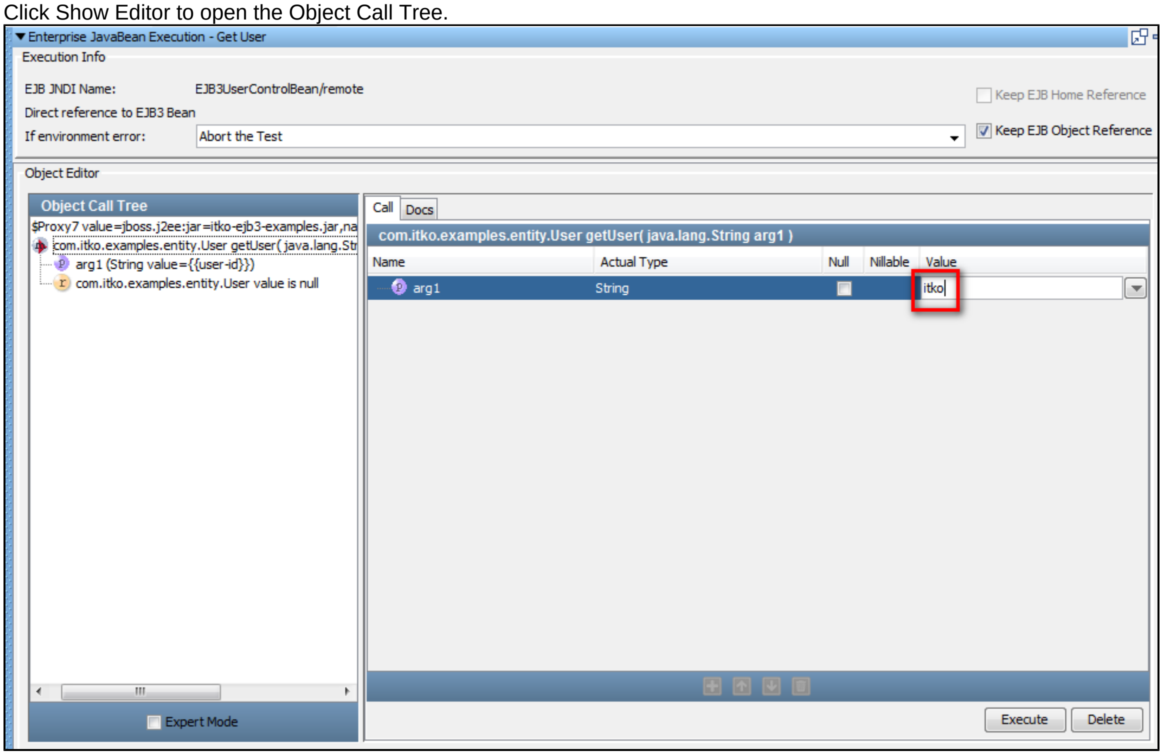
2. Click Next. On the next screen, click Finish.



Please confirm/edit the following Properties, as they are used to execute this Object. Then press Show Editor.

Key	Value
user-id	user-510446439

3. Click Show Editor to open the Object Call Tree.



EJB JNDI Name: EJB3UserControlBean/remote

Direct reference to EJB3 Bean

If environment error: Abort the Test

□ Keep EJB Home Reference

☑ Keep EJB Object Reference

```
$Proxy7 value=jboss.j2ee:jar=itko-ejb3-examples.jar,na
  com.itko.examples.entity.User getUser( java.lang.Str
    arg1 (String value={{user-id}})
    com.itko.examples.entity.User value is null
```

com.itko.examples.entity.User getUser(java.lang.String arg1)

Name	Actual Type	Null	Nillable	Value
arg1	String	□		itko

4. Enter an input parameter "itko" in the value field.
5. Click Execute to execute this method.

The returned value upon executing the getLogin method will be stored in the property getUserObject. Notice that in this case the returned value is an object (of type UserState). You also can add an assertion here.

In this example, you could also call a method on the returned object to get the actual login value for this user, and save the login in another property.

> ⚠ Inline filters (and assertions) do not result in a filter being added to the test step in the element tree. Inline filter management is always done in the Complex Object Editor.

For more details on the Complex Object Editor, see Using Complex Object Editor.

Deleting a Filter

To delete a filter

- Select the filter in the Filter Elements tab and right-click to open a menu. Click Delete to delete the filter. Or,
- Select the filter in the Filter Elements tab and click the Delete ✘ icon on the toolbar.

You cannot delete a test step that has filters attached to it.

Reordering a Filter

To reorder a filter

1. Select the filter in the Filter Elements tab.
2. Click the Move up ↑ or Move down ↓ icon on the toolbar.

Dragging and Dropping a Filter

You can drag and drop filters in the model editor from one test step to another.

To drag and drop a filter

1. Click the filter attached to a test step; for example, Step1.
2. Drag and drop that filter to another test step in the model editor; for example, Step2.
3. The dragged filter will then be applied to Step2.

Types of Filters

This section describes each of the filters that are available in LISA.

Regular expressions are used for comparison purposes in several filters. For more information about regular expressions, see http://psoug.org/reference/regexp.html.

The following filters are available.

Utility Filters
Database Filters
Messaging_ESB Filters
HTTP_HTML Filters
XML Filters
Web 2.0 Filters
Java Filters
VSE Filters
Pathfinder Filters

Utility Filters

These are the filters available in the Utility Filters list for any test step.

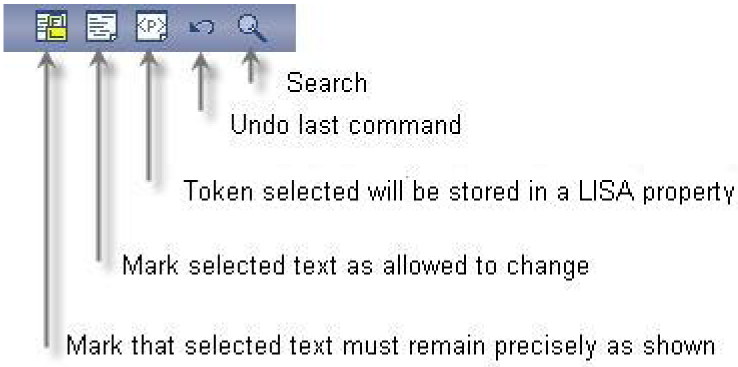
Create Property Based on Surrounding Values

The **Create Property Based on Surrounding Values** filter lets you read textual content and filter it for information to be stored in LISA properties. It can be used on text, and on XML and HTML treated as text. It uses a "paint the screen" technique.

"Paint the screen" gives you great flexibility to define what in the text buffer you want to parse out as properties. There are three ways to mark the text:

- Text that must appear in the response precisely as shown: a **Must** block.
- Text that does not have to appear in the response, or can change: an **Any** block.
- Text that will be stored in a LISA property: a **Property** block. Property blocks must always be bounded by **Must** blocks.

The text is marked using the icons at the bottom of the editor:


Search
Undo last command
Token selected will be stored in a LISA property
Mark selected text as allowed to change
Mark that selected text must remain precisely as shown

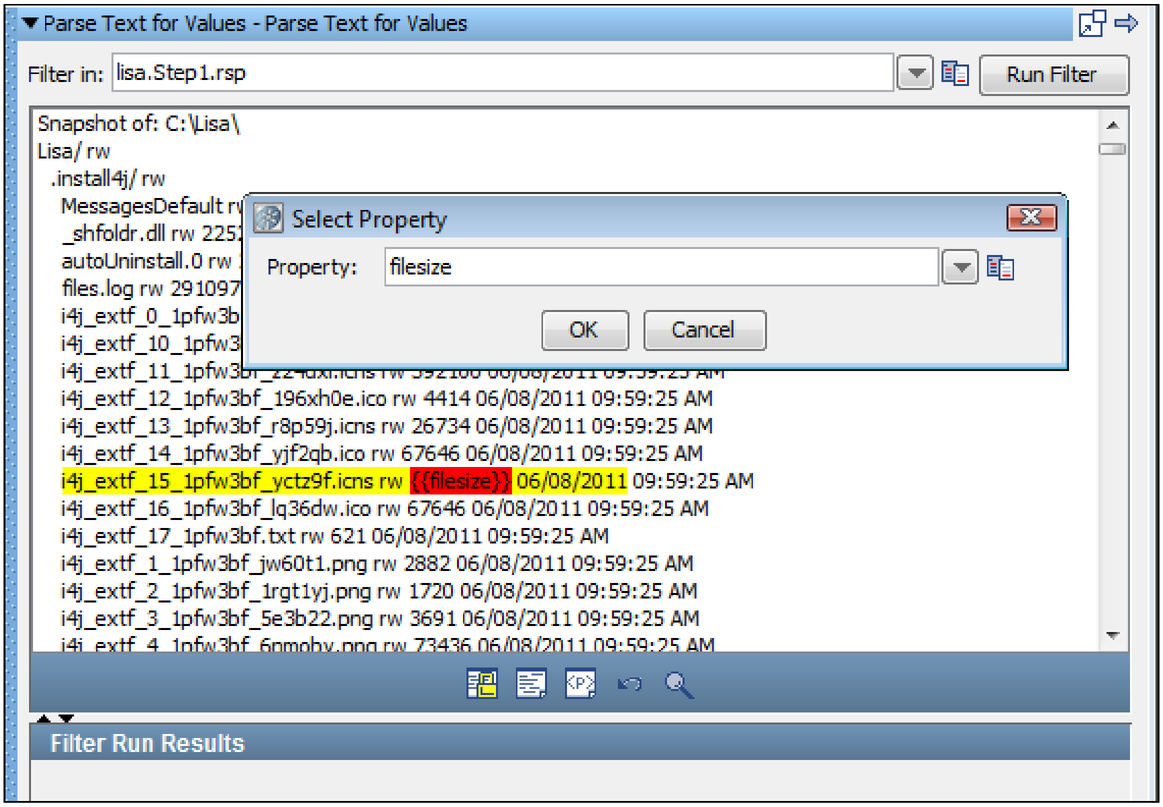
This technique is best explained by example.

In the following example, the goal is to store the size of a particular file in a property.

The text is marked using the editor icons, by selecting text and then clicking the appropriate icon.

- Yellow background indicates text that must appear as shown (indicated by the arrows from "Text uses ...."). This is a **Must** block and is marked using the **Must** icon.
- Red background identifies the text that will be stored in the property entered into the dialog (indicated by a single arrow). This is a **Property** block and is marked using the **Property** icon.

⚠ **Property** blocks must always be bounded by **Must** blocks.



Parse Text for Values - Parse Text for Values
Filter in: lisa.Step1.rsp
Run Filter
Select Property
Property: filesize
OK
Cancel
Filter Run Results

```
Snapshot of: C:\Lisa\
Lisa/ rw
 .install4j/ rw
  MessagesDefault r
  _shfoldr.dll rw 225
  autoUninstall.0 rw
  files.log rw 291097
  i4j_extf_0_1pfw3b
  i4j_extf_10_1pfw3
  i4j_extf_11_1pfw3
  i4j_extf_12_1pfw3bf_196xh0e.ico rw 4414 06/08/2011 09:59:25 AM
  i4j_extf_13_1pfw3bf_r8p59j.icns rw 26734 06/08/2011 09:59:25 AM
  i4j_extf_14_1pfw3bf_yjf2qb.ico rw 67646 06/08/2011 09:59:25 AM
  i4j_extf_15_1pfw3bf_yctz9f.icns rw {{filesize}} 06/08/2011 09:59:25 AM
  i4j_extf_16_1pfw3bf_lq36dw.ico rw 67646 06/08/2011 09:59:25 AM
  i4j_extf_17_1pfw3bf.txt rw 621 06/08/2011 09:59:25 AM
  i4j_extf_1_1pfw3bf_jw60t1.png rw 2882 06/08/2011 09:59:25 AM
  i4j_extf_2_1pfw3bf_1rgt1yj.png rw 1720 06/08/2011 09:59:25 AM
  i4j_extf_3_1pfw3bf_5e3b22.png rw 3691 06/08/2011 09:59:25 AM
  i4j_extf_4_1pfw3bf_6nmoby.png rw 73436 06/08/2011 09:59:25 AM
```

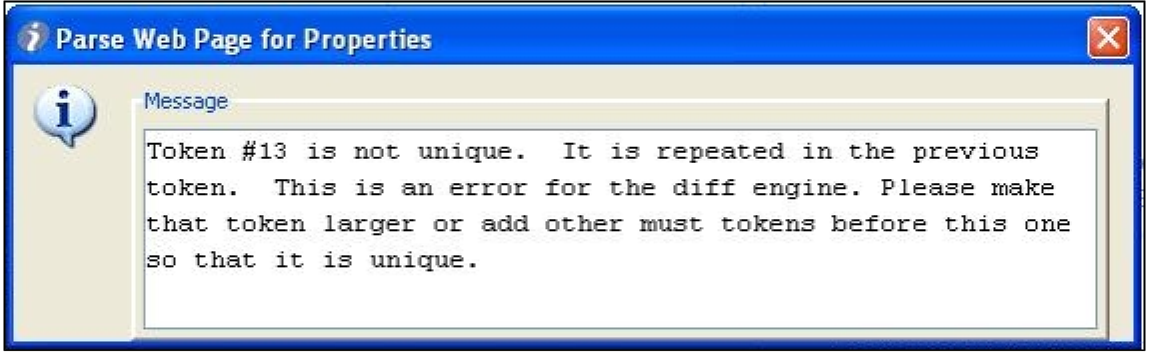
This screen shows the contents of the text buffer. The goal is to parse out the file size of the Simulator.exe file. The file size is the number that appears after "Simulator.exe". The boundaries are set around the file size, and the **Must** icon has been clicked. The actual file size text inside the selected content was selected, and the **Property** icon was clicked. The property name was then entered into the dialog. The actual value of the file size has been replaced with the name of the LISA property.

When this filter is run, the property "filesize" will be assigned the size of the Simulator.exe file.

You can repeat this process on this text buffer to have as many properties defined as you like.

Handling Non-unique Tokens

If you see the following error message, your selected token is not unique; the selection you just made is repeated in the token before it.

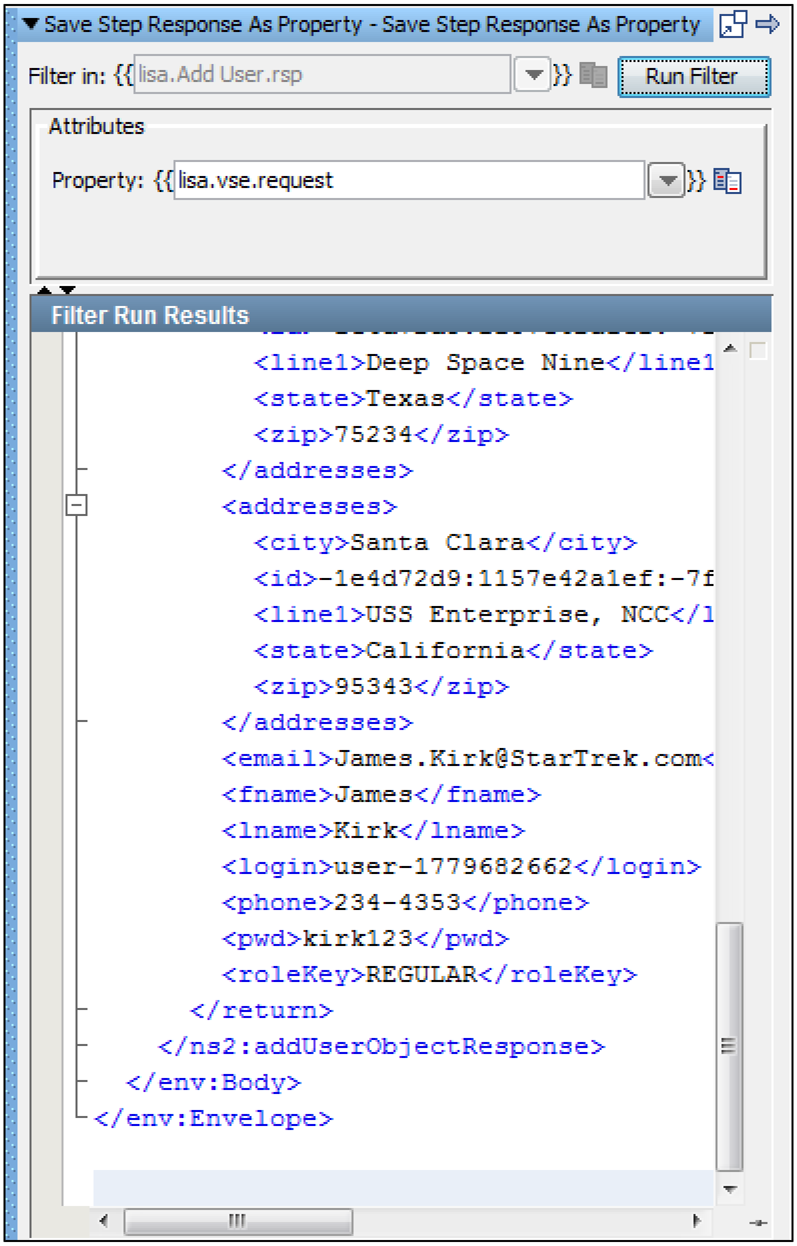


Token #13 is not unique. It is repeated in the previous token. This is an error for the diff engine. Please make that token larger or add other must tokens before this one so that it is unique.

To solve this in most cases, simply create another token to make the prior token a **Must** token also. In other cases, when this does not work, a judicious placement of another **Must** block between the two duplicate tokens will avoid the error. This will work because LISA can distinguish between the two duplicate tokens based on their relative location.

Store Step Response

(Also known as Save Step Response as Property)
The **Store Step Response** filter lets you save the last response as a property, for future use.


Save Step Response As Property - Save Step Response As Property
Run Filter
Attributes
Filter Run Results

Filter in: {{lisa.Add User.rsp}}

Property: {{lisa.vse.request}}

```
          <line1>Deep Space Nine</line1
          <state>Texas</state>
          <zip>75234</zip>
        </addresses>
        <addresses>
          <city>Santa Clara</city>
          <id>-1e4d72d9:1157e42a1ef:-7f
          <line1>USS Enterprise, NCC</l
          <state>California</state>
          <zip>95343</zip>
        </addresses>
        <email>James.Kirk@StarTrek.com<
        <fname>James</fname>
        <lname>Kirk</lname>
        <login>user-1779682662</login>
        <phone>234-4353</phone>
        <pwd>kirk123</pwd>
        <roleKey>REGULAR</roleKey>
      </return>
    </ns2:addUserObjectResponse>
  </env:Body>
</env:Envelope>
```

Enter the following parameters:

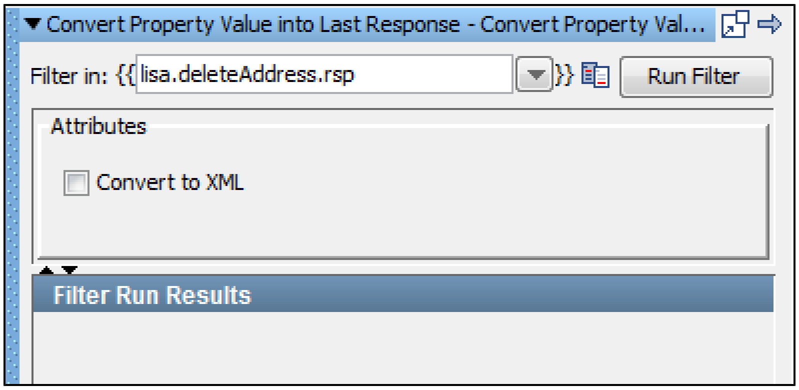
- **Filter in:** Where to apply the filter. The previous illustration shows list.Add User.rsp, which means that the filter will be applied to the response of the Add User step. You cannot change this value for this filter.
- **Property:** The name of the property to store the last response.
- **Filter Run Results:** Displays the property and values set as a result of running the filter.

Click the Run Filter button to execute the filter and see the results in the Filter Run Results.

Override "Last Response" Property

(Also known as Convert Property Value into Last Response)

The **Override "Last Response" Property** filter lets you replace the current value of the last response with the value of an existing LISA property. For example, assume you execute an EJB, but you eventually get a value back after making some method calls on the EJB. Instead of leaving the EJB object to be the last response, it may make more sense, in your test case, to make the result from one of your method calls the last response. You can first save the return value from that call as a property using a filter, and then you can use that property in this filter.



Convert Property Value into Last Response - Convert Property Val...

Filter in: {{lisa.deleteAddress.rsp}}

Attributes

□ Convert to XML

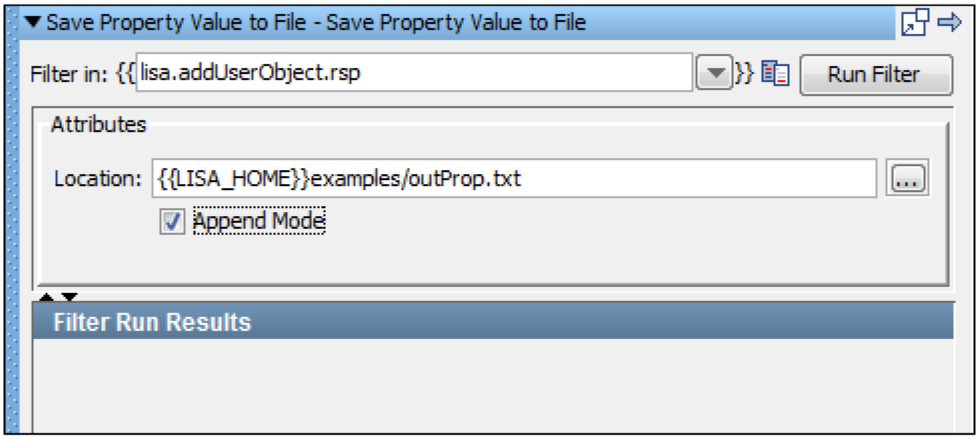
Enter the following parameters:

- **Filter in:** The name of the property you want considered as the step's last response. The property should be in the pull-down menu; if not you can enter it. It must be an existing property.
- **Convert to XML**: Select this if you want the response to be converted to valid XML.
- **Filter Run Results:** Displays the property and values set as a result of running the filter.

Click the Run Filter button to execute the filter and see the results in the Filter Run Results.

Save Property Value to File

The **Save Property Value to File** filter lets you save the value of an existing property to a file in your file system.



Save Property Value to File - Save Property Value to File

Filter in: {{lisa.addUserObject.rsp}}

Attributes

Location: {{LISA_HOME}}examples/outProp.txt

☑ Append Mode

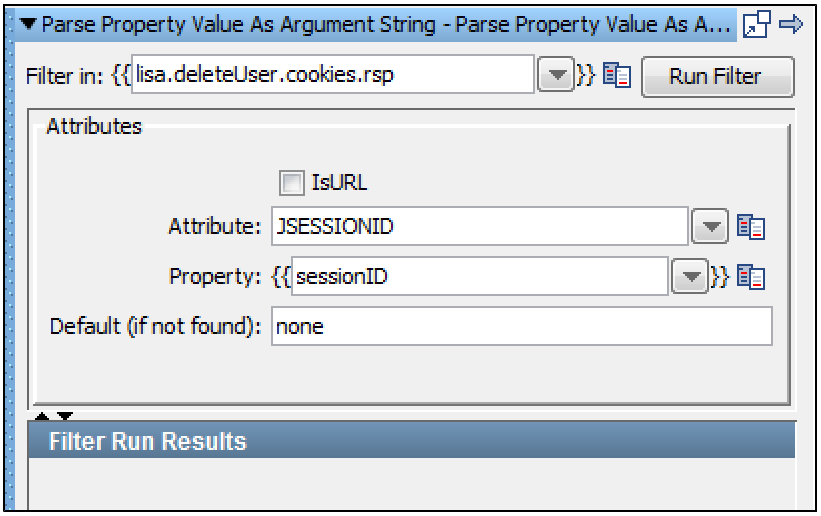
Enter the following parameters:

- **Filter in:** The name of the property whose value you want to write to the file.
- **Location:** The path name of the file to write the value to. You can browse to the file. You can use properties in the location.
- **Append Mode:** Select this check box if you want to append the information to an existing file.
- **Filter Run Results:** Displays the property and values set as a result of running the filter.

Click the Run Filter button to execute the filter and see the results in the Filter Run Results.

Parse Property Value as Argument String

The **Parse Property Value As Argument String** filter lets you store the text of a specific attribute in a property. This is very useful as a second filter, where you parse a filtered value for information.



Parse Property Value As Argument String - Parse Property Value As A...

Filter in: {{lisa.deleteUser.cookies.rsp}}

Attributes

□ IsURL

Attribute: JSESSIONID

Property: {{sessionID}}

Default (if not found): none

Enter the following parameters:

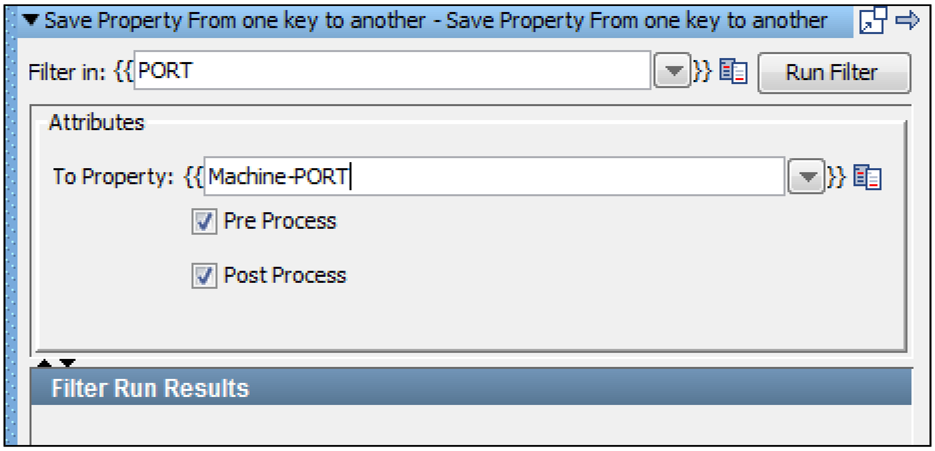
- **Filter in**: The name of the existing property to parse. For example, to parse the "lisa.deleteUser.cookies.rsp" property to return the value of the SESSIONID attribute, enter **lisa.deleteUser.cookies.rsp**.
- **IsURL**: Select this check box if the property value is a URL.
- **Attribute**: The attribute to retrieve. The example shows the JSESSIONID attribute.
- **Property**: The name of the property to store the text of the attribute. The example shows **sessionID**.
- **Default (if not found)**: The default value to use if the attribute is not found.
- **Filter Run Results:** Displays the property and values set as a result of running the filter.

Click the Run Filter button to execute the filter and see the results in the Filter Run Results.

Save Property from one key to another

The **Save Property from one key to another** filter copies values from one key to another by reference where normal Java rules apply.

This filter simply optimizes BeanShell overhead for simple property copy.



Save Property From one key to another - Save Property From one key to another

Filter in: {{PORT}}

Attributes

To Property: {{Machine-PORT}}

☑ Pre Process

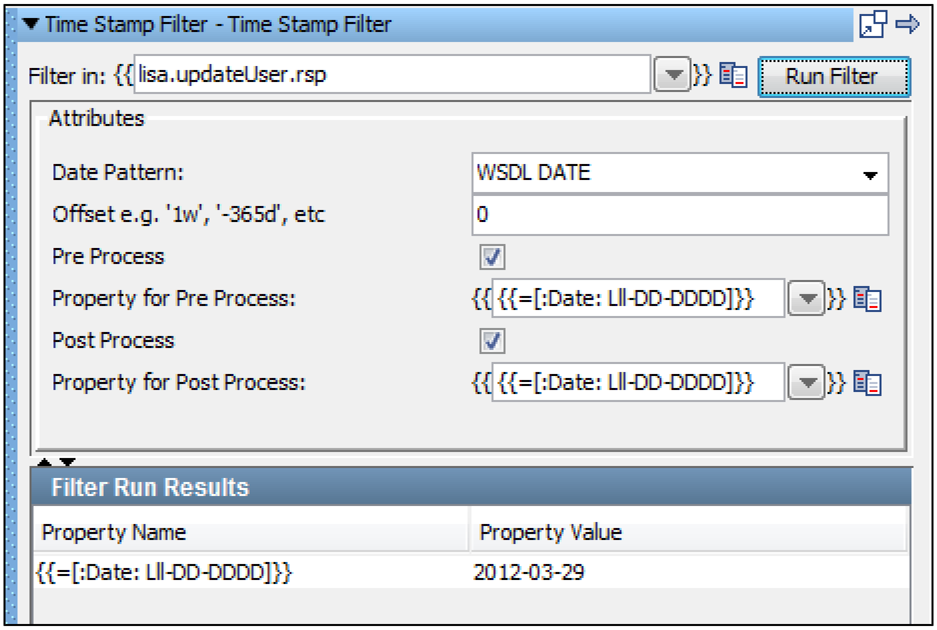
☑ Post Process

Enter the following parameters:

- **Filter in**: This field accepts the property content of which will be copied in some other property. This is the input source property.
- **To Property**: This is the property name where input property content will be copied.
- **Run Filter**: Lets you test the filter immediately while developing test steps rather than waiting until the test case is completed.

Time Stamp Filter

The **Time Stamp** filter is used to assign the current time and date to the property so that you can use this property in the following test steps.



Filter in: {{lisa.updateUser.rsp}}

Attributes

Date Pattern: WSDL DATE

Offset e.g. '1w', '-365d', etc 0

Pre Process ☑

Property for Pre Process: {{ {{=[:Date: Lll-DD-DDDD]}} }}

Post Process ☑

Property for Post Process: {{ {{=[:Date: Lll-DD-DDDD]}} }}

Filter Run Results

Property Name	Property Value
{{=[:Date: Lll-DD-DDDD]}}	2012-03-29

Enter the following parameters:

- **Filter in**: The name of the existing step.
- **Date Pattern**: Select the date pattern you want to display.
- **Offset**: Used to offset the date to an appropriate (future or past) date based on the current date.
- **Pre Process**: When enabled, generates a time stamp before the step runs.
- **Property for Pre Process**: The property to store the pre time stamp.
- **Post Process**: When enabled, generates a time stamp after the step runs.
- **Property for Post Process**: The property to store the post time stamp.
- **Filter Run Results:** Displays the property and values set as a result of running the filter.

Click the Run Filter button to execute the filter and see the results in the Filter Run Results.

Database Filters

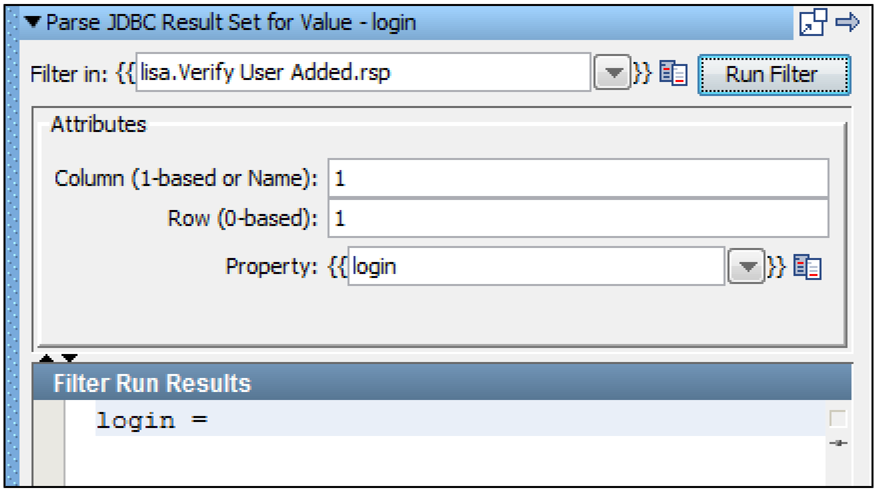
These are the filters available in the Database Filters list for any test step.

Extract Value from JDBC Result Set

The **Extract Value from JDBC Result Set** filter lets you store the text of a specific JDBC result set value in a property.

This filter can be created in two ways; either as a manual filter from the filter list or by using the embedded filter commands on a result set response.

Creating the filter manually



Filter in: {{lisa.Verify User Added.rsp}}

Column (1-based or Name): 1

Row (0-based): 1

Property: {{login}}

```
login =
```

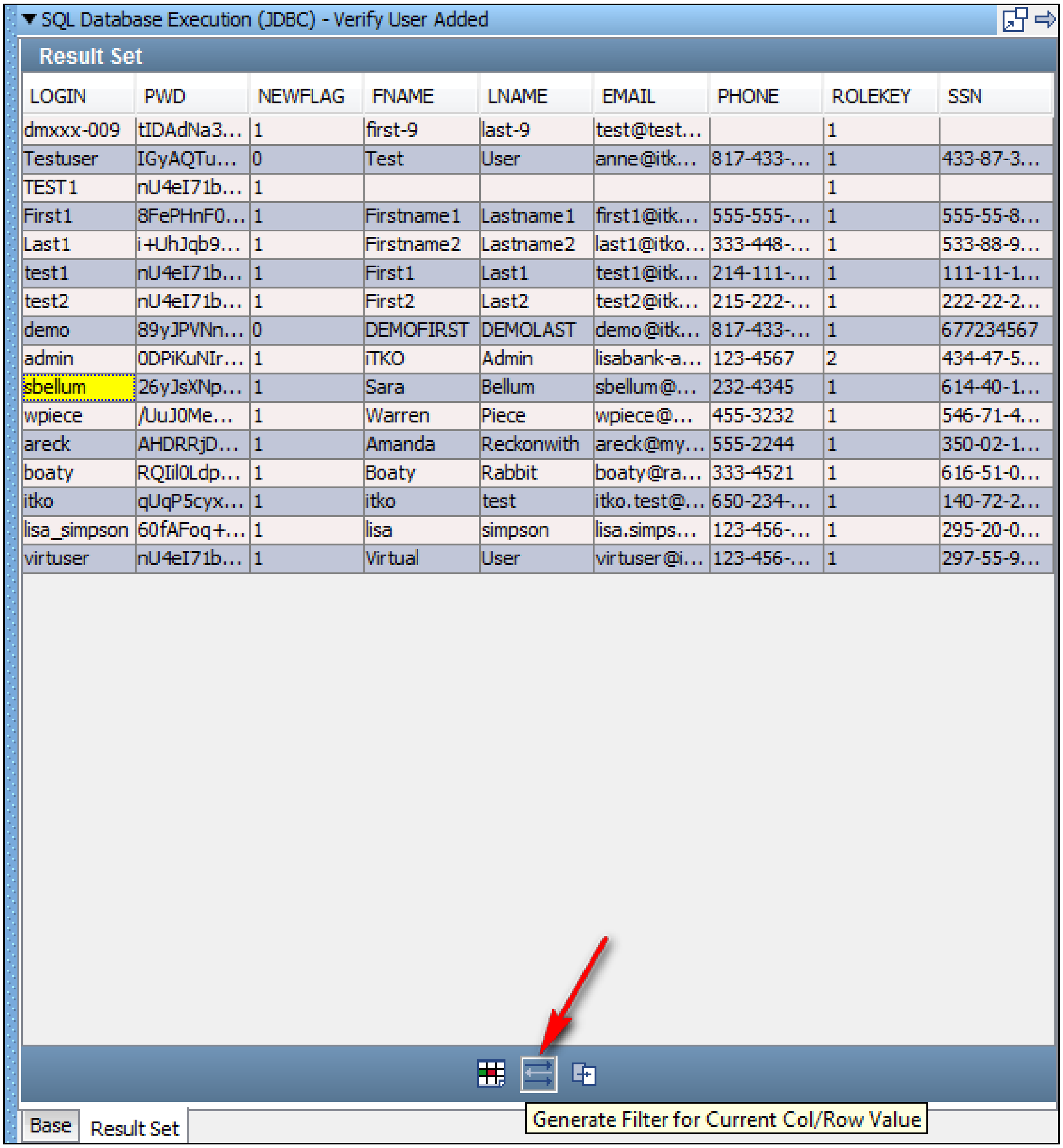
Enter the following parameters:

- **Filter in**: The name of the property you want considered as the step's last response. The property should be in the pull down menu; if not you can enter it. It must be an existing property.
- **Column (1-based or name):** The index or name of the column (field).
- **Row (0-based):** The row to retrieve the value. This is a 0-based index.
- **Property:** The name of the property where the value in the cell at the row/column intersection is stored.
- **Filter Run Results:** Displays the property and values set as a result of running the filter.

Click the Run Filter button to execute the filter and see the results in the Filter Run Results.

Creating the filter from a result set response

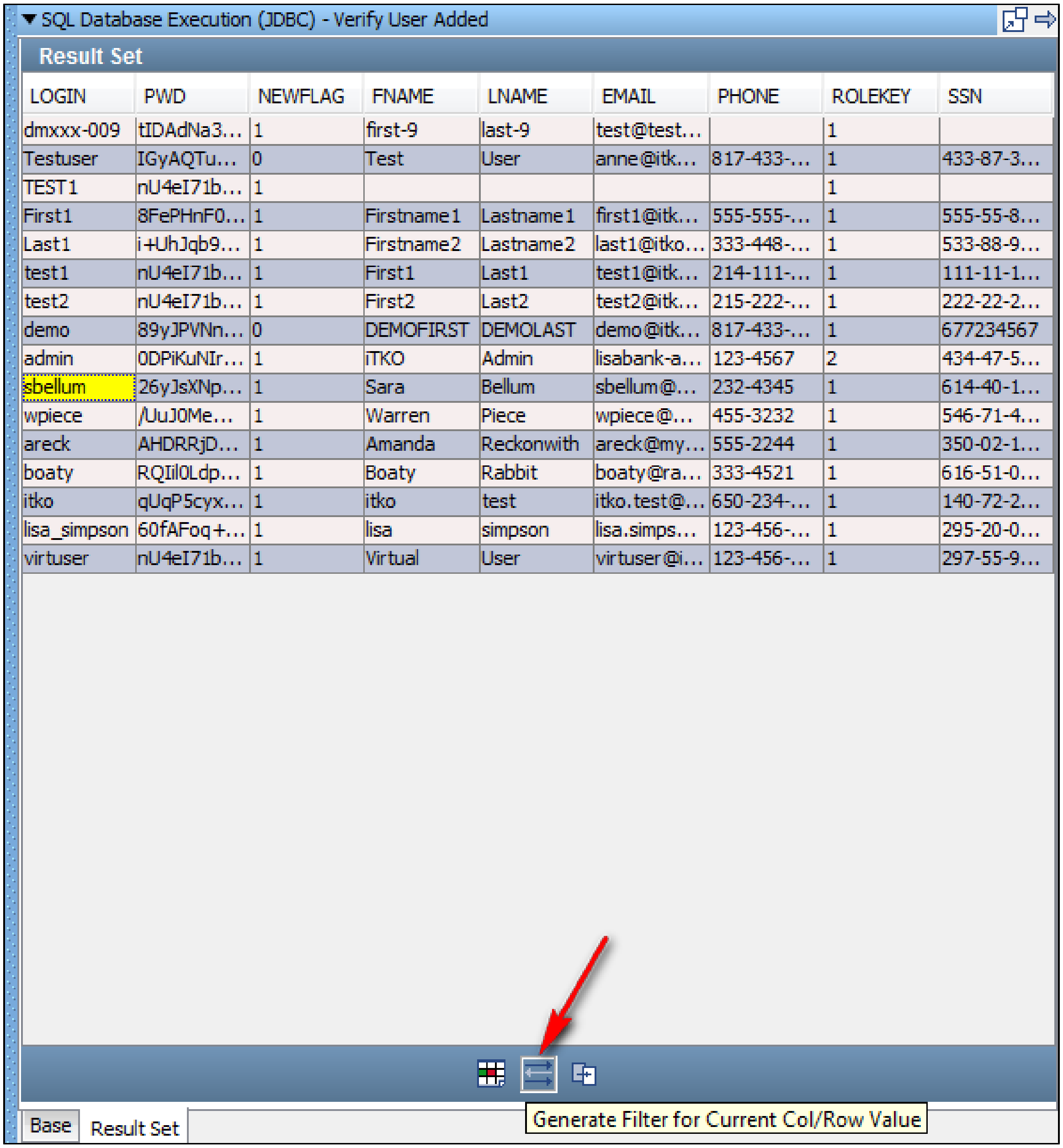
Display the step response that contains the result set. From within the result set select the cell of the value you want to store in the filter.



LOGIN	PWD	NEWFLAG	FNAME	LNAME	EMAIL	PHONE	ROLEKEY	SSN
dmxxx-009	tIDAdNa3…	1	first-9	last-9	test@test…		1	
Testuser	IGyAQTu…	0	Test	User	anne@itk…	817-433-…	1	433-87-3…
TEST1	nU4eI71b…	1					1	
First1	8FePHnF0…	1	Firstname1	Lastname1	first1@itk…	555-555-…	1	555-55-8…
Last1	i+UhJqb9…	1	Firstname2	Lastname2	last1@itko…	333-448-…	1	533-88-9…
test1	nU4eI71b…	1	First1	Last1	test1@itk…	214-111-…	1	111-11-1…
test2	nU4eI71b…	1	First2	Last2	test2@itk…	215-222-…	1	222-22-2…
demo	89yJPVNn…	0	DEMOFIRST	DEMOLAST	demo@itk…	817-433-…	1	677234567
admin	0DPiKuNIr…	1	iTKO	Admin	lisabank-a…	123-4567	2	434-47-5…
sbellum	26yJsXNp…	1	Sara	Bellum	sbellum@…	232-4345	1	614-40-1…
wpiece	/UuJ0Me…	1	Warren	Piece	wpiece@…	455-3232	1	546-71-4…
areck	AHDRRjD…	1	Amanda	Reckonwith	areck@my…	555-2244	1	350-02-1…
boaty	RQIil0Ldp…	1	Boaty	Rabbit	boaty@ra…	333-4521	1	616-51-0…
itko	qUqP5cyx…	1	itko	test	itko.test@…	650-234-…	1	140-72-2…
lisa_simpson	60fAFoq+…	1	lisa	simpson	lisa.simps…	123-456-…	1	295-20-0…
virtuser	nU4eI71b…	1	Virtual	User	virtuser@i…	123-456-…	1	297-55-9…

Click the Generate Filter icon ⇄, shown by the arrow in the previous example.

In the dialog, enter the property key:



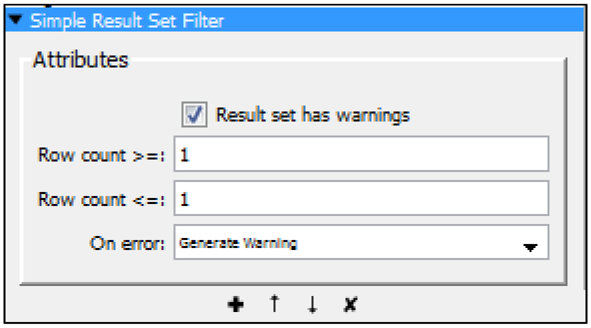
LOGIN	PWD	NEWFLAG	FNAME	LNAME	EMAIL	PHONE	ROLEKEY	SSN
dmxxx-009	tIDAdNa3...	1	first-9	last-9	test@test...		1	
Testuser	IGyAQTu...	0	Test	User	anne@itk...	817-433-...	1	433-87-3...
TEST1	nU4eI71b...	1					1	
First1	8FePHnF0...	1	Firstname1	Lastname1	first1@itk...	555-555-...	1	555-55-8...
Last1	i+UhJqb9...	1	Firstname2	Lastname2	last1@itko...	333-448-...	1	533-88-9...
test1	nU4eI71b...	1	First1	Last1	test1@itk...	214-111-...	1	111-11-1...
test2	nU4eI71b...	1	First2	Last2	test2@itk...	215-222-...	1	222-22-2...
demo	89yJPVNn...	0	DEMOFIRST	DEMOLAST	demo@itk...	817-433-...	1	677234567
admin	0DPiKuNIr...	1	iTKO	Admin	lisabank-a...	123-4567	2	434-47-5...
sbellum	26yJsXNp...	1	Sara	Bellum	sbellum@...	232-4345	1	614-40-1...
wpiece	/UuJ0Me...	1	Warren	Piece	wpiece@...	455-3232	1	546-71-4...
areck	AHDRRjD...	1	Amanda	Reckonwith	areck@my...	555-2244	1	350-02-1...
boaty	RQIil0Ldp...	1	Boaty	Rabbit	boaty@ra...	333-4521	1	616-51-0...
itko	qUqP5cyx...	1	itko	test	itko.test@...	650-234-...	1	140-72-2...
lisa_simpson	60fAFoq+...	1	lisa	simpson	lisa.simps...	123-456-...	1	295-20-0...
virtuser	nU4eI71b...	1	Virtual	User	virtuser@i...	123-456-...	1	297-55-9...

Click the OK button.

LISA will create the filter to store the value **sbellum** in the property **theLogin**.

Simple Result Set Filter

The **Simple Result Set** filter is used to count the number of rows in a Result Set Response.



☑ Result set has warnings

Row count >=: 1

Row count <=: 1

On error: Generate Warning

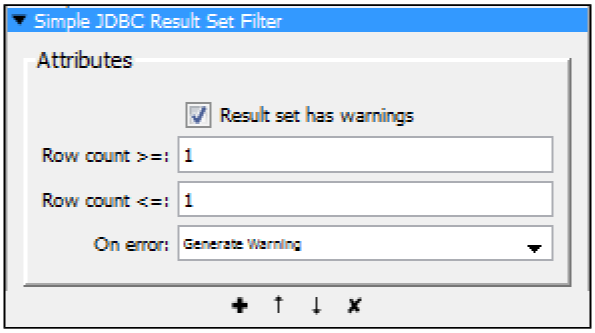
For more information see Size of JDBC Result Set.

Size of JDBC Result Set

(Also known as Simple JDBC Result Set Filter)

The **Size of JDBC Result Set** filter lets you check that the result set returned in each JDBC-based step matches the criteria specified. It is a simple filter to handle most common database errors automatically.

This filter does not affect non-JDBC steps, and is usually used as a global filter in a test case.



Simple JDBC Result Set Filter

Attributes

☑ Result set has warnings

Row count >=: 1

Row count <=: 1

On error: Generate Warning

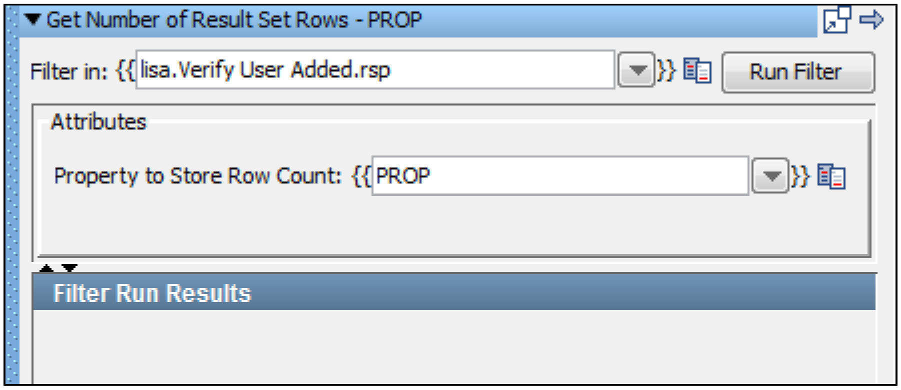
Enter the following parameters:

- **Result Set Has Warnings**: Some databases return warnings in the result set. If your database supports this feature and you want to make a warning fire the **On error** step for this filter, make sure the **Result Set Has Warnings** check box is selected.
- **Row Count At Least (>=)**: The minimum number of rows in the result set. If the result set contains less than this value, the filter sets the next step to the value specified in **On Error** step.
- **Row Count No More Than (<=)**: The maximum number of rows in the result set. If the result set contains more than this value, the filter sets the next step to the value specified in **On Error** step.
- **On Error**: The step to execute if the conditions for this filter are not met.

> ⚠ This filter can serve the purpose of a general global assertion because you can choose a next step based on the presence of an error.

Set Size of a Result Set to a Property

The **Set Size of a Result Set to a Property** filter lets you store the count of a result set to a property provided.



Get Number of Result Set Rows - PROP

Filter in: {{lisa.Verify User Added.rsp}}

Attributes

Property to Store Row Count: {{PROP}}

Filter Run Results

Enter the following parameters:

- **Filter in**: The name of the property you want considered as the step's last response. The property should be in the pull-down menu; if not you can enter it. It must be an existing property.
- **Property to Store Row Count:** User-provided property name to store the row count. The default property name is **PROP**.
- **Filter Run Results:** Displays the property and values set as a result of running the filter.

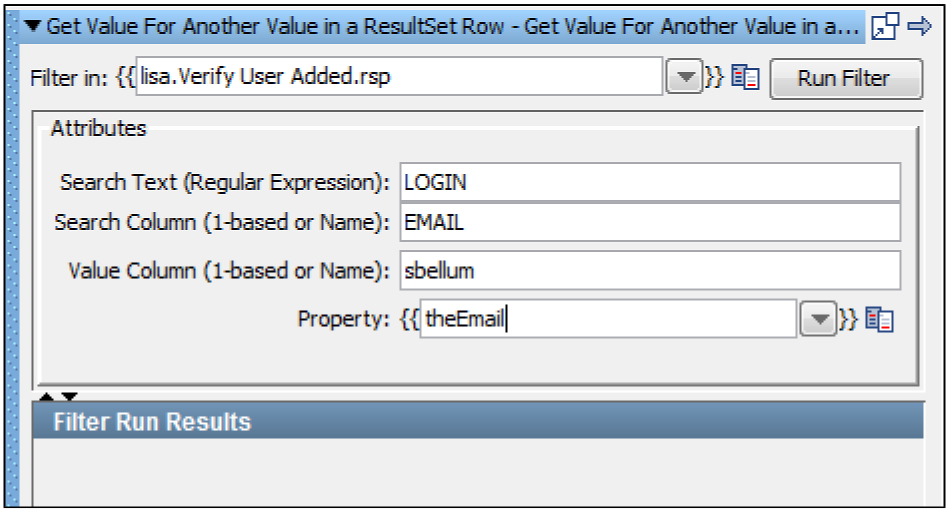
Click the Run Filter button to execute the filter and see the results in the Filter Run Results.

Get Value For Another Value in a ResultSet Row

The **Get Value for Another Value in a ResultSet Row** filter lets you search a column (field) in a result set for a particular value. If the value is found the value in another column (field) and the same row is placed in a property.

This filter can be created in two ways, either as a manual filter from the filter list, or by using the embedded filter commands on a result set response.

Creating the filter manually


Get Value For Another Value in a ResultSet Row - Get Value For Another Value in a...

Filter in: {{lisa.Verify User Added.rsp}}

Search Text (Regular Expression): LOGIN

Search Column (1-based or Name): EMAIL

Value Column (1-based or Name): sbellum

Property: {{theEmail}}

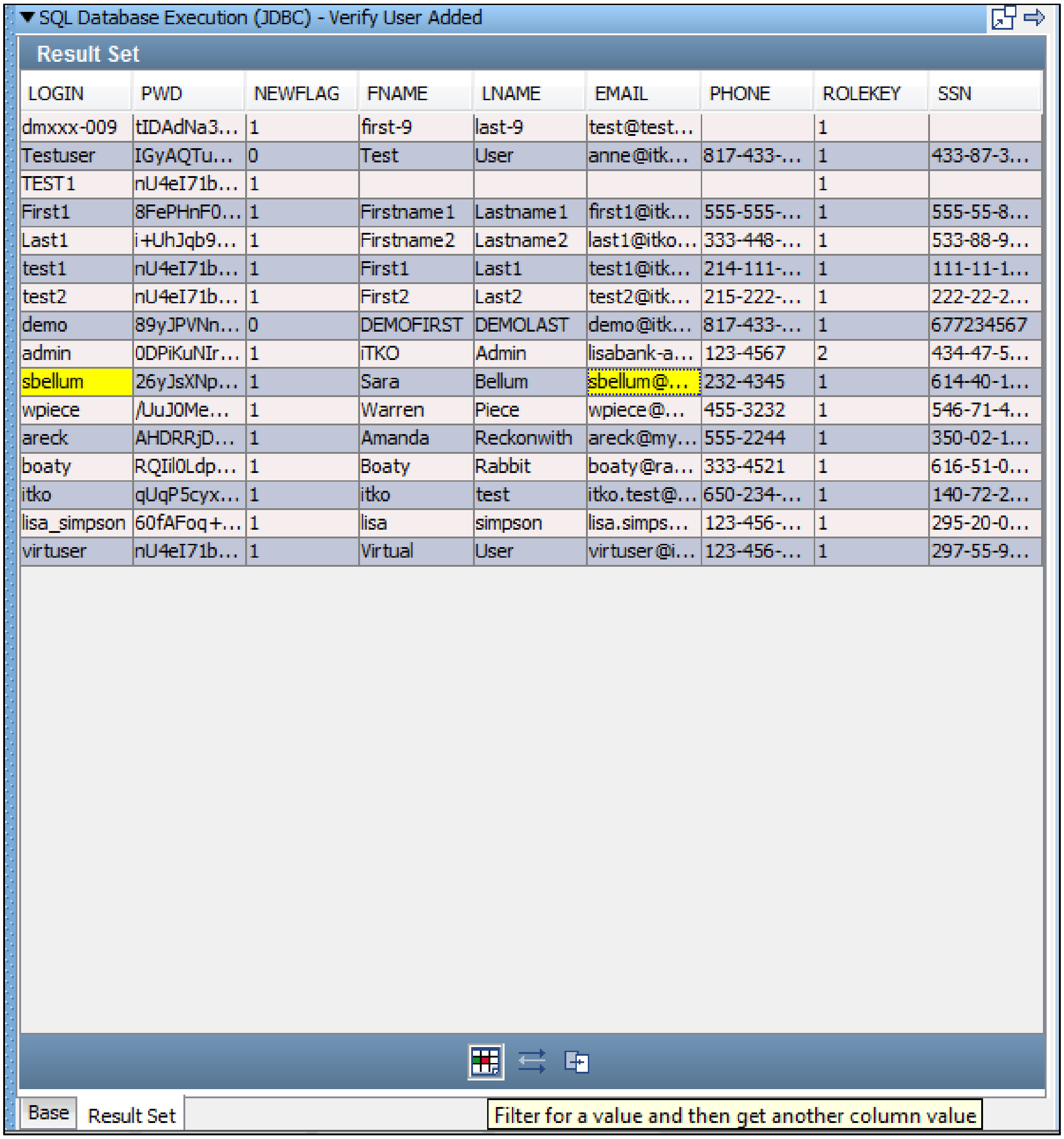
Enter the following parameters:

- **Filter in:** The name of the property you want considered as the step's last response. The property should be in the pull-down menu; if not you can enter it. It must be an existing property.
- **Search Text (Regular Expression):** The search string.
- **Search Column (1-based or Name):** The index or name of the column to search.
- **Value Column (1-based or Name):** The index or name of the column to extract the property value.
- **Property:** The name of the property to store the value.
- **Filter Run Results:** Displays the property and values set as a result of running the filter.

Click the Run Filter button to execute the filter and see the results in the Filter Run Results.

Creating the filter from a result set response

Display the step response that contains the result set. From within the result set select the two values in different columns, using the Ctrl key.

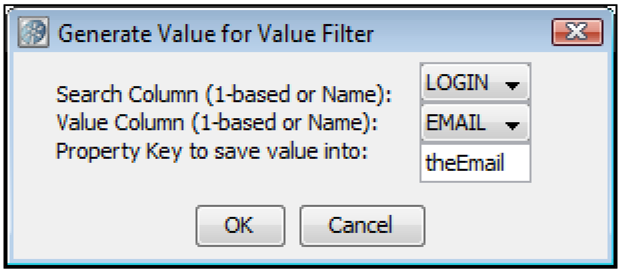


Result Set

LOGIN	PWD	NEWFLAG	FNAME	LNAME	EMAIL	PHONE	ROLEKEY	SSN
dmxxx-009	tIDAdNa3...	1	first-9	last-9	test@test...		1	
Testuser	IGyAQTu...	0	Test	User	anne@itk...	817-433-...	1	433-87-3...
TEST1	nU4eI71b...	1					1	
First1	8FePHnF0...	1	Firstname1	Lastname1	first1@itk...	555-555-...	1	555-55-8...
Last1	i+UhJqb9...	1	Firstname2	Lastname2	last1@itko...	333-448-...	1	533-88-9...
test1	nU4eI71b...	1	First1	Last1	test1@itk...	214-111-...	1	111-11-1...
test2	nU4eI71b...	1	First2	Last2	test2@itk...	215-222-...	1	222-22-2...
demo	89yJPVNn...	0	DEMOFIRST	DEMOLAST	demo@itk...	817-433-...	1	677234567
admin	0DPiKuNIr...	1	iTKO	Admin	lisabank-a...	123-4567	2	434-47-5...
sbellum	26yJsXNp...	1	Sara	Bellum	sbellum@...	232-4345	1	614-40-1...
wpiece	/UuJ0Me...	1	Warren	Piece	wpiece@...	455-3232	1	546-71-4...
areck	AHDRRjD...	1	Amanda	Reckonwith	areck@my...	555-2244	1	350-02-1...
boaty	RQIil0Ldp...	1	Boaty	Rabbit	boaty@ra...	333-4521	1	616-51-0...
itko	qUqP5cyx...	1	itko	test	itko.test@...	650-234-...	1	140-72-2...
lisa_simpson	60fAFoq+...	1	lisa	simpson	lisa.simps...	123-456-...	1	295-20-0...
virtuser	nU4eI71b...	1	Virtual	User	virtuser@i...	123-456-...	1	297-55-9...

Select **Filter for a value** and then get another column value filter using the Filter icon.

In the dialog that opens, select or reassign the columns for the search and the value, then enter the Property Key:



Search Column (1-based or Name): LOGIN
Value Column (1-based or Name): EMAIL
Property Key to save value into: theEmail

Click OK.

LISA will create exactly the same filter as the one that was created manually in the previous example.

In the example, **sbellum** will be searched for, and if found, the value in the **EMAIL** column of that row will be placed in the property **theEmail**.

Messaging_ESB Filters

Messaging_ESB Filters

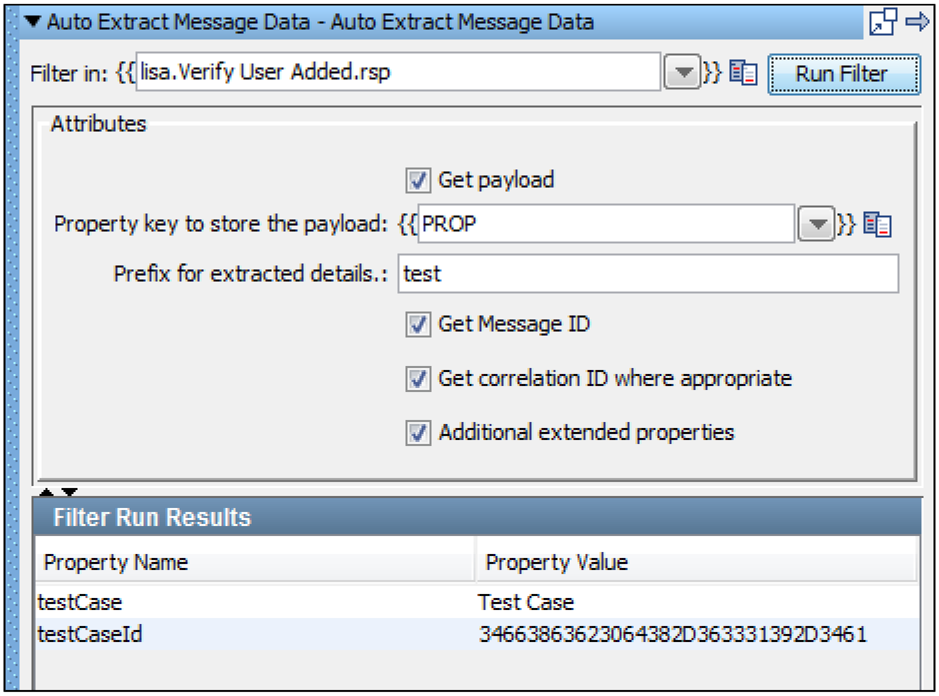
These are the filters available in the Messaging/ESB Filters list for any test step:

Extract Payload and Properties from Messages

There are several internal properties of messages that LISA will auto extract into properties in the test step using the **Extract Payload and Properties from Messages** filter. You can also select to auto extract the payload into a property. This is a fast way to get data from a message.

Different messaging platforms impose various restrictions and can be seen as warnings at execution.

The property names can default to **lisa.stepName.message** or you can specify the prefix. You can specify an exact name for the payload.



Auto Extract Message Data - Auto Extract Message Data

Filter in: {{lisa.Verify User Added.rsp}}

Run Filter

Attributes

☑ Get payload

Property key to store the payload: {{PROP}}

Prefix for extracted details.: test

☑ Get Message ID

☑ Get correlation ID where appropriate

☑ Additional extended properties

Filter Run Results

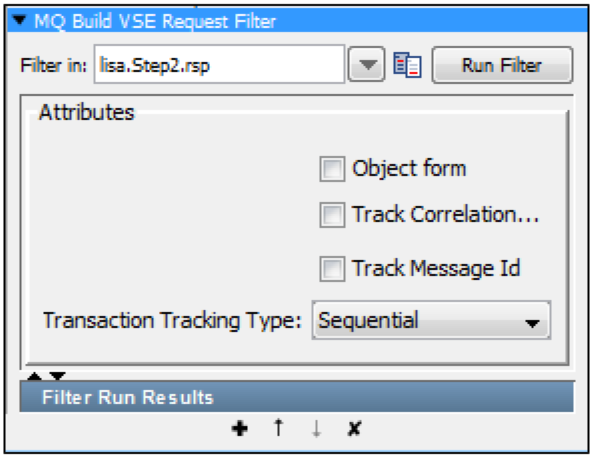
Property Name	Property Value
testCase	Test Case
testCaseId	34663863623064382D363331392D3461

Enter the following parameters:

- **Filter in**: The name of the property you want considered as the step's last response. The property should be in the pull-down menu; if not you can enter it. It must be an existing property.
- **Get Payload**: Select this if you require payload.
- **Property key to store the Payload**: Enter or select the property key to be used as payload.
- **Prefix for extracted details**: Enter the prefix to be attached to the property name in the result.
- **Get Message ID**: Select to get the Message ID.
- **Get Correlation ID**: Select to get the correlation ID.
- **Additional Extended Properties**: Select to get any additional extended properties.
- **Run Filter**: Click the Run filter button to run and execute the filter. The results can be seen in the Filter Run Results section.

Convert a MQ Message to a VSE Request

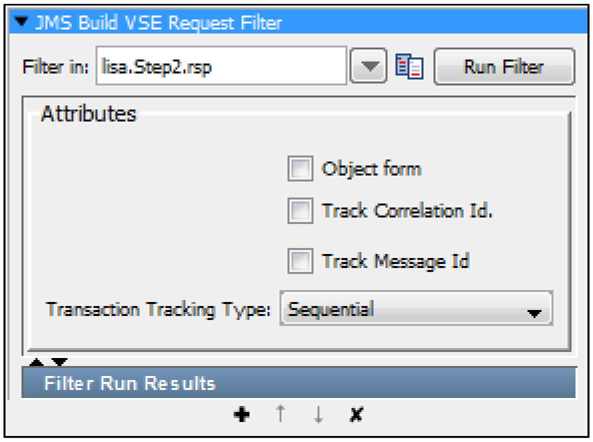
The **Convert a MQ Message to a VSE Request** filter is automatically added from the LISA Virtualize recorder. It serves the specific purpose that enables proper functioning with recordings. It should be used carefully and if it is added to a step in a VSE model, it should not usually be removed or edited.



- **Filter In**: The name of the property you want considered as the step's last response. The property should be in the pull-down menu; if not you can enter it. It must be an existing property.
- **Object Form**: Select to get the Object Form.
- **Track Correlation ID**: Select to track the correlation ID.
- **Track Message ID**: Select to track the message ID.
- **Transaction Tracking Type**: Select the appropriate tracking type from - Sequential, Correlation ID, Message ID or Message ID to Correlation ID.
- **Run Filter**: Click the Run Filter button to run and execute the filter. The results can be seen in the Filter Run Results section.

Convert a JMS Message to a VSE Request

The **Convert a JMS Message to a VSE Request** filter is automatically added from the LISA Virtualize recorder. It serves the specific purpose that enables proper functioning with recordings. It should be used carefully and if it isadded to a step in a VSE model, it should not usually be removed or edited.



- **Filter In**: The name of the property you want considered as the step's last response. The property should be in the pull-down menu; if not, you can enter it. It must be an existing property.

- **Object Form**: Select to get the Object Form.

- **Track Correlation ID**: Select to track the correlation ID.

- **Track Message ID**: Select to track the message ID.

- **Transaction Tracking Type**: Select appropriate tracking type from: Sequential, Correlation ID, Message ID or Message ID to Correlation ID.

- **Run Filter**: Click the Run Filter button to run and execute the filter. The results can be seen in the Filter Run Results section.

HTTP_HTML Filters

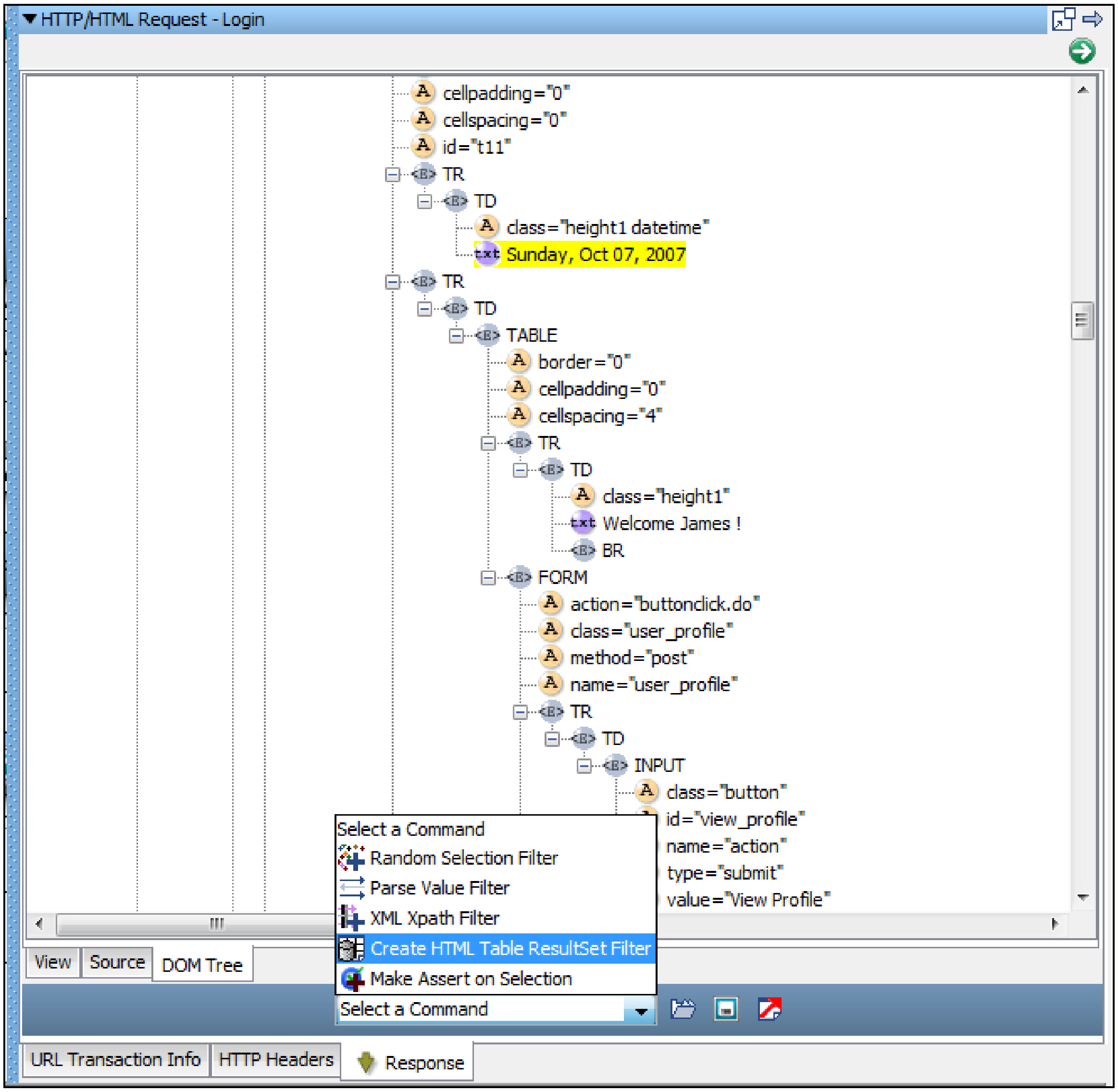
These are the filters available in the HTTP/HTML Filters list for any test step.

Create Resultset from HTML Table Rows

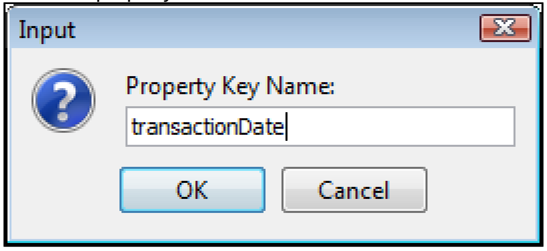
The **Create Resultset from HTML Table Rows** filter lets you create a result set (for example, a JDBC result set) from an HTML table returned in the HTML response. The columns and rows of an HTML table can be selected, and LISA will create a result set from them. The result set can then be used to generate assertions in the same way as it would in a database step.

Although you can create this filter by selecting it from the filter list and filling in the parameters, it is far easier to create it directly from the HTTP/HTML Request step response using one of the filter commands available to that step. This is the approach we take here. The parameters produced here, that is, the ones you would have needed to calculate to manually create this filter, are shown later in this section.

1. To create a filter on a table, record a web page that contains the table, go to the appropriate HTML step, and view it from the DOM tree. Select the values that are to be placed in the table, using the Cntl key, to select multiple fields. You must select one example value from each column in the table that you want to use in the result set.



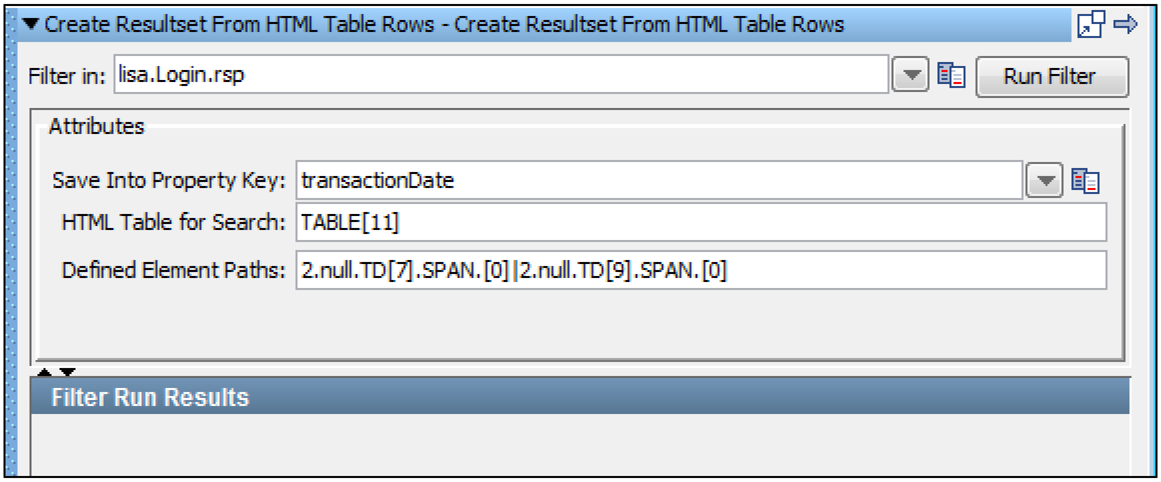
2. When it is highlighted, select **Create HTML Table Results Filter**.
3. Enter the property name in the window.



Property Key Name: transactionDate

4. The property will now be available in the test case.

LISA added this property to the current step. In the following screenshot are the parameters that LISA calculated for this step. These are the parameters you would have had to supply to manually create this filter.



Create Resultset From HTML Table Rows - Create Resultset From HTML Table Rows

Filter in: lisa.Login.rsp

Run Filter

Attributes

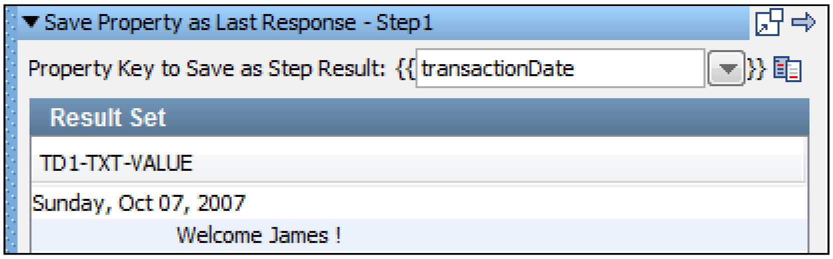
Save Into Property Key: transactionDate

HTML Table for Search: TABLE[11]

Defined Element Paths: 2.null.TD[7].SPAN.[0]|2.null.TD[9].SPAN.[0]

Filter Run Results

To show the results of this filter we added a step of type **Save Property as Last Response** and put in the property created by the filter. The result set panel displays the results.



Save Property as Last Response - Step1

Property Key to Save as Step Result: {{transactionDate}}

Result Set

TD1-TXT-VALUE
Sunday, Oct 07, 2007
Welcome James !

If you are editing an existing test case you may need to replay the test case to generate the property from the filter using the **Replay test case to a specific point** command. The **Replay test case to a specific point** command is activated using the Replay icon on the toolbar, or from the Command menu. You can now use the embedded filters and assertions that are available at the bottom of the result set window of this step.

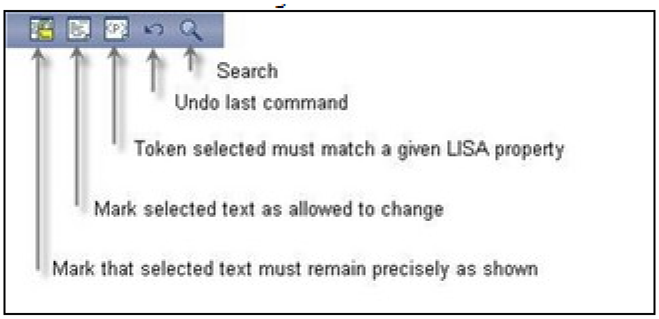
Parse Web Page for Properties

The **Parse Web Page for Properties** filter lets you view a rendered web page to create properties from the HTML content. It uses the "paint the screen" technique.

"Paint the screen" gives you great flexibility to define what in the HTML you want to parse out as properties. There are three ways to mark the text:

1. Text that must appear in the response precisely as shown: a **Must** block.
2. Text that is not required to appear in the response, or can change: an **Any** block.
3. Text that will be stored in a LISA property: a **Property** block.

The text is marked using the icons at the bottom of the editor:

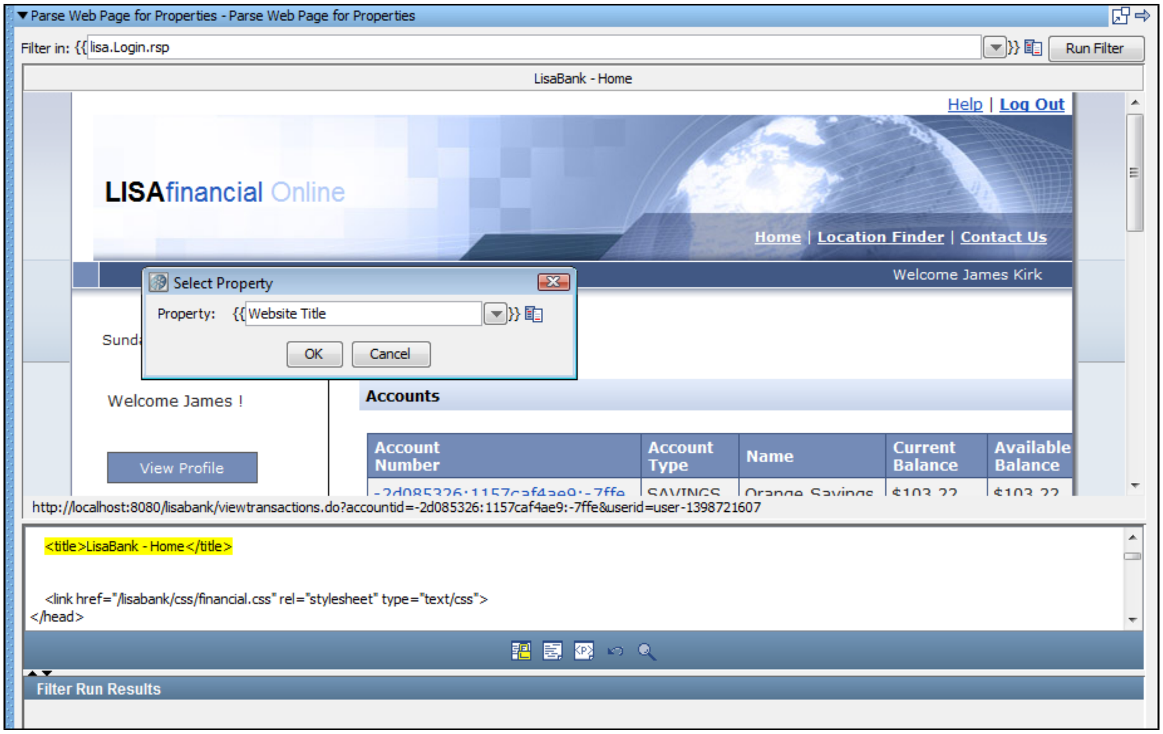

Search
Undo last command
Token selected must match a given LISA property
Mark selected text as allowed to change
Mark that selected text must remain precisely as shown

This technique is best explained by example. In the following example, see the following example. Assume that we expect that the company name "ITKO" will change from user to user, and therefore needs to be stored as a LISA property.

We have marked the text using the editor icons, by selecting text and then clicking on the appropriate icon.

- Yellow background indicates text that must appear as shown (indicated by the arrows from "Text uses ...."). This is a **Must** block and is marked using the **Must** icon.
- Red background identifies the text that will be stored in the property entered into the dialog (indicated by a single arrow). This is a **Property** block and is marked using the **Property** icon.

> ⚠ **Property** blocks must always be bounded by **Must** blocks.



▼ Parse Web Page for Properties - Parse Web Page for Properties

Filter in: {{lisa.Login.rsp}}

LisaBank - Home

Select Property

Property: {{Website Title}}

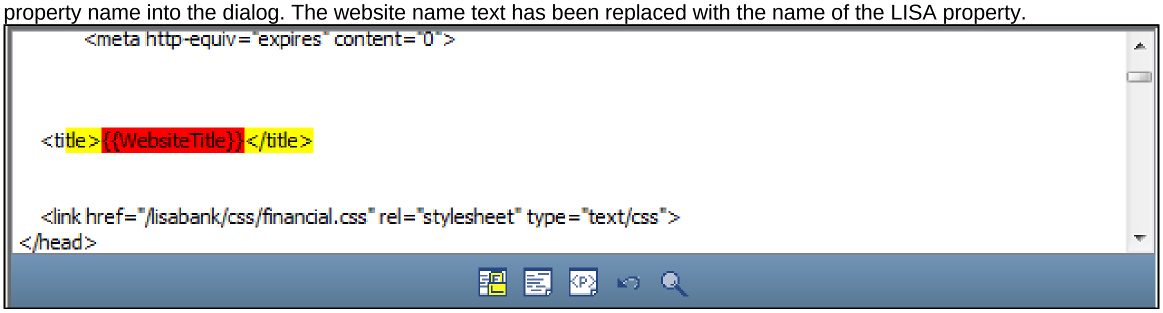
http://localhost:8080/lisabank/viewtransactions.do?accountid=-2d085326:1157caf4ae9:-7ffe&userid=user-1398721607

```
<title>LisaBank - Home</title>


<link href="/lisabank/css/financial.css" rel="stylesheet" type="text/css">
</head>
```

Filter Run Results

This screen shows the HTML rendered in a browser in the top panel, and the actual HTML text in the bottom panel. We want to parse out the website title in the **title** field. We have set the boundaries around that, and clicked the "Must" icon.

Then we selected the website name text, "LISABank - Home," inside the highlighted content, and clicked the **Property** icon . We entered the property name into the dialog. The website name text has been replaced with the name of the LISA property.



```
        <meta http-equiv="expires" content="0">



    <title>{{WebsiteTitle}}</title>



    <link href="/lisabank/css/financial.css" rel="stylesheet" type="text/css">
</head>
```

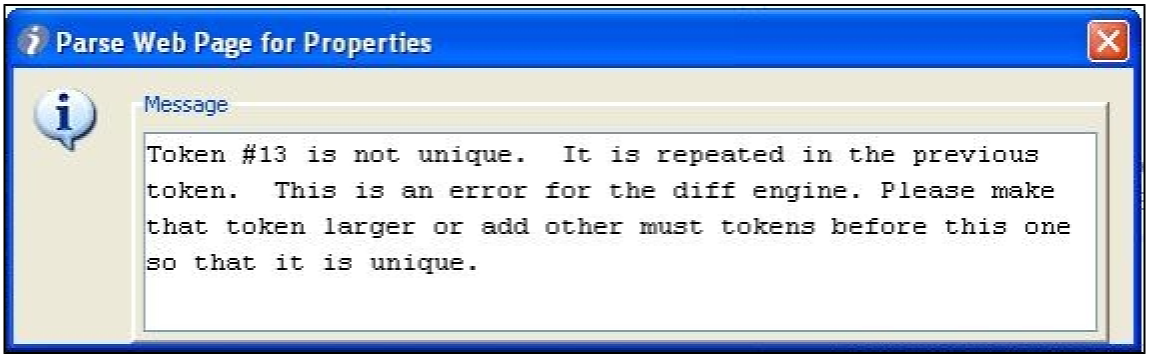
Frequently you can do this purely from the web page view by selecting the content in the web browser. At times, it will be easier to click on the web browser in the area that you want to select, then make your actual selection in the HTML panel.

Now when this filter is run, the property **WebsiteTitle** will be assigned the current value that appears on the HTML page. The website title can change location in the text buffer and it will still be located and parsed for the property.

You can repeat this process on this text buffer to have as many properties defined as you like.

Handling Non-unique Tokens

If you see an error message such as the following:



```
Token #13 is not unique.  It is repeated in the previous
token.  This is an error for the diff engine. Please make
that token larger or add other must tokens before this one
so that it is unique.
```

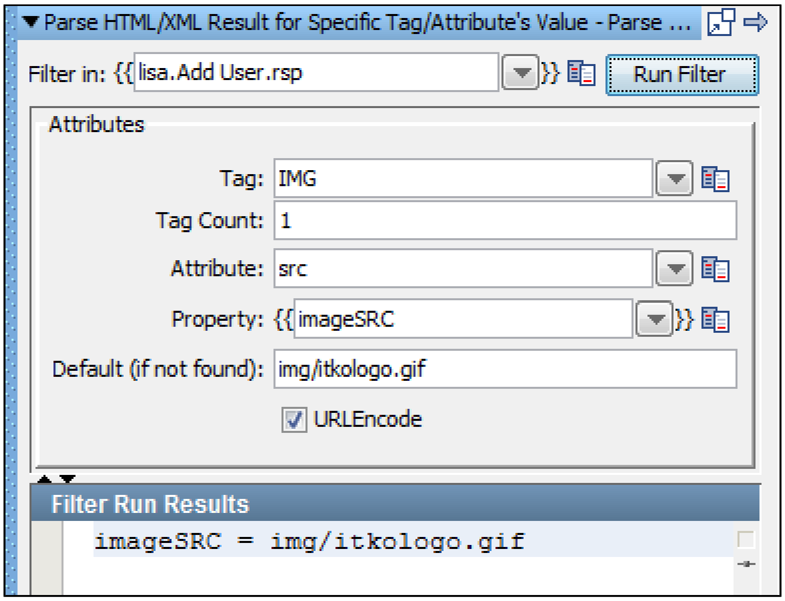
LISA is telling you that your selected token is not unique; the selection you just made is repeated in the token before it. To solve this in most cases, simply create another token to make the prior token a **Must** token also. In other cases, when this does not work, a judicious placement of another **Must** block in between the two duplicate tokens should work. This will work because LISA can distinguish between the two duplicate tokens based on their relative location.

Parse HTML_XML Result for Specific Tag_Attributes Values

The **Parse HTML for Specific Tag/Attribute's Values** filter lets you parse the HTML response for a given attribute of a given tag.

This filter can be created in two ways: either as a manual filter from the filter list or by using the embedded filter commands on a result set response.

Creating the filter manually


Parse HTML/XML Result for Specific Tag/Attribute's Value - Parse ...
Run Filter
Attributes
Filter Run Results

Filter in: {{lisa.Add User.rsp}}

Tag: IMG

Tag Count: 1

Attribute: src

Property: {{imageSRC}}

Default (if not found): img/itkologo.gif

☑ URLEncode

```
imageSRC = img/itkologo.gif
```

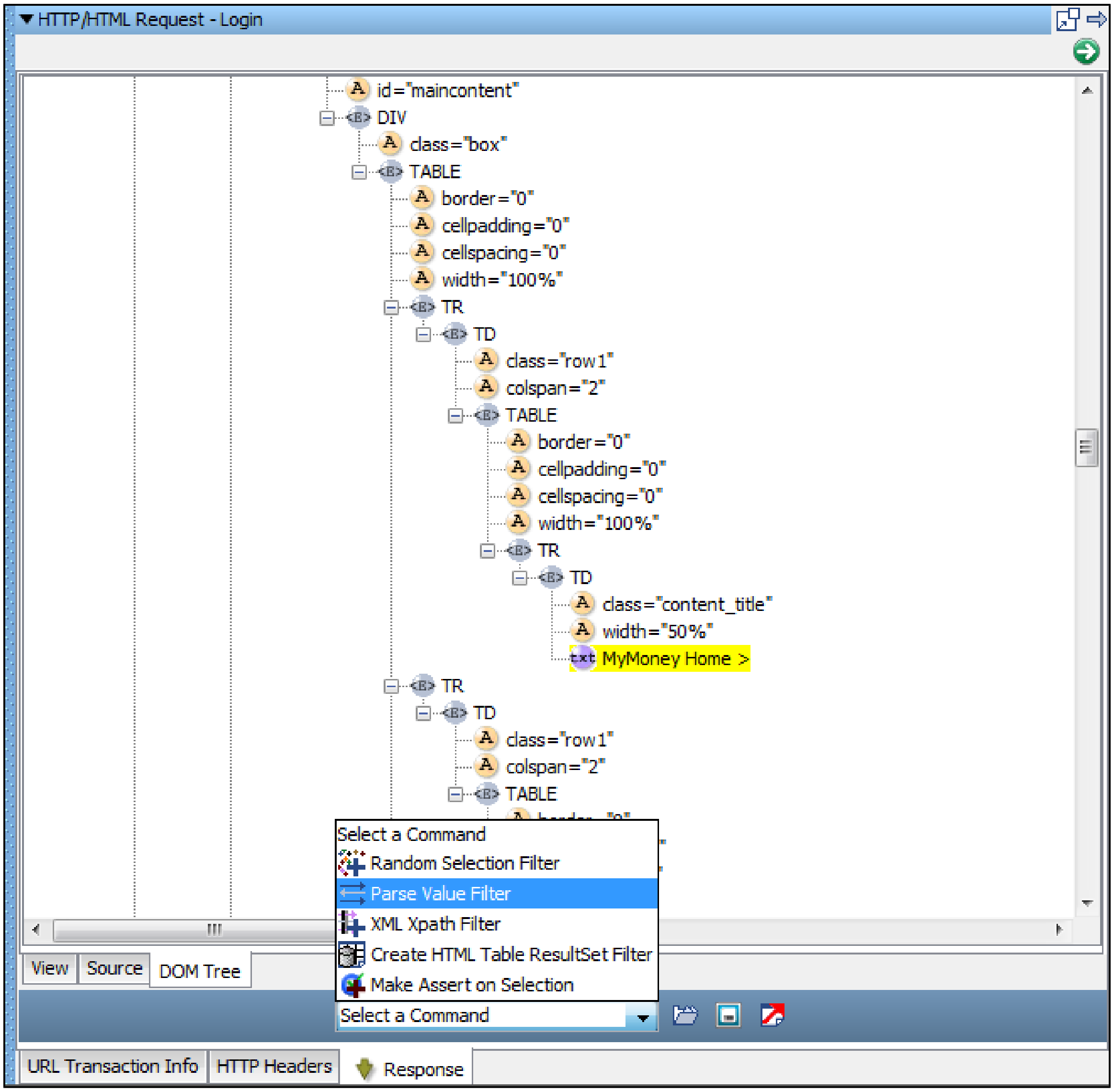
1. Enter the following parameters:

- **Filter in**: The name of the property you want considered as the step's last response. The property should be in the pull-down menu; if not, you can enter it. It must be an existing property.
- **Tag:** The name of the HTML tag; for an image tag enter **IMG**.
- **Tag Count:** The occurrence of the tag from the top of response; for the first image tag enter **1**.
- **Attribute:** The name of the attribute to filter; for the source attribute enter **src**.
- **Property:** The property in which to store the value.
- **Default (if not found):** The value to use if the attribute value is not found.
- **URLEncode:** When checked, property value is URLEncoded.
- **Filter Run Results:** Displays the property and values set as a result of running the filter.

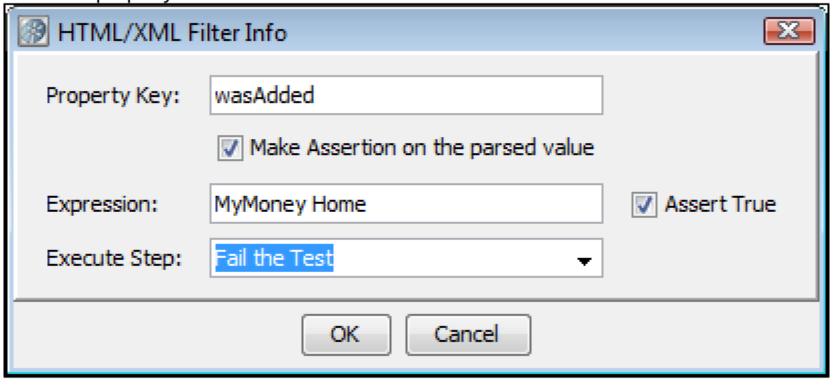
2. Click the Run Filter button to execute the filter and see the results in the Filter Run Results.

Creating the filter from the HTTP/HTML request step response page

1. Display the step response that contains the HTML response.



2. From the DOM Tree view select the attribute whose value you want to store in a property.
3. When it is highlighted, select **Parse Value Filter**.
4. Enter the property name in the window.



HTML/XML Filter Info

Property Key: wasAdded

☑ Make Assertion on the parsed value

Expression: MyMoney Home ☑ Assert True

Execute Step: Fail the Test

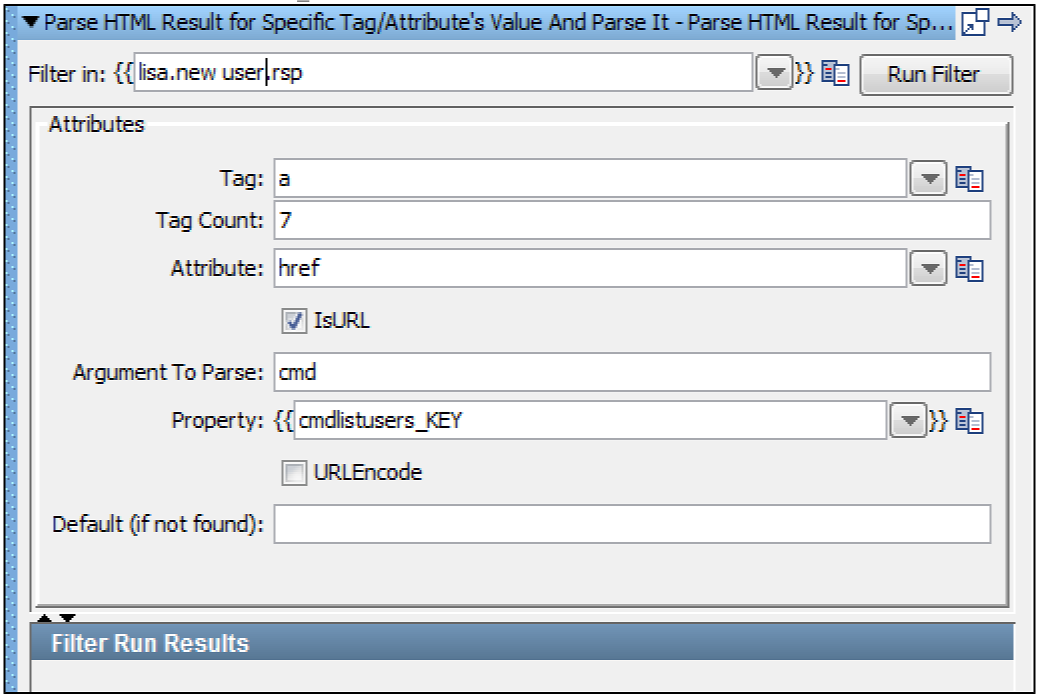
5. Assertions can be also be added here.

Parse HTML Result for Specific Tag_Attribute's Value and Parse It

The **Parse HTML Result for Specific Tag/Attribute's Value and Parse It** filter is really a combination of two other filters: **Parse HTML Result for Attribute's Value** and **Parse Property Value as Argument String**.

This filter is designed to find a certain attribute in a web page, and then further parse that attribute. If the attribute is a URL, and not just a name-value pair, there is a function for handling that information.

In this example, we see the filter finds the seventh anchor tag's "href" attribute, which is a URL. The filter takes the "cmd" parameter and stores that value in the **cmdlist users_KEY**.



Parse HTML Result for Specific Tag/Attribute's Value And Parse It - Parse HTML Result for Sp...

Filter in: {{lisa.new user.rsp}}

Attributes

Tag: a

Tag Count: 7

Attribute: href

☑ IsURL

Argument To Parse: cmd

Property: {{cmdlistusers_KEY}}

☐ URLEncode

Default (if not found):

Filter Run Results

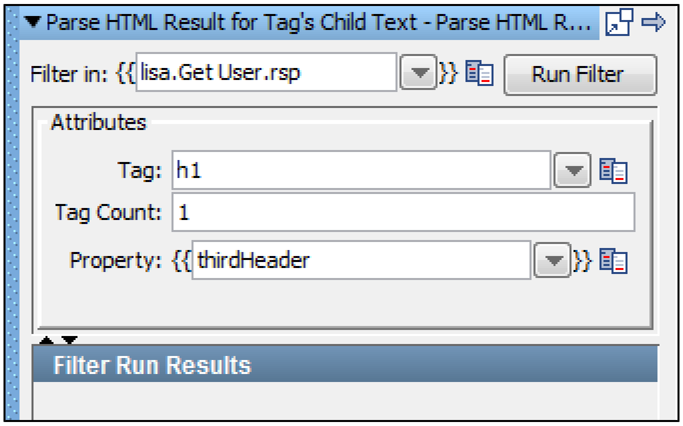
Enter the following parameters:

- **Filter in**: The name of the property you want considered as the step's last response. The property should be in the pull-down menu; if not, you can enter it. It must be an existing property.
- **Tag:** The name of the HTML tag; for an anchor tag enter **a**.
- **Tag Count:** The occurrence of the tag from the top of response; for the seventh anchor tag enter **7**.
- **Attribute:** The name of the attribute to filter; for the href attribute enter **href**.
- **IsURL:** Select the check box if the attribute value is a URL.
- **Argument to Parse:** The name of the argument to parse for its value; in this example, **cmd**.
- **Property:** The property in which to store the value; in this example, **cmdlist users_KEY**.
- **URLEncode:** When checked, property value is URLEncoded.
- **Default (if not found):** The value to use if the attribute value is not found.
- **Filter Run Results:** Displays the property and values set as a result of running the filter.

Click the Run Filter button to execute the filter and see the results in the Filter Run Results.

Parse HTML Result for Tag's Child Text

The **Parse HTML Result for Tag's Child Text** filter lets you store the text of a tag's child text in a property.



Parse HTML Result for Tag's Child Text - Parse HTML R...

Filter in: {{lisa.Get User.rsp}}

Attributes

Tag: h1

Tag Count: 1

Property: {{thirdHeader}}

Filter Run Results

Enter the following parameters:

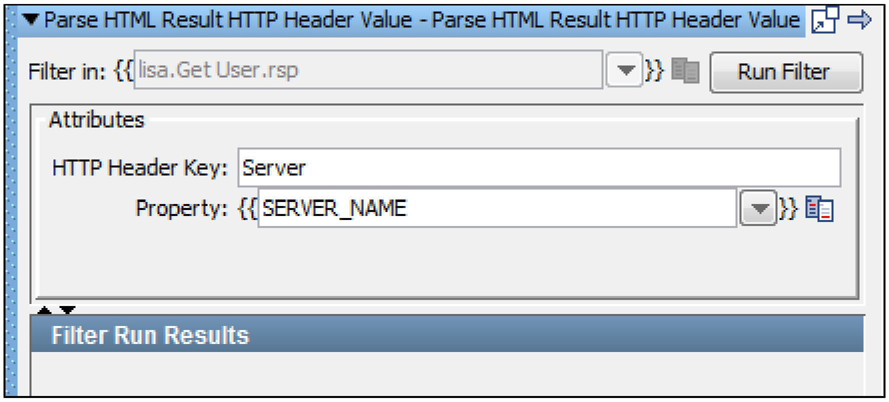
- **Filter in**: The name of the property you want considered as the step's last response. The property should be in the pull-down menu; if not, you can enter it. It must be an existing property.

- **Tag:** The type of the tag. For example, for an h1 tag enter **h1**.
- **Tag Count:** The occurrence of the tag. For the child text of the third h1 tag, enter **3**.
- **Property:** The name of the property to store the text.
- **Filter Run Results:** Displays the property and values set as a result of running the filter.

Click the Run Filter button to execute the filter and see the results in the Filter Run Results.

Parse HTML Result for HTTP Header Value

The **Parse HTML Result for HTTP Header Value** filter lets you store the value of a returned HTTP header key in a property.


Parse HTML Result HTTP Header Value - Parse HTML Result HTTP Header Value
Run Filter
Attributes
Filter Run Results

Filter in: {{lisa.Get User.rsp}}

HTTP Header Key: Server

Property: {{SERVER_NAME}}

A common use of this filter is saving the HTTP header **Server** in a property named SERVER_NAME.

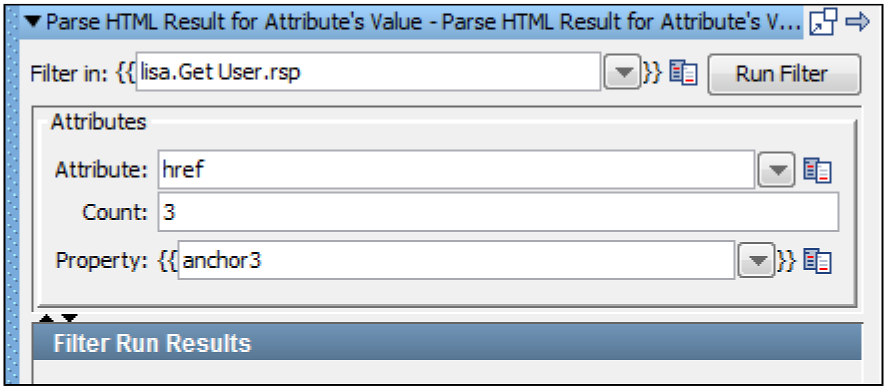
Enter the following parameters:

- **Filter in**: The name of the property you want considered as the step's last response. The property should be in the pull-down menu; if not, you can enter it. It must be an existing property.
- **HTTP Header Key:** The name of the HTTP header; for example, **Server.**
- **Property:** The property to store the header value.
- **Filter Run Results:** Displays the property and values set as a result of running the filter.

Click the Run Filter button to execute the filter and see the results in the Filter Run Results.

Parse HTML Result for Attribute's Value

The **Parse HTML Result for Attribute's Value** filter lets you store the text of a specific attribute in a property. The attribute can occur anywhere in the result, including scripting code.


Parse HTML Result for Attribute's Value - Parse HTML Result for Attribute's V...
Run Filter
Attributes
Filter Run Results

Filter in: {{lisa.Get User.rsp}}

Attribute: href

Count: 3

Property: {{anchor3}}

The parameters set are:

- **Filter in**: The name of the property you want considered as the step's last response. The property should be in the pull-down menu; if not, you can enter it. It must be an existing property.
- **Attribute:** The type of attribute to retrieve. For example, if you want the URL of an anchor tag, enter **href**.

- **Count:** The occurrence of the tag. For example, if you want the URL of the third anchor tag on the page, enter **3**.
- **Property**: The property to store the text of the attribute; in this example, **anchor3**.
- **Filter Run Results:** Displays the property and values set as a result of running the filter.

Click the Run Filter button to execute the filter and see the results in the Filter Run Results.

Parse HTML Result for LISA Tags

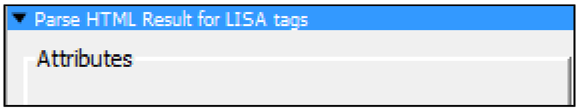
The **Parse HTML Result for LISA Tags** filter provides a way for developers to test-enable their web applications. For an in-depth study on test-enabling, see the *Developer's Guide (SDK)*.

This filter provides the ability to insert "LISAPROP" tags into your web page. The LISAPROP tag has two attributes: name and value. The LISAPROP tags do not show up in your web pages. They function only to discretely provide valuable information about your web page to a tester. An example of a LISAPROP might be:

<LISAPROP name="FIRST_USER" value="sbellum">.

If a tester has installed this type of filter, the property **FIRST_USER** will automatically be assigned the value **sbellum**. This removes any need for the tester to parse for this value. This type of filter helps a developer make the testing easier.

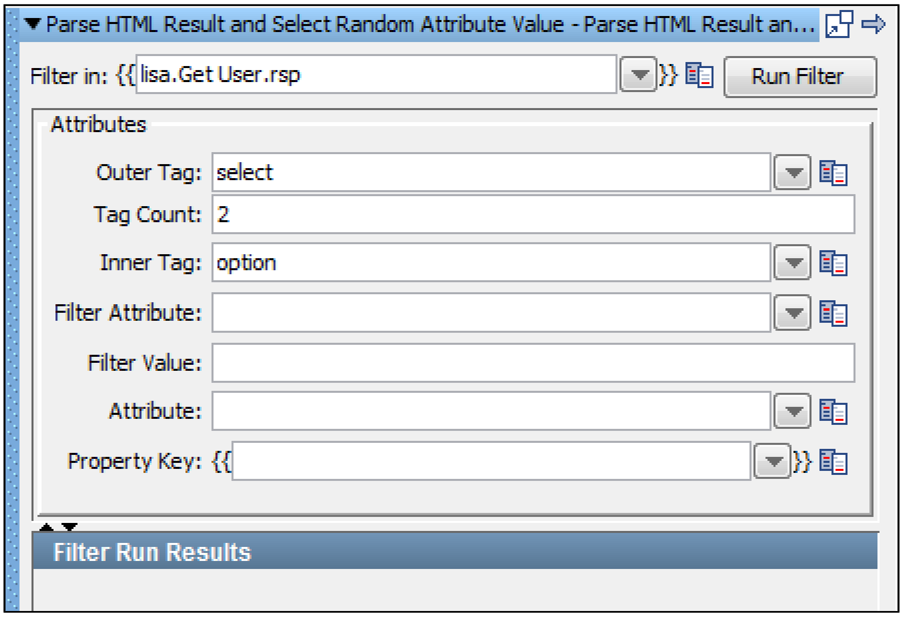
Frequently a web page will not contain the information needed to perform proper validation, or that information is very difficult to parse. Even when it is there, the parsing can become incorrect because of subtle changes in the HTML that is generated. This LISAPROP filter can resolve many tedious parsing issues for web testing.



There are no parameters required.

Parse HTML Result and Select Random Attribute Value

The **Parse HTML Result and Select Random Attribute Value** filter lets you store the text of a random selection from a set in a property. The attribute can occur anywhere in the result, including scripting code.


Parse HTML Result and Select Random Attribute Value - Parse HTML Result an...
Run Filter
Attributes
Filter Run Results

Filter in: {{lisa.Get User.rsp}}

Outer Tag: select

Tag Count: 2

Inner Tag: option

Filter Attribute:

Filter Value:

Attribute:

Property Key: {{}}

Enter the following parameters:

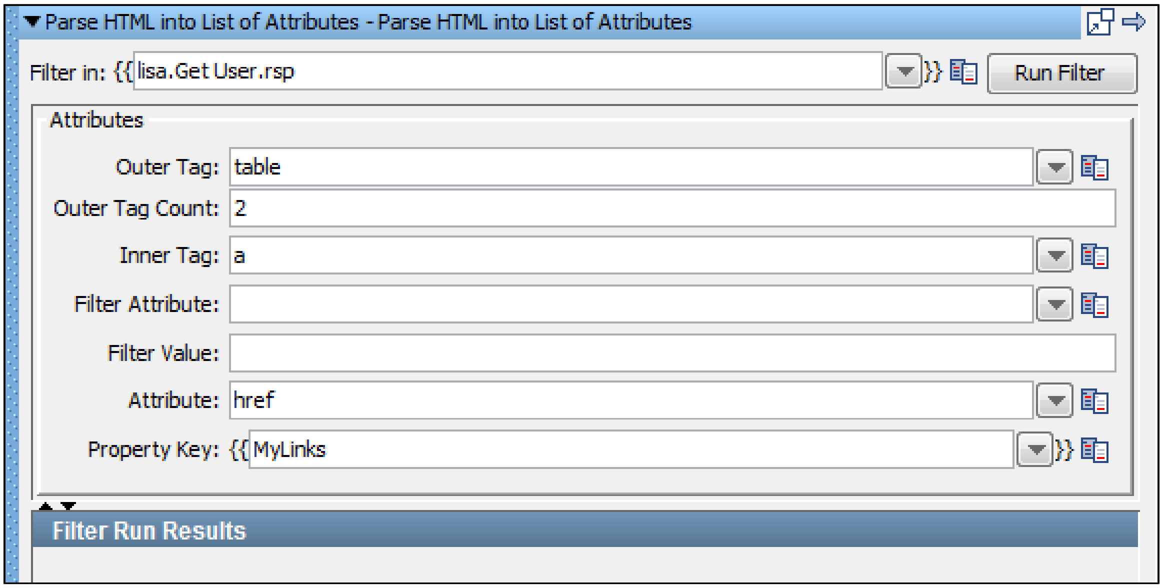
- **Filter in**: The name of the property you want considered as the step's last response. The property should be in the pull-down menu; if not, you can enter it. It must be an existing property.
- **Outer Tag**: The outer element that contains the list from which to pick. For example, to select a drop-down menu, you would enter the text **select**.
- **Tag Count**: The occurrence of the outer tag. For example, to select the second drop-down menu, you would enter the text "2".

- **Inner Tag:** The tag to randomly pick the attribute from. To pick a random item in the drop-down menu, you would enter the text **option**.
- **Filter Attribute:** Optional field to specify attribute names that should not appear in the pick list.
- **Filter Value:** Optional field to specify attribute values that should not appear in the pick list.
- **Attribute:** The attribute from which to retrieve text. If this is blank, the child text of the inner tag is returned.
- **Property Key:** The property to store the text of the attribute.
- **Filter Run Results:** Displays the property and values set as a result of running the filter.

Click the Run Filter button to execute the filter and see the results in the Filter Run Results.

Parse HTML into List of Attributes

The **Parse HTML into List of Attributes** filter lets you store the text of a set of attributes, as a list, in a property.


Parse HTML into List of Attributes - Parse HTML into List of Attributes
Run Filter
Attributes
Filter Run Results

Filter in: {{lisa.Get User.rsp}}

Outer Tag: table

Outer Tag Count: 2

Inner Tag: a

Filter Attribute:

Filter Value:

Attribute: href

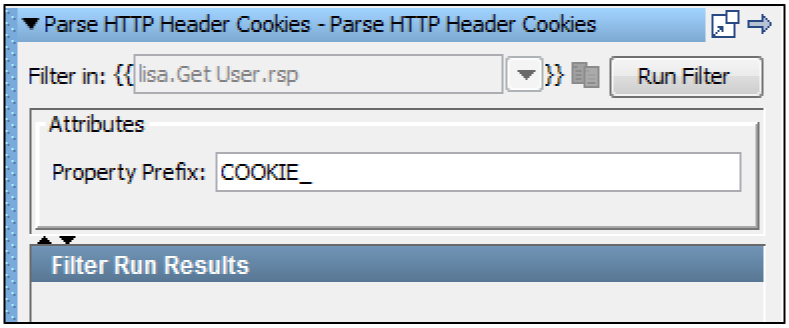
Property Key: {{MyLinks}}

- **Filter in**: The name of the property you want considered as the step's last response. The property should be in the pull-down menu. If not, you can enter it. It must be an existing property.
- **Outer Tag:** The outer element that contains the list of tags to parse. For example, to store all the links from all the anchor tags in a table, enter **table**.
- **Outer Tag Count:** The occurrence of the outer tag. For the second table, enter **2**.
- **Inner Tag:** The tag to retrieve the values from. For example, for all the anchor tags in the table enter **a**.
- **Filter Attribute:** Optional field to specify attribute names that should not appear in the pick list.
- **Filter Value:** Optional field to specify attribute values that should not appear in the pick list.
- **Attribute:** The attribute of the Inner Tag to retrieve the text from. If this is blank, the child text of the Inner Tag is returned. To store all the links from all of the anchor tags in a table, enter **href**.
- **Property Key:** The name of the property to store the text of the attribute.
- **Filter Run Results:** Displays the property and values set as a result of running the filter.

Click the Run Filter button to execute the filter and see the results in the Filter Run Results.

Parse HTTP Header Cookies

The **Parse HTTP Header Cookies** filter lets you parse the HTTP header for cookie values, and store them in a property starting with a specific prefix.



Parse HTTP Header Cookies - Parse HTTP Header Cookies

Filter in: {{lisa.Get User.rsp}}

Attributes

Property Prefix: COOKIE_

Enter the following parameters:

- **Filter in**: The name of the property you want considered as the step's last response. The property should be in the pull-down menu; if not, you can enter it. It must be an existing property.
- **Property Prefix:** A text string that will be prefixed to the cookie name to provide the property name to use. The full names of these properties are therefore dependent on the names of the cookies that have been returned. The cookie names can be identified in the property tab of the Interactive Test Run (ITR).
- **Filter Run Results:** Displays the property and values set as a result of running the filter.

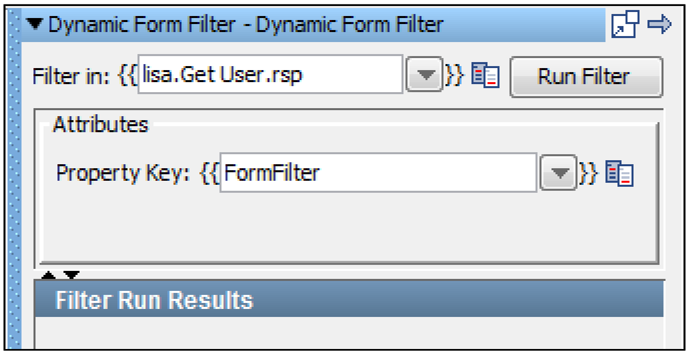
Click the Run Filter button to execute the filter and see the results in the Filter Run Results.

Dynamic Form Filter

The **Dynamic Form** filter identifies dynamically generated forms in HTML responses and parses them into a set of properties. The property key that you enter becomes part of the property name for each form element in each form. This is easier to understand by examining the example that follows.

You might test an HTML page with two dynamically generated forms:

```
<form name="F001" action="index.jsp"> <input type="text" name="0001A" value="default" /> <input
type="text" name="0001B" value="" /></form>
<form name="F002" action="orders.jsp"> <input type="text" name="0002A" value=Key"" /> <input
type="text" name="0002B" value="" /></form>
```



Dynamic Form Filter - Dynamic Form Filter

Filter in: {{lisa.Get User.rsp}}

Attributes

Property Key: {{FormFilter}}

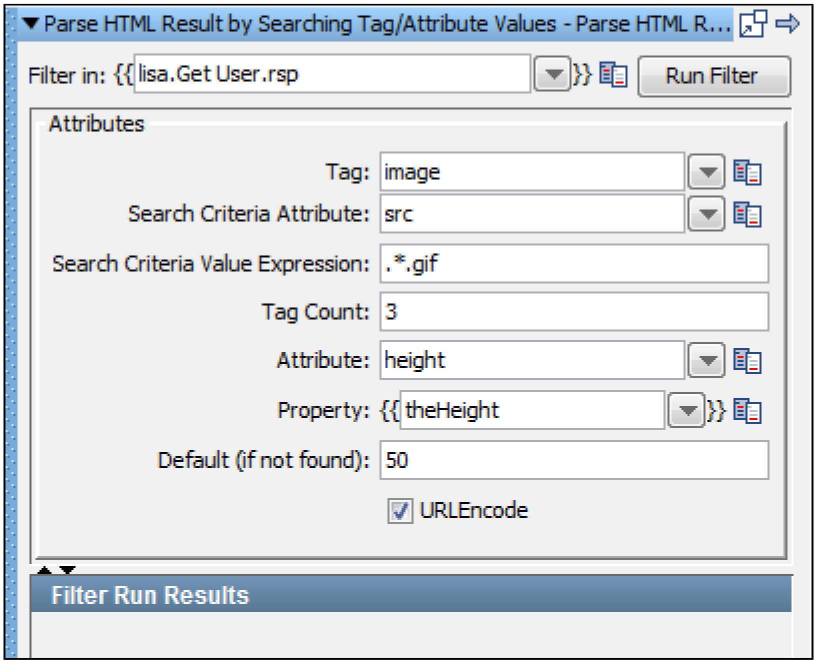
Using a property key of **FormTest** in the filter panel would create the following key-value pairs:

Key	Value
FormTest.Form1.text1.name	0001A
FormTest.Form1.text1.value	default
FormTest.Form1.text2.name	0001B
FormTest.Form1.text2.value	

FormTest.Form2.text1.name	0002A
FormTest.Form2.text1.value	
FormTest.Form2.text2.name	0002B
FormTest.Form2.text2.value	

Parse HTML Result by Searching Tag_Attribute Values

The **Parse HTML Result by Searching Tag/Attribute Values** filter lets you filter the value of a tag attribute by searching for the name and value of another attribute in that tag. If more than one tag fits the criteria you can specify which one you want.



Parse HTML Result by Searching Tag/Attribute Values - Parse HTML R...

Filter in: {{lisa.Get User.rsp}}

Attributes

Tag: image

Search Criteria Attribute: src

Search Criteria Value Expression: .*.gif

Tag Count: 3

Attribute: height

Property: {{theHeight}}

Default (if not found): 50

☑ URLEncode

The parameters set are:

- **Filter in**: The name of the property you want considered as the step's last response. The property should be in the pull-down menu; if not, you can enter it. It must be an existing property.
- **Tag:** The name of the tag to search.
- **Search Criteria Attribute:** The attribute to search for.
- **Search Criteria Value Expression:** The attribute expression to search for.
- **Tag Count:** The specific tag to use from those that satisfy the search criteria.
- **Attribute:** The attribute whose value you want.
- **Property:** The property in which to store the value.
- **Default (if not found):** The value to use if search is not successful.
- **URLEncode:** When selected, the value is URLEncoded.
- **Filter Run Results:** Displays the property and values set as a result of running the filter.

Click the Run Filter button to execute the filter and see the results in the Filter Run Results.

XML Filters

These are the filters available in the XML Filters list for any test step:

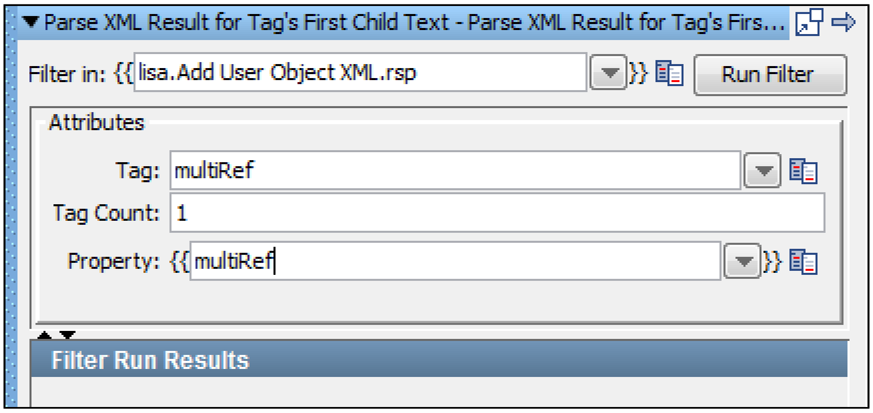
Parse text from XML

(Also known as Parse XML Result for Tag's First Child Text)

The **Parse text from XML** filter stores the text of a tag's child text in a property. To define a **Parse text from XML** filter, set the type of the filter and set the three attributes.

This filter can be created in two ways: as a manual filter from the filter list or by using the embedded filter commands on an XML response.

Creating the filter manually



Parse XML Result for Tag's First Child Text - Parse XML Result for Tag's Firs...

Filter in: {{lisa.Add User Object XML.rsp}}

Attributes

Tag: multiRef

Tag Count: 1

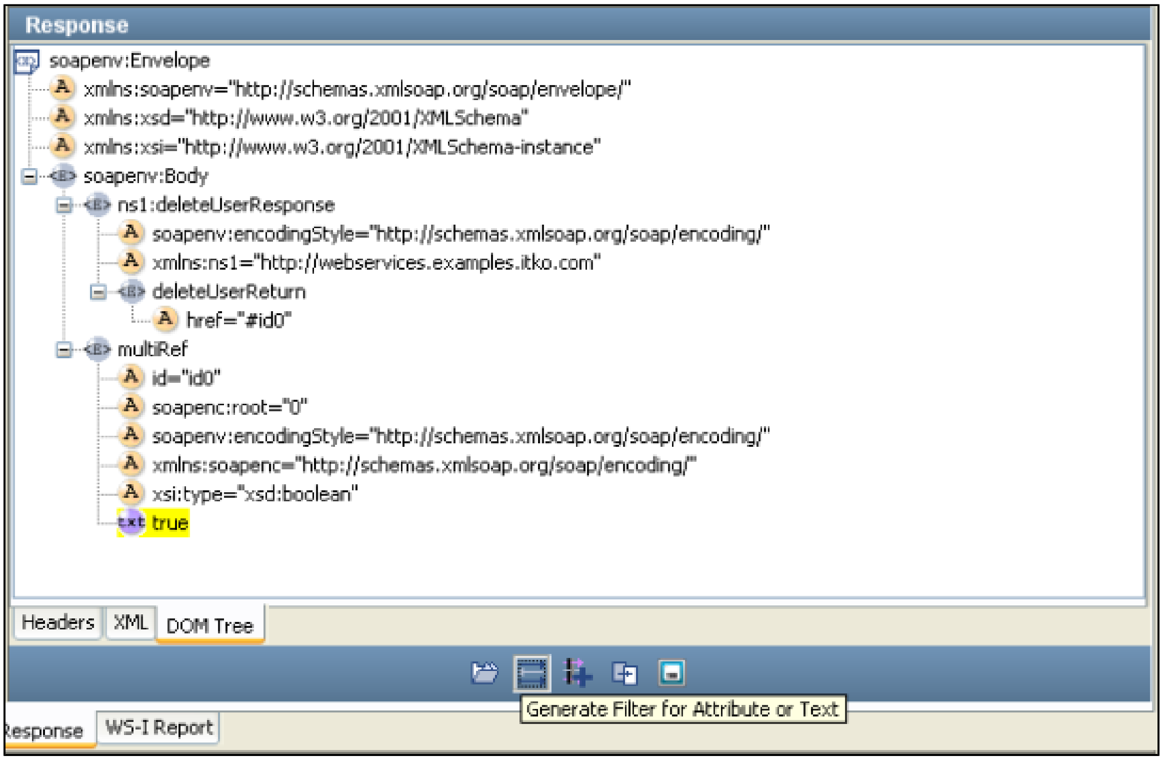
Property: {{multiRef}}

1. Enter the following parameters:

- **Filter in:** Where to apply the filter. This illustration shows **lisa.Add User Object XML.rsp**, which means that the filter will be applied to the response of **Add User Object XML**. You can edit this value for this filter.
- **Tag:** The type of the tag. For example, if you want the child text of the multiRef tag, enter **multiRef**.
- **Tag Count:** The occurrence of the tag. For example, if you want the child text of the first multiRef tag, set the Count to **1**.
- **Property:** The name of the property to store the text.
- **Filter Run Results:** Displays the property and values set as a result of running the filter.

2. Click the Run Filter button to execute the filter and see the results in the Filter Run Results.

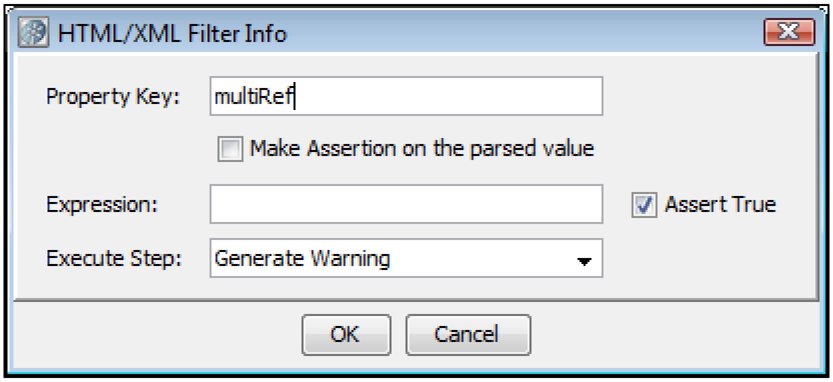
Creating the filter directly from the response page



Response

```
soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    soapenv:Body
        ns1:deleteUserResponse
            soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            xmlns:ns1="http://webservices.examples.itko.com"
            deleteUserReturn
                href="#id0"
        multiRef
            id="id0"
            soapenc:root="0"
            soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
            xsi:type="xsd:boolean"
            true
```

1. From the DOM Tree view select the attribute whose value you want to store in a property.
2. After it is selected, select **Generate Filter for Attribute or Text.**
3. Enter the property name in the dialog window.


HTML/XML Filter Info

Property Key: multiRef

□ Make Assertion on the parsed value

Expression:

☑ Assert True

Execute Step: Generate Warning

Assertions can also be added here.

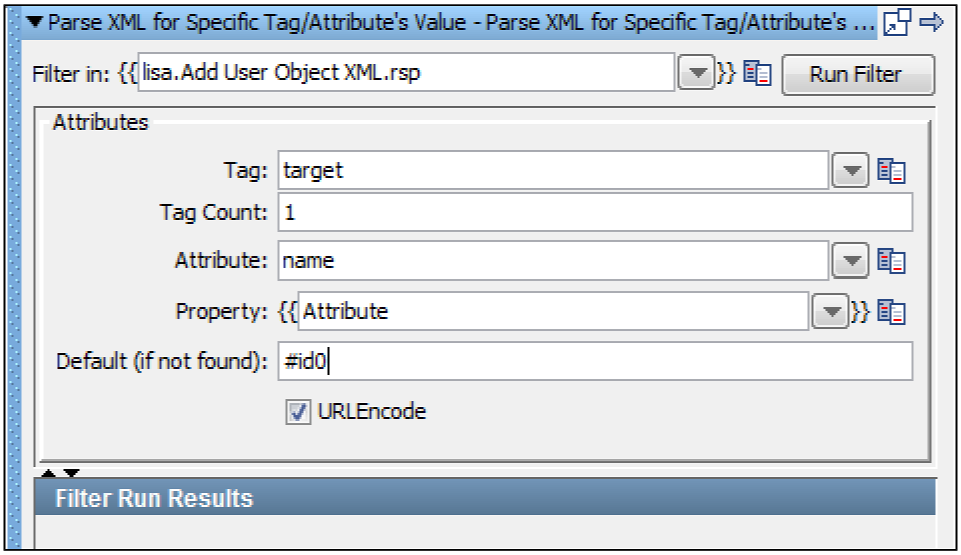
Read Attribute from XML Tag

(Also known as Parse XML for specific Tag/Attribute's Value)

The **Read Attribute from XML Tag** filter lets you store the text of a specific attribute in a property. The attribute can occur anywhere in the result.

This filter can be created in two ways: either as a manual filter from the filter list, or by using the embedded filter commands on an XML response.

Creating the filter manually


Parse XML for Specific Tag/Attribute's Value - Parse XML for Specific Tag/Attribute's ...

Filter in: {{lisa.Add User Object XML.rsp}}

Attributes

Tag: target

Tag Count: 1

Attribute: name

Property: {{Attribute}}

Default (if not found): #id0

☑ URLEncode

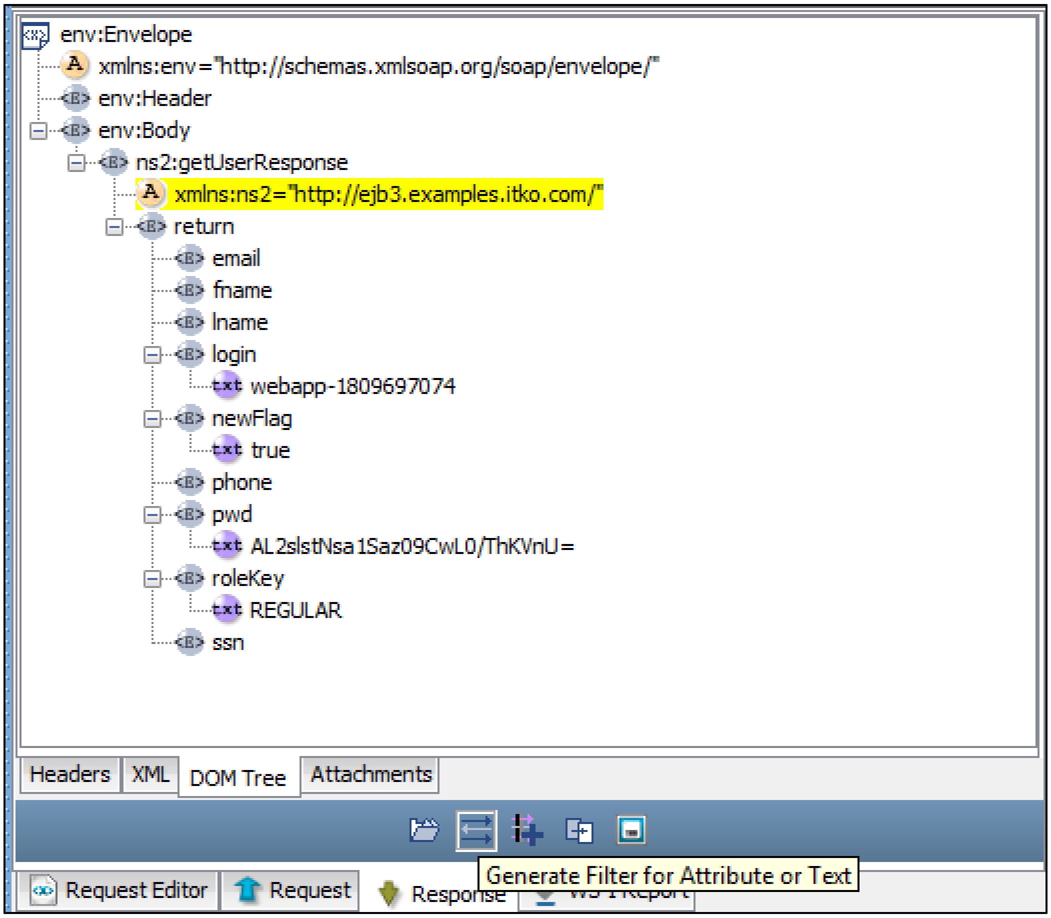
Enter the following parameters:

- **Filter in**: Where to apply the filter. The previous illustration shows **lisa.Add User Object XML.rsp**, which means that the filter will be applied to the response of the Add User Object step.
- **Tag**: The name of the XML tag; for example, **target**.
- **Tag Count**: The occurrence of the tag from the top of response; for the first tag enter **1**.
- **Attribute**: The name of the attribute to filter; for the href attribute enter **href**.
- **Property**: The property in which to store the value.
- **Default**: The value to use if the attribute value is not found.
- **URLEncode**: When checked, property value is URLEncoded.
- **Filter Run Results:** Displays the property and values set as a result of running the filter.

Click the Run Filter button to execute the filter and see the results in the Filter Run Results.

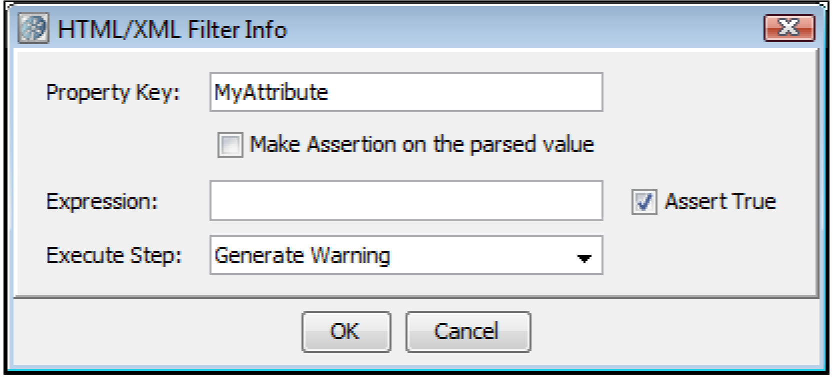
Creating the filter from the response page

1. Display the step response that contains the XML.



```
env:Envelope
  xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
  env:Header
  env:Body
    ns2:getUserResponse
      xmlns:ns2="http://ejb3.examples.itko.com/"
      return
        email
        fname
        lname
        login
          webapp-1809697074
        newFlag
          true
        phone
        pwd
          AL2slstNsa1Saz09CwL0/ThKVnU=
        roleKey
          REGULAR
        ssn
```

2. From the DOM Tree view, select the attribute whose value you want to store in a property.
3. When it is highlighted, select **Generate Filter for Attribute or Text**.
4. Enter the property name in the window.



HTML/XML Filter Info

Property Key: MyAttribute

□ Make Assertion on the parsed value

Expression:

☑ Assert True

Execute Step: Generate Warning

You can also add an assertion at this point if you want. A Property Value Expression Assertion can be added to this step.

Parse XML Result for LISA Tag

The **Parse XML Result for LISA Tags** filter provides a way for developers to test-enable their XML applications. For an in-depth study on test-enabling, see the *Developer's Guide (SDK)*.

This filter provides the ability to insert LISAPROP tags into your XML page. The LISAPROP tag has two attributes: name and value. The LISAPROP tags function only to discretely provide valuable information about your XML to a tester. An example of a LISAPROP might be:

<LISAPROP name="FIRST_USER" value="sbellum">.

If a tester has installed this type of filter, the property "FIRST_USER" will automatically be assigned the value "sbellum". This removes any need on behalf of the tester to parse for this value. This type of filter helps a developer make the testing easier.

The XML may not contain the information needed to perform proper validation, or that information is very difficult to parse. Even when it is there, the parsing can become incorrect due to subtle changes in the XML that is generated. This LISAPROP filter can resolve many tedious parsing issues.

There are no parameters required.

Choose Random XML Attribute

The **Choose Random XML Attribute** filter lets you store the text of a random selection from a set in a property. The attribute can occur anywhere in the result. This filter works exactly like Parse HTML Result and Select Random Attribute Value.

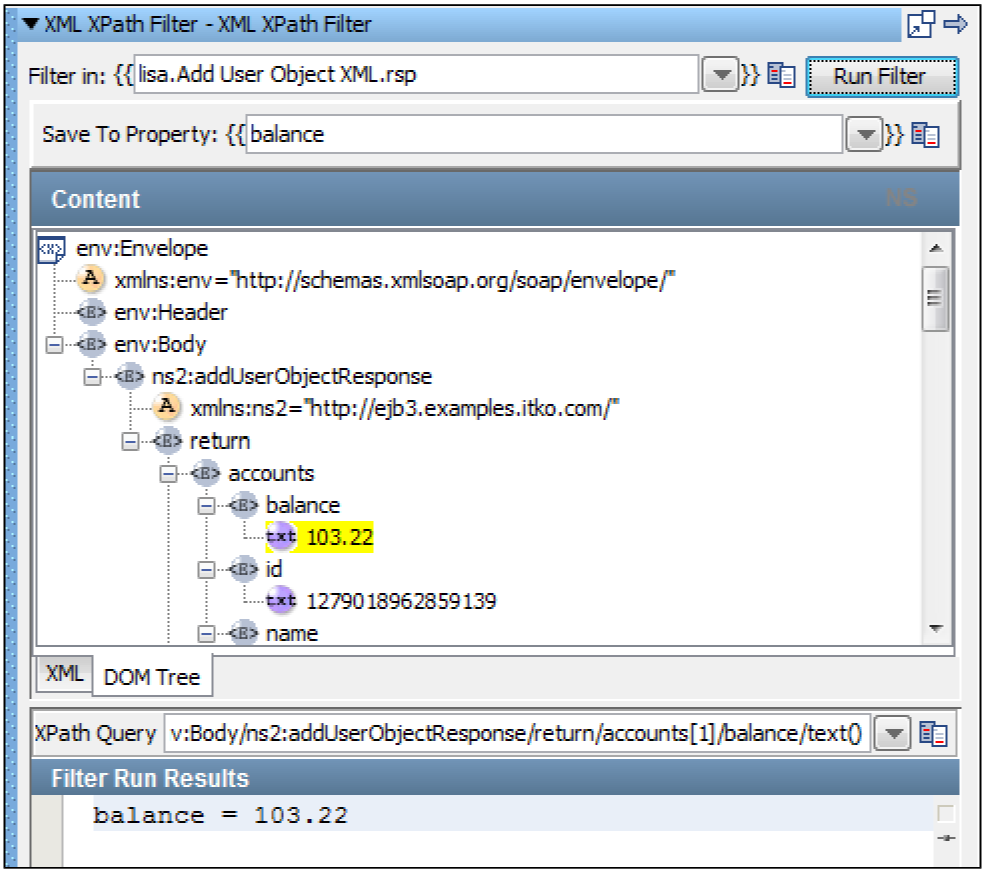
XML XPath Filter

The **XML XPath** filter lets you use an XPath query that will be run on a property, or the last response and store it in a property. When this filter is selected, the last response is loaded into the content panel.

The response can be viewed as an XML document or as a DOM tree. However, the XPath selection can be made only from the DOM tree.

Construct the XPath query by using one of the following methods:

- Manually enter the XPath expression in the XPath Query text box.
- Select an element from the DOM tree and let LISA construct the XPath expression.
- Select an element from the DOM tree, and then edit the XPath that is constructed. For example, you may want to modify it to use a LISA property, or a counter data set.



Filter in: {{lisa.Add User Object XML.rsp}}

Save To Property: {{balance}}

```
env:Envelope
  xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
  env:Header
  env:Body
    ns2:addUserObjectResponse
      xmlns:ns2="http://ejb3.examples.itko.com/"
      return
        accounts
          balance
            103.22
          id
            1279018962859139
          name
```

XPath Query v:Body/ns2:addUserObjectResponse/return/accounts[1]/balance/text()

```
balance = 103.22
```

Enter the following parameters:

- **Filter In**: Enter the last response or a named property.
- **Save To Property**: The property in which to store the result of the XPath query.

Now construct the XPath query using one of the methods described earlier.

After an XPath query has been constructed, test it by clicking Run Filter. The results of the query appear in the **Filter Run Results** pane.

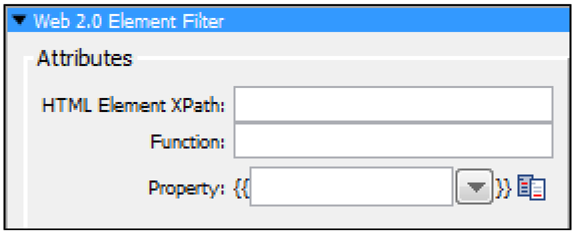
The **lisa.xml.xpath.computeXPath.alwaysUseLocalName** property controls whether the XPath **local-name()** function is always used during XPath generation. The default value is **false**, which means that the **local-name()** function is used only when necessary. To generate an XPath that will work regardless of an XML node's namespace, set the value to **true**.

Web 2.0 Filters

These are the filters available in the Web 2.0 Filters list for any test step:

Web 2.0 Element Filter

The **Web 2.0 Element Filter** lets you retrieve an HTML element from the response and store it in a property.

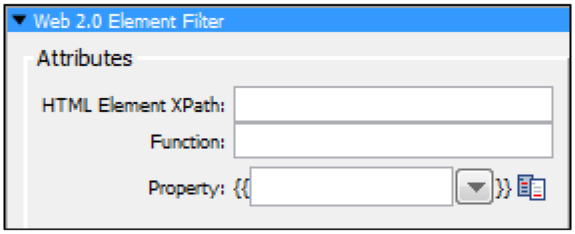


Enter the following parameters:

- **HTML Element XPath**: The XPath expression that uniquely identifies the DOM element that was the target of the event.
- **Function**: Name of predefined Web 2.0 function.
- **Property**: The name of the property to store the result.

Web 2.0 Text Filter

The **Web 2.0 Text Filter** lets you retrieve text from the response using an XPath expression and then a regular expression, and then store the text in a property.

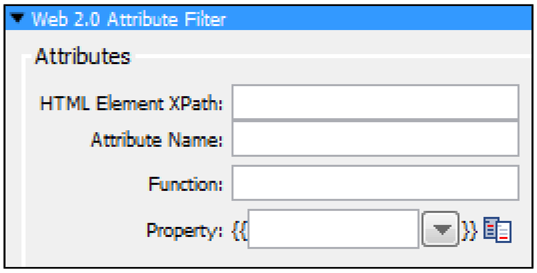


Enter the following parameters:

- **HTML Element XPath**: The XPath expression that uniquely identifies the DOM element that was the target of the event.
- **Regular Expression**: A regular expression that is applied to the result of the filter to further control what gets returned.
- **Function**: The name of a predefined Web 2.0 function.
- **Property**: The name of the property to store the result.

Web 2.0 Attribute Filter

The **Web 2.0 Attribute Filter** lets you retrieve an HTML attribute value from the response and store it in a property.

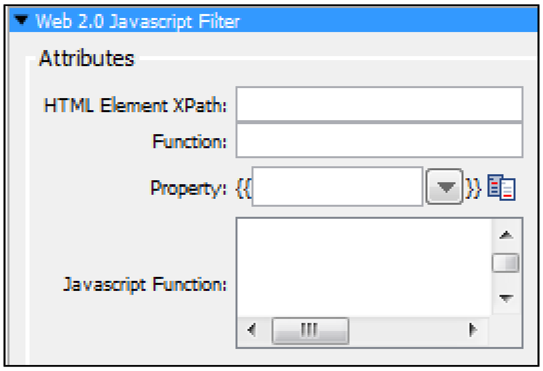


Enter the following parameters:

- **HTML Element XPath**: The XPath expression that uniquely identifies the DOM element that was the target of the event.
- **Attribute Name**:The optional DOM attribute name used to execute DOM attribute filters.
- **Function**: The name of a predefined Web 2.0 function.
- **Property**: The name of the property to store the last response.

Web 2.0 JavaScript Filter

The **Web 2.0 JavaScript Filter** lets you retrieve arbitrary information from the response using a JavaScript expression.

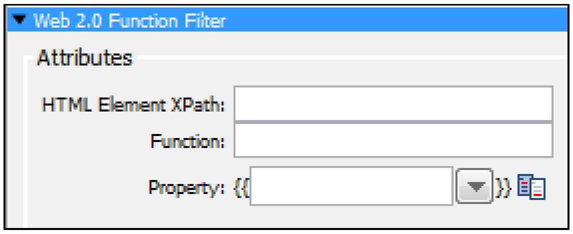


Enter the following parameters:

- **HTML Element XPath**: The XPath expression that uniquely identifies the DOM element that was the target of the event.
- **Javascript Function**: A snippet of valid JavaScript code that gets executed by the filter. It should return an object.
- **Function**: The name of a predefined Web 2.0 function.
- **Property**: The name of the property to store the result.

Web 2.0 Function Filter

The **Web 2.0 Function Filter** lets you execute a predefined Web 2.0 function.

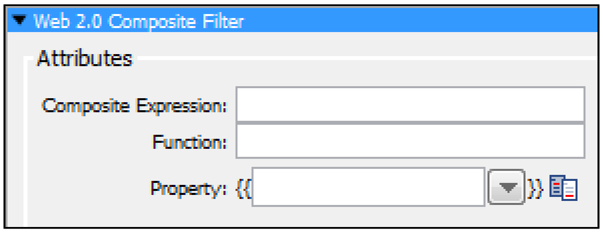


Enter the following parameters:

- **HTML Element XPath**: The XPath expression that uniquely identifies the DOM element that was the target of the event.
- **Function**: The name of predefined Web 2.0 function.
- **Property**: The name of the property to store the last response.

Web 2.0 Composite Filter

The **Web 2.0 Composite Filter** lets you combine other Web 2.0 filters using a string or an arithmetic expression.



Enter the following parameters:

- **Composite Expression**: Expression with Web 2.0 filters using a string or arithmetic expression.
- **Function**: Name of a predefined Web 2.0 function.
- **Property**: The name of the property to store the result.

Java Filters

These are the filters available in the Java Filters list for any test step.

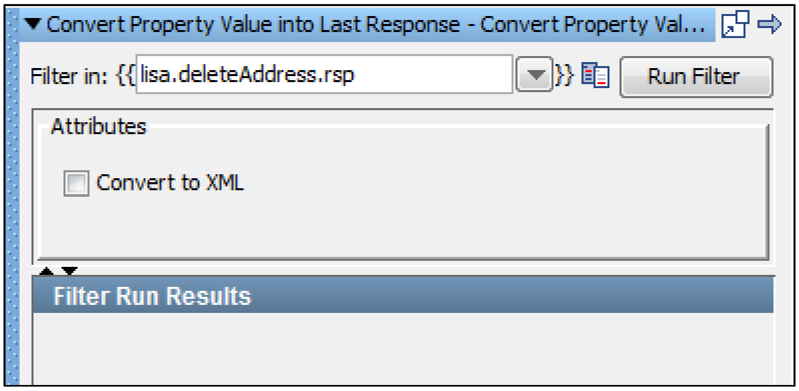
Override "Last Response" Property
Save Property Value to File
Store Step Response

Java Override "Last Response" Property Filter

There is a special property known as Last Response, which contains the response from the previous step. For example, if the previous response was an HTTP step, the last response will be a web page that was returned.

The **Override "Last Response" Property** filter should be used if you want the last response to be something other than the default value. This filter lets you replace the current value of the last response with the value of an existing LISA property.

Click the filter to open its editor.



Convert Property Value into Last Response - Convert Property Val...

Filter in: {{lisa.deleteAddress.rsp}}

□ Convert to XML

Enter the following parameters:

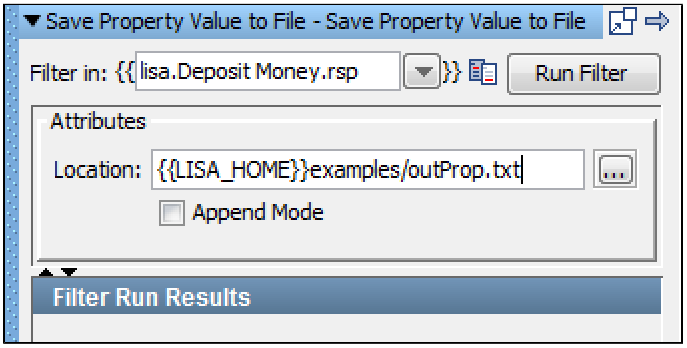
- **Filter in**: The name of the property you want considered as the step's last response. The property should be in the pull-down menu; if not, you can enter it. It must be an existing property.
- **Convert to XML**: Check this if you want the response to be converted to valid XML.
- **Filter Run Results**: Displays the property and values set as a result of running the filter.

Click the Run Filter button to execute the filter and see the results in the Filter Run Results section.

For more detailed information see Utility Filters: Override Last Response Property.

Java Save Property Value to File Filter

The **Save Property Value to File** filter lets you save the value of an existing property to a file in your file system.



Save Property Value to File - Save Property Value to File

Filter in: {{lisa.Deposit Money.rsp}}

Location: {{LISA_HOME}}examples/outProp.txt

□ Append Mode

Enter the following parameters:

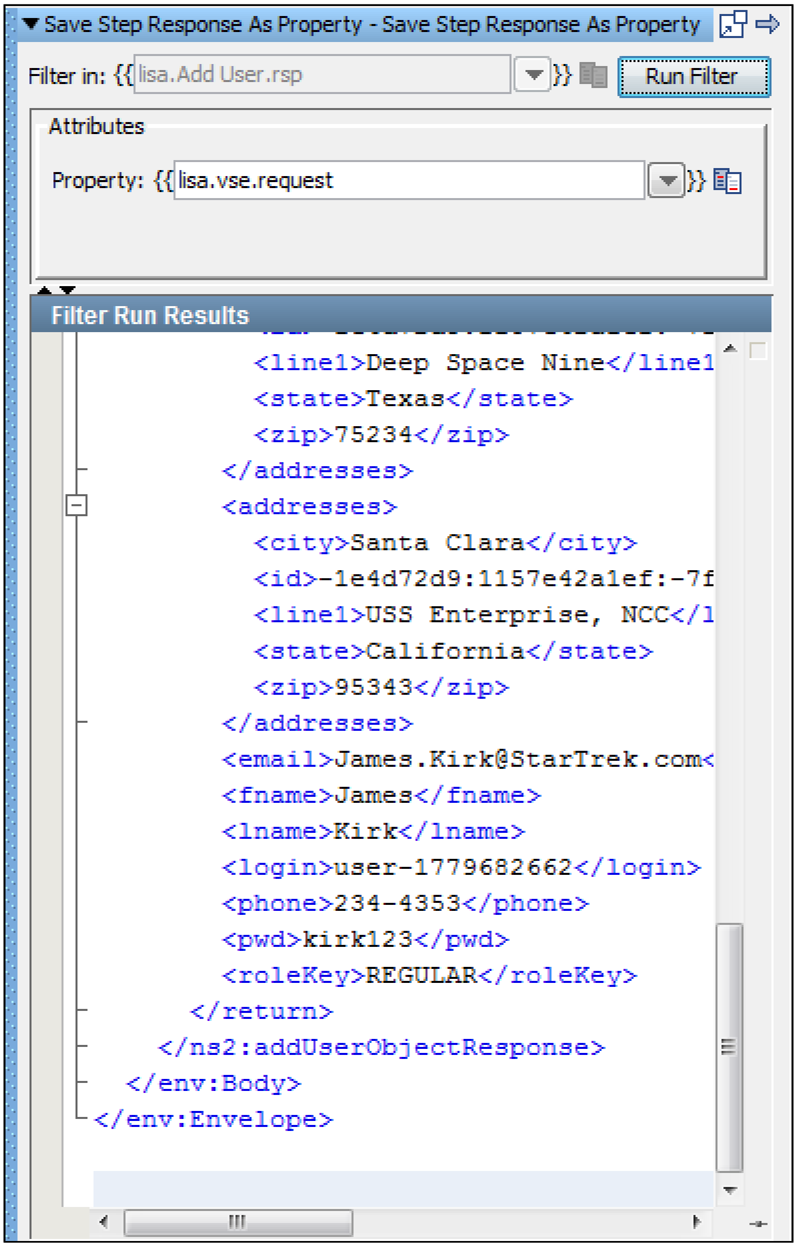
- **Filter in:** The name of the property whose value you want to write to the file.
- **Location:** The path name of the file to write the value to. You can browse to the file. You can use properties in the location.
- **Append Mode:** Select this check box if you want to append the information to an existing file.
- **Filter Run Results:** Displays the property and values set as a result of running the filter.

Click the Run Filter button to execute the filter and see the results in the Filter Run Results.

Java Store Step Response Filter

(Also known as Save Step Response as Property Filter)

The **Store Step Response as Property** filter lets you save the last response as a property, for future use.


Save Step Response As Property - Save Step Response As Property
Run Filter
Attributes
Filter Run Results

Filter in: {{lisa.Add User.rsp}}

Property: {{lisa.vse.request}}

```
          <line1>Deep Space Nine</line1
          <state>Texas</state>
          <zip>75234</zip>
        </addresses>
        <addresses>
          <city>Santa Clara</city>
          <id>-1e4d72d9:1157e42a1ef:-7f
          <line1>USS Enterprise, NCC</l
          <state>California</state>
          <zip>95343</zip>
        </addresses>
        <email>James.Kirk@StarTrek.com<
        <fname>James</fname>
        <lname>Kirk</lname>
        <login>user-1779682662</login>
        <phone>234-4353</phone>
        <pwd>kirk123</pwd>
        <roleKey>REGULAR</roleKey>
      </return>
    </ns2:addUserObjectResponse>
  </env:Body>
</env:Envelope>
```

Enter the following parameter:

- **Filter in:** Enter the response to apply the filter to. The previous illustration shows **lisa.Add User.rsp**, which means that the filter will be applied to the response of **Add User**. You cannot change this value for this filter.
- **Property:** The name of the property to store the last response.
- **Filter Run Results:** Displays the property and values set as a result of running the filter.

Click the Run Filter button to execute the filter and see the results in the Filter Run Results.

VSE Filters

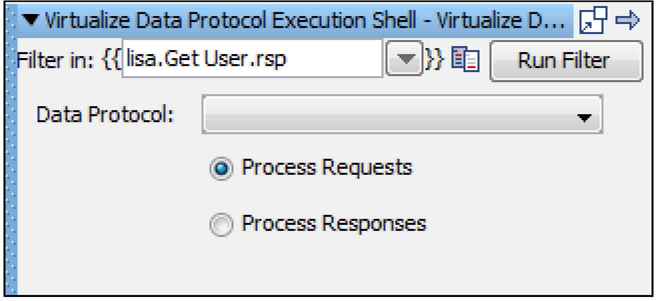
This is the filter available in the VSE Filters list for any test step.

Data Protocol Filter

Data Protocol Filter

The **Data Protocol** filter is used on protocol-specific listen steps for virtual models. It provides the necessary wrapper for a data protocol to act as a filter, the appropriate way for things to work in the run-time side of VSE.

1. Click the filter to open its editor.



Filter in: {{lisa.Get User.rsp}}

Data Protocol:

● Process Requests

○ Process Responses

2. Enter the following parameters:

- **Filter in:** Enter the response to apply the filter to. The previous illustration shows **lisa.Get User.rsp**, which means that the filter will be applied to the response of **Get User**. You cannot change this value for this filter.
- **Data Protocol**: Select the appropriate data protocol to be used from the drop-down list.
- **Process Requests**: Check to see the process request.
- **Process Responses**: Check to see the process response.

3. Click the Run Filter button to execute the filter and see the results in the Filter Run Results section.

Pathfinder Filters

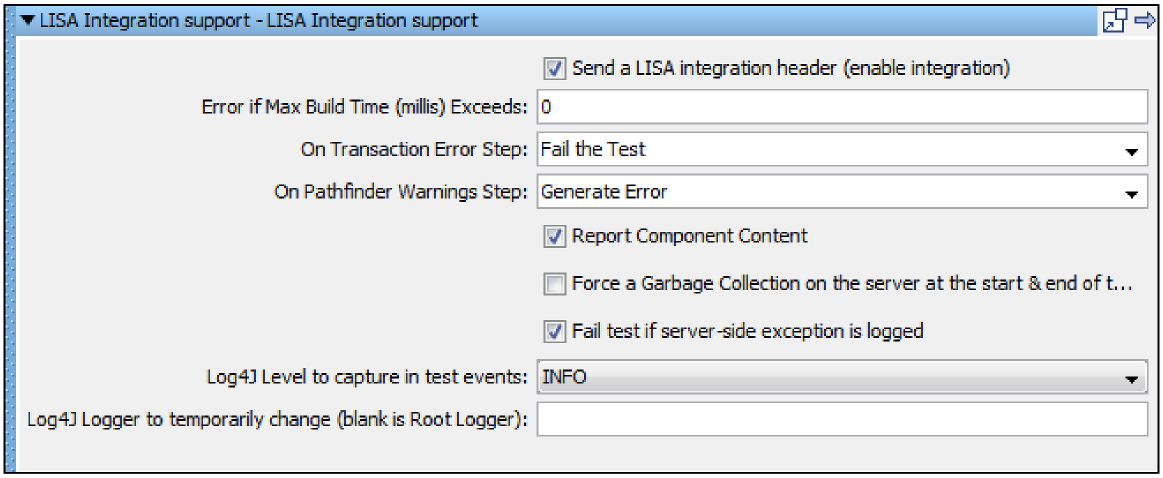
These are the filters available in the Pathfinder Filters list for any test step.

LISA Integration Support for Pathfinder
LISA Integration Support for webMethods Integration Server

LISA Integration Support for Pathfinder

The **LISA Integration Support for Pathfinder** filter is a common filter to enable Pathfinder for all the technologies supported by LISA. This filter collects additional information from a Pathfinder application.

Currently LISA supports integration with web services, JMS, servlets, EJB and Java objects.



☑ Send a LISA integration header (enable integration)

Error if Max Build Time (millis) Exceeds: 0

On Transaction Error Step: Fail the Test

On Pathfinder Warnings Step: Generate Error

☑ Report Component Content

☐ Force a Garbage Collection on the server at the start & end of t...

☑ Fail test if server-side exception is logged

Log4J Level to capture in test events: INFO

Log4J Logger to temporarily change (blank is Root Logger):

Enter the following parameters:

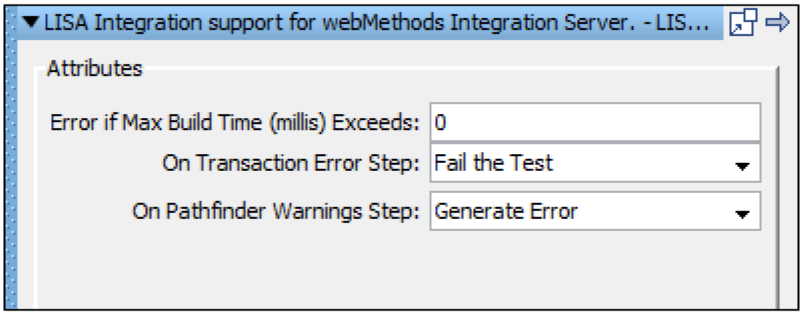
- **Error if Max Build Time (millis) Exceeds**: Enter build time in milliseconds. If it exceeds the time specified, an error will be generated.
- **On Transaction Error Step**: Select the step to redirect to on transaction error after filter is set to run.
- **On Pathfinder Warning Step**: Select the step to redirect to on Pathfinder Warning Step after filter is set to run.
- **Report Component Content** check box: Generate a report of the component content.
- **Force a Garbage Collection on the server at the start & end of the request** check box: Forces a garbage collection on the server at

the start and end of the request.

- **Fail test if server-side exception is logged** check box: Fail the test case if exception is thrown at the server side.
- **Log4J level to capture in the test events**: Select the log4J level that is to be captured in the test events.
- **Log4J Logger to temporarily change (blank is Root Logger)**: Enter the name of the logger.

LISA Integration Support for webMethods Integration Server

The **LISA Integration Support for webMethods Integration Server** filter collects additional information from Pathfinder-enabled webMethods Integration Server.


LISA Integration support for webMethods Integration Server. - LIS...
Attributes

Error if Max Build Time (millis) Exceeds: 0
On Transaction Error Step: Fail the Test
On Pathfinder Warnings Step: Generate Error

Enter the following parameters:

- **Error if Max Build Time(millis) Exceeds**: Enter build time in milliseconds. If it exceeds the time specified, an error will be generated.
- **On Transaction Error Step**: Select the step to redirect to on transaction error after filter is set to run.
- **On Pathfinder Warning Step**: Select the step to redirect to on Pathfinder Warning Step after filter is set to run.

Assertions

An assertion is a LISA code element that runs after a step and all its filters have run, to verify that the results from running the step match expectations.

The result of an assertion is a Boolean value (true or false).

The outcome may determine whether the test step passes or fails, and also determines the next step to run in the test case. An assertion is used to dynamically alter the test case workflow by introducing conditional logic (branching) into the workflow – very much like an 'if' conditional block programming.

For example, you might create an assertion for a JDBC step that helps ensure that only one row in the result set contains a specific user name. If the results of the JDBC step contain more than one row, the assertion changes the next step to execute. In this way, **an assertion provides conditional functionality.**

The test case flow is usually modeled with one of the following two possibilities:

- The next step defined for each step is the next logical step in the test case – in which case the assertions are pointing to failure; or
- The next step is set to fail, and the assertions all point to the next logical step.

The choice will depend, for the most part, on the actual logic being employed.

⚠ If an assertion references an unresolved property, a model definition error will be raised. The model definition error will not cause the test to terminate, but it will caution the test author that an unresolved property was encountered. The problem created by an unresolved property is that an assertion cannot give the proper verdict because the assertion does not have enough information to do so, resulting in false positives or false negatives. (Most assertions return "false" as the verdict if an unresolved property is encountered, but that is not an enforced rule.) By running a test in the ITR and inspecting the test events panel for model definition errors, a test author can determine if any unresolved properties exist.

You can add as many assertions as you need, giving you the capability to build a workflow of any complexity that you need. **Nothing except assertions can change the LISA workflow**.

⚠ Assertions are executed in the order that they appear, and the workflow logic will usually depend on the order that the assertions are applied.

After an assertion fires, the next step can be configured and is determined by that assertion, and the remaining assertions are ignored. An **event** is generated every time an assertion is evaluated and fired.

Global and Step Assertions

Like filters, assertions can be applied as a global assertion, that is, to the entire test case cycle or as a step assertion, where they will be applied only to a particular step.

The following topics are available in this chapter.

Adding an Assertion

There are several ways to add assertions into the test case.

All methods except the first imply using assertions that are available for selection in the specific test step editor.

Adding an Assertion Manually

To add an assertion manually

Select the assertion type from a list and enter the parameters for the assertion.

There are two types of manual assertions:

- Global assertions are defined at the test case level. This type of assertion will be applicable to all the steps in the test case and is automatically run for every step in the test case, unless a given node is instructed otherwise.
- Step assertions are defined at the test step level. This type of assertion will be applicable only to that step and will execute for that step only.

Adding a Global Assertion

To add a global assertion

Open a test case and in the right panel click the **Global Assertions** element.

You can apply the following types of global assertions:

HTTP

- Simple Web Assertion
- Check Links on Web Responses

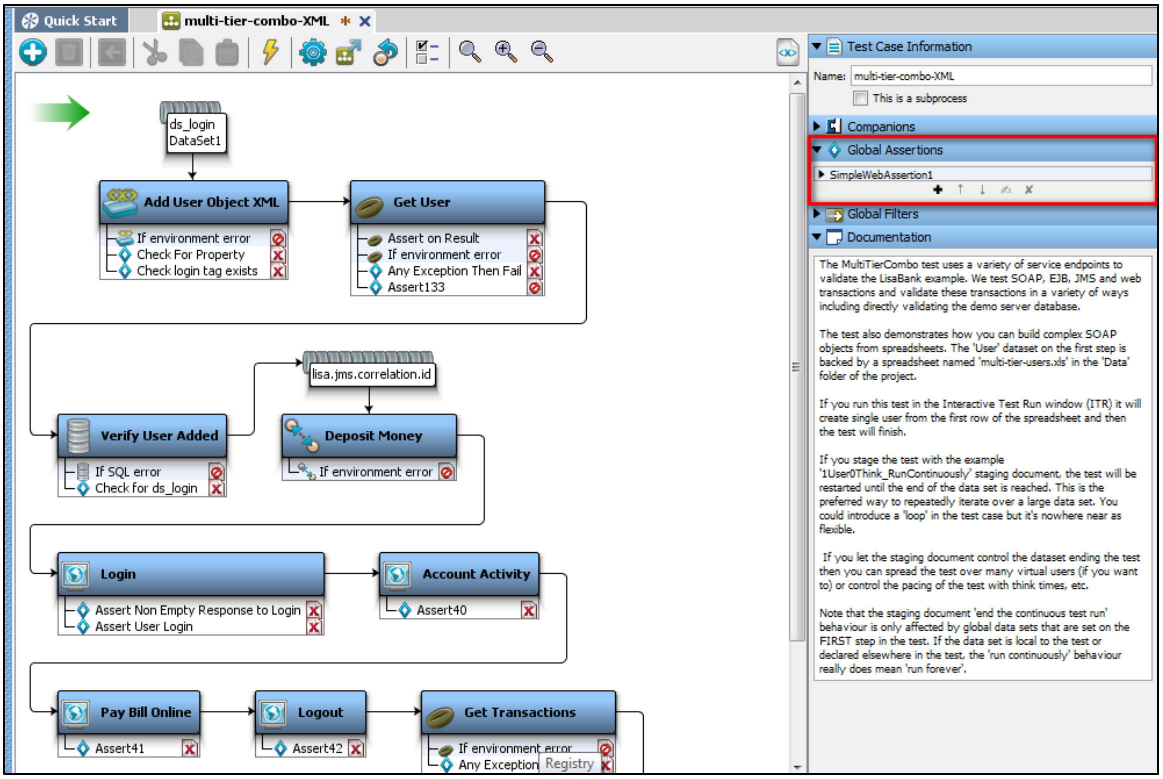
XML

- Ensure Step Response Time

Other

- Ensure Result Contains Expression
- Ensure Step Response Time
- Scan a File for Content

The following image shows a global assertion applied to the **multi-tier-combo** test case.



Name: multi-tier-combo-XML

□ This is a subprocess

The MultiTierCombo test uses a variety of service endpoints to validate the LisaBank example. We test SOAP, EJB, JMS and web transactions and validate these transactions in a variety of ways including directly validating the demo server database.

The test also demonstrates how you can build complex SOAP objects from spreadsheets. The 'User' dataset on the first step is backed by a spreadsheet named 'multi-tier-users.xls' in the 'Data' folder of the project.

If you run this test in the Interactive Test Run window (ITR) it will create single user from the first row of the spreadsheet and then the test will finish.

If you stage the test with the example '1User0Think_RunContinuously' staging document, the test will be restarted until the end of the data set is reached. This is the preferred way to repeatedly iterate over a large data set. You could introduce a 'loop' in the test case but it's nowhere near as flexible.

If you let the staging document control the dataset ending the test then you can spread the test over many virtual users (if you want to) or control the pacing of the test with think times, etc.

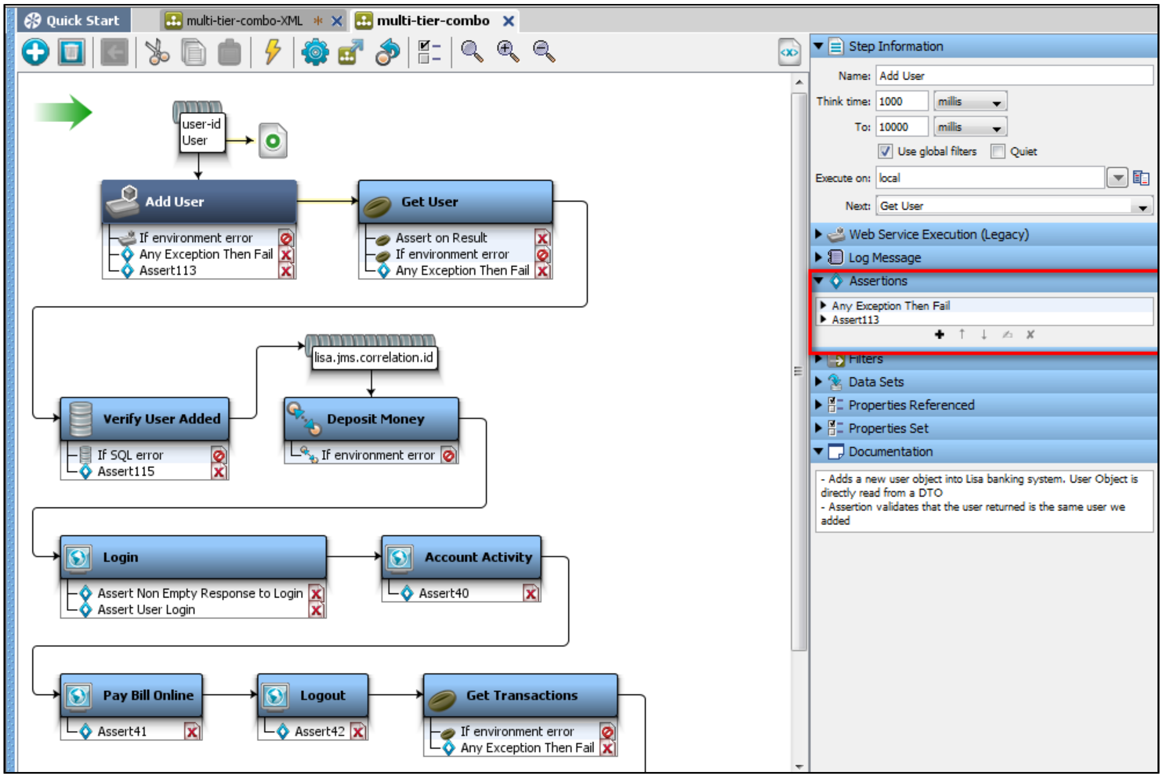
Note that the staging document 'end the continuous test run' behaviour is only affected by global data sets that are set on the FIRST step in the test. If the data set is local to the test or declared elsewhere in the test, the 'run continuously' behaviour really does mean 'run forever'.

Adding a Step Assertion

To add a step assertion

Select the step for which you want to apply the assertion and in the right panel click the **Assertion** element, or right-click the step and select **Add Assertion** and select the appropriate assertion for the step.

The following image shows step assertions applied to the **Add user** step in the **multi-tier-combo** test case.



Name: Add User

Think time: 1000 millis

To: 10000 millis

☑ Use global filters ☐ Quiet

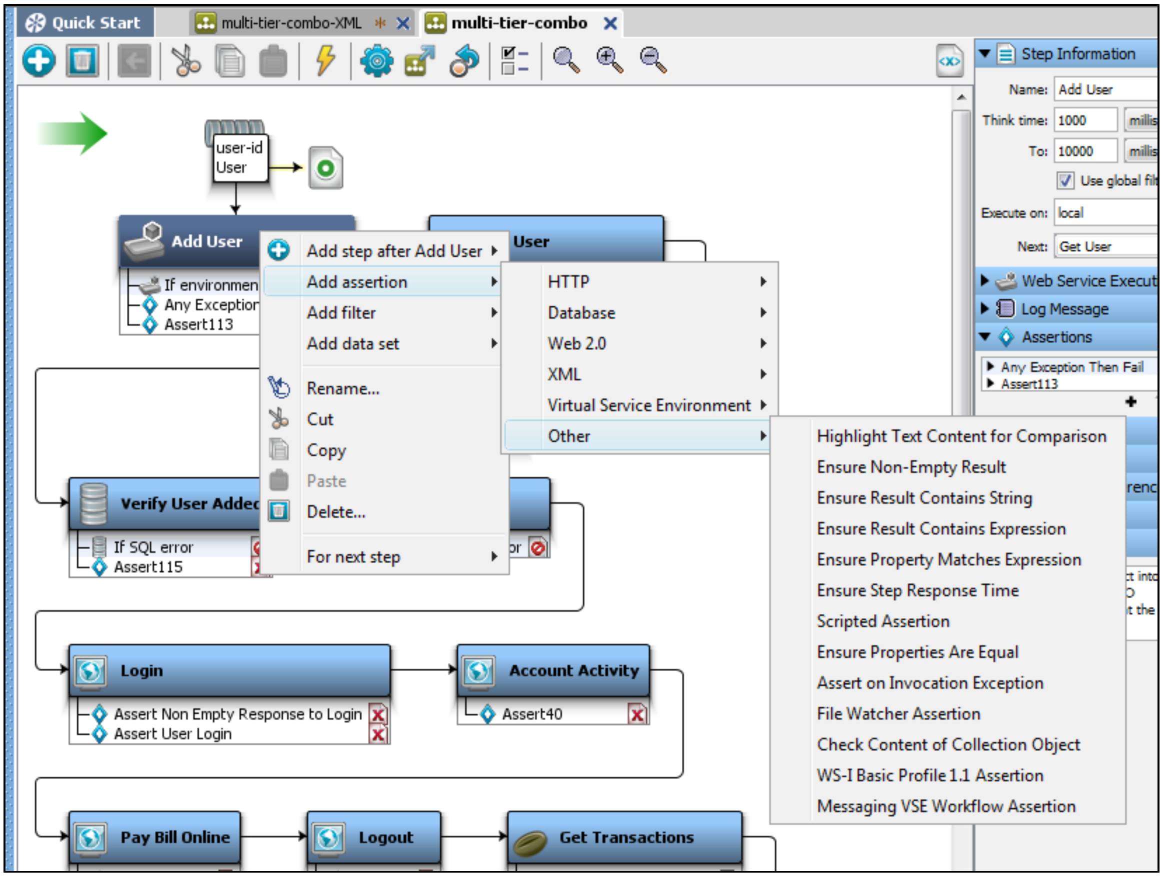
Execute on: local

Next: Get User

- Adds a new user object into Lisa banking system. User Object is directly read from a DTO
- Assertion validates that the user returned is the same user we added

To add an assertion, click the **Add** ✚ icon on the assertion toolbar.

Or, you can right-click a step in the model editor to add an assertion. The assertion panel opens up and shows a menu of assertions that can be applied to the step.



Name: Add User

Think time: 1000 millis

To: 10000 millis

☑ Use global fil

Execute on: local

Next: Get User

Selecting/Editing an Assertion

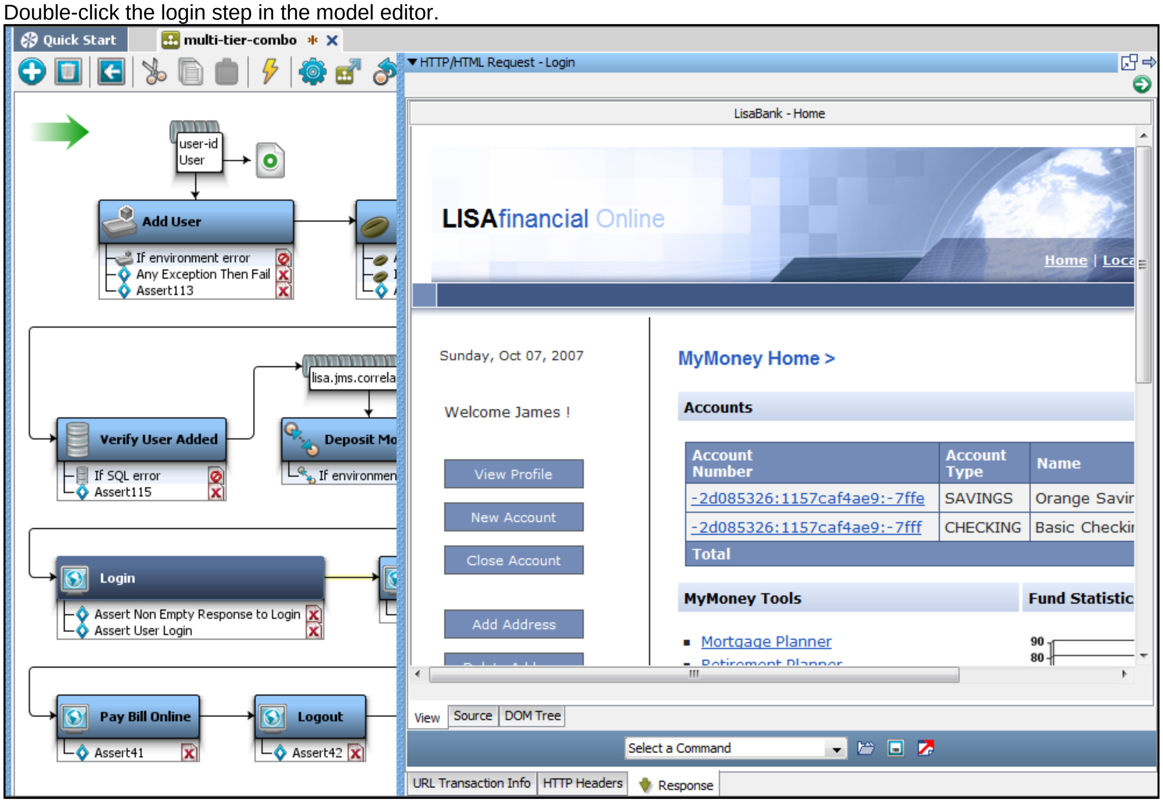
- Click the step in the model editor to which the assertion is applied and/or click the assertion related to that step in the Assertion tab.
- Double-click the assertion to open the assertion editor. The editor is unique for each type of assertion.

Adding an Assertion from an HTTP Response

When you have access to the response from an HTTP-based step, you can use the response to add an assertion directly.

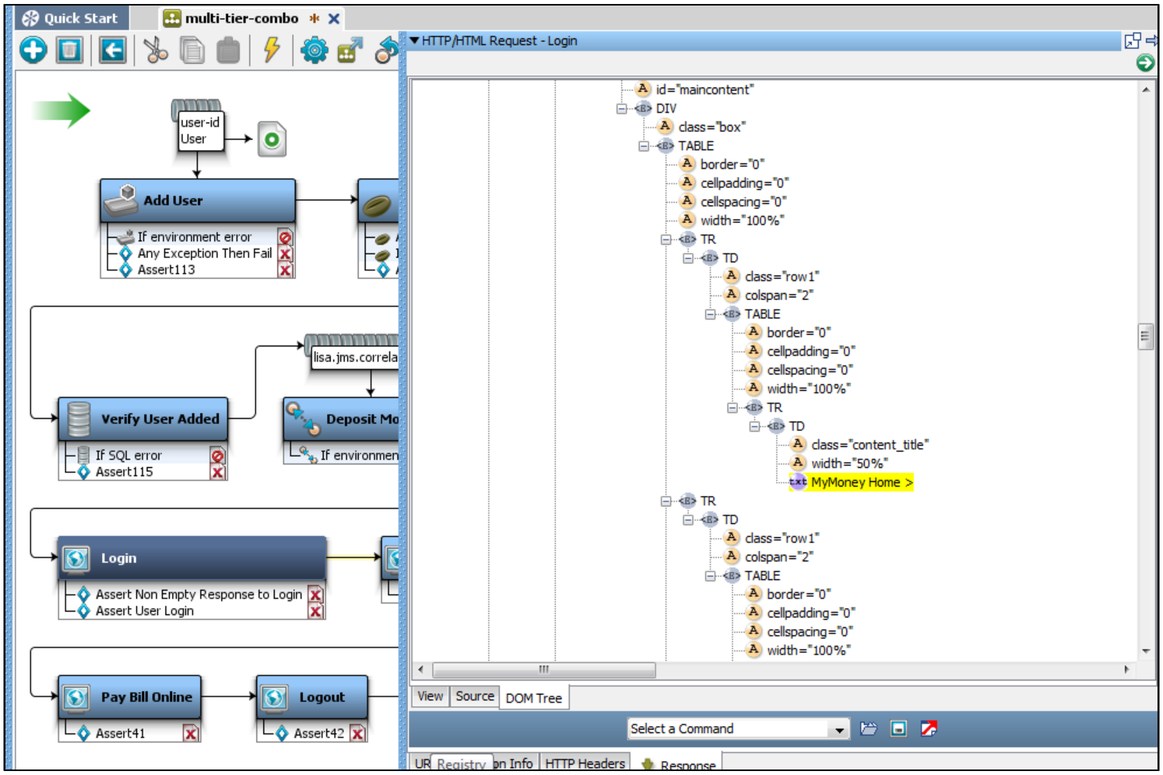
This example of the HTTP/HTML response uses the login step in the multi-tier-combo test case. The point of this example is to test whether the text "MyMoney Home" appears in the response.

1. Run the multi-tier-combo test case in the ITR.
2. Double-click the login step in the model editor.

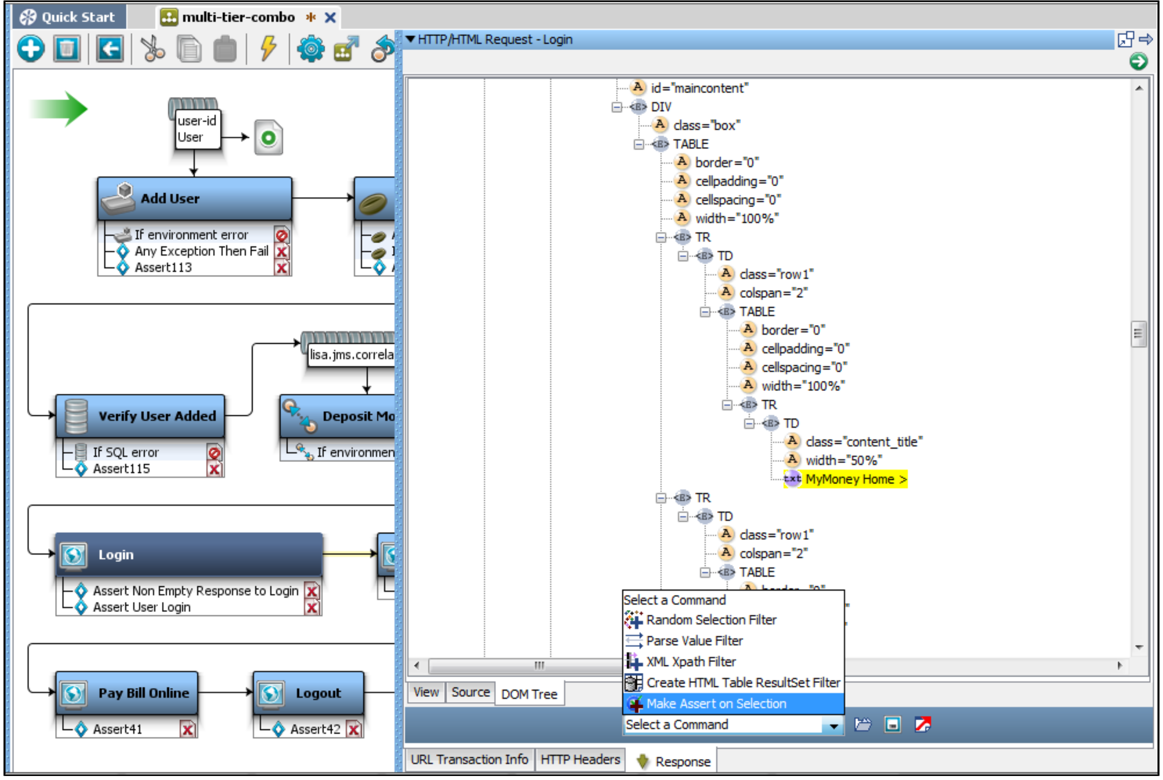


Account Number	Account Type	Name
-2d085326:1157caf4ae9:-7ffe	SAVINGS	Orange Savir
-2d085326:1157caf4ae9:-7fff	CHECKING	Basic Checkir
Total		

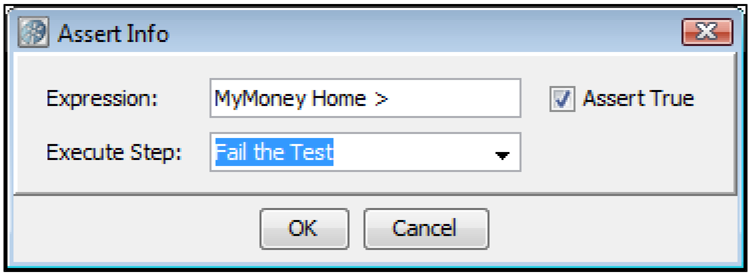
3. Select the text "MyMoney Home" in the View tab.
4. Click the DOM Tree tab to view and make sure that this text is selected in the tree.



5. From the Select a Command pull-down menu at the bottom of the panel, select Make Assert on Selection.



6. In the window that is displayed, enter the expression that the selected text should match with, and select the appropriate assertion behavior.

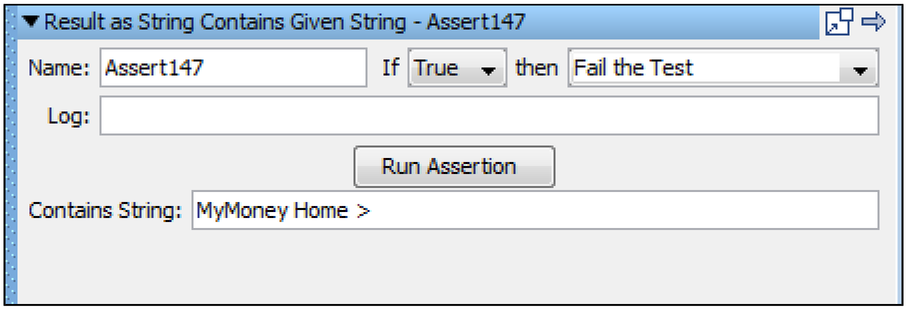

Assert Info

Expression: MyMoney Home > ☑ Assert True

Execute Step: Fail the Test

7. In this example, the assertion fires if the text "MyMoney Home" is not present, and then redirects to the fail step.
8. Click OK to save the assertion.

The assertion that was generated can be seen as an assertion in the login step.


Result as String Contains Given String - Assert147

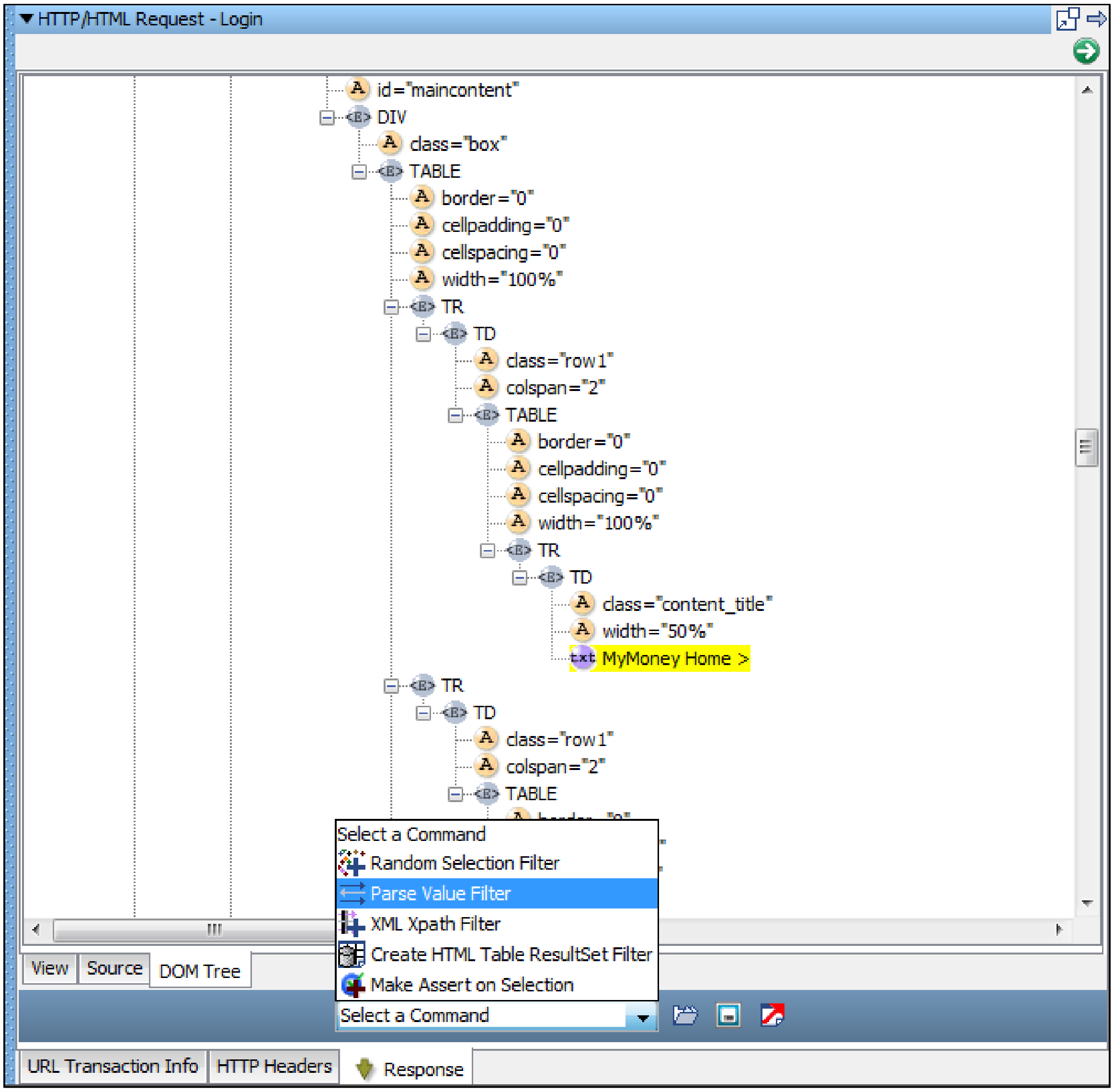
Name: Assert147 If True then Fail the Test

Log:

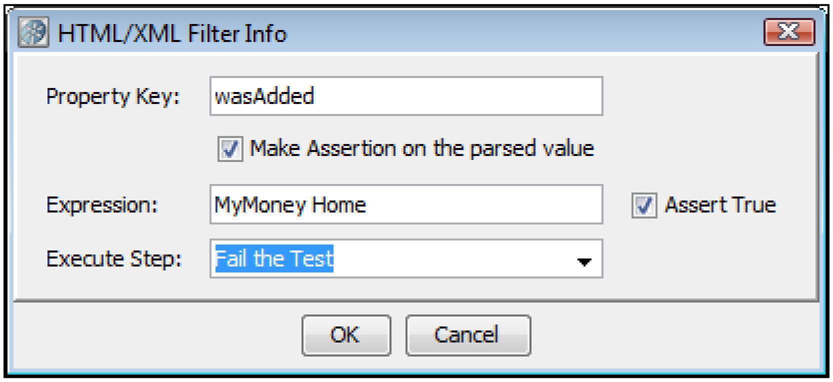
Contains String: MyMoney Home >

Running one Filter and one Assertion

Alternatively, if you wanted a filter to capture the value "MyMoney Home," and then run it as an assertion, you can use the Parse Value Filter, which can do both things.



The window displayed by the Parse Value Filter shows Property Key value is the filter to be applied and Expression is the assertion to be fired.


HTML/XML Filter Info

Property Key: wasAdded

☑ Make Assertion on the parsed value

Expression: MyMoney Home ☑ Assert True

Execute Step: Fail the Test

As a result, one filter and one assertion will be added to the login step and can be seen in the model editor.

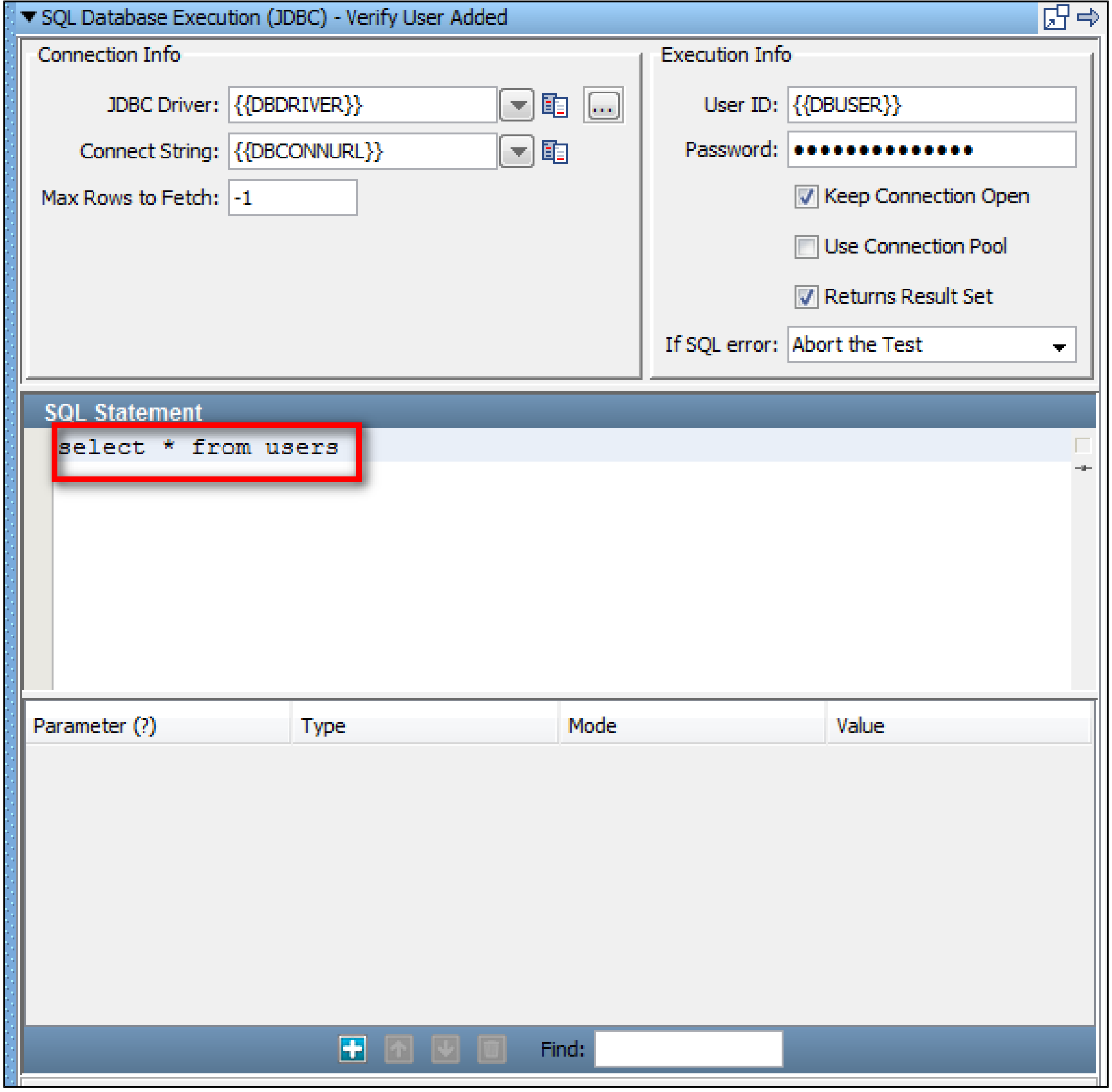
✓ The same assertion capabilities are available when an HTML response is displayed in the step editor.

Adding an Assertion from a JDBC Result Set

When you have access to the Result Set response from a JDBC step, you can use the response to add an assertion directly. The following is an example of how to add an assertion this way.

Here is an example for a result set response, using the response of Verify User Added step in multi-tier-combo test case in the examples directory (multi-tier-combo.tst).

1. Select the Verify User Added step, and double-click it to open its editor window. Edit the SQL statement to read **select * from users**.



SQL Database Execution (JDBC) - Verify User Added

Connection Info

JDBC Driver: {{DBDRIVER}}

Connect String: {{DBCONNURL}}

Max Rows to Fetch: -1

Execution Info

User ID: {{DBUSER}}

Password: ••••••••••••••

☑ Keep Connection Open

□ Use Connection Pool

☑ Returns Result Set

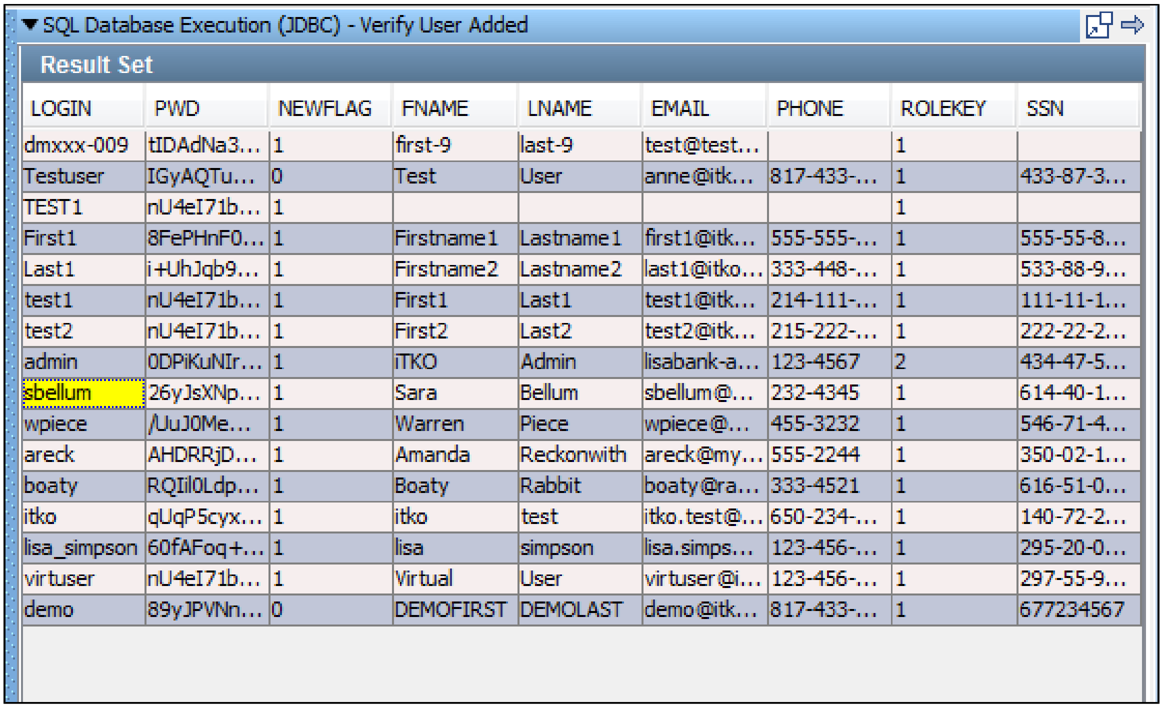
If SQL error: Abort the Test

SQL Statement

```
select * from users
```

Parameter (?)	Type	Mode	Value

2. Click the Test/Execute SQL button to run the query.
3. Select the Result Set tab and click the cell in the result set that represents the information that you want to test for (for example, **sbellum**).

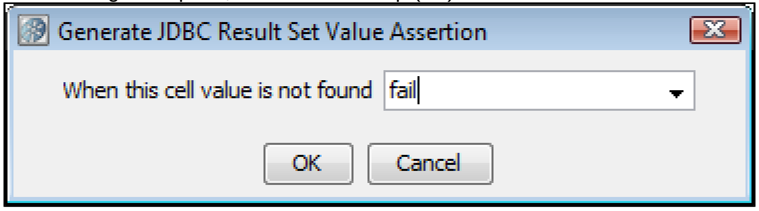


SQL Database Execution (JDBC) - Verify User Added

Result Set

LOGIN	PWD	NEWFLAG	FNAME	LNAME	EMAIL	PHONE	ROLEKEY	SSN
dmxxx-009	tIDAdNa3...	1	first-9	last-9	test@test...		1	
Testuser	IGyAQTu...	0	Test	User	anne@itk...	817-433-...	1	433-87-3...
TEST1	nU4eI71b...	1					1	
First1	8FePHnF0...	1	Firstname1	Lastname1	first1@itk...	555-555-...	1	555-55-8...
Last1	i+UhJqb9...	1	Firstname2	Lastname2	last1@itko...	333-448-...	1	533-88-9...
test1	nU4eI71b...	1	First1	Last1	test1@itk...	214-111-...	1	111-11-1...
test2	nU4eI71b...	1	First2	Last2	test2@itk...	215-222-...	1	222-22-2...
admin	0DPiKuNIr...	1	iTKO	Admin	lisabank-a...	123-4567	2	434-47-5...
sbellum	26yJsXNp...	1	Sara	Bellum	sbellum@...	232-4345	1	614-40-1...
wpiece	/UuJ0Me...	1	Warren	Piece	wpiece@...	455-3232	1	546-71-4...
areck	AHDRRjD...	1	Amanda	Reckonwith	areck@my...	555-2244	1	350-02-1...
boaty	RQIil0Ldp...	1	Boaty	Rabbit	boaty@ra...	333-4521	1	616-51-0...
itko	qUqP5cyx...	1	itko	test	itko.test@...	650-234-...	1	140-72-2...
lisa_simpson	60fAFoq+...	1	lisa	simpson	lisa.simps...	123-456-...	1	295-20-0...
virtuser	nU4eI71b...	1	Virtual	User	virtuser@i...	123-456-...	1	297-55-9...
demo	89yJPVNn...	0	DEMOFIRST	DEMOLAST	demo@itk...	817-433-...	1	677234567

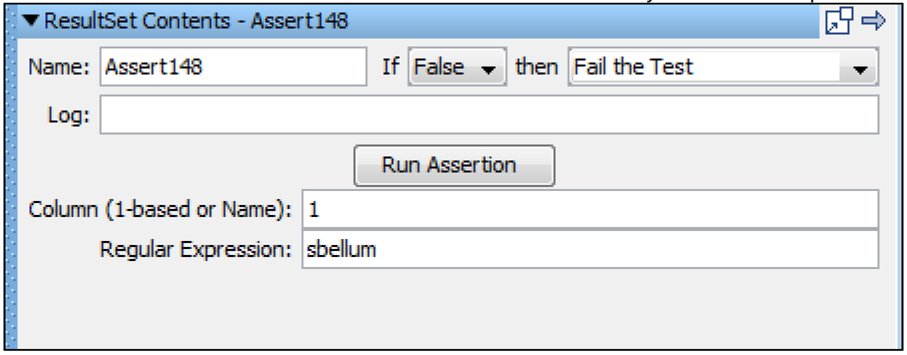
4. Click the Generate Assertions for Cell's Value icon in the toolbar below the Result set window.
   We want to test that **sbellum** appears in a cell in the **LOGIN** column.
5. In the dialog that opens, enter the test step (fail) to redirect if the value is not found:



Generate JDBC Result Set Value Assertion

When this cell value is not found: fail

LISA will create an assertion named Result Set Contents in the Verify User Added step.



ResultSet Contents - Assert148

Name: Assert148 If False then Fail the Test

Log:

Column (1-based or Name): 1

Regular Expression: sbellum

The same assertion capabilities are available when a JDBC result set is displayed in the step editor.

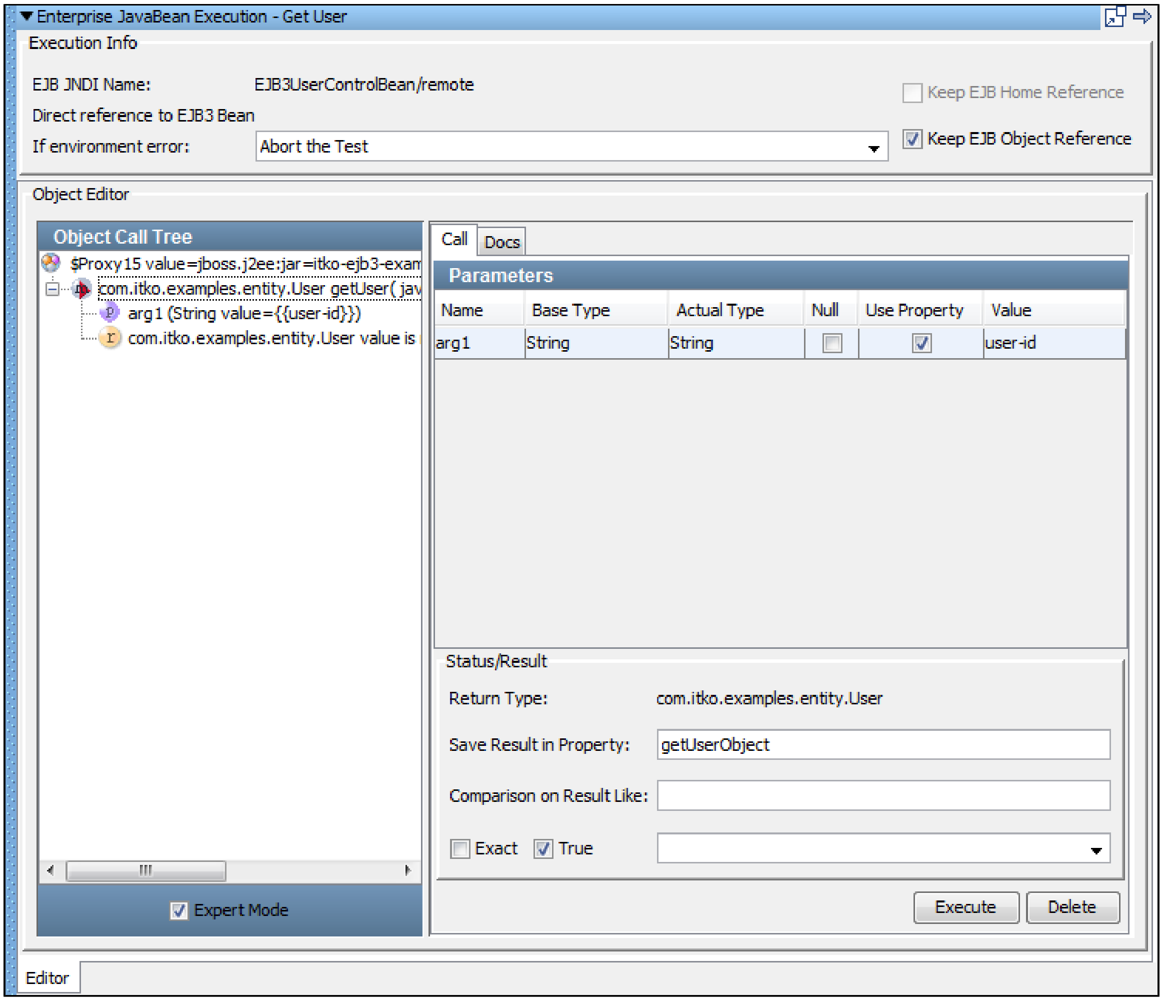
Adding an Assertion for Returned Java Object

When the result of your test step is a Java object, you can use the inline assertion panel in the Complex Object Editor to add an assertion on the returned value from the method call directly. The following is an example of how to add an assertion this way.

Here is an example of an object in the Complex Object Editor:

This example uses the get user (an EJB step) step in multi-tier-combo test case in the examples directory (multi-tier-combo.tst).

We have entered an input parameter **itko**, and executed the method call getUser. We are now about to execute the getPwd call on the UserState object that was returned from that call.



Enterprise JavaBean Execution - Get User

Execution Info

EJB JNDI Name: EJB3UserControlBean/remote

Direct reference to EJB3 Bean

If environment error: Abort the Test

□ Keep EJB Home Reference

☑ Keep EJB Object Reference

Object Editor

Object Call Tree

```
$Proxy15 value=jboss.j2ee:jar=itko-ejb3-exam
com.itko.examples.entity.User getUser( jav
    arg1 (String value={{user-id}})
    com.itko.examples.entity.User value is
```

☑ Expert Mode

Parameters

Name	Base Type	Actual Type	Null	Use Property	Value
arg1	String	String	□	☑	user-id

Status/Result

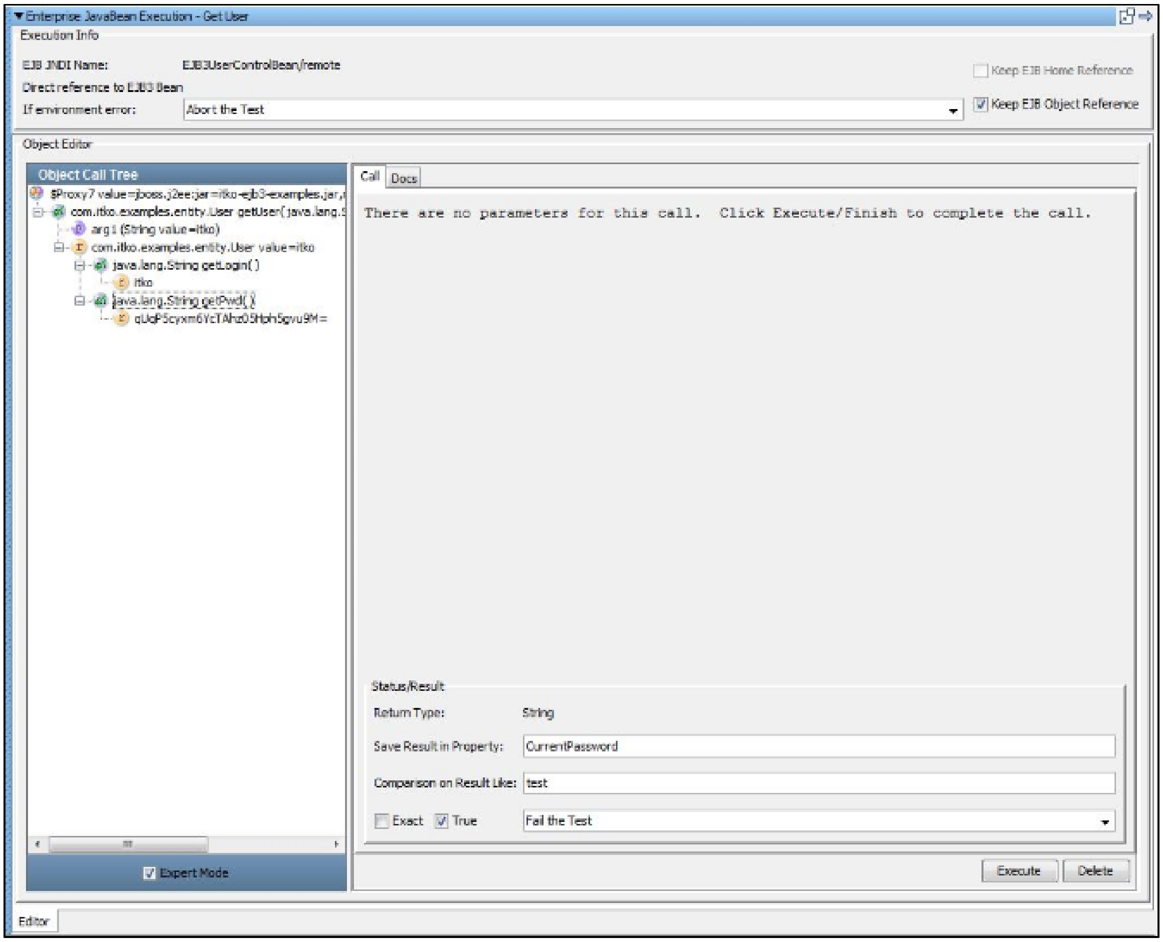
Return Type: com.itko.examples.entity.User

Save Result in Property: getUserObject

Comparison on Result Like:

□ Exact ☑ True

1. Select the Expert Mode check box in the left pane, to open the Status/Result pane where you can add the assertion.



EJB JNDI Name: EJB3UserControlBean/remote

Direct reference to EJB3 Bean

If environment error: Abort the Test

□ Keep EJB Home Reference

☑ Keep EJB Object Reference

```
$Proxy7 value=jboss.j2ee:jar=itko-ejb3-examples.jar,
com.itko.examples.entity.User getUser( java.lang.S
  arg1 (String value=itko)
  com.itko.examples.entity.User value=itko
    java.lang.String getLogin( )
      itko
    java.lang.String getPwd( )
      qUqP5cyxm6YcTAhz05Hph5gvu9M=
```

```
There are no parameters for this call.  Click Execute/Finish to complete the call.
```

Status/Result

Return Type: String

Save Result in Property: CurrentPassword

Comparison on Result Like: test

□ Exact ☑ True Fail the Test

2. The returned value upon executing the getPwd method will be stored in the property CurrentPassword.
3. Add an assertion that tests to see if the returned value is equal to the string "test". If it is not that, then redirect to the Fail step.
4. Click Execute to execute this step.

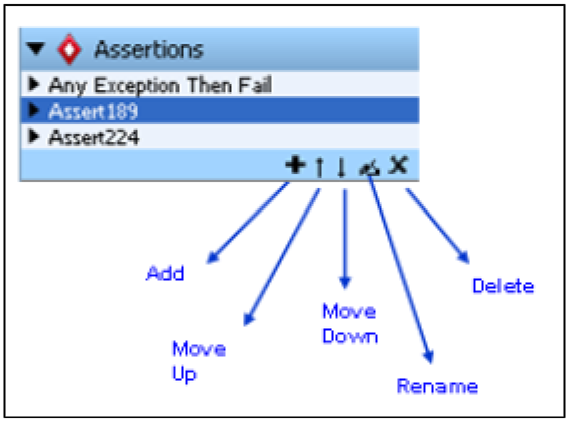
> ⚠ In-line assertions (and filters) do not result in an assertion being added to the test step. In-line assertion management is always done in the Complex Object Editor.

For old test cases, all inline assertions that were set to the Fail step, will now change to the Abort step.

For more details on the Complex Object Editor, see Complex Object Editor (COE).

Assertions Toolbar

All the elements have their own toolbar to add/delete/reorder at the bottom of the element.



Deleting an Assertion

To delete an assertion

- Right-click any assertion in the Elements tab to open a menu. Click Delete to delete the assertion.
- Select the assertion in the test step and right-click to open a menu. Click Delete to delete the assertion.
- Select the assertion in the Elements tab and click the Delete icon on the toolbar.

Reordering an Assertion

You may need to reorder assertions, because assertions are evaluated in the order in which they appear. Thus, changing the order of the assertions can affect the workflow.

To reorder an assertion

- Select the assertion in the Elements tab and click the Move Up ↑ or Move Down ↓ icon on the toolbar.
- Drag and drop the assertion in the model editor to the target destination.

Renaming an Assertion

To rename an assertion

- Select the assertion and right-click to open a menu. Click Rename to rename the assertion.
- Select the assertion and click the Rename icon on the toolbar.

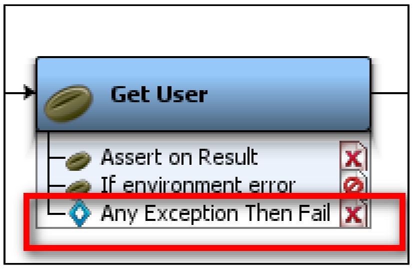
Dragging and Dropping an Assertion

You can drag and drop assertions in the model editor from one test step to another.

1. Click the assertion in one test step; for example, **Step1**.
2. Select and drag the assertion to other test step in the model editor; for example, **Step2**.
3. The dragged assertion will then be applied to **Step2**.

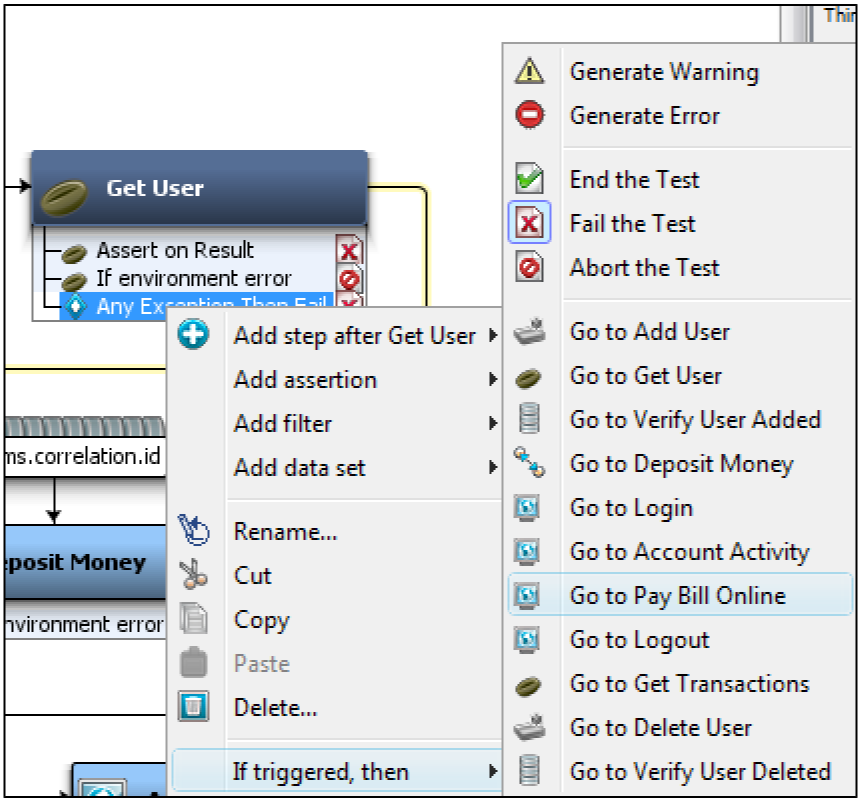
Configuring the Next Step of an Assertion

An assertion added to a step can be seen in the model editor.



After the assertion is added to a step, you can select its next step to be executed, if you want the workflow to be altered.

1. Right-click the assertion in the Model Editor to open the menu.



2. Select **If triggered, then**and do one of the following:
    - Select to generate a warning or error.
    - Select to end, fail, or abort the step.
    - Select the next step to be executed.

Types of Assertions

This section describes each of the assertions that are available in LISA.

Regular expressions are used for comparison purposes in many assertions. For more information about regular expressions, visit http://download.oracle.com/javase/tutorial/essential/regex/.

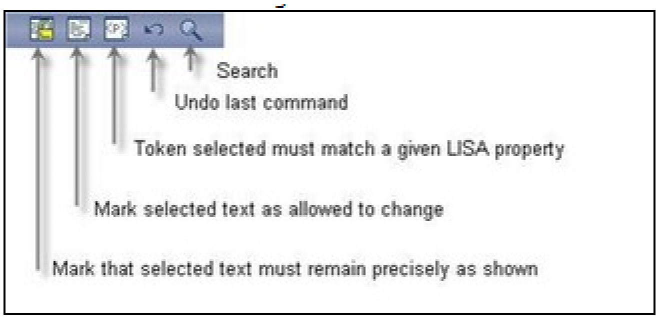
HTTP Assertions

The following assertions are available in the HTTP assertions list for any test step:

Highlight HTML Content for Comparison

The **Highlight HTML Content for Comparison** assertion lets you make a comparison based on the contents on an HTML page. This assertion uses the "paint the screen" technique specifically designed to work with HTML pages. For example, if there is a large HTML document, then you can identify the data before and after the "content of interest". Then you simply identify what the "content of interest" will be compared against (usually this would be an expected value supplied in a data set).

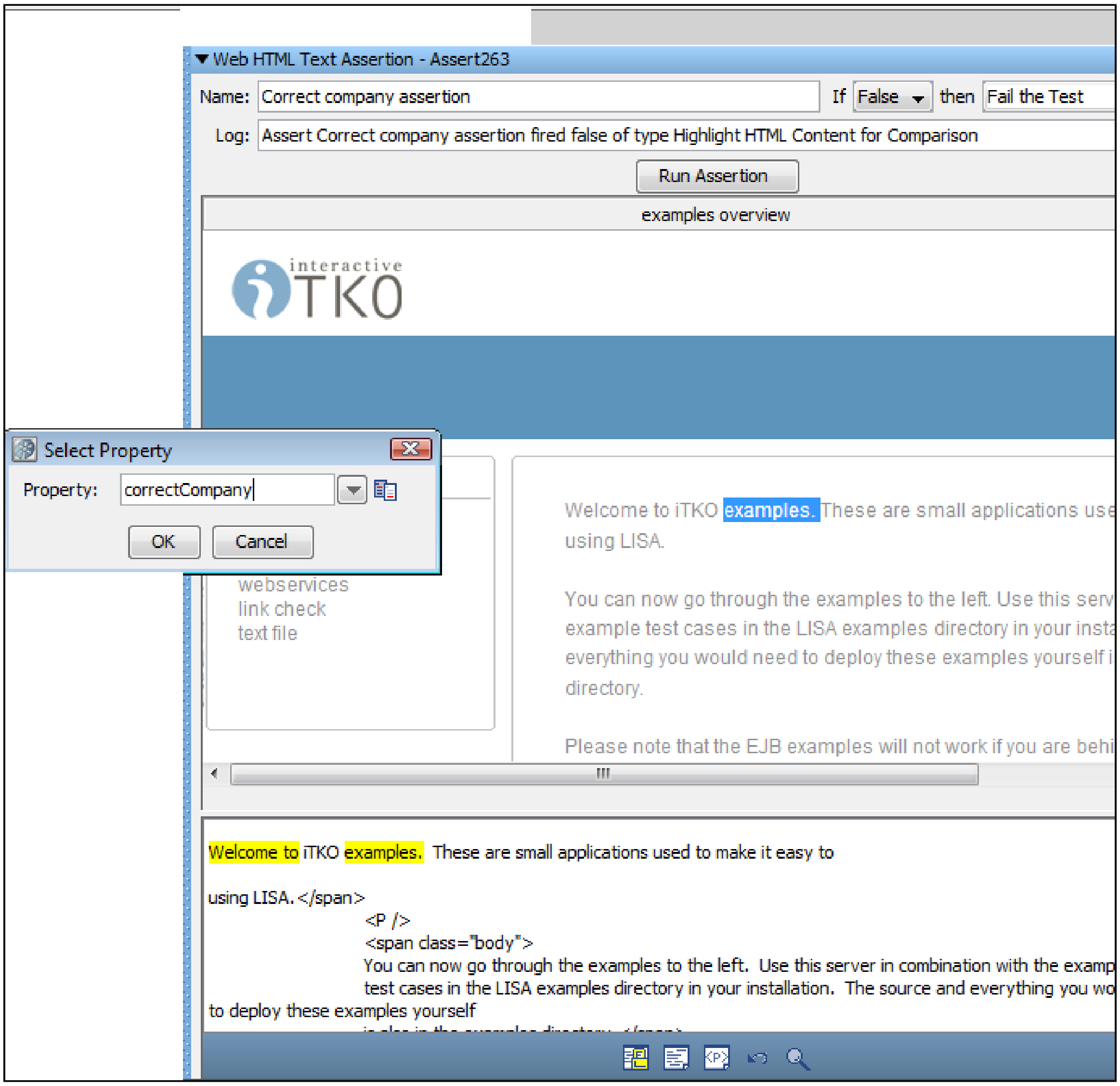
The text is marked using the icons at the bottom of the editor:


Search
Undo last command
Token selected must match a given LISA property
Mark selected text as allowed to change
Mark that selected text must remain precisely as shown

This technique is best explained by example.

In the following example, we want to make sure that the company name, currently ITKO, that appears in the phrase "Welcome to ITKO examples" matches the value in a specified LISA property. We have marked the text, using the buttons shown previously, by selecting text and then clicking the appropriate icon.

- Yellow background indicates text that must appear as shown.
- White background indicates text that need not be present, or can change.
- Red background identifies the text that must match the property entered into the dialog.



Web HTML Text Assertion - Assert263

Name: Correct company assertion

If False then Fail the Test

Log: Assert Correct company assertion fired false of type Highlight HTML Content for Comparison

Select Property

Property: correctCompany

Welcome to iTKO examples. These are small applications use
using LISA.

You can now go through the examples to the left. Use this serv
example test cases in the LISA examples directory in your insta
everything you would need to deploy these examples yourself i
directory.

Please note that the EJB examples will not work if you are behi

```
Welcome to iTKO examples.  These are small applications used to make it easy to

using LISA.</span>
                    <P />
                    <span class="body">
                    You can now go through the examples to the left.  Use this server in combination with the examp
                    test cases in the LISA examples directory in your installation.  The source and everything you wo
to deploy these examples yourself
```

This screen shows the HTML rendered in a browser in the top panel, and the actual HTML text in the bottom panel. We want to make the phrases "Welcome to" and "examples" required. We have set the boundaries around that, and clicked the Must icon. Then we selected the company name text, "ITKO", inside the highlighted content, and clicked the Property icon. We entered the property name **correctCompany** into the dialog. This property will be compared to the text that appears between the two bounding phrases. The company name text has been replaced with the name of the LISA property.

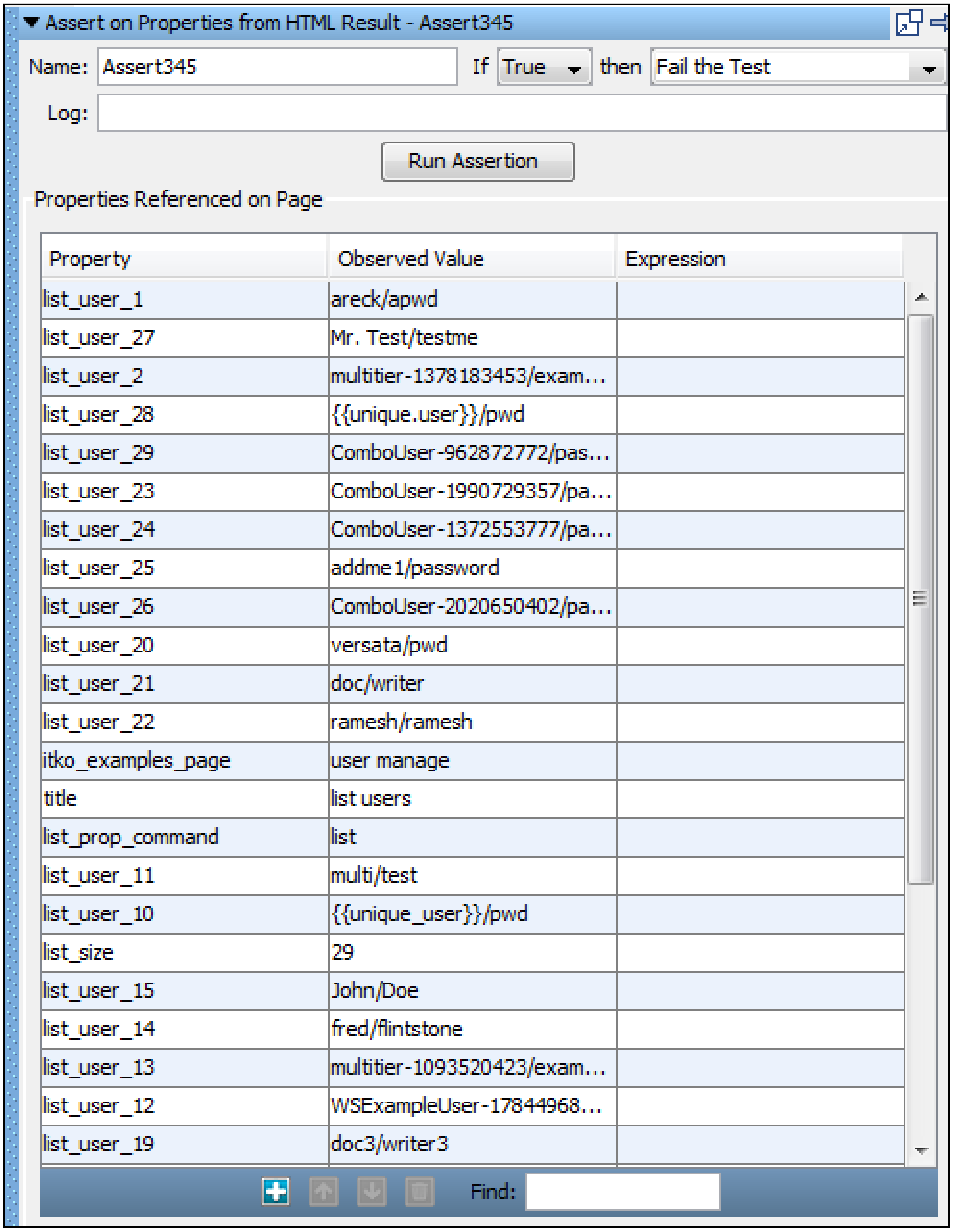
Click the **Run Assertion** button to execute an assertion.

When this assertion is run, the value of the property **correctCompany** will be inserted between the phrases "Welcome to" and "examples" and the resulting phrase will be compared to the corresponding phrase in the HTML response. The phrase "Welcome to `correctCompany` examples" can change location in the HTML and it will still be located.

Check HTML for Properties in Page

The **Check HTML for Properties in Page** assertion is useful for web testing when there is property data in the web page that might be used for the assertion. The property data is made available for assertion by parsing the web page for meta tags, title tags, hidden form fields, and other tags that the product can automatically parse, including <lisaprop> tags and the LISA Integration API.

Here is a sample of the available properties table.


Assert on Properties from HTML Result - Assert345
Run Assertion
Properties Referenced on Page
Find:

Name: Assert345 If True then Fail the Test

Log:

Property	Observed Value	Expression
list_user_1	areck/apwd	
list_user_27	Mr. Test/testme	
list_user_2	multitier-1378183453/exam...	
list_user_28	{{unique.user}}/pwd	
list_user_29	ComboUser-962872772/pas...	
list_user_23	ComboUser-1990729357/pa...	
list_user_24	ComboUser-1372553777/pa...	
list_user_25	addme1/password	
list_user_26	ComboUser-2020650402/pa...	
list_user_20	versata/pwd	
list_user_21	doc/writer	
list_user_22	ramesh/ramesh	
itko_examples_page	user manage	
title	list users	
list_prop_command	list	
list_user_11	multi/test	
list_user_10	{{unique_user}}/pwd	
list_size	29	
list_user_15	John/Doe	
list_user_14	fred/flintstone	
list_user_13	multitier-1093520423/exam...	
list_user_12	WSExampleUser-17844968...	
list_user_19	doc3/writer3	

Enter the following parameters:

- **Name**: Enter the name of the assertion.
- **If**: Select the behavior of the assertion using the drop-down box.

- **then**: Select the step to redirect to if the assertion fires.

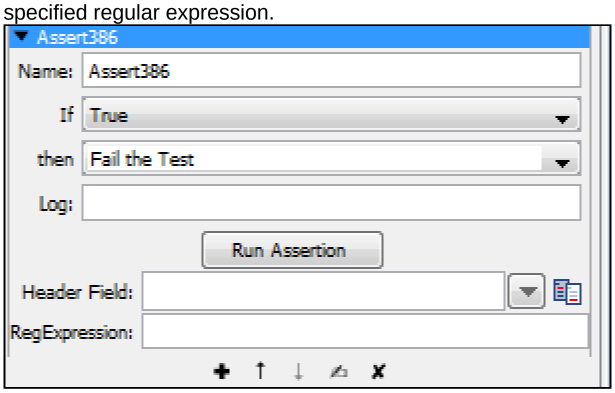
- **Log**: The text that will be printed out as event text if the assertion fired.

Click the Run Assertion button to execute an assertion.

> ⚠ You may be prompted to install the **Parse HTML for LISA Tag** filter.

Ensure HTTP Header Contains Expression

The **Ensure HTTP Header Contains Expression** assertion lets you check that a specific HTTP result header contains a field that matches a specified regular expression.



Assert386

Name: Assert386

If True

then Fail the Test

Log:

Header Field:

RegExpression:

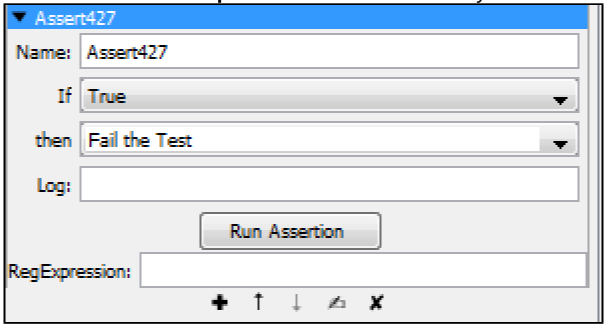
Enter the following parameters:

- **Name**: The name of this assertion. This will help you identify events for this assertion.
- **If**: Select the behavior of the assertion using the drop-down box.
- **then**: Select the step to redirect to if the assertion fires.
- **Log**: The text that will be printed out as event text if the assertion fires.
- **Header Field**: The name of the header field.
- **RegExpression**: The regular expression that must appear in the header field.

Click the Run Assertion button to execute an assertion.

Check HTTP Response Code

The **Check HTTP Response Code** assertion lets you check that the HTTP response code matches a specified regular expression.



Assert427

Name: Assert427

If True

then Fail the Test

Log:

RegExpression:

Enter the following parameters:

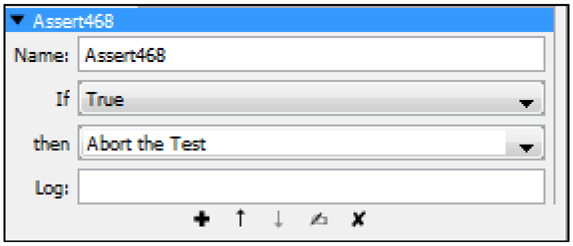
- **Name**: The name of this assertion. This will help you identify events for this assertion.
- **If**: Select the behavior of the assertion using the drop-down box.
- **then**: Select the step to redirect to if the assertion fires.
- **Log**: The text that will be printed out as event text if the assertion fires.
- **RegExpression**: The regular expression that must appear in the response code. For example, to check that the HTTP response code is in the 400-499 range, set the RegExpression to **4\d\d**.

Click the Run Assertion button to execute an assertion.

Simple Web Assertion

The **Simple Web Assertion** reads the return code from the web application.

If the application returns code 404 (page not found), 500 (server error) or any other error then this assertion returns true.


Assert468

Name: Assert468

If: True

then: Abort the Test

Log:

The **multi-tier-combo** test case in the examples project has this type of assertion.

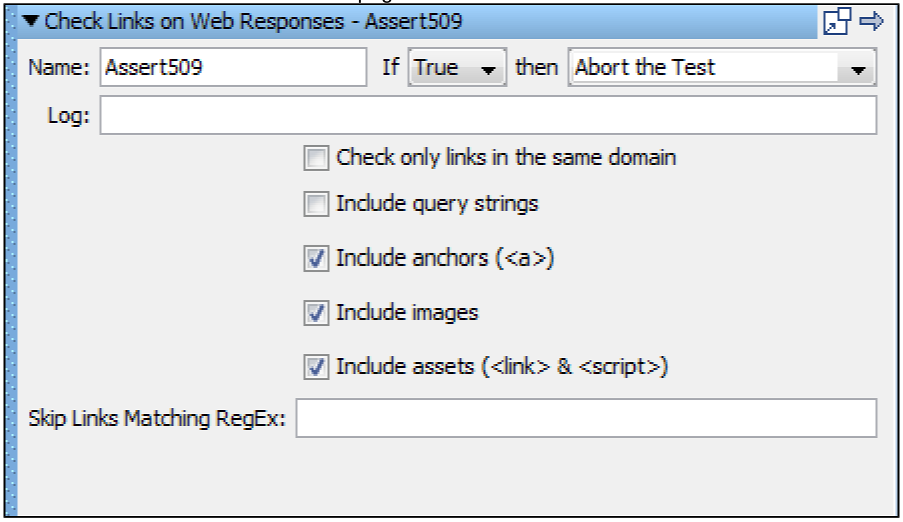
Enter the following parameters:

- **Name**: The name of this assertion. This will help you identify events for this assertion.
- **If**: Select the behavior of the assertion using the drop-down list.
- **then**: Select the step to redirect to if the assertion fires.
- **Log**: The text that will be printed out as event text if the assertion fired.

Click the Run Assertion button to run and execute the assertion.

Check Links on Web Responses

The **Check Links on Web Responses** assertion checks every link on the returned web page to make sure that it contains a valid page and does not return an HTTP error like a 404 error, or others. This is commonly used to make sure that the links are working properly across the application and there are no inactive links on the page.


Check Links on Web Responses - Assert509

Name: Assert509 If: True then: Abort the Test

Log:

☐ Check only links in the same domain

☐ Include query strings

☑ Include anchors (<a>)

☑ Include images

☑ Include assets (<link> & <script>)

Skip Links Matching RegEx:

Enter the following parameters:

- **Name**: Enter the name of the assertion.
- **If**: Select the behavior of the assertion using the drop-down box.
- **then**: Select the step to redirect to if the assertion fires.
- **Log**: The text that will be printed out as event text if the assertion fires.

The following criteria can be checked for the links:

- **Check only links in the same domain**: Checks only links in the current domain of the returned web page.
- **Include query strings**: If any query strings are present on the returned web page then those are checked.
- **Include anchors (<a>)**: Any anchor links in the current web page are checked.
- **Include images**: All the images on the returned web page are checked.
- **Include assets (<link> & <script>)**: Current web page is checked for script and links.
- **Skip Links Matching RegEx**: Enter a RegEx expression for any links you want to skip.

Database Assertions

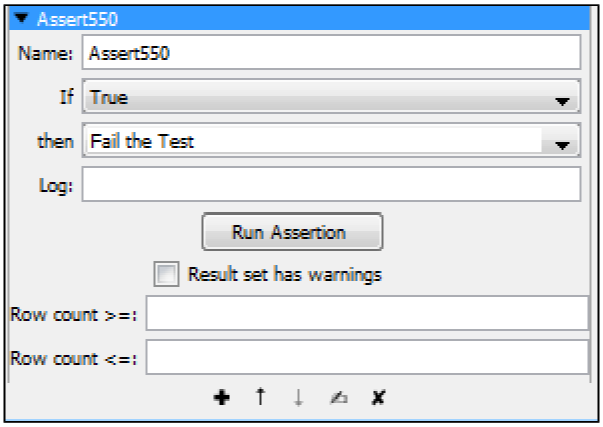
The following assertions are available in the Database Assertions list for any test step.

Ensure Result Set Size
Ensure Result Set Contains Expression

Ensure Result Set Size

The **Ensure Result Set Size** assertion will count the number of rows in a result set and verify that the size falls between an upper and lower value.

An example of this assertion could be checking to make sure the number of rows in an HTML table matches a supplied expected value from a data set.


Assert550

Name: Assert550

If True

then Fail the Test

Log:

□ Result set has warnings

Row count >=:

Row count <=:

Enter the following parameters:

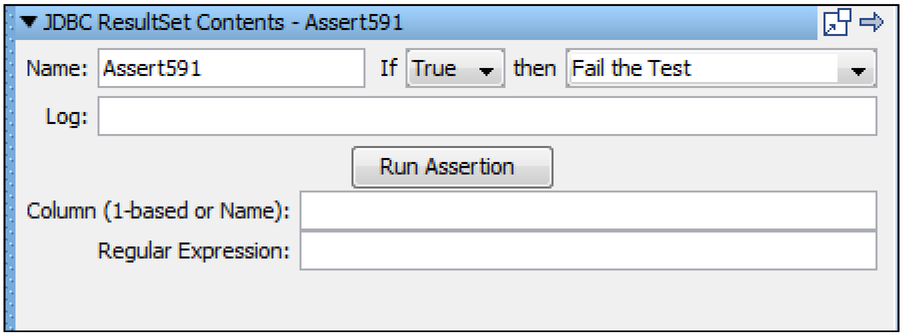
- **Name**: The name of this assertion. This will help you identify events for this assertion.
- **If**: Select the behavior of the assertion using the drop-down box.
- **then:** Select the step to redirect to if the assertion fires.
- **Log**: The text that will be printed out as event text if the assertion fires.
- **Result set has warnings**: If checked, the database may return warnings in the result set. Check with your system administrator to determine whether your database supports warnings in the result set.
- **Row Count >=**: The minimum number of rows in the result set. **-1** indicates no minimum.
- **Row Count <=**: The maximum number of rows in the result set. **-1** indicates no maximum.

Click the Run Assertion button to execute an assertion.

For example, to make sure that a Database Assertion step returns one and only one row, set the **Row Count >=** field to **1** and the **Row Count <=** field to **1**.

Ensure Result Set Contains Expression

The **Ensure Result Set Contains Expression** assertion will check a particular column in a result set and ensure that the supplied expression matches at least one value.


JDBC ResultSet Contents - Assert591

Name: Assert591

If True

then Fail the Test

Log:

Column (1-based or Name):

Regular Expression:

Enter the following parameters:

- **Name**: The name of this assertion. This will help you identify events for this assertion.
- **If**: Select the behavior of the assertion using the drop-down box.
- **then**: Select the step to redirect to if the assertion fires.
- **Log**: The text that will be printed out as event text if the assertion fires.
- **Column**: The column that contains the text to check. This can be a column name or an index.
- **Regular Expression**: The regular expression to match in the column.

Click the Run Assertion button to execute an assertion.

For example, to check that at least one of the rows returned from a query has a login value that starts with **wp**, set the **Column** field to **login** and the **Regular Expression** field to **wp.***.

Web 2.0 Assertions

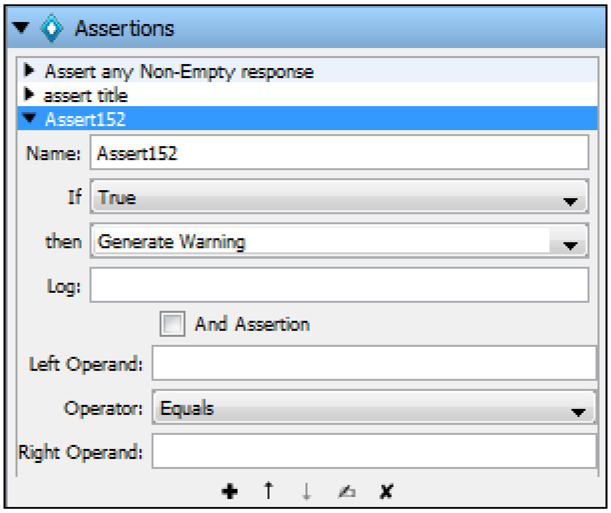
The following assertions are available in the Web 2.0 assertions list for any test step.

Web 2.0 Basic Assertion
Web 2.0 Validation Assertion
Web 2.0 Branching Assertion

Web 2.0 Basic Assertion

The **Web 2.0 Basic Assertion** is intended to be created from the DOM web browser. The basic assertion provides the ability to compare properties; for example, comparing a bank balance before and after making a deposit. The basic assertion can be used to make sure the {{ `after}}EQUALS{{before + deposit`}}.

The Web 2.0 Basic Assertion evaluates a unary or binary expression using predefined operators and properties. Typically, you will use a variable created by a filter on the left side and then equal it or match it to a constant value or another variable on the right side.



Assertions

- Assert any Non-Empty response
- assert title
- Assert152

Name: Assert152

If: True

then: Generate Warning

Log:

□ And Assertion

Left Operand:

Operator: Equals

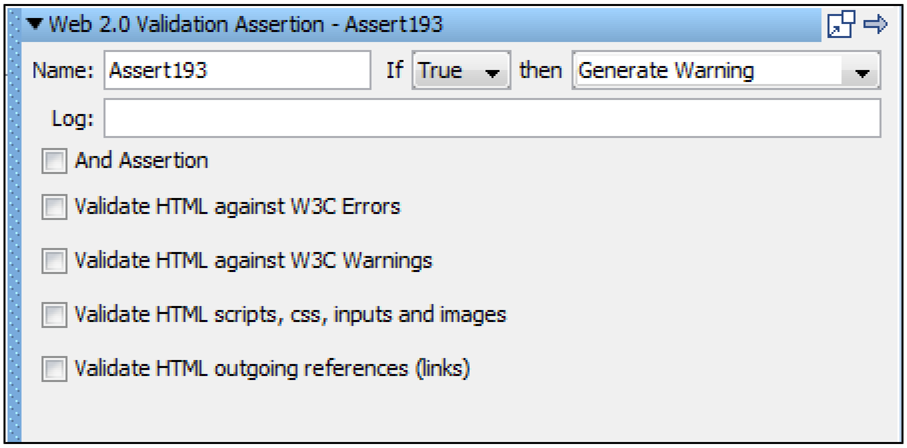
Right Operand:

Enter the following parameters:

- **If**: Select the behavior of the assertion using the drop-down box.
- **then**: Select the step to redirect to if the assertion fires.
- **Log**: The text that will be printed out as event text if the assertion fires.
- **And Assertion**: Deprecated; not being used.
- **Left Operand**: Left side of the expression for comparison.
- **Operator**: One of the following operators:
  - **Equals**: Compare the two sides of the expression for equality.
  - **Not Equals**: Compare the two sides of the expression for non-equality.
  - **Matches**: Attempt to match (as regular expression) the left or the right side of the expression against the other side.
  - **Not Matches**: Attempt not to match (as regular expression) the left or the right side of the expression against the other side.
  - **Less Than**: Compare the two sides of the expression as numeric values for order. Returns false on non-numeric values.
  - **More Than**: Compare the two sides of the expression as numeric values for order. Returns false on non-numeric values.
  - **Exists**: Verify if the entity represented by the left side exists in the current page context.
  - **Evaluate**: Evaluate arbitrary JavaScript code in the context of the current page. Must return a Boolean.
- **Right Operand**: Right side of the expression for comparison, or blank if a unary comparison.

Web 2.0 Validation Assertion

The Web 2.0 Validation Assertion is intended to be created from the DOM web browser. This assertion validates the entire HTML result to verify it complies with W3C standards. It also validates the page for errors, warnings, or broken links. This is important in Web 2.0 applications because there can be many tools to generate the page, so it is good to verify it is standards compliant.



Web 2.0 Validation Assertion - Assert193

Name: Assert193 If True then Generate Warning

Log:

☐ And Assertion

☐ Validate HTML against W3C Errors

☐ Validate HTML against W3C Warnings

☐ Validate HTML scripts, css, inputs and images

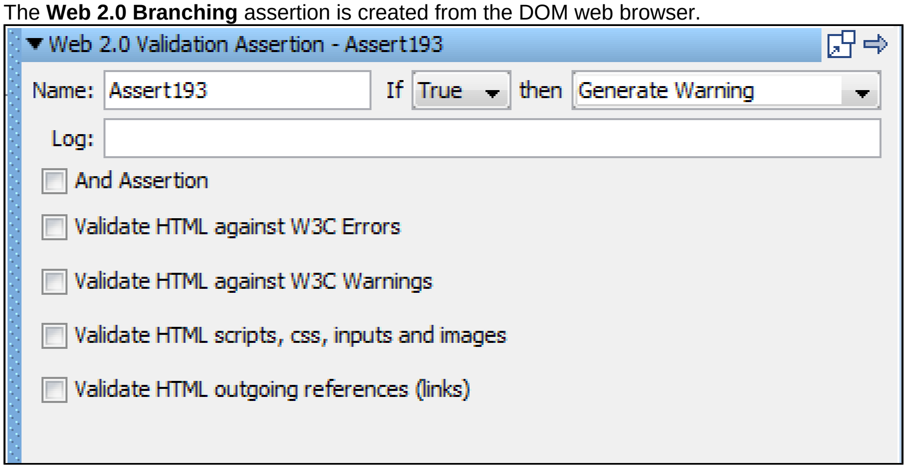
☐ Validate HTML outgoing references (links)

Select one or more of the following validations:

- **And Assertion:** Deprecated; not being used.
- **Validate HTML against W3C Errors**: HTML page is evaluated for W3C errors.
- **Validate HTML against W3C Warnings**: HTML page is evaluated for W3C warnings.
- **Validate HTML scripts, css, input and images**: Page is evaluated for images, css scripts and inputs.
- **Validate HTML outgoing references (links)**: HTML Page is evaluated for valid links.

Web 2.0 Branching Assertion

The **Web 2.0 Branching** assertion is created from the DOM web browser.



Web 2.0 Validation Assertion - Assert193

Name: Assert193 If True then Generate Warning

Log:

☐ And Assertion

☐ Validate HTML against W3C Errors

☐ Validate HTML against W3C Warnings

☐ Validate HTML scripts, css, inputs and images

☐ Validate HTML outgoing references (links)

Select one or more of the following validations:

- **And Assertion:** Deprecated; not being used.
- **Validate HTML against W3C Errors**: HTML page is evaluated for W3C errors.
- **Validate HTML against W3C Warnings**: HTML page is evaluated for W3C warnings.
- **Validate HTML scripts, css, input and images**: Page is evaluated for images, css scripts and inputs.
- **Validate HTML outgoing references (links)**: HTML Page is evaluated for valid links.

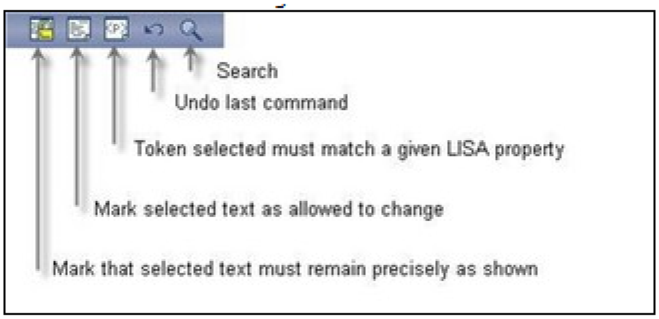
XML Assertions

The following assertions are available in the XML assertions list for any test step.

Highlight Text Content for Comparison

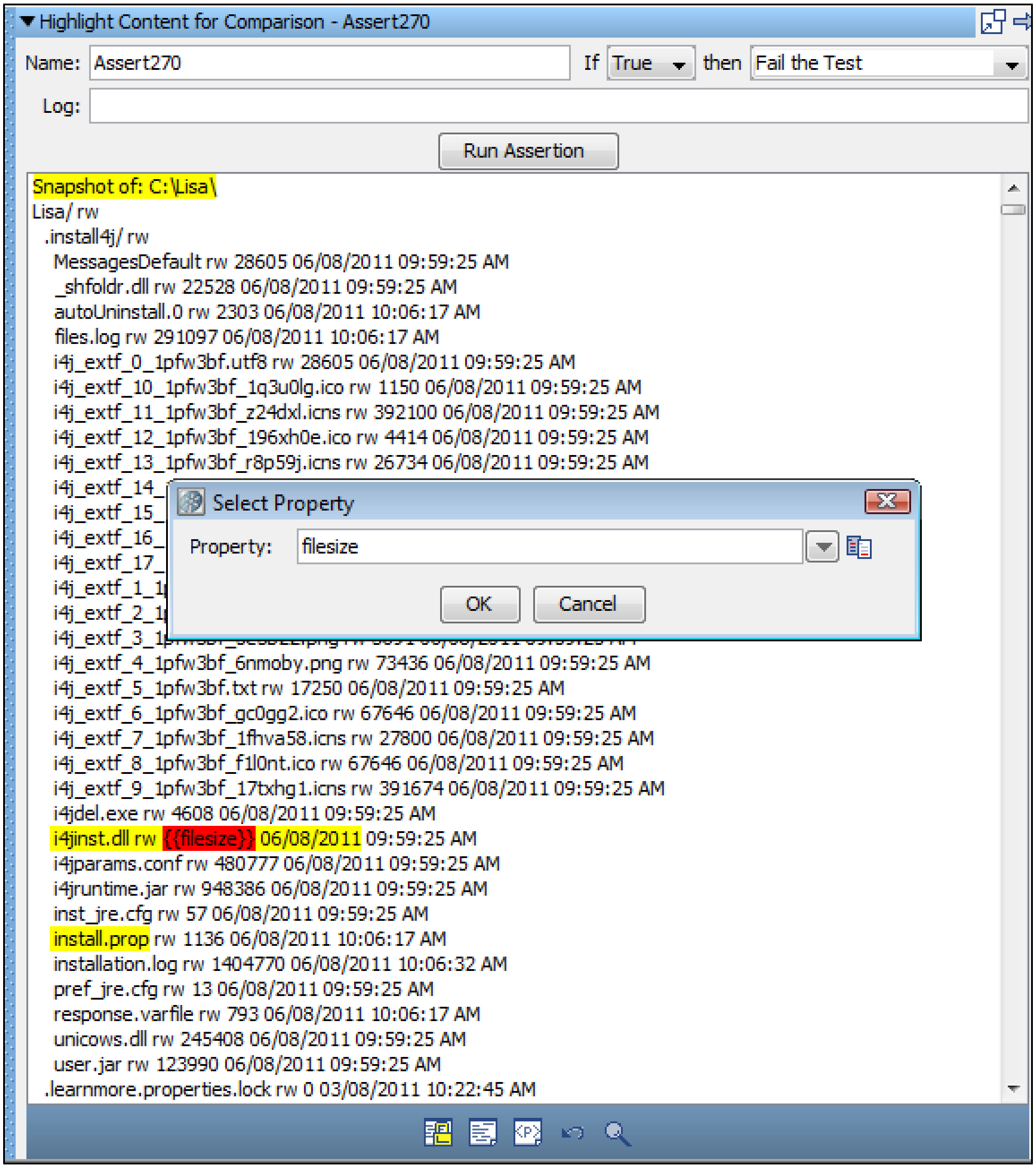
The **Highlight Text Content for Comparison** assertion uses the "paint the screen" technique specifically designed to work with HTML pages. For example, if there is a large HTML document, then you identify the data before and after the content of interest. Then, you simply identify what the content of interest will be compared against (usually this would be an expected value supplied in a data set).

Mark the text with the icons at the bottom of the editor:


Search
Undo last command
Token selected must match a given LISA property
Mark selected text as allowed to change
Mark that selected text must remain precisely as shown

In the following example, we want to make sure that certain files appear in the buffer, and one of the file sizes needs to be compared to the value of a property. We have marked the text using the three icons shown in the previous illustration, by selecting text and then clicking on the appropriate icon.

- Yellow background indicates text that must appear as shown.
- White background indicates text that need not be present, or can change.
- Red background identifies the text that must match the property entered into the dialog.



Highlight Content for Comparison - Assert270

Name: Assert270 If True then Fail the Test

Log:

```
Snapshot of: C:\Lisa\
Lisa/ rw
 .install4j/ rw
  MessagesDefault rw 28605 06/08/2011 09:59:25 AM
  _shfoldr.dll rw 22528 06/08/2011 09:59:25 AM
  autoUninstall.0 rw 2303 06/08/2011 10:06:17 AM
  files.log rw 291097 06/08/2011 10:06:17 AM
  i4j_extf_0_1pfw3bf.utf8 rw 28605 06/08/2011 09:59:25 AM
  i4j_extf_10_1pfw3bf_1q3u0lg.ico rw 1150 06/08/2011 09:59:25 AM
  i4j_extf_11_1pfw3bf_z24dxl.icns rw 392100 06/08/2011 09:59:25 AM
  i4j_extf_12_1pfw3bf_196xh0e.ico rw 4414 06/08/2011 09:59:25 AM
  i4j_extf_13_1pfw3bf_r8p59j.icns rw 26734 06/08/2011 09:59:25 AM
  i4j_extf_14_
  i4j_extf_15_
  i4j_extf_16_
  i4j_extf_17_
  i4j_extf_1_1
  i4j_extf_2_1
  i4j_extf_3_1
  i4j_extf_4_1pfw3bf_6nmoby.png rw 73436 06/08/2011 09:59:25 AM
  i4j_extf_5_1pfw3bf.txt rw 17250 06/08/2011 09:59:25 AM
  i4j_extf_6_1pfw3bf_gc0gg2.ico rw 67646 06/08/2011 09:59:25 AM
  i4j_extf_7_1pfw3bf_1fhva58.icns rw 27800 06/08/2011 09:59:25 AM
  i4j_extf_8_1pfw3bf_f1l0nt.ico rw 67646 06/08/2011 09:59:25 AM
  i4j_extf_9_1pfw3bf_17txhg1.icns rw 391674 06/08/2011 09:59:25 AM
  i4jdel.exe rw 4608 06/08/2011 09:59:25 AM
  i4jinst.dll rw {{filesize}} 06/08/2011 09:59:25 AM
  i4jparams.conf rw 480777 06/08/2011 09:59:25 AM
  i4jruntime.jar rw 948386 06/08/2011 09:59:25 AM
  inst_jre.cfg rw 57 06/08/2011 09:59:25 AM
  install.prop rw 1136 06/08/2011 10:06:17 AM
  installation.log rw 1404770 06/08/2011 10:06:32 AM
  pref_jre.cfg rw 13 06/08/2011 09:59:25 AM
  response.varfile rw 793 06/08/2011 10:06:17 AM
  unicows.dll rw 245408 06/08/2011 09:59:25 AM
  user.jar rw 123990 06/08/2011 09:59:25 AM
 .learnmore.properties.lock rw 0 03/08/2011 10:22:45 AM
```

Select Property

Property: filesize

The set of tokens shown in the previous illustration can be read this way:

- The buffer must start with the phrase in yellow: "Snapshot of: C:\Lisa\".
- There are a number of files that may or may not be in the buffer in the next token, but because it is an "Any" token, the variance is immaterial.
- The file "i4jinst.dll" and "rw" attributes must appear.
- The red `filesize` means that the value associated with the property key **filesize** will be swapped into the expression, then the comparison made.
- The text "06/08/2011" must appear.
- The file "install.prop" must appear.
- The buffer can have any amount of content afterward.

After you have finished the markup, enter the following parameters:

- **Name**: The name of this assertion. This will help you identify events for this assertion.

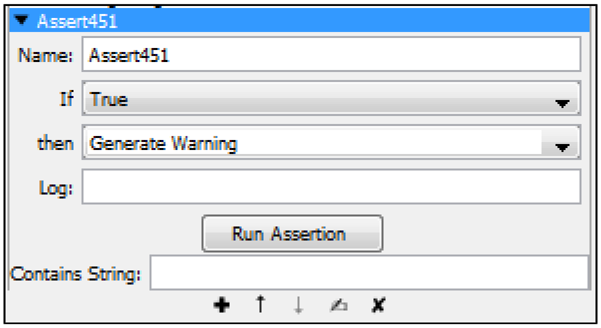
- **If**: Select the behavior of the assertion using the drop-down box.
- **then**: Select the step to redirect to if the assertion fires.
- **Log**: The text that will be printed out as event text if the assertion fires.

Click the Run Assertion button to execute an assertion.

> ⚠ **Property** blocks must always be bounded by **Must** blocks.

Ensure Result Contains String

The **Ensure Result Contains String** assertion lets you search the response (as text) for a string.



Assert451

Name: Assert451

If: True

then: Generate Warning

Log:

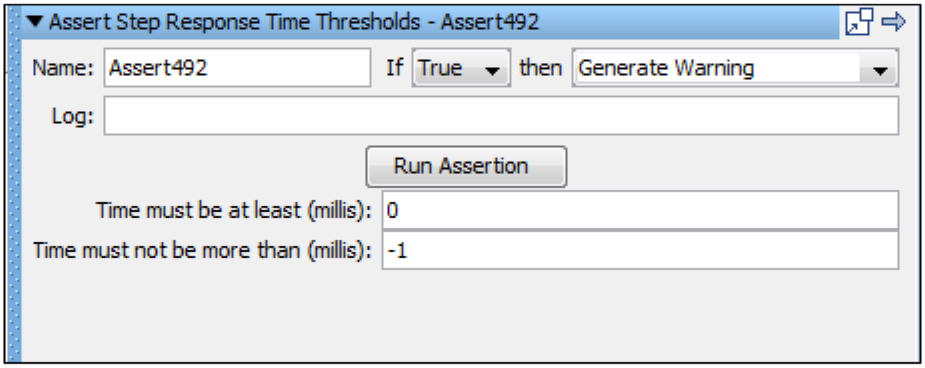
Contains String:

Enter the following parameters:

- **Name**: The name of this assertion. This will help you identify events for this assertion.
- **If**: Select the behavior of the assertion using the drop-down box.
- **then**: Select the step to redirect to if the assertion fires.
- **Log**: The text that will be printed out as event text if the assertion fires.
- **Contains String**: The string to search for in the step result - this can contain a property.

Ensure Step Response Time

The **Ensure Step Response Time** assertion lets you define upper and lower bounds on the response time and assert that the response time is within those bounds.



Assert Step Response Time Thresholds - Assert492

Name: Assert492

If: True

then: Generate Warning

Log:

Time must be at least (millis): 0

Time must not be more than (millis): -1

Enter the following parameters:

- **Name**: The name of this assertion. This will help you identify events for this assertion.
- **If**: Select the behavior of the assertion using the drop-down box.
- **then**: Select the step to redirect to if the assertion fires.
- **Log**: The text printed out as event text if the assertion fires.
- **Time must be at least (millis)**: Enter the lower bound in milliseconds.
- **Time must not be more than (millis)**: Enter the upper bound in milliseconds. This is ignored if set to **-1**.

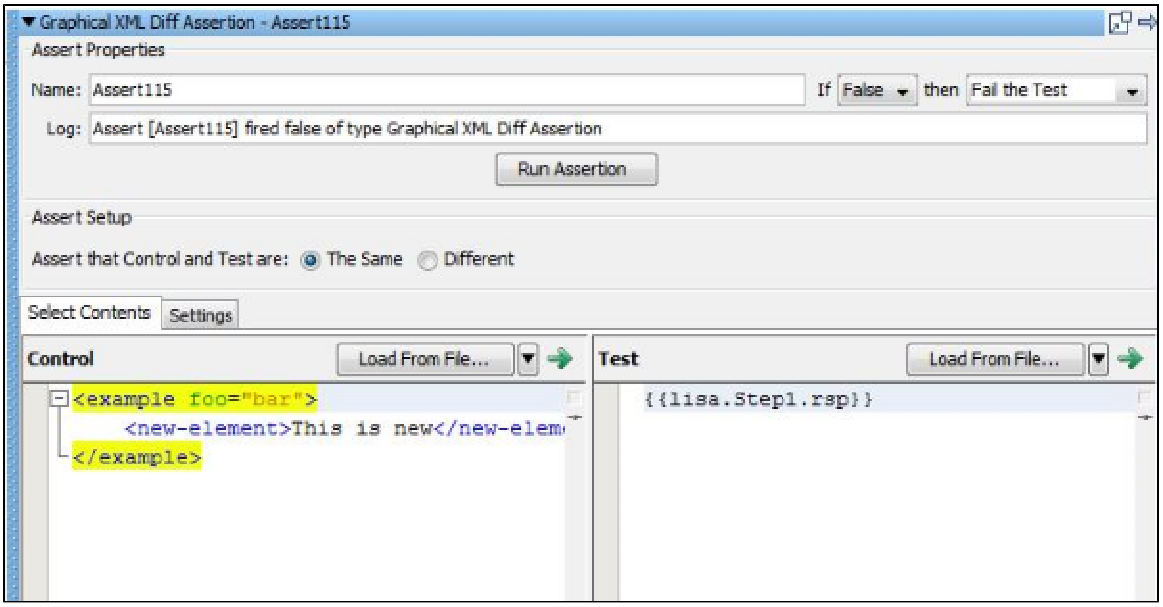
⚠ Parameters can contain properties.

Graphical XML Side-by-Side Comparison

The **Graphical XML Side-by-Side Comparison** assertion lets you compare a test XML value received from a test with a control XML value. The assertion can return true if the responses are the same or different. This provides a flexible ability to compare XML documents at various steps in a business process to make sure they match expected criteria. This is known as "exclusive" testing where an entire response is compared except for a few values known to change.

The assertion editor works by comparing a left and right side of XML to each other. The left side is known as Control Content and the right side as Test Content. Control Content serves as the expected XML returned from a web service in the application under test, while the Test Content should be actual content, for example. By default, Test Content is loaded from the last response of the test step associated with the assertion, signified by the empty LISA property key. Otherwise, any valid LISA property key can be used and the Test Content will be loaded from it.

Alternatively, in test case authoring mode, XML can be loaded from a file or entered manually for Control and Test Content so that a quick graphical diff can be performed.



Name: Assert115 If False then Fail the Test

Log: Assert [Assert115] fired false of type Graphical XML Diff Assertion

Assert that Control and Test are: ● The Same ○ Different

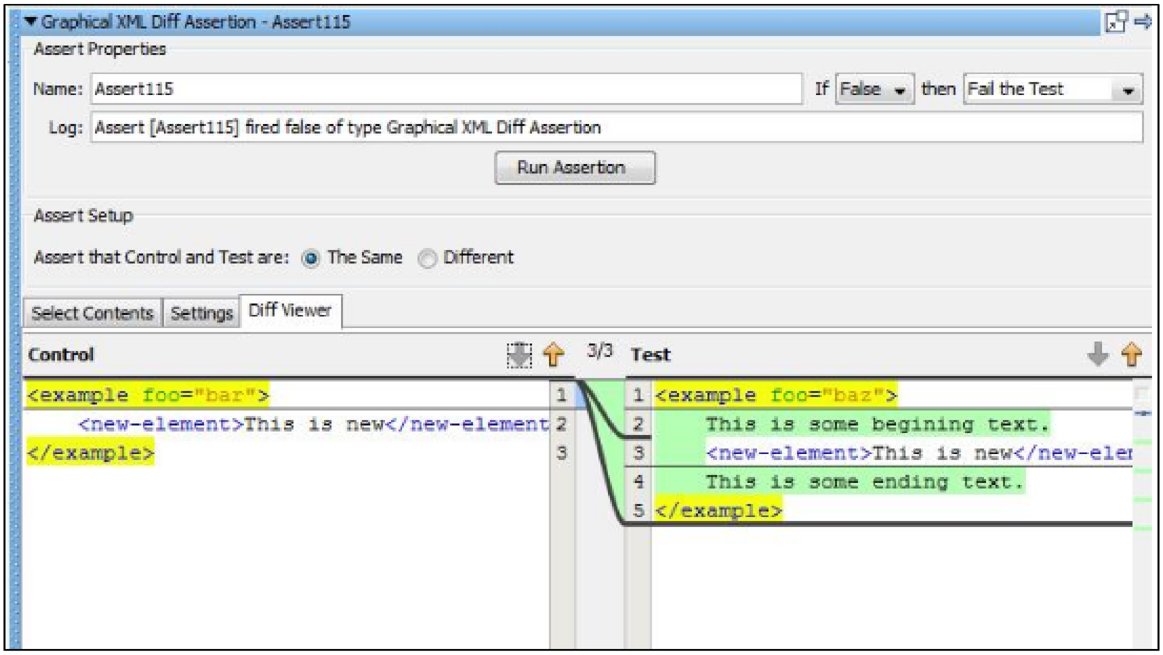
Control

```
<example foo="bar">
    <new-element>This is new</new-elem
</example>
```

Test

```
{{lisa.Step1.rsp}}
```

After a diff is executed, the results appear in the visualizer in the Diff Viewer tab in the assertion editor.



Graphical XML Diff Assertion - Assert115

Assert Properties

Name: Assert115 If False then Fail the Test

Log: Assert [Assert115] fired false of type Graphical XML Diff Assertion

Assert Setup

Assert that Control and Test are: ● The Same ○ Different

```
<example foo="bar">
    <new-element>This is new</new-element
</example>
```

```
<example foo="baz">
    This is some begining text.
    <new-element>This is new</new-eler
    This is some ending text.
</example>
```

Output During Execution

When an assertion is executed, the diff results are logged as test events.

An `Info message` EventID containing the XML diff results is always logged.

If the assertion fires, an `Assertion fired` EventID containing the XML diff results is logged.

The diff results are reported in a format resembling the original UNIX diff utility. An example of a text diff report is:

```
Assert [Assert1] fired false of type Graphical XML Diff Assertion
XML is [Different]
=====
1,2[ELEMENT_NAME_CHANGED]1,2
<! <test2>
<! </test2>
---
>! <test>
>! </test>
```

Each difference is displayed with a heading of the format:

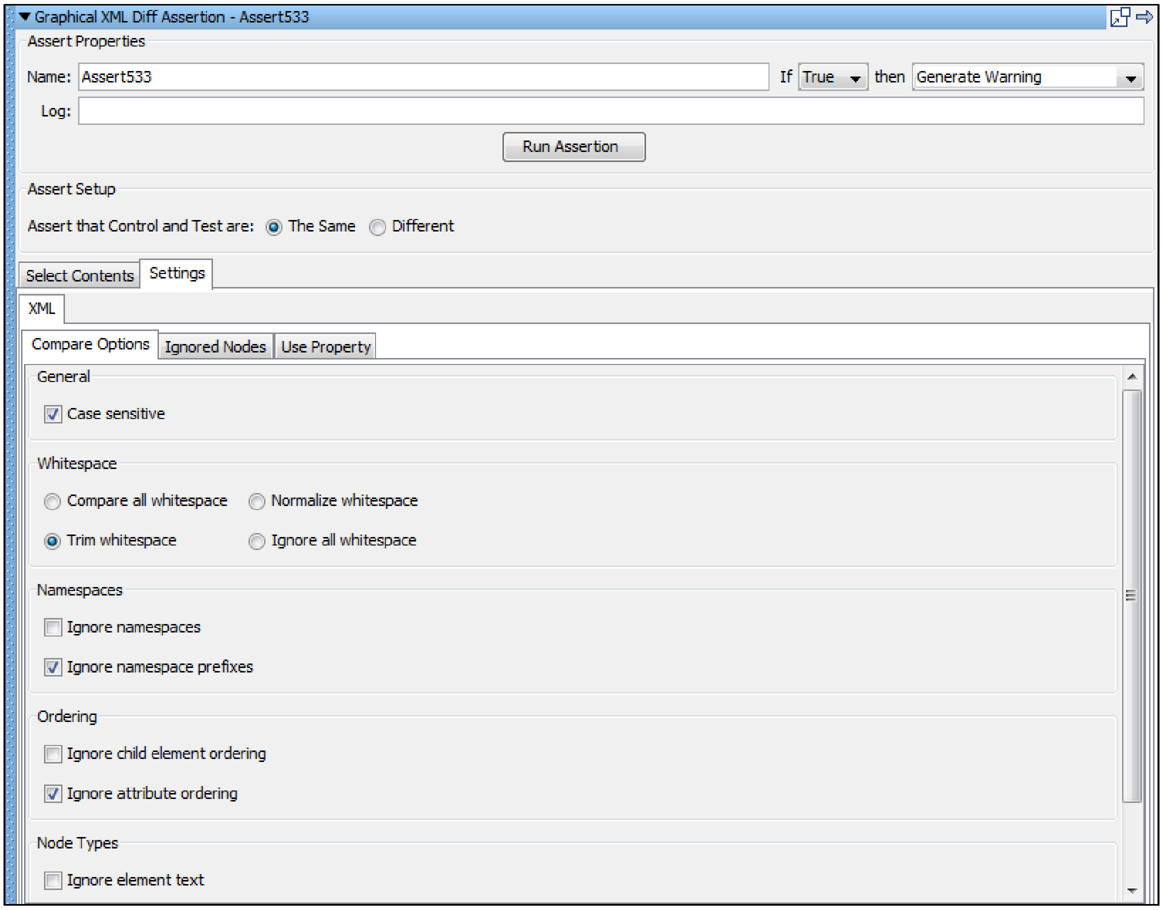
```
<First Start Line>, <First End Line>'['<Diff Type>']'<Second Start Line>,<Second End Line>
```

Then the difference in the first content is displayed, followed by the separator '---', followed by the difference in the second content.

The + character signifies addition, - a deletion, and ! a change. When these characters are present, they indicate an actual change occurred on the line of content, as opposed to a context line.

XML Compare Options

The following comparison options are available for use by the diff engine:



Assert Properties

Name: Assert533 If True then Generate Warning

Log:

Assert Setup

Assert that Control and Test are: ● The Same ○ Different

General

☑ Case sensitive

Whitespace

○ Compare all whitespace ○ Normalize whitespace

● Trim whitespace ○ Ignore all whitespace

Namespaces

☐ Ignore namespaces

☑ Ignore namespace prefixes

Ordering

☐ Ignore child element ordering

☑ Ignore attribute ordering

Node Types

☐ Ignore element text

General

- **Case sensitive**: Whether case sensitivity should be used during the comparison (enabled by default).

Whitespace

- **Trim whitespace**: During a comparison, all leading and trailing whitespace will be removed from element text and attribute values (enabled by default).
- **Collapse whitespace**: In addition to trimming whitespace, any sequence of one or more whitespace characters inside text is converted to a single space character.
- **Normalize whitespace**: Any sequence of one or more whitespace characters is converted to a single space character.
- **Ignore all whitespace**: All whitespace is ignored during the comparison.

Namespaces

- **Ignore namespaces**: The namespace value of an element or attribute is ignored.
- **Ignore namespace prefixes**: The namespace prefix of an element or attribute is ignored (enabled by default).

Ordering

- **Ignore child element ordering**: Ignore the order of child elements in the XML document.
- **Ignore attribute ordering**: Ignore the order of attributes in the XML document (enabled by default).

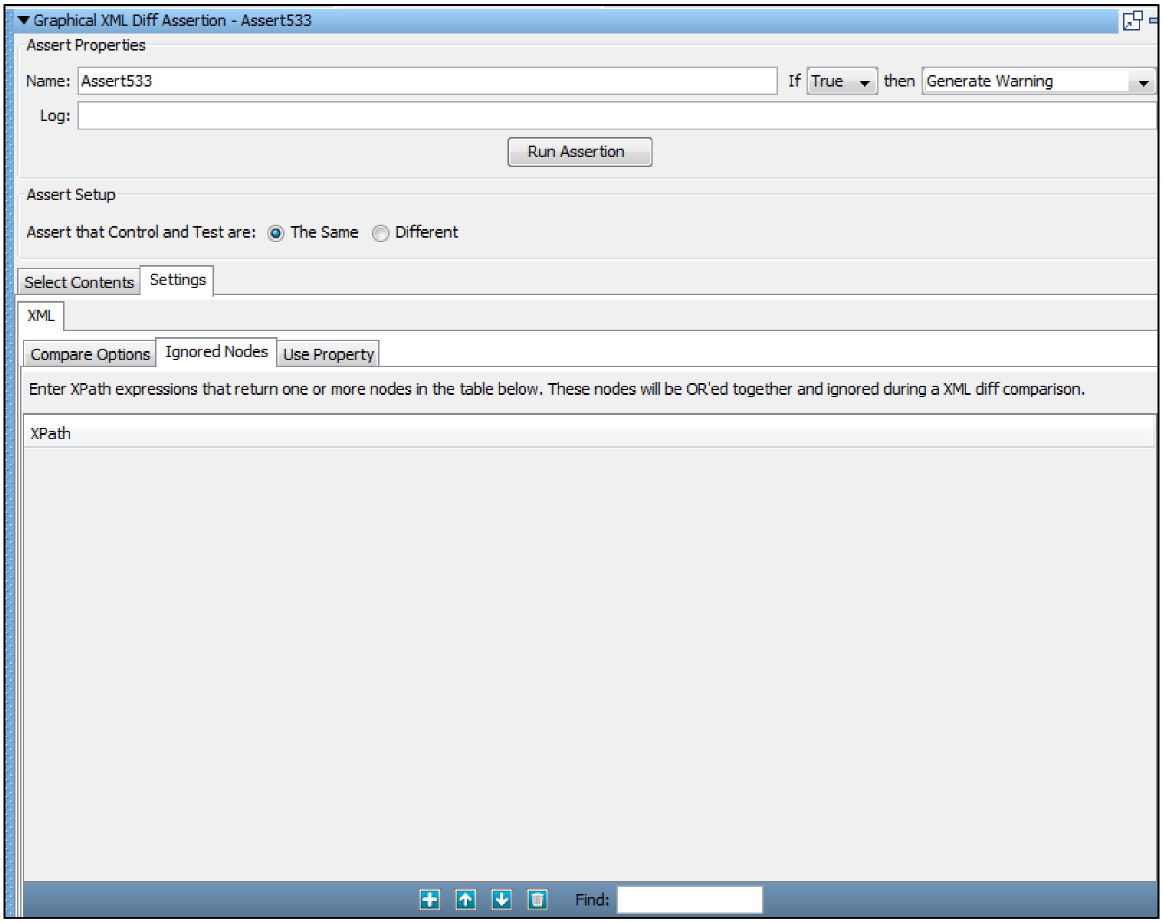
Node Types

- **Ignore element text**: Ignore all element text.
- **Ignore attribute values**: Compare attribute names but ignore attribute values.
- **Ignore attributes**: Ignore attribute names and values.

Ignored Nodes

Ignored nodes are created from a list of XPaths that are executed against the left and right documents. Each evaluated XPath that returns a node set is aggregated. When the diff occurs, any node that is found in the aggregate set is ignored.

Ignored node XPaths can be any arbitrary query that returns a node set. For example, the XPath "//*" excludes all nodes in an XML document. "/example/text()" excludes the first text node child of the "example" element in an XML document. "/example/@myattr" is the XPath to ignore the "myattr" attribute, including the attribute text value, belonging to the "example" element in an XML document.



Graphical XML Diff Assertion - Assert533

Assert Properties

Name: Assert533 If True then Generate Warning

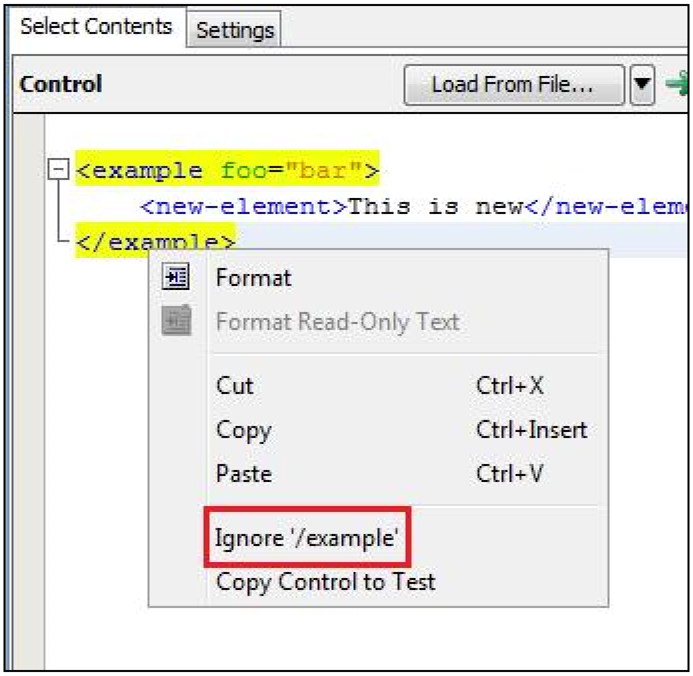
Log:

Assert Setup

Assert that Control and Test are: ● The Same ○ Different

Enter XPath expressions that return one or more nodes in the table below. These nodes will be OR'ed together and ignored during a XML diff comparison.

A right-click menu item also lets a node be selected directly inside a given XML document and its XPath will be added to the Ignored Nodes list.



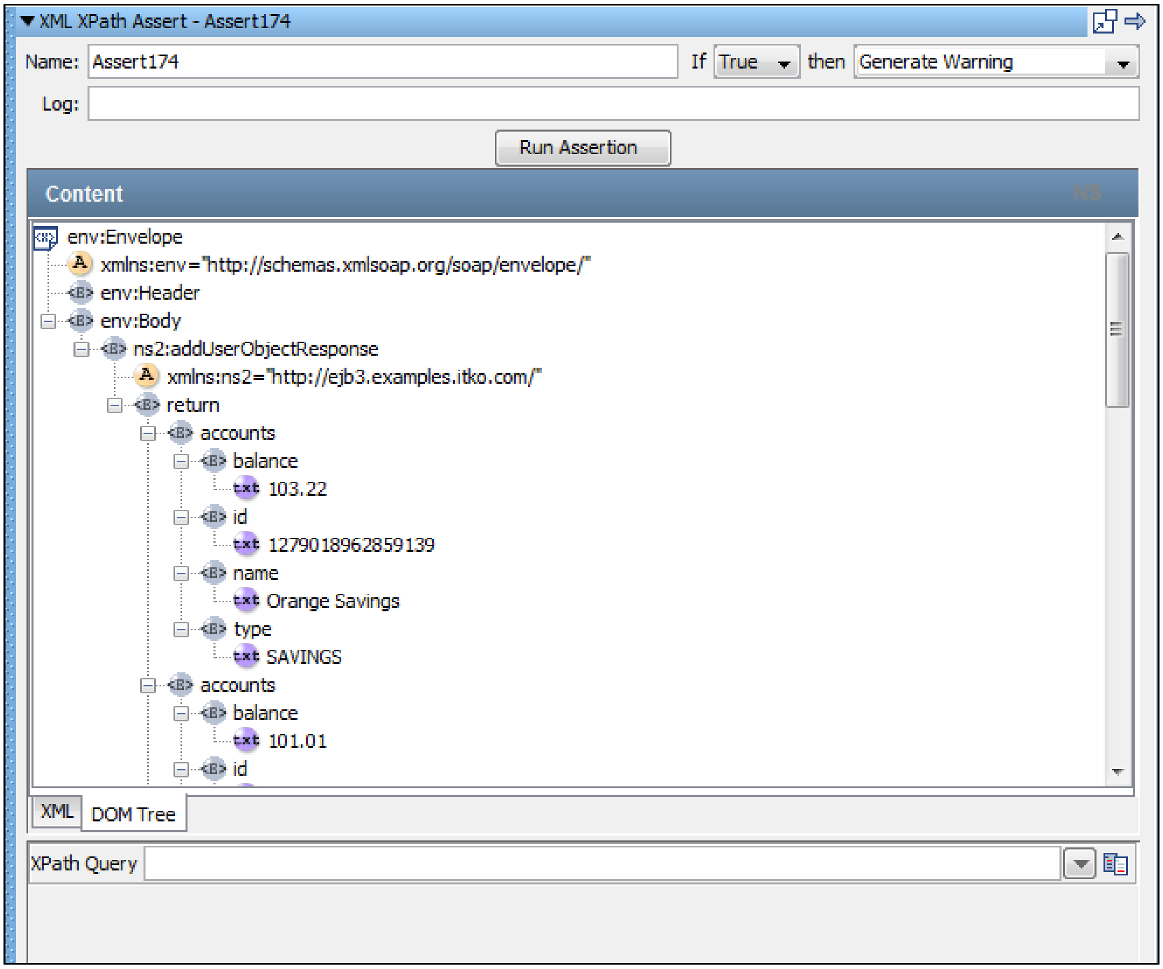
```
<example foo="bar">
    <new-element>This is new</new-elem
</example>
```

XML XPath Assertion

The **XML XPath** assertion lets you use an XPath query that will be run on the response. When this assertion is selected, the last response is loaded into the content panel.

XPath can be thought of as the "SQL for XML." It is a powerful query language that makes parsing XML simple. XPath assertions are useful when you need to validate a web service response in a more sophisticated way than simply parsing the entire result for a given string. For example, you might want to make sure the second and third order item contains "ITKO" and the value of those line items is greater than 10.

The response can be viewed as an XML document or as a DOM Tree. However, the XPath selection can only be made from the DOM Tree view.



Name: Assert174 If True then Generate Warning

Log:

```
env:Envelope
  xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
  env:Header
  env:Body
    ns2:addUserObjectResponse
      xmlns:ns2="http://ejb3.examples.itko.com/"
      return
        accounts
          balance
            103.22
          id
            1279018962859139
          name
            Orange Savings
          type
            SAVINGS
        accounts
          balance
            101.01
          id
```

There are three ways to construct the XPath query:

- Manually enter the XPath expression in the XPath Query text box.
- Select an element from the DOM tree and let LISA construct the XPath expression.
- Select an element from the DOM tree, and then edit the XPath that LISA constructs. For example, you may want to modify it to use a LISA property, or a counter data set.

Enter the following parameters:

- **Name**: The name of this assertion. This will help you identify events for this assertion.
- **If**: Select from the drop-down True or False.
- **Then**: Select from the drop-down Fail the Test, Abort the Test, End the Test, or Go To a step.
- **Log**: The text that will be printed out as event text if the assertion fires.

Now construct the XPath query using one of the methods described previously.

After an XPath query has been constructed, test it by clicking the Run Assertion button on the top of the panel. The results of the query are displayed in the Query Results panel.

The previous example uses the fourth occurrence of the **<wsdl:part>** tag.

It is common to select an XPath node in a web service result and compare the node to a text value to quickly assert that a response contains the expected value. For this common use case, you can add an equality operator to the end of the initial expression that is provided. For example, if we select the new password returned in the response (**BobPass**):

LISA builds the following XPath expression:

string(/env:Envelope/env:Body/ns2:updatePasswordResponse/[name()='return']/**[name()='pwd'])=**

If we add **='NewPassword',** we are comparing the string of the result to the value of the LISA property we used to set the new password. If the equality test does not match, the assertion fails.

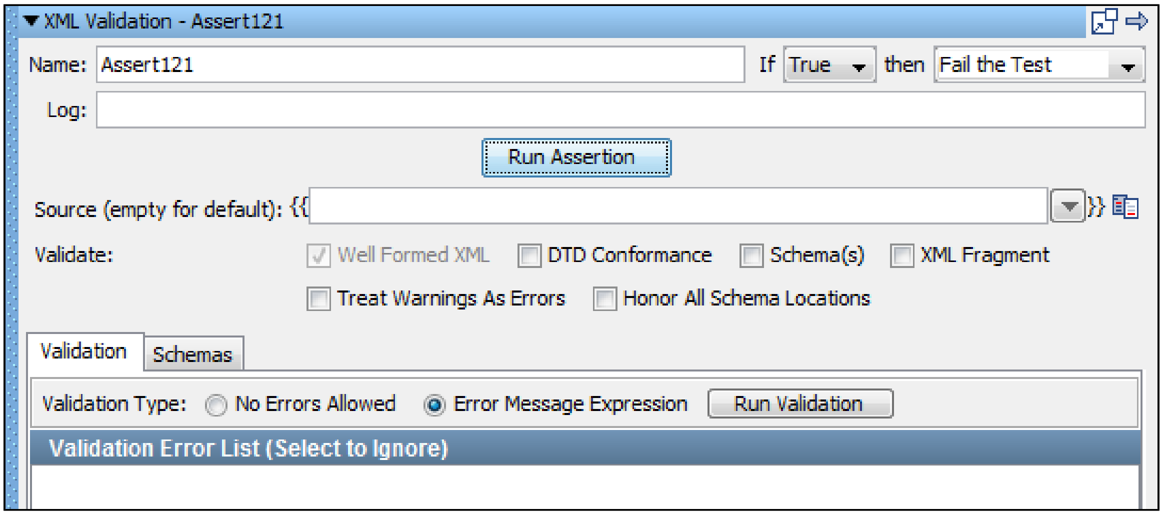
So the entire XPath expression becomes:

string(/env:Envelope/env:Body/ns2:updatePasswordResponse/[name()='return']/**[name()='pwd'])='NewPassword'**

Here we are checking the web service response, looking for the new password, and making sure it matches what we think it should match.

Ensure XML Validation

The **Ensure XML Validation** assertion lets you validate an XML document. You can check to see if the XML document is well-formed, you can validate against a Document Type Definition (DTD), or you can validate against one or more schemas. If you have an XML fragment, you can choose to have LISA add the XML declaration tag. You can also specify that warnings should be treated as errors. You enter the XML to validate as a property.



XML Validation - Assert121

Name: Assert121 If True then Fail the Test

Log:

Source (empty for default): {{ }}

Validate: ☑ Well Formed XML □ DTD Conformance □ Schema(s) □ XML Fragment

□ Treat Warnings As Errors □ Honor All Schema Locations

Validation Type: ○ No Errors Allowed ● Error Message Expression

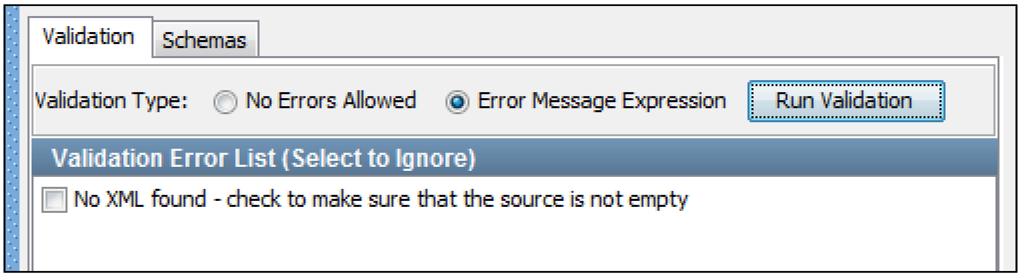
Validation Error List (Select to Ignore)

Enter the following parameters:

- **Name**: The name of this assertion. This will help you identify events for this assertion.
- **If**: Select the behavior of the assertion using the drop-down menu.
- **then**: Select the step to redirect to if the assertion fires.
- **Log**: The text that will be printed out as event text if the assertion fires.
- **Source**: The property that contains the XML. If this is left blank, the last response is used.
- **Validate**: Multiple validation options can be selected:
  - **Well Formed XML**: Check that XML is well-formed.
  - **DTD Conformance**: Check conformance with a DTD.
  - **Schema(s)**: Check conformance with one or more schemas.
  - **XML Fragment**: If XML is a fragment, an XML declaration will be added to the top of XML fragment.
  - **Treat Warnings As Errors**: Warning will be reported as errors.
  - **Honor All Schema Locations**: If you have multiple imports for the same namespace, this option will open each schema location instead of just the first one.

Click the Run Assertion button to execute an assertion.

Validation Tab



Validation Type: ○ No Errors Allowed ● Error Message Expression

Validation Error List (Select to Ignore)

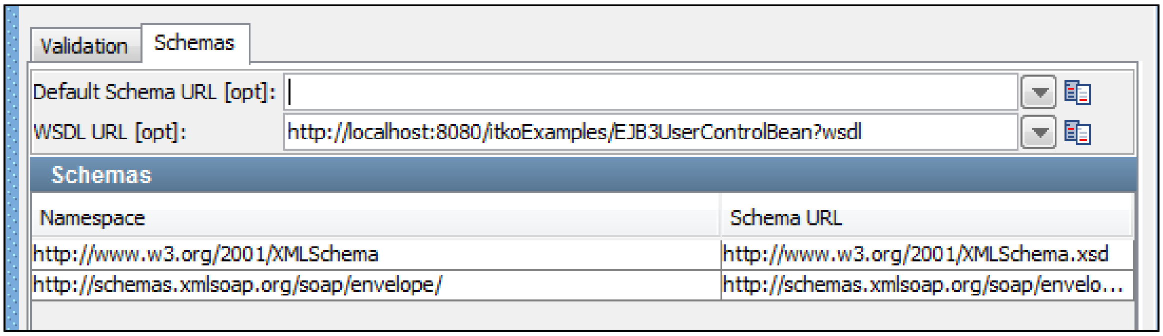
□ No XML found - check to make sure that the source is not empty

You can run the validation by clicking the Run Validation button. Any resulting validation errors are displayed in the **Validation Error List**. Now you can use the Validation Type option buttons to choose how to handle the errors:

- **No Errors Allowed**: The validation fails on any error.
- **Error Message Expressions**: Errors can be marked to be ignored in the validation. The errors to ignore can be checked in the Validation

Error List if you choose this option. An error is shown in the previous example that can be ignored.

Schemas Tab



Default Schema URL [opt]:
WSDL URL [opt]: http://localhost:8080/itkoExamples/EJB3UserControlBean?wsdl

Schemas

Namespace	Schema URL
http://www.w3.org/2001/XMLSchema	http://www.w3.org/2001/XMLSchema.xsd
http://schemas.xmlsoap.org/soap/envelope/	http://schemas.xmlsoap.org/soap/envelo...

Enter the information for each schema you want to use in the validation. You can also specify the default schema:

- **Default Schema URL**: Optionally, specify the default schema's URL.
- **WSDL URL**: Optionally, specify a URL of a WSDL.

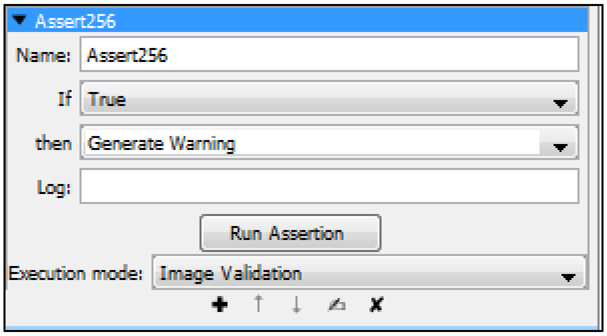
Virtual Service Environment Assertion

The following assertion is available in the Virtual Service Environment assertions list for any test step.

Assert on Execution Mode

Assert on Execution Mode

The **Assert on Execution Mode** assertion will check the current execution mode to its reference and fire if they match. This assertion is primarily used to control step flow for virtual service models.



Name: Assert256
If True
then Generate Warning
Log:
Execution mode: Image Validation

Enter the following parameters:

- **Name**: The name of this assertion. This will help you identify events for this assertion.
- **If**: Select the behavior of the assertion using the drop-down box.
- **then**: Select the step to redirect to if the assertion fires.
- **Log**: The text that will be printed out as event text if the assertion fires.
- **Execution Mode**: Select the execution mode from the available options in the drop-down. For more information about execution modes, see the section "Set the execution mode for the selected virtual service**"** in the *Virtualize Guide*.

Click the Run Assertion button to run and execute the assertion.

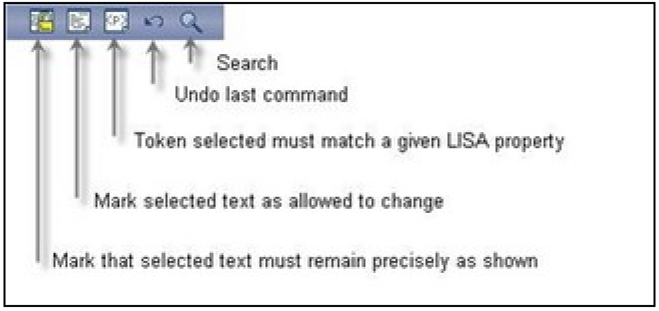
Other Assertions

The following assertions are available in the Other assertions list for any test step.

Highlight Text Content for Comparison Assertion

The **Highlight Text Content for Comparison** assertion uses the "paint the screen" technique specifically designed to work with HTML pages. For example, if there is a large HTML document, you identify the data before and after the content of interest. Then, you simply identify what the "content of interest" will be compared against (usually this is an expected value supplied in a data set).

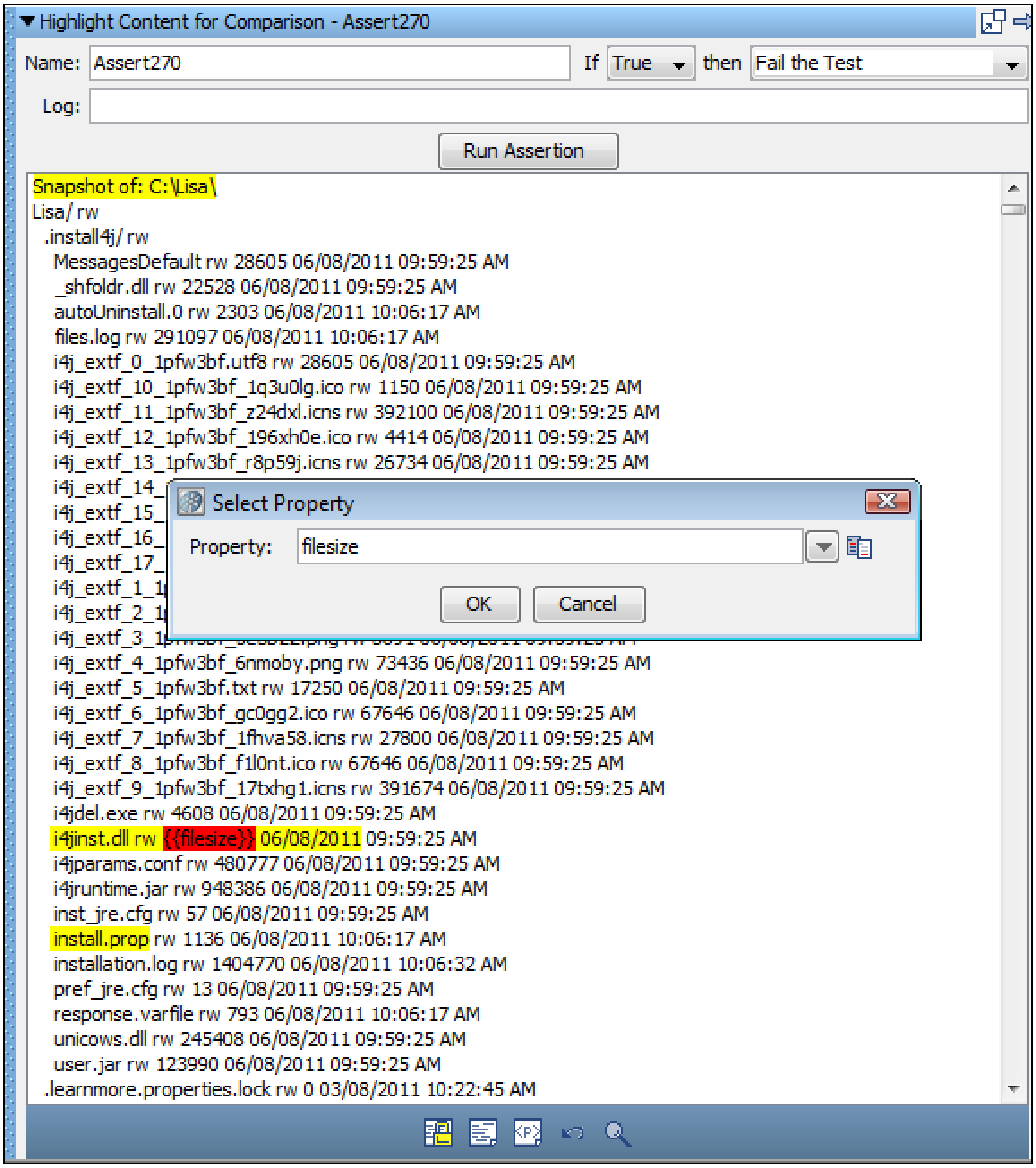
The text is marked using the icons at the bottom of the editor:


Search
Undo last command
Token selected must match a given LISA property
Mark selected text as allowed to change
Mark that selected text must remain precisely as shown

This technique is best explained by example.

In the following example, we want to make sure that certain files appear in the buffer, and one of the file sizes needs to be compared to the value of a property. The text is marked using the three icons shown in the previous image, by selecting text and clicking on the appropriate icon.

- Yellow background indicates text that must appear as shown.
- White background indicates text that need not be present, or can change.
- Red background identifies the text that must match the property entered into the dialog.



Name: Assert270 If True then Fail the Test

Log:

```
Snapshot of: C:\Lisa\
Lisa/ rw
 .install4j/ rw
  MessagesDefault rw 28605 06/08/2011 09:59:25 AM
  _shfoldr.dll rw 22528 06/08/2011 09:59:25 AM
  autoUninstall.0 rw 2303 06/08/2011 10:06:17 AM
  files.log rw 291097 06/08/2011 10:06:17 AM
  i4j_extf_0_1pfw3bf.utf8 rw 28605 06/08/2011 09:59:25 AM
  i4j_extf_10_1pfw3bf_1q3u0lg.ico rw 1150 06/08/2011 09:59:25 AM
  i4j_extf_11_1pfw3bf_z24dxl.icns rw 392100 06/08/2011 09:59:25 AM
  i4j_extf_12_1pfw3bf_196xh0e.ico rw 4414 06/08/2011 09:59:25 AM
  i4j_extf_13_1pfw3bf_r8p59j.icns rw 26734 06/08/2011 09:59:25 AM
  i4j_extf_14_
  i4j_extf_15_
  i4j_extf_16_
  i4j_extf_17_
  i4j_extf_1_1
  i4j_extf_2_1
  i4j_extf_3_1
  i4j_extf_4_1pfw3bf_6nmoby.png rw 73436 06/08/2011 09:59:25 AM
  i4j_extf_5_1pfw3bf.txt rw 17250 06/08/2011 09:59:25 AM
  i4j_extf_6_1pfw3bf_gc0gg2.ico rw 67646 06/08/2011 09:59:25 AM
  i4j_extf_7_1pfw3bf_1fhva58.icns rw 27800 06/08/2011 09:59:25 AM
  i4j_extf_8_1pfw3bf_f1l0nt.ico rw 67646 06/08/2011 09:59:25 AM
  i4j_extf_9_1pfw3bf_17txhg1.icns rw 391674 06/08/2011 09:59:25 AM
  i4jdel.exe rw 4608 06/08/2011 09:59:25 AM
  i4jinst.dll rw {{filesize}} 06/08/2011 09:59:25 AM
  i4jparams.conf rw 480777 06/08/2011 09:59:25 AM
  i4jruntime.jar rw 948386 06/08/2011 09:59:25 AM
  inst_jre.cfg rw 57 06/08/2011 09:59:25 AM
  install.prop rw 1136 06/08/2011 10:06:17 AM
  installation.log rw 1404770 06/08/2011 10:06:32 AM
  pref_jre.cfg rw 13 06/08/2011 09:59:25 AM
  response.varfile rw 793 06/08/2011 10:06:17 AM
  unicows.dll rw 245408 06/08/2011 09:59:25 AM
  user.jar rw 123990 06/08/2011 09:59:25 AM
 .learnmore.properties.lock rw 0 03/08/2011 10:22:45 AM
```

Property: filesize

The set of tokens shown here can be read this way:

- The buffer must start with the phrase in yellow: "Snapshot of: C:\Lisa\".
- There are a number of files that may or may not be in the buffer in the next token, but because it is an "Any" token the variance is immaterial.
- The file "i4jinst.dll" and "rw" attributes must appear.
- The red `filesize` means that the value associated with the property key **filesize** will be swapped into the expression, then the comparison made.
- The text "06/08/2011" must appear.
- The file "install.prop" must appear.
- The buffer can have any amount of content afterward.

After you have finished the markup, enter the following parameters:

- **Name**: The name of this assertion. This will help you identify events for this assertion.

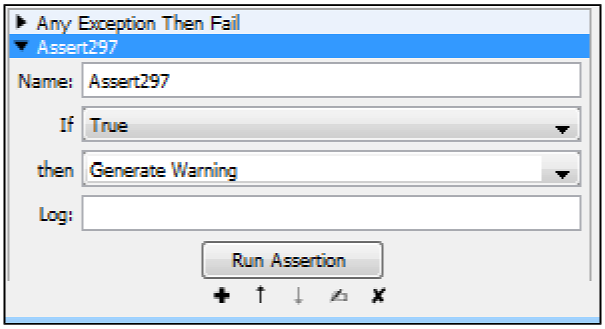
- **If**: Select the behavior of the assertion using the drop-down box.
- **then**: Select the step to redirect to if the assertion fires.
- **Log**: The text that will be printed out as event text if the assertion fires.

Click the Run Assertion button to execute an assertion.

> ⚠ **Property** blocks must always be bounded by **Must** blocks.

Ensure Non-Empty Result Assertion

The **Ensure Non-Empty Result** assertion checks the return from the step to verify that some value has been returned. If there is no response (timeout) or a null value returned, this assertion will return true.



Name: Assert297

If: True

then: Generate Warning

Log:

Enter the following parameters:

- **Name**: The name of this assertion. This will help you identify events for this assertion.
- **If**: Select the behavior of the assertion using the drop-down menu.
- **then**: Select the step to redirect to if the assertion fires.
- **Log**: The text printed out as event text if the assertion fires.

Click the Run Assertion button to execute an assertion.

No other attributes are required.

> ⚠ Use this assertion with caution, as it implements no content validation.

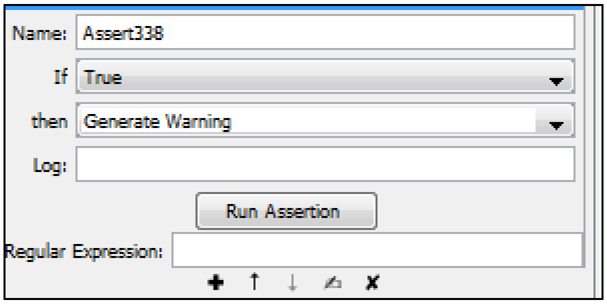
Ensure Result Contains String Assertion

The **Ensure Result Contains String** assertion will return true if the value being searched is found anywhere inside the response. This is typically used to make sure the response contains a required value such as a unique id that was supplied during the request.

For more information see Ensure Result Contains String.

Ensure Result Contains Expression Assertion

The **Ensure Result Contains Expression** assertion lets you check that a specified regular expression occurs somewhere in the result, as text.



Name: Assert338

If True

then Generate Warning

Log:

Regular Expression:

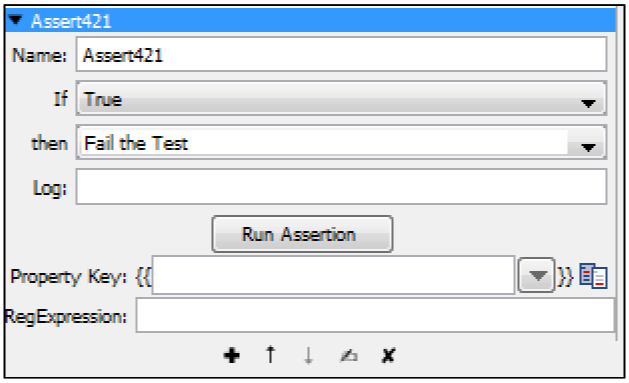
Enter the following parameters:

- **Name**: The name of this assertion. This will help you identify events for this assertion.
- **If**: Select the behavior of the assertion using the drop-down menu.
- **then**: Select the step to redirect to if the assertion fires.
- **Log**: The text that will be printed out as event text if the assertion fires.
- **Regular Expression**: The regular expression to search for in the step result. For example, to check that a number in the 400s appears somewhere in the result, set this parameter to **4/d/d**.

Click the Run Assertion button to execute an assertion.

Ensure Property Matches Expression Assertion

The **Ensure Property Matches Expression** assertion lets you check that the current value of a property matches a specified regular expression.



Assert421

Name: Assert421

If True

then Fail the Test

Log:

Property Key: {{ }}

RegExpression:

Enter the following parameters:

- **Name**: The name of this assertion. This will help you identify events for this assertion.
- **If**: Select the behavior of the assertion using the drop-down box.
- **then**: Select the step to redirect to if the assertion fires.
- **Log**: The text that will be printed out as event text if the assertion fires.
- **Property Key**: The name of the property to be checked. Enter the property name, select from the drop-down list of properties, select an existing string pattern, or create a new string pattern.
- **RegExpression**: The regular expression that must appear in the current value of the property.

Click the Run Assertion button to execute an assertion.

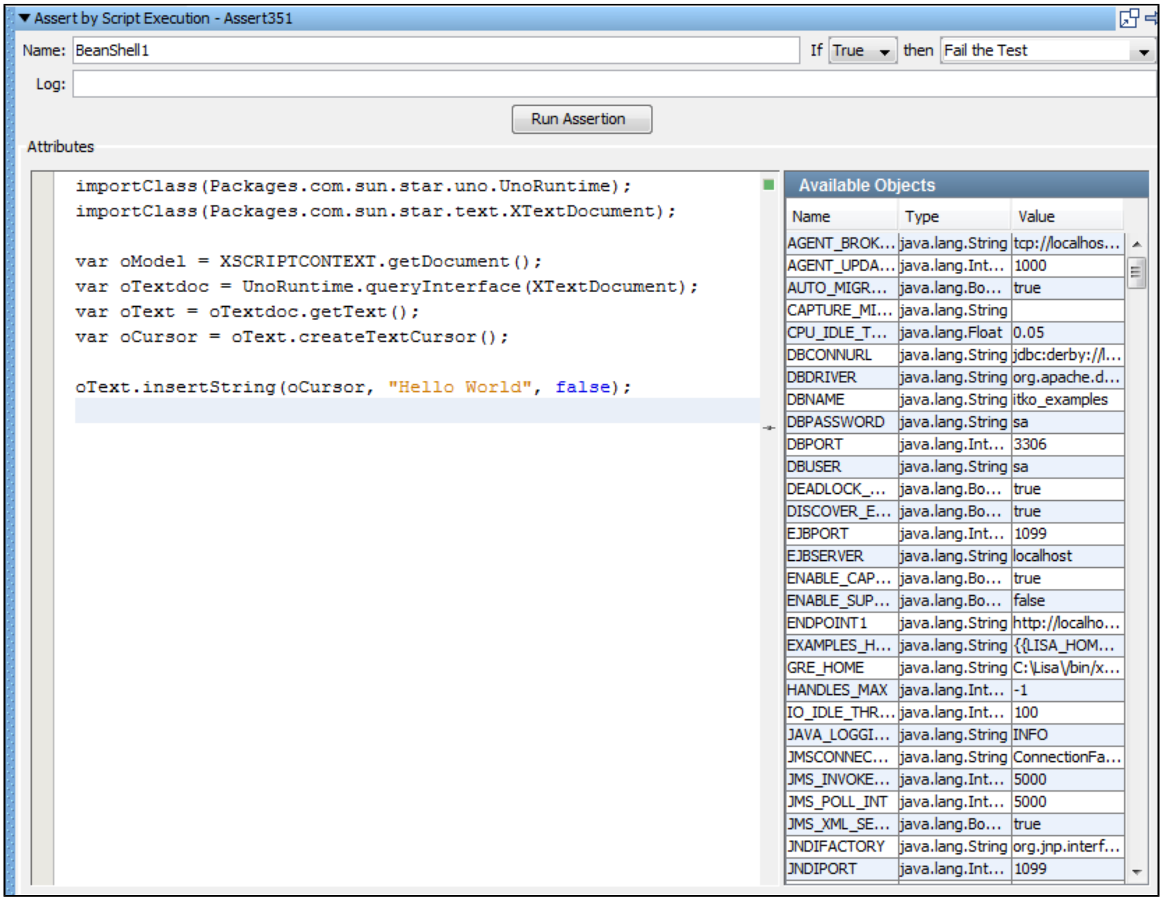
Ensure Step Response Time Assertion

The **Ensure Step Response Time** assertion lets you define an upper and lower threshold for application response time. If the performance is either too fast or too slow, the test case can be failed by using this assertion. Sometimes an application that returns a response very quickly can be a sign that the transaction was not properly processed.

For more information see Ensure Step Response Time.

Scripted Assertion

The **Scripted Assertion** assertion lets you write and execute Java script in the BeanShell interpreter. The result must be a Boolean, or **false** is returned.



Name: BeanShell1 If True then Fail the Test

Log:

```
importClass(Packages.com.sun.star.uno.UnoRuntime);
importClass(Packages.com.sun.star.text.XTextDocument);

var oModel = XSCRIPTCONTEXT.getDocument();
var oTextdoc = UnoRuntime.queryInterface(XTextDocument);
var oText = oTextdoc.getText();
var oCursor = oText.createTextCursor();

oText.insertString(oCursor, "Hello World", false);
```

Name	Type	Value
AGENT_BROK...	java.lang.String	tcp://localhos...
AGENT_UPDA...	java.lang.Int...	1000
AUTO_MIGR...	java.lang.Bo...	true
CAPTURE_MI...	java.lang.String	
CPU_IDLE_T...	java.lang.Float	0.05
DBCONNURL	java.lang.String	jdbc:derby://l...
DBDRIVER	java.lang.String	org.apache.d...
DBNAME	java.lang.String	itko_examples
DBPASSWORD	java.lang.String	sa
DBPORT	java.lang.Int...	3306
DBUSER	java.lang.String	sa
DEADLOCK_...	java.lang.Bo...	true
DISCOVER_E...	java.lang.Bo...	true
EJBPORT	java.lang.Int...	1099
EJBSERVER	java.lang.String	localhost
ENABLE_CAP...	java.lang.Bo...	true
ENABLE_SUP...	java.lang.Bo...	false
ENDPOINT1	java.lang.String	http://localho...
EXAMPLES_H...	java.lang.String	{{LISA_HOM...
GRE_HOME	java.lang.String	C:\Lisa\/bin/x...
HANDLES_MAX	java.lang.Int...	-1
IO_IDLE_THR...	java.lang.Int...	100
JAVA_LOGGI...	java.lang.String	INFO
JMSCONNEC...	java.lang.String	ConnectionFa...
JMS_INVOKE...	java.lang.Int...	5000
JMS_POLL_INT	java.lang.Int...	5000
JMS_XML_SE...	java.lang.Bo...	true
JNDIFACTORY	java.lang.String	org.jnp.interf...
JNDIPORT	java.lang.Int...	1099

Enter the following parameters:

- **Name**: The name of this assertion. This will help you identify events for this assertion.
- **If**: Select the behavior of the assertion using the drop-down box.
- **then**: Select the step to redirect to if the assertion fires.
- **Log**: The text printed out as event text if the assertion fired.

Click the Run Assertion button to execute an assertion.

Enter your script into the script editor on the left.

All the objects available for use in the script editor are listed in the Available Objects panel on the right. This includes primitive types of data like strings and numbers, but also includes objects like any EJB response objects that have been executed in the test case. Double-click an entry in the **Available Objects** table to paste that variable name into the editor area.

Click Test to open a window with the result of the script execution or a description of the errors that occurred.

When you save the test case, the assertion is checked for syntax errors.

Some things to remember:

- If you use LISA properties - {{`someprop`}} - in a script, it will be substituted for the actual value of the property at run time before the script is executed.

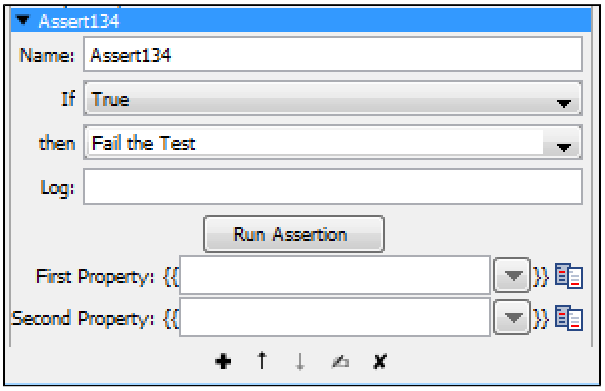
- If you need to get access to a property that has a "." in the name, these are imported into the script environment replacing "." with "_". So {{`foo.bar`}} in a script is the same as **foo_bar**.

- You can produce a LISA log event inside a script step or assertion by using the **testExec** object. To produce a LISA log event, code the following line, as opposed to using the log4j logger. The **testExec.log()** method causes an actual LISA event to be raised. You can see the event in the ITR.

```
testExec.log("Got here");
```

Ensure Properties Are Equal Assertion

The **Ensure Properties Are Equal** assertion lets you compare the values of two properties to make sure that they are same. Typically this is used with a data set and supplied "expected value" to make sure that the application functionality is correct.



Assert134

Name: Assert134

If True

then Fail the Test

Log:

First Property: {{ }}

Second Property: {{ }}

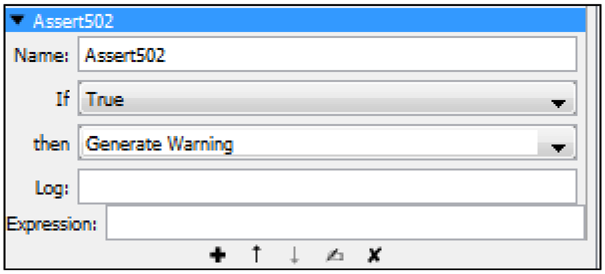
Enter the following parameters:

- **Name**: The name of this assertion. This will help you identify events for this assertion.
- **If**: Select from the drop-down True or False.
- **Then**: Select from the drop-down Fail the Test, Abort the Test, End the Test, or Go To a step.
- **Log**: The text printed out as event text if the assertion fires.
- **First Property**: The first property in the comparison. Enter the property name, select from the drop-down list of properties, select an existing string pattern, or create a new string pattern.
- **Second Property**: The second property in the comparison. Enter the property name, select from the drop-down list of properties, select an existing string pattern, or create a new string pattern.

Click the Run Assertion button to test the assertion.

Assert on Invocation Exception Assertion

The **Assert on Invocation Exception** assertion lets you alter the test flow based on the occurrence of a Java exception. The assertion will assert true if a particular Java exception is returned in the response.



Assert502

Name: Assert502

If True

then Generate Warning

Log:

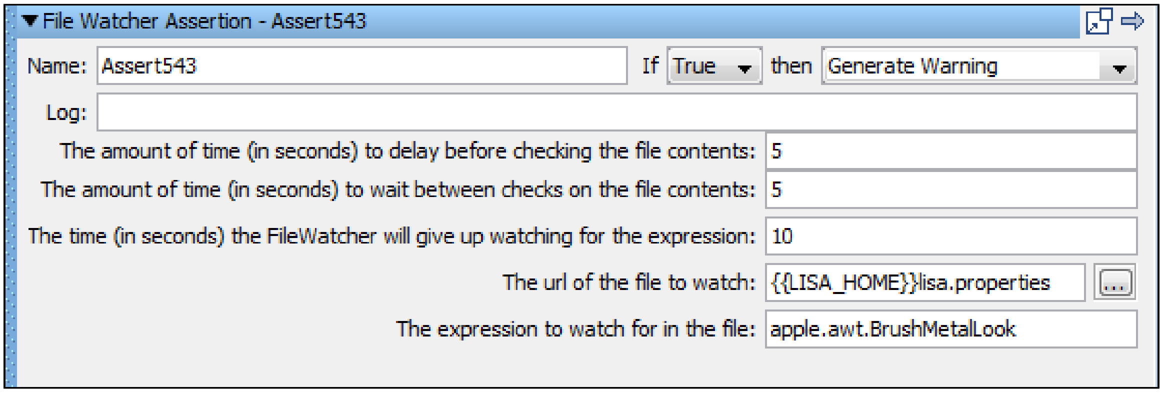
Expression:

Enter the following parameters:

- **Name**: The name of this assertion. This will help you identify events for this assertion.
- **Log**: The text printed out as event text if the assertion fired.
- **Assert**: Select the behavior of the assertion using the option buttons.
- **Execute**: Select the step to redirect to if the assertion fires.
- **Expression**: The expression to search for in the invocation exception. It can be a regular expression. It is common to use the expression '.*'

File Watcher Assertion Assertion

The **File Watcher** assertion lets you monitor a file for given content, and react to the presence (or absence) of a given expression. This assertion runs in the background while your test case is executing.



File Watcher Assertion - Assert543

Name: Assert543 If True then Generate Warning

Log:

The amount of time (in seconds) to delay before checking the file contents: 5

The amount of time (in seconds) to wait between checks on the file contents: 5

The time (in seconds) the FileWatcher will give up watching for the expression: 10

The url of the file to watch: {{LISA_HOME}}lisa.properties

The expression to watch for in the file: apple.awt.BrushMetalLook

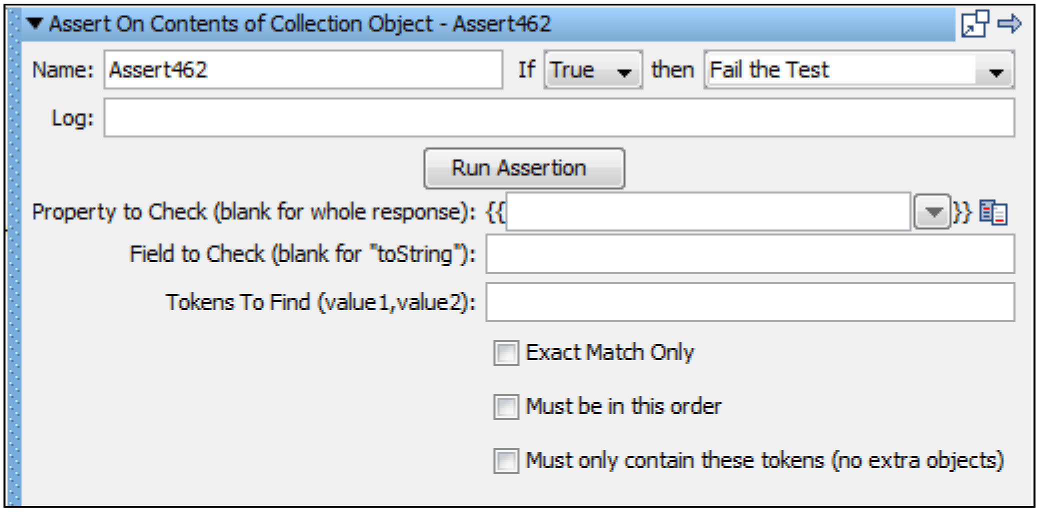
Enter the following parameters:

- **Name**: The name of this assertion. This will help you identify events for this assertion.
- **Log**: The text that will be printed out as event text if the assertion fired.
- **The amount of time (in seconds) to delay before checking the file contents**: The number of seconds to wait before checking the file at the beginning of the step that contains this assertion.
- **The amount of time (in seconds) to wait between checks on the file contents**: The number of seconds to wait between each check.
- **The time (in seconds) the File Watcher will give up watching for the expression**: The total number of seconds this assertion will check for the expression.
- **The url of the file to watch**: The URL or path to the file being watched.
- **The expression to watch for in the file**: The regular expression being watched for in the response.

> ⚠ The times are in seconds and must be integers. They default to 0.

Check Content of Collection Object Assertion

The **Check Content of Collection Object** assertion lets you make simple assertions on the contents of a collection. This is a useful way to find out if certain tokens are in the collection, with the option of adding some simple constraints. For example, if one of the pieces of the data returned from a bank web service is a list of accounts (checking, savings, loan, and so on), this assertion can check to make sure all the account IDs match expected values.



Assert On Contents of Collection Object - Assert462

Name: Assert462 If True then Fail the Test

Log:

Property to Check (blank for whole response): {{ }}

Field to Check (blank for "toString"):

Tokens To Find (value1,value2):

□ Exact Match Only

□ Must be in this order

□ Must only contain these tokens (no extra objects)

Enter the following parameters:

- **Name**: The name of this assertion. This will help you identify events for this assertion.
- **If**: Select from the drop-down True or False.
- **Then**: Select from the drop-down Fail the Test, Abort the Test, End the Test, or Go To a step.
- **Log**: The text printed out as event text if the assertion fired.

- **Property to Check (blank for whole response)**: The name of the property holding the object you want to use in the assertion. Leave this blank to use the last response. The object must be of type **Java Collection** or **Array**.

- **Field to Check (blank for "toString")**: Enter the name of a field and LISA will call its get method.

- **Tokens To Find (value1, value2)**: Comma-delimited string of tokens to check for.

- **Exact Match Only**: The token names much match exactly.

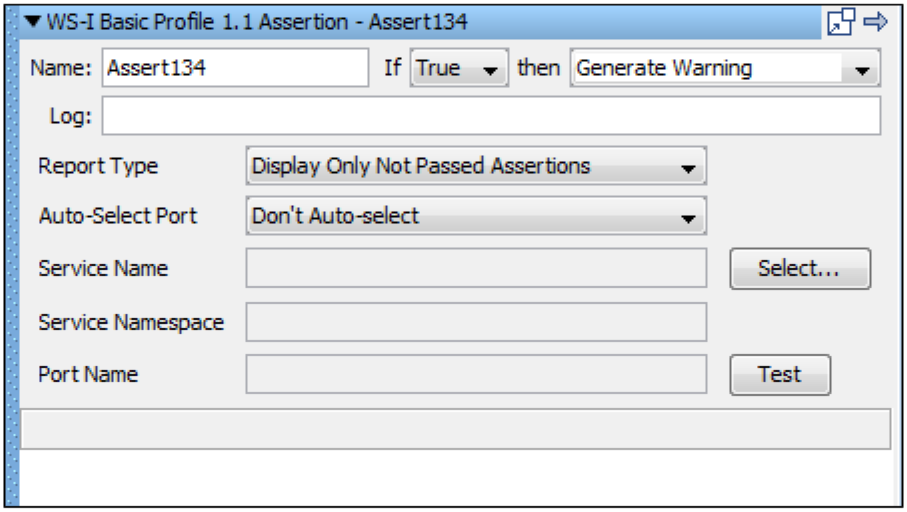
- **Must be in this order**: Select this check box if the tokens must be found in same order as the order in the **Tokens To Find** string.

- **Must only contain these tokens (no extra objects)**: The tokens in the **Tokens To Find** string must be the only tokens found.

Check the Run Assertion button to test the assertion.

WS-I Basic Profile 1.1 Assertion

The **WS-I Basic Profile 1.1** assertion lets you get a WS-I Basic Profile compliance report for a specific web service. This report is delivered in the standard format specified by the WS-I Basic Profile specification.



WS-I Basic Profile 1.1 Assertion - Assert134

Name: Assert134 If True then Generate Warning

Log:

Report Type: Display Only Not Passed Assertions

Auto-Select Port: Don't Auto-select

Service Name

Service Namespace

Port Name

Enter the following parameters:

- **Name**: The name of this assertion. This will help you identify events for this assertion.

- **Log**: The text that will be printed out as event text if the assertion fired.

- **Report Type**: Select the level of (WS-I) assertions to include in the report. There are four levels:
  - **Display All Assertions**
  - **Display All But Info Assertions**
  - **Display Only Failed Assertions**
  - **Display Only Not Passed Assertions**

- **Auto-Select Port**: Determine how the port will be selected – a specific port or **Don't Auto-select**.

- **Service Name**: Select a service name from the list. This may auto-populate from the step.

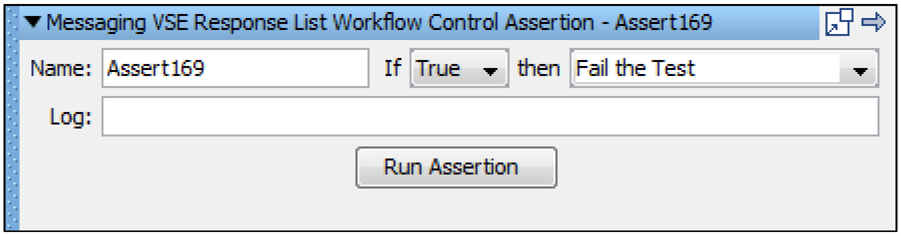
- **Service Namespace**: Is automatically populated based on the service name.

- **Port Name**: Is automatically populated based on the service name.

Click Test to run the assertion.

Messaging VSE Workflow Assertion

The **Messaging VSE Workflow Assertion** assertion is automatically added from the VSE recorder. It serves a specific purpose that enables proper function with VSE recordings. Do not use it unless you know what you are doing, and if has been added to a step in a VSE model, do not remove or edit it.



Messaging VSE Response List Workflow Control Assertion - Assert169

Name: Assert169 If True then Fail the Test

Log:

Click Run Assertion to run the assertion.

Data Sets

A **data set** is a collection of values that can be used to set properties in a test case while a test is running. This provides a mechanism to introduce external test data to a test case.

Data sets are often rows of data that can be inserted into LISA properties as **Name-Value** pairs. But this is not always the case; sometimes a data set will return a single property value.

Data sets can be created internal to LISA, or externally: for instance, in a file or a database table.

While a test is running, LISA will assign properties to the steps specified in the data set editor. When the last data value(s) are read from the data set, the data can be re-used starting at the top of the data set, or the test can be re-directed to any step in the test case.

Data sets can be global or local.

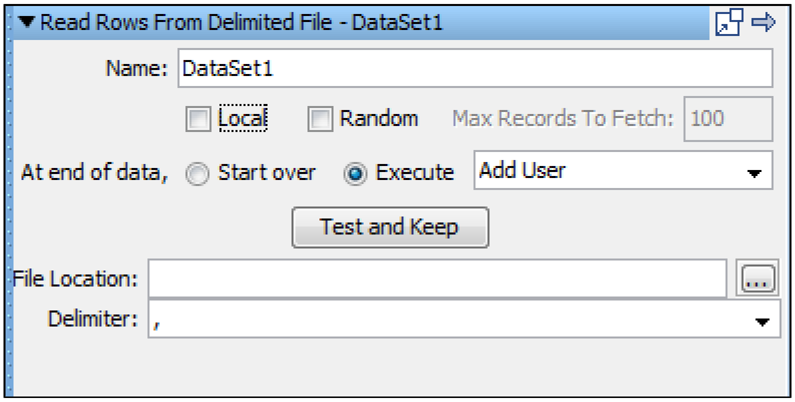
The following topics are available in this chapter.

Global and Local Data Sets

Data sets can be global or local.

Global Data Sets

By default, a data set is global.



Read Rows From Delimited File - DataSet1

Name: DataSet1

□ Local □ Random Max Records To Fetch: 100

At end of data, ○ Start over ● Execute Add User

File Location:

Delimiter: ,

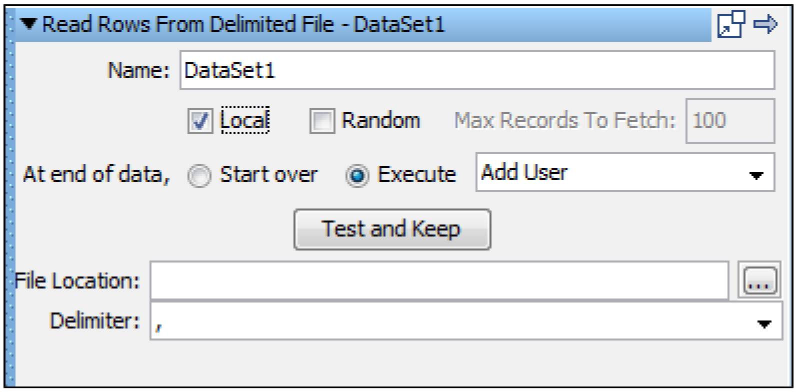
The coordinator server is responsible to provide data to all the test steps.

Here, all the model instances share a single instance of the data set.

The global data set is shared and applied to all instances of the model, even if they are run in different simulators.

Local Data Sets

You can make the data set local by selecting the Local check box while building the data set.



Read Rows From Delimited File - DataSet1

Name: DataSet1

☑ Local ☐ Random Max Records To Fetch: 100

At end of data, ○ Start over ● Execute Add User

Test and Keep

File Location:

Delimiter: ,

Each instance gets (essentially) its own copy of the data set.

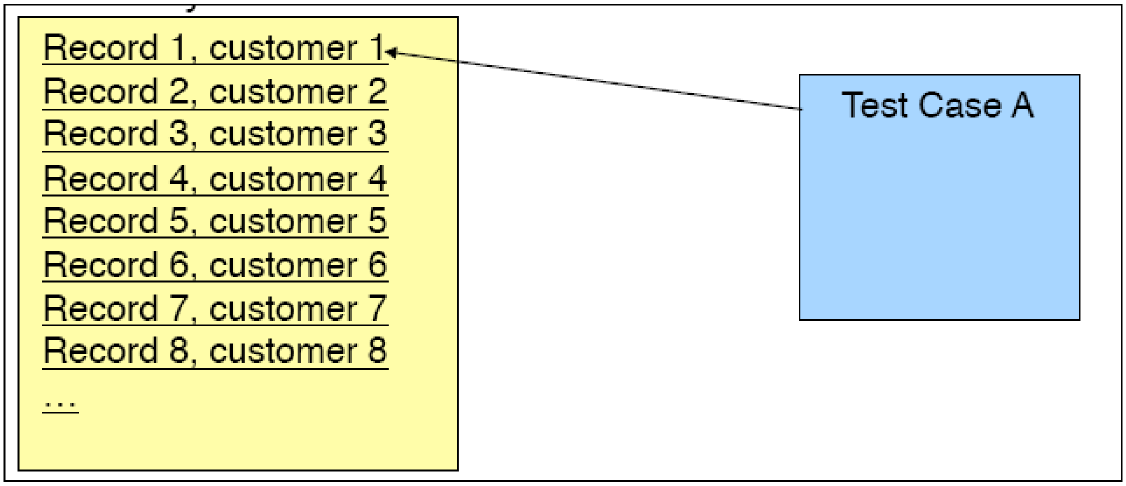
A local data set will provide one copy of the data set to each instance being run.

Example

There are three concurrent virtual users, a local data set with 100 rows of data, and a test case that loops over 100 rows of data and stops. Each virtual user will see all the 100 rows of data in the data set.

For a local data set

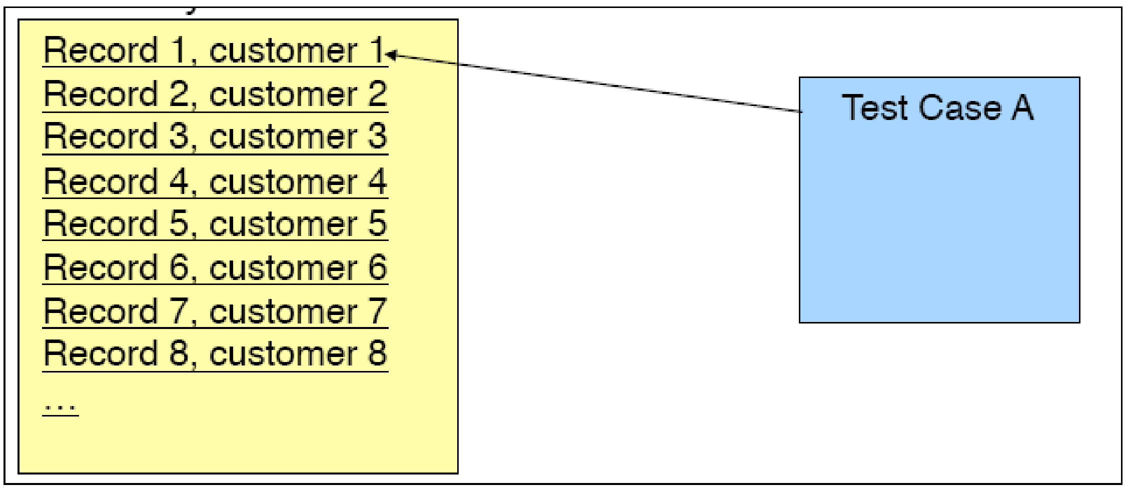
A single run (1 vuser):Test Case A, will get the first row of data: Record 1, customer 1



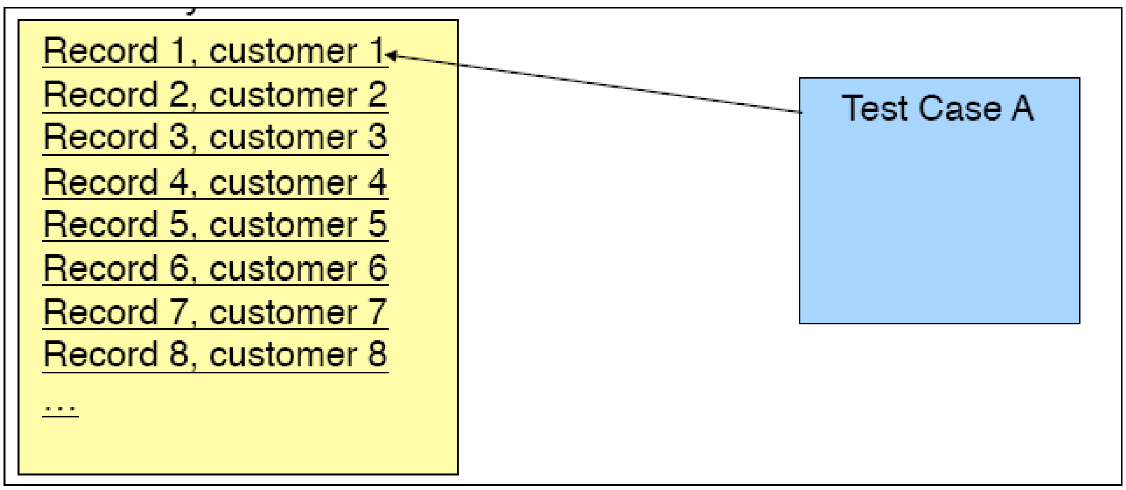
A data set that is shared across virtual users is **global**.

For a global data set

When **Test Case A** is staged with three virtual users or staged to run continuously, each test case will get the next row of data. LISA will share a data set across multiple runs given the instructions specified in the staging document.



Each virtual user (instance) gets its own copy of the data set is **Local**.



When local is marked and Test Case A has a loop, the test will read every row of data in the data set. Each run will get its own copy of the data set.

Test Case Looping

Often the data in a data set drives the number of times a test case is run.

This can be implemented several ways:

- A test is set to finish when all the data in the data set has been exhausted.
- A test is set to re-use the data set when the data has been exhausted.
- A test step can call itself, or a series of steps can be configured in a loop that runs until the data in the data set has been exhausted.

A numeric counter data set can be used to cause a specific step to execute a fixed number of times, which can be set at the step-level or test case-level.

For example, consider a test where we want to test the login functionality of our application using 100 user ID/password pairs. This can be achieved by having a single step call itself until the data set (with 100 rows of data) has been exhausted; at which point it can redirect to the next natural step in the test case, or perhaps the End step. Alternatively, you can use a counter data set with the user ID/password data set.

If a test case contains a global data set on the first step and the data set is set to end the test when the data set is drained, all instances of the test will end for a staged run, overriding any other staging parameters such as steady state time. Local data sets will not end the staged run in this fashion nor will data sets on steps other than the first test.

Random Data Sets

A random data set is a special type of data set, which can be thought as a wrapper around another data set.

In the case of random data sets, you can choose a data set to be randomized for specific steps, and also choose the maximum number of records for randomization.

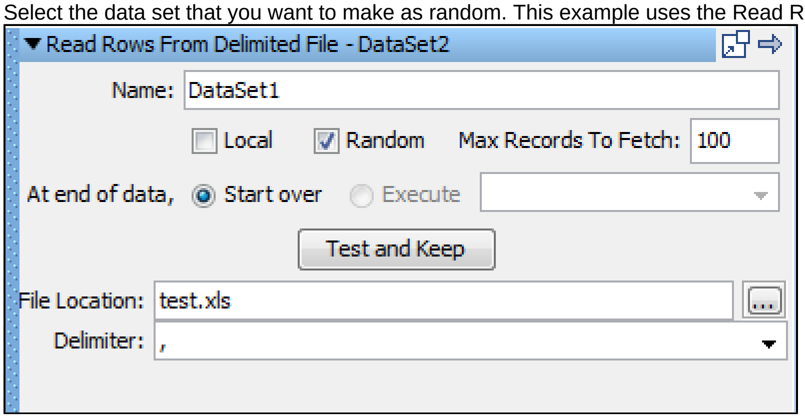
When a random wraps a data set, N copies of the data set are added to the random's list, where N = Max Number of rows. So when N=10, 10 copies of the rows from the data set are made.

When a step references the random data set, a random row is selected from the data set.

If you have a random data set named "RAND1," and it reads one column named "Fruit," and the maxRows is set to "'10," `RAND1` will be the random number generated to pick a row; it will be in the range 0 through N-1. `Fruit` will be the value of "Fruit" column found in that row.

To make a data set random

1. Select the data set that you want to make as random. This example uses the Read Rows from Delimited File data set.



Read Rows From Delimited File - DataSet2

Name: DataSet1

□ Local ☑ Random Max Records To Fetch: 100

At end of data, ● Start over ○ Execute

Test and Keep

File Location: test.xls

Delimiter: ,

2. Select the Random check box.
3. Enter the Max Record to Fetch number. This number is the maximum number of records that you want from the data set. If this is larger than the records in the data set, the smaller number will be used to create the random set.

Example Scenarios

To show the behavior of tests using data sets, consider the following scenarios:

A test has fifteen virtual users, and a data set has two rows of data.

Scenario 1: At the end of data -> Start over

The first user would read the first row of data, the second user would read the second row. The third user would start over and read the first row, and so forth. This would continue until all fifteen users have run the test. The first row was read eight times, the second row seven times.

Scenario 2: At end of data -> Execute End step

The first user would read the first row of data, the second user would read the second row. The third user through the fifteenth user would start the test, and immediately jump to the **End** step.

A test has 100 virtual users, and the data set has 1500 rows of data. Tests are running concurrently.

Scenario 1: At the end of data -> Start over

When the 100 users start, they would read the first 100 rows of data. Depending on the staging document, as cycles end the users would start new runs of the test case and consume more rows of the data set. If all rows are consumed, and the test run has not ended, it would start over with the first row again until the test run ends.

Scenario 2: At end of data -> Execute End step

The test run would end after 1500 cycles.

A test has ten virtual users, and the data set has 10,000 rows of data. The staging document specifies that the test should run for two minutes.

The ten users will start and read the first ten rows of data. They will continue consuming rows of data from the 10,000 rows, and depending on how fast they run would determine how far down the data set they go. At the two minute mark, the test would end.

Adding a Data Set

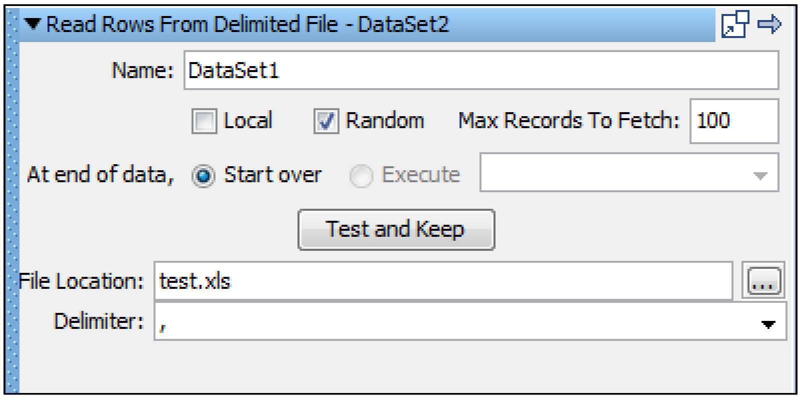
To add a data set

1. Select a test case in the model editor.
2. Expand the Data Sets tab in the right panel.

3. Click the Add icon to open the data set panel listing the Common Data Sets.
4. Click the required data set to open the appropriate Data Set Editor. The editor is specific to each data set.

Data Set Editor Example

The following image shows the editor for the Read rows from a delimited data file data set.



Read Rows From Delimited File - DataSet2

Name: DataSet1

☐ Local ☑ Random Max Records To Fetch: 100

At end of data, ● Start over ○ Execute

File Location: test.xls

Delimiter: ,

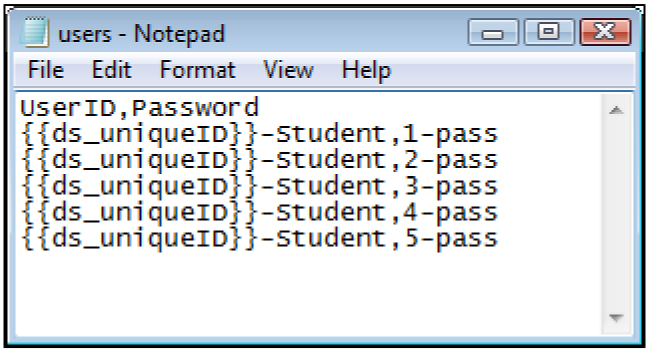
The top panel of the data set editor is common to all data set types. It consists of:

- **Name**: The name of the data set.
- **Local**: Select the check box if you want a local data set. The default is global (not checked).
- **Random:** Select the Random check box if you want to make this a random data set and enter the Max records to fetch number.
- **At end of data**: Instructions for how to proceed after all the data has been read. There are two options:
  - **Start over**: Continue reading data from the top of the data set.
  - **Execute**: Select the step to execute after all the data has been read. The pull-down menu has been pre-populated with all the available steps in the test case.
- **Test and Keep**: After all the parameters have been entered click this button to test the data set, and to load it into the appropriate steps in the test case

The bottom panel of the data set editor is specific to the data set being created. For the Read rows from a delimited data file data set, enter the following:

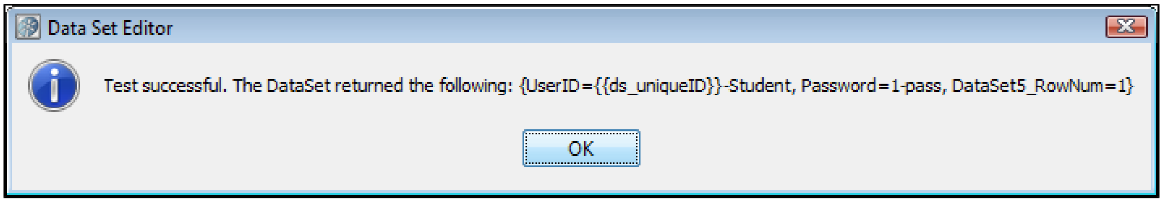
- **File Location**: Enter the full path name of the text file, or browse to file with the browse button. You can use a property in the path name (for example, `LISA_HOME`).
- **Delimiter**: Enter the delimiter being used. Any value is allowed as a delimiter. Common delimiters are provided in the drop-down menu.

This data set requires a delimited text file. In the following example, the first line specifies the property names: userid and password. Subsequent lines list the data values to be used for these properties.



```
UserID,Password
{{ds_uniqueID}}-Student,1-pass
{{ds_uniqueID}}-Student,2-pass
{{ds_uniqueID}}-Student,3-pass
{{ds_uniqueID}}-Student,4-pass
{{ds_uniqueID}}-Student,5-pass
```

When you click Test and Keep, the first row of data is loaded, and a message confirms that the data set can be read and shows the first row of data.



Test successful. The DataSet returned the following: {UserID={{ds_uniqueID}}-Student, Password=1-pass, DataSet5_RowNum=1}

This example shows a data set named DataSet1 that will be used until all the rows have been used, and then the **end** step will be executed.

Ending a Test Case by a Data Set

The following conditions must be met for a data set to end a test run:

- The data set must be global.
- The data set must be set "At end of data: Execute end".
- The data set must be increased on the first step of the test case.

Be careful when you apply a global data set to the first step in a test case as it will pull down the entire staged run before completing the cycles.

Deleting a Data Set

To delete a data set

1. Do one of the following:
    - Select the data set in the Elements panel, and click the Delete icon on the toolbar.
    - Right-click the data set in the model editor and select Delete.
      A message box appears to confirm the deletion of the data set.
2. Click Yes.

Reordering a Data Set

To reorder a data set

1. Select the data set in right panel.
2. Click the up or down arrow on the toolbar to move the selected data set up or down.

Renaming a Data Set

To rename a data set

- Select the data set in the Elements panel and click the Rename icon on the toolbar.

- Select the data set in the model editor and right-click to open a menu to rename.

Moving a Data Set

You can move data sets in the Model Editor from one test step to another by using drag and drop.

To move a data set

1. Click the data set attached to a test step: for example, Step 1.
2. Select the data set and drag it to the target test step: for example, Step 2, in the Model Editor and leave it there.
3. The dragged data set will then be applied to Step 2.

Data Set Next Step Selection

You can select the next step or the end step to be executed after the data set has fired.

To select the next step

1. Right-click the data set in the Model Editor to open the menu.
2. Click the At end menu and select the next step to be executed.

Data Sets and Properties

Data sets can use properties.

There are some major uses of properties:

- When specifying the location of an external data set, we can use a property, perhaps LISA_HOME rather than a hard coded value for the path name. For example - **LISA_HOME/myTests/myDataset.csv**

This increases the portability of the data set. Properties used in this way must be specified as a system property, or be defined in a Configuration, because they usually have to be available early in the test run. You can define a dummy value in your Configuration and modify it at a later date. This will allow the file to be found at design time. This use of properties is important when you are running your tests on LISA Server.

- Data set values can contain properties. These will be evaluated when the value is read. For example, we could set up a login value in the data set as student _1. Then, if the current value of **student** is **Bart**, the resulting data value becomes **Bart_1**.

- Local data sets can use properties from the test case, but global data sets cannot because they are shared by all virtual users.

Types of Data Sets

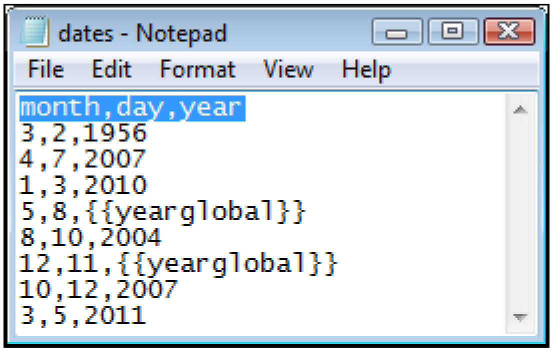
The types of data sets are documented as they appear on the menu.

Read Rows from a Delimited Data File Data Set
Create your own Data Sheet Data Set
Create your own Set of Large Data Data Set
Read Rows from a JDBC Table Data Set
Create a Numeric Counting Data Set
Read Rows from Excel File Data Set
Read DTO's from Excel File
Unique Code Generator Data Set
Random Code Generator Data Set
Message_Correlation ID Generator Data Set
Load a Set of File Names Data Set
XML Data Set

Read Rows from a Delimited Data File Data Set

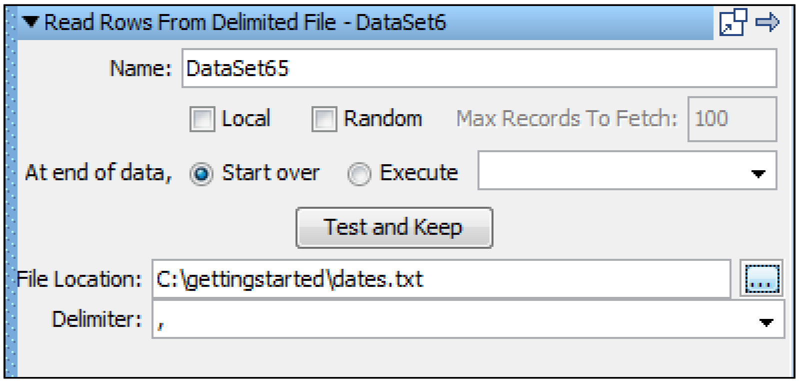
The **Read Rows from a Delimited Data File** data set assigns values to properties based on the contents of a text file. This is the most commonly used type of data set in LISA. The first line of the text file specifies the names of the properties into which the data values will be stored. Subsequent lines list the data values to be used for these properties. The text file is created using a simple text editor.

Following is an example of a comma-delimited data file. The first row, which is highlighted, shows the property names in this data set.



```
month,day,year
3,2,1956
4,7,2007
1,3,2010
5,8,{{yearglobal}}
8,10,2004
12,11,{{yearglobal}}
10,12,2007
3,5,2011
```

The Data Set Editor is used to define the data set.



Name: DataSet65

☐ Local ☐ Random Max Records To Fetch: 100

At end of data, ● Start over ○ Execute

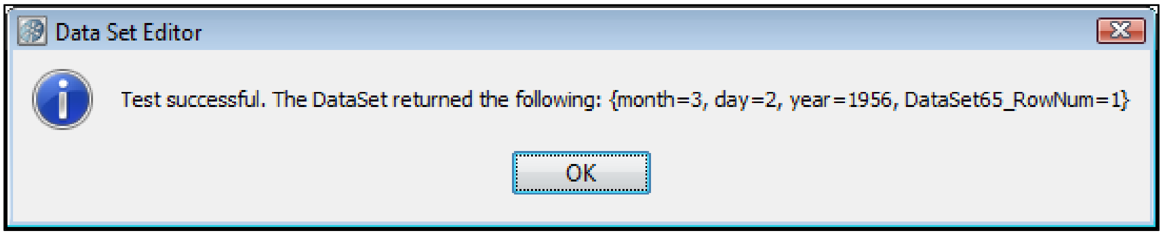
File Location: C:\gettingstarted\dates.txt

Delimiter: ,

In the Data Set Editor enter the following:

- **Name**: Enter the name of the data set.
- **At End Of Data**: Select whether you want to start over and read values from the start of the data set, or execute the step you select in the pull-down menu.
- **Local**: Designate whether this should act as a global or local data set. Global is the default. Local data sets are created one-per-simulator; global data sets are created once and shared by all simulators.
- **File Location**: The full path name of the text file, or browse to file with the **Browse** button.
- **Delimiter**: The delimiter being used. Any character is allowed as a delimiter. There are common delimiters in the drop-down menu.

Click the Test and Keep button to test and load the data. You will get a message that confirms that the data set can be read, and shows the first set of data.

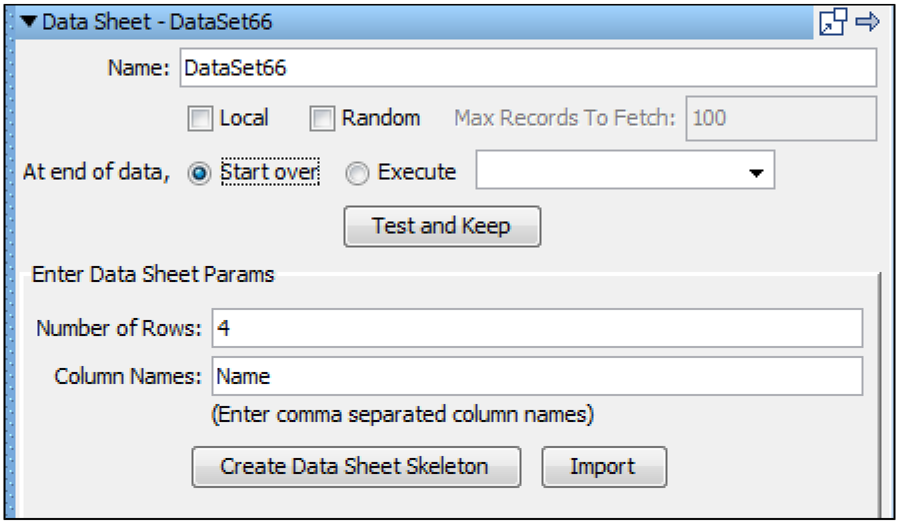


Test successful. The DataSet returned the following: {month=3, day=2, year=1956, DataSet65_RowNum=1}

Create your own Data Sheet Data Set

The **Data Sheet** data set lets you generate your data set data within LISA, without requiring any reference to an external file.

The Data Sheet consists of a data table; the column headings specify the property names and the table rows specify the data values for those properties. The table skeleton is built by specifying the number of rows and the column names (properties).



Name: DataSet66

□ Local □ Random Max Records To Fetch: 100

At end of data, ● Start over ○ Execute

Number of Rows: 4

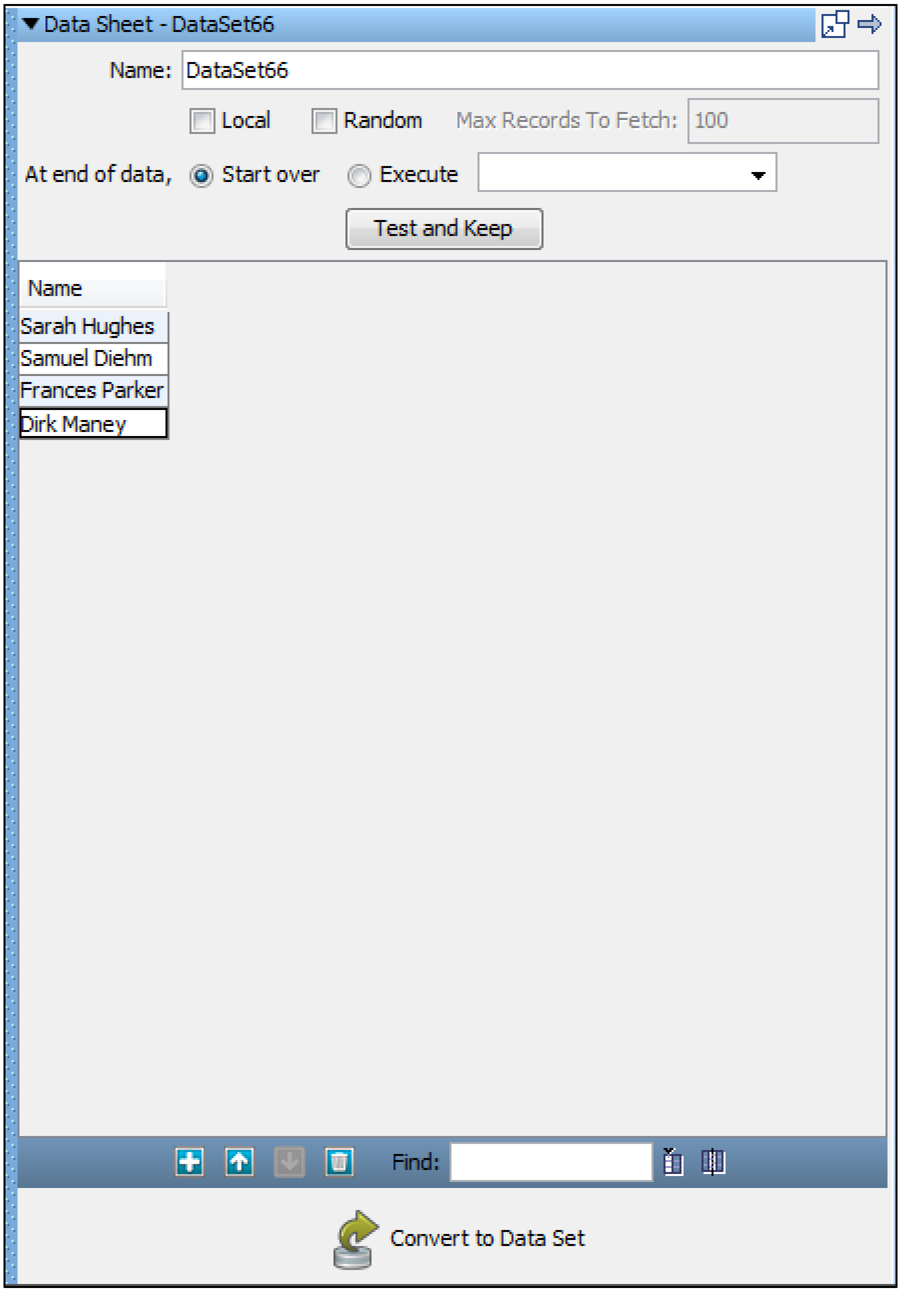
Column Names: Name

(Enter comma separated column names)

In the Data Set Editor enter the following:

- **Name**: The name of the data set.
- **At End Of Data**: Select whether you want to start over and read values from the start of the data set, or execute the step you select in the pull-down menu.
- **Local**: Whether this should act as a global or local data set. Global is the default. Local data sets are created one-per-simulator; global data sets are created once and shared by all simulators.
- **Number of Rows**: Initial estimate of number of rows (it can be modified later).
- **Column Names**: Comma delimited list of column names (these are also the property names).

Click the Create Data Sheet Skeleton button.
Enter your data into the table. Following is an example of a data sheet:



Data Sheet - DataSet66

Name: DataSet66

□ Local □ Random Max Records To Fetch: 100

At end of data, ● Start over ○ Execute

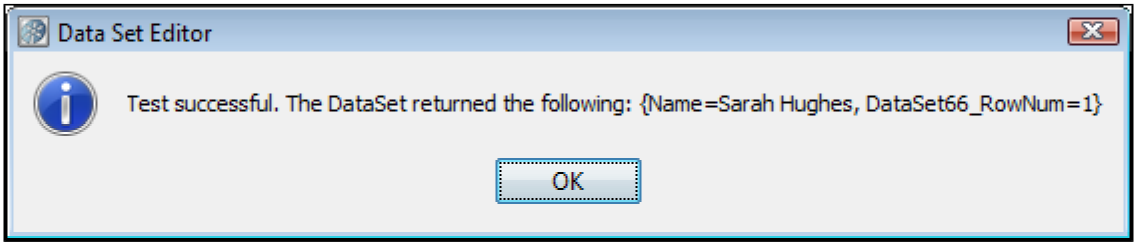
Name
Sarah Hughes
Samuel Diehm
Frances Parker
Dirk Maney

Find:

Use the functions in the bottom toolbar to create and modify the table:

- **Add**: Add rows to the table.
- **Delete**: Delete the selected row.
- **Up and Down**: Select a row and move it up or down in the table.
- **Add Column**: Add a column to your table.
- **Delete Column**: Delete a column. Select a cell in the column you want deleted, making sure that the cell is selected for editing, and click this button.

Click the Test & Keep button to test and load the data. You will get a message that confirms that the data set can be read, and shows the first set of data:

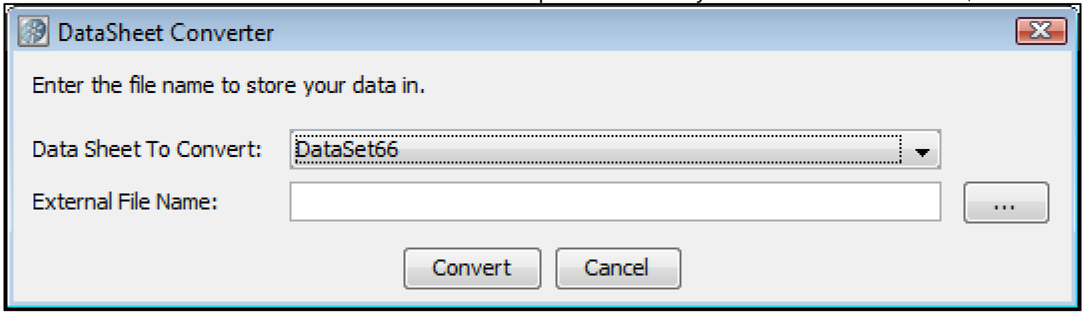


Data Set Editor

Test successful. The DataSet returned the following: {Name=Sarah Hughes, DataSet66_RowNum=1}

You can also:

- Double-click the column header to sort the rows
- Right-click on the column name to change the column name
- Select a cell, and then right-click on a cell to select the toolbar functions and to Launch Extended View to edit the cell
- Click the Convert to Data Set button at the bottom of the panel to convert your data sheet to a data set, or a file on the file system.



DataSheet Converter

Enter the file name to store your data in.

Data Sheet To Convert: DataSet66

External File Name:

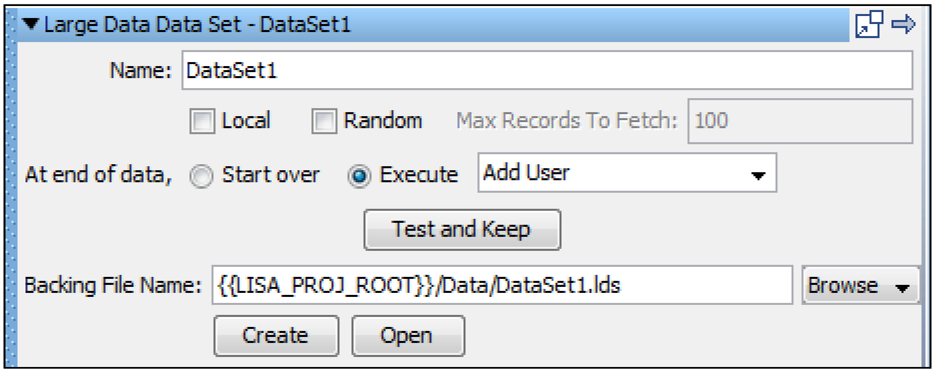
- **Data Sheet to Convert**: By default, the data sheet you just created, or select from the pull-down list.
- **External File Name**: The name and path of the file you want to create from the datasheet.

> ⚠ Moving rows up or down in the table may affect the outcome of a test. The order of columns will not affect the outcome of the test.

Select the steps that will use the data sheet.

Create your own Set of Large Data Data Set

The **Create your own Set of Large Data** data set lets you define a custom data table that can be arbitrarily large. The data can have any number of rows and columns. A backing file name is the file in which all the data is stored.



Large Data Data Set - DataSet1

Name: DataSet1

□ Local □ Random Max Records To Fetch: 100

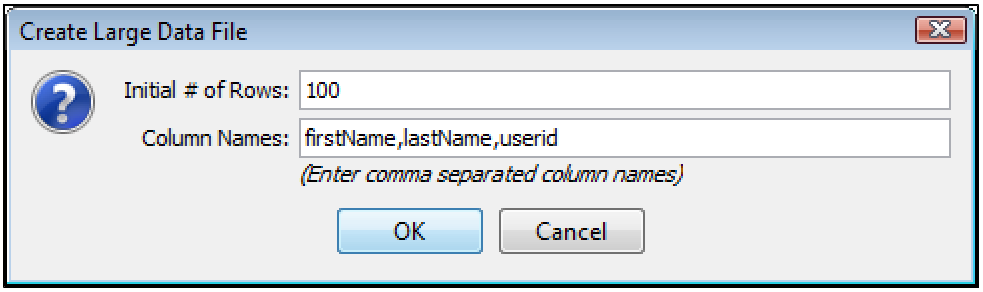
At end of data, ○ Start over ● Execute Add User

Backing File Name: {{LISA_PROJ_ROOT}}/Data/DataSet1.lds

Enter the following parameters:

- **Name**: The data set name.
- **Local**: Whether this should act as a global or local data set. Global is the default. Local data sets are created one-per-simulator; global data sets are created once and shared by all simulators.
- **Random**: Whether the record after the current record (sequential access) will be read, or a random record will be read. Sequential reading is the default.
- **Max Records to Fetch**: The upper bound on the number of records to fetch for random access. This text field is disabled if the Random check box is not selected.
- **At end of data, Start over or Execute**: Select whether you want to start over and read values from the start of the data set, or execute the step you select in the pull-down menu
- **Backing file name**: This is the name of the file in which data is stored. This file is created automatically and data that you supply is inserted in it.

The Create button will bring up a panel that lets you specify additional parameters for file creation.

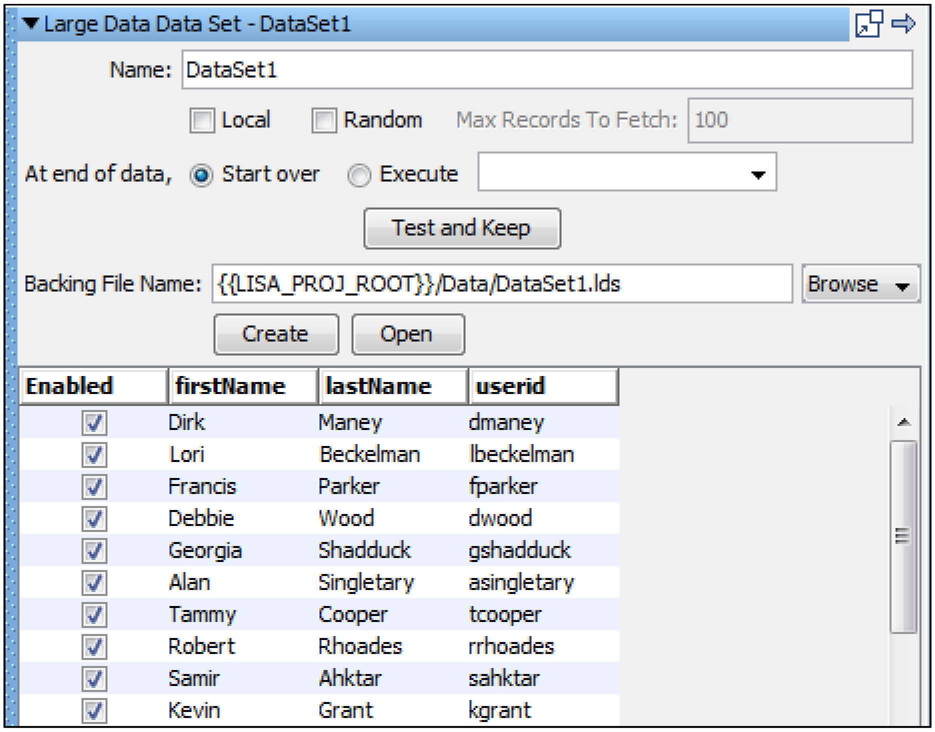


Initial # of Rows: 100

Column Names: firstName,lastName,userid

(Enter comma separated column names)

- **Initial # of Rows**: This field gives the initial number of rows to be created. You can add additional rows using the editor.
- **Column Names**: This field expects column names separated by commas that will go into the data set.



Name: DataSet1

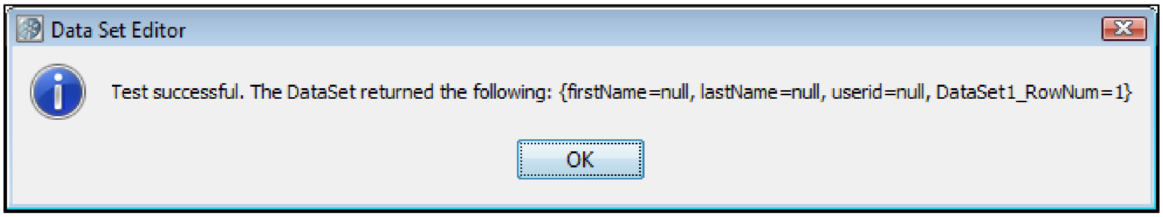
□ Local □ Random Max Records To Fetch: 100

At end of data, ● Start over ○ Execute

Backing File Name: {{LISA_PROJ_ROOT}}/Data/DataSet1.lds

Enabled	firstName	lastName	userid
☑	Dirk	Maney	dmaney
☑	Lori	Beckelman	lbeckelman
☑	Francis	Parker	fparker
☑	Debbie	Wood	dwood
☑	Georgia	Shadduck	gshadduck
☑	Alan	Singletary	asingletary
☑	Tammy	Cooper	tcooper
☑	Robert	Rhoades	rrhoades
☑	Samir	Ahktar	sahktar
☑	Kevin	Grant	kgrant

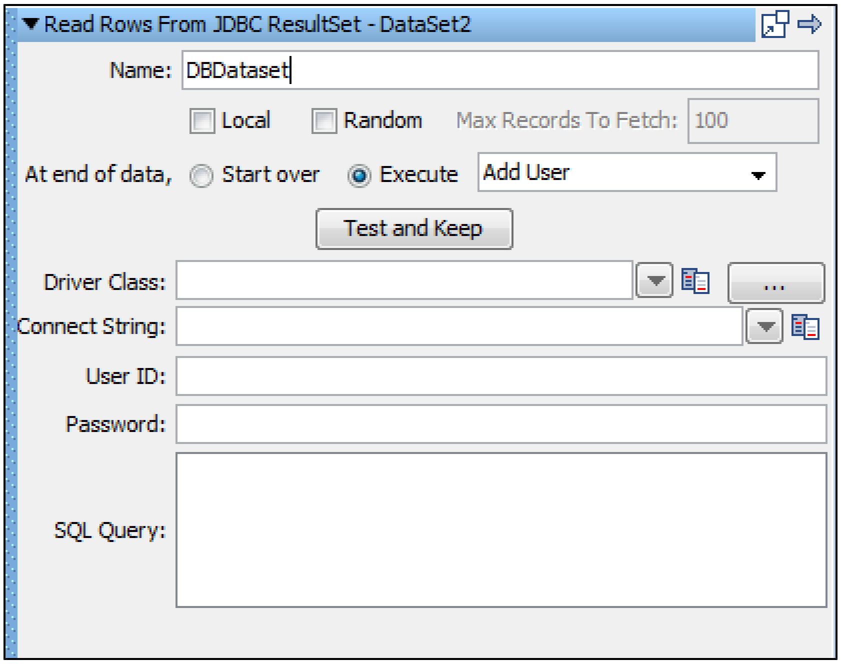
Pressing the Test and Keep button after entering data in the columns that we created before, data will be copied in the backing file created.



Test successful. The DataSet returned the following: {firstName=null, lastName=null, userid=null, DataSet1_RowNum=1}

Read Rows from a JDBC Table Data Set

The **Read Rows from a JDBC Table** data set is used to read source test case data from a database. The data is read using a JDBC driver (which must be supplied by the user). Each column in the table of data will be represented as a LISA property. The data set will then loop across the rows returned from the SQL query.



Read Rows From JDBC ResultSet - DataSet2

Name: DBDataset

□ Local □ Random Max Records To Fetch: 100

At end of data, ○ Start over ● Execute Add User

Driver Class:

Connect String:

User ID:

Password:

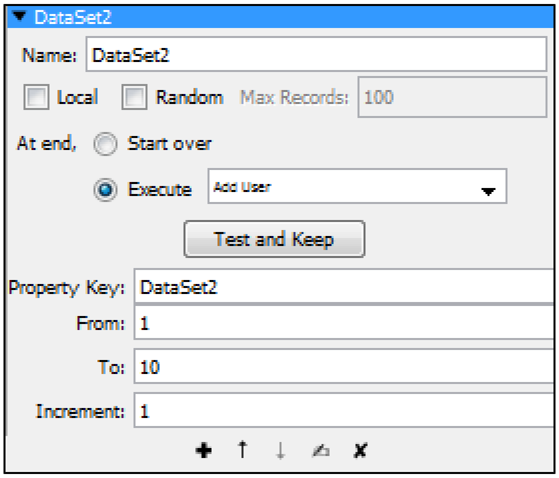
SQL Query:

In the Data Set Editor, enter the following:

- **Name**: The name of the data set.
- **Local:** Whether this should act as a global or local data set. Global is the default. Local data sets are created one-per-simulator; global data sets are created once and shared by all simulators.
- **Random:** Whether the record after the current record (sequential access) will be read, or a random record will be read. Sequential reading is the default.
- **Max Records to Fetch:** The upper bound on the number of records to fetch for random access. This text field is disabled if the Random check box is not selected.
- **At End Of Data**: Select whether you want to Start Over and read values from the start of the data set, or Execute the step you select in the pull-down menu.
- **Driver Class**: Enter or select the full package name of the appropriate driver class. Standard driver classes are available in the pull-down menu.
- **Connect String**: This is the standard JDBC URL for your database. Enter or select the URL. JDBC URL templates for common database managers are available in the pull-down menu.
- **User ID**: Enter a user ID (if it is required by the database).
- **Password**: Enter a password (if it is required by the database).
- **SQL Query**: The SQL query used to create the data set.
  Click the Test and Keep button to test and load the data. You will get a dialog window that confirms that the data set can be read, and shows the first set of data.

Create a Numeric Counting Data Set

The **Create a Numeric Counting** data set assigns a number to a property. The number assigned starts at a given value and changes by a fixed step every time the data set is used until it exceeds a known limit. This data set is used to simulate a "for" loop, or to set the number of times something will occur. For example, you might want to make 100 calls to the same step. The following example illustrates how to do this using the Create a Numeric Counting Data Set.



Name: DataSet2

☐ Local ☐ Random Max Records: 100

At end, ○ Start over

● Execute Add User

Property Key: DataSet2

From: 1

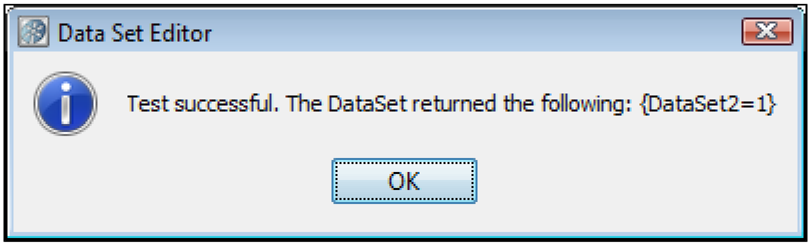
To: 10

Increment: 1

In the Data Set Editor enter the following:

- **Name**: The name of the data set.
- **Local:** Whether this should act as a global or local data set. Global is the default. Local data sets are created one-per-simulator; global data sets are created once and shared by all simulators.
- **Random:** Whether the record after the current record (sequential access) will be read, or a random record will be read. Sequential reading is the default.
- **Max Records:** The upper bound on the number of records to fetch for random access. This text field is disabled if the Random check box is not selected.
- **At End Of Data**: Select whether you want to start over and read values from the start of the data set, or execute the step you select in the pull-down menu.
- **Property Key**: The name of the property into which the counter value will be stored.
- **From**: The initial counter value.
- **To**: The final counter value.
- **Increment**: The step increment for the counter. The counter data set can be used to count backwards by assigning a negative increment.

Click the Test and Keep button to test and load the data. You will get a message that confirms that the data set can be read, and shows the first set of data.



Test successful. The DataSet returned the following: {DataSet2=1}

Select the steps that will use the data set.

In our example, the data set is configured to start at 1, increment by 1 until it exceeds 10.

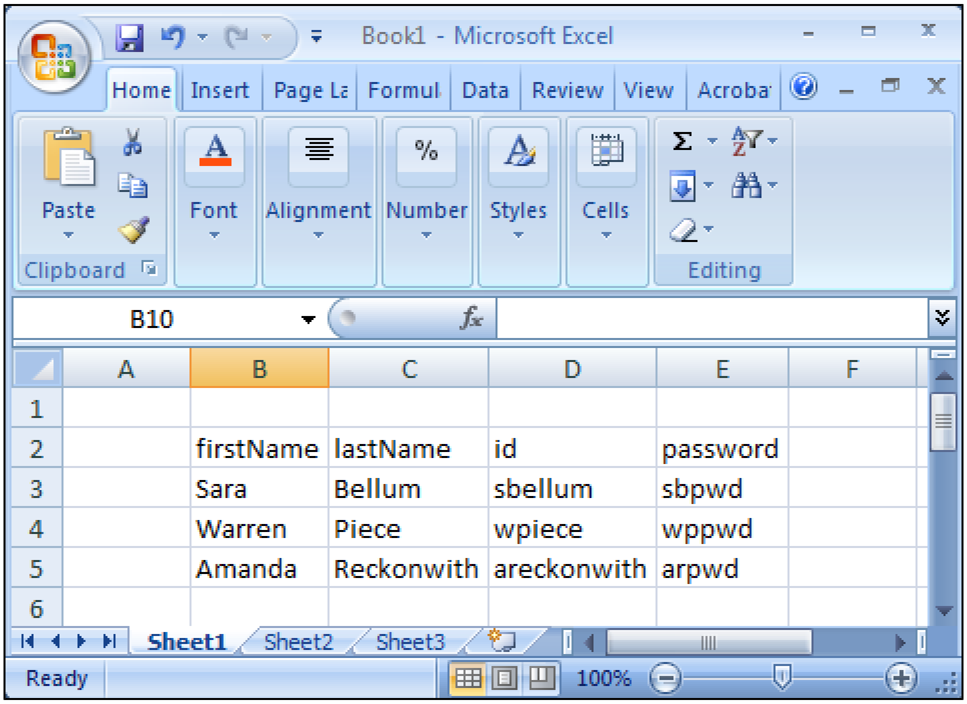
To iterate on a single step, change that step's "next step" to be itself. Do this by going to the step's Base Step Info and selecting Next for the Step element.

Read Rows from Excel File Data Set

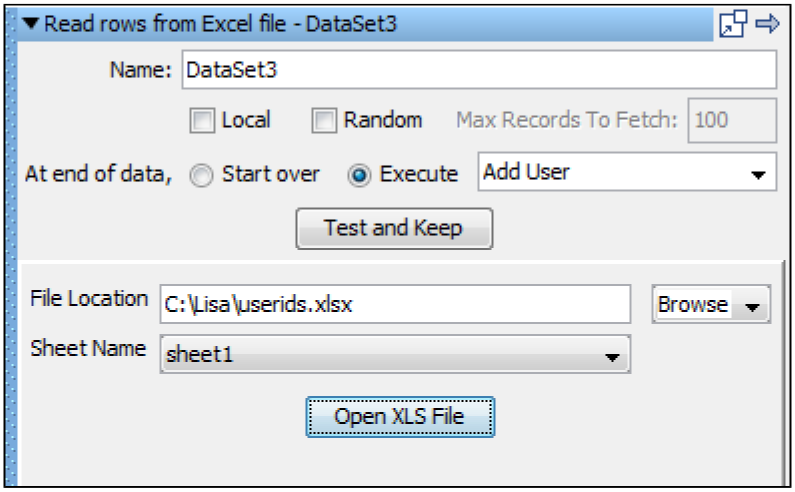
The **Read Rows from Excel File** data set assigns values to properties based on the contents of an Excel spreadsheet.

The first non-blank row of the Excel spreadsheet specifies the names of the properties to which the data values will be assigned. Subsequent rows list the data values to be used for these properties. The first full row of empty cells is treated as the end of data.

For example, you might want to test a set of first name, last name, user ID and password combinations.



	A	B	C	D	E	F
1						
2		firstName	lastName	id	password	
3		Sara	Bellum	sbellum	sbpwd	
4		Warren	Piece	wpiece	wppwd	
5		Amanda	Reckonwith	areckonwith	arpwd	
6						



Name: DataSet3

☐ Local ☐ Random Max Records To Fetch: 100

At end of data, ○ Start over ● Execute Add User

File Location C:\Lisa\userids.xlsx

Sheet Name sheet1

1. In the Data Set Editor enter the following parameters:

- **Name**: Enter the name of the data set.
- **At End Of Data**: Select whether you want to start over and read values from the start of the data set, or execute the step you select in the pull-down menu.
- **Local**: Select whether this should act as a global or local data set. Global is the default. Local data sets are created one-per-simulator; global data sets are created once and shared by all simulators.
- **File Location**: Enter the full path name of the Excel file, or browse to file with the browse button. You can use a property in the path name (for instance `LISA_HOME`).
- **Sheet Name**: Enter the name of the sheet in the Excel spreadsheet.

2. Click the Test and Keep button to test and load the data. You will get a message that confirms that the data set can be read, and shows the first set of data.

3. Select the steps that will use the data set.

The previous example shows a data set named **DataSet3** that will be used as many times as required. Data is loaded from Excel file.

You can click the Open XLS File button to open and edit the Excel spreadsheet.

> ⚠ As of LISA 6.0.5, Excel 2007 spreadsheets are supported for Excel File data sets and Excel DTO data sets. Excel DTO data sets are still created using the XLS format.

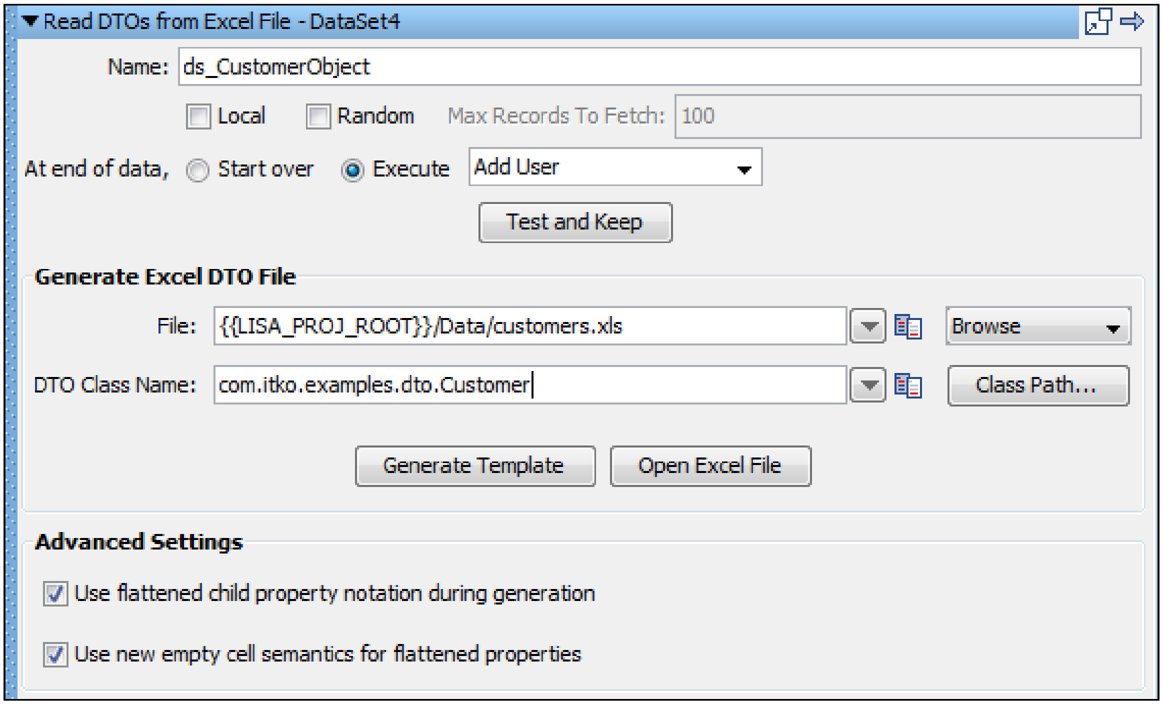
Read DTOs from Excel File Data Set

The **Read DTOs from Excel File** data set lets you parameterize Java data transfer objects (DTOs) in your test steps and gives you an easy way, through Excel spreadsheets, to provide data values for those parameters.

The Read DTO's from Excel file data set assigns values to the properties of a DTO and stores the object in a LISA property. This property can then be used whenever the DTO is required as a parameter. The data in the data set can be simple data types like numbers or strings, or complex data types such as DTOs, arrays and collections. The data represented in Excel will be converted into the proper data types automatically when needed. The only complex part of using this data set is the initial creation of the Excel spreadsheet. Fortunately, this is done for you. Given the package name of the DTO, a template is created, using one or more Excel sheets, that represent the object. Data types such as primitives, strings, arrays of primitives and simple individual DTOs can be represented on a single sheet. More complex data types, such as arrays of objects, require additional Excel sheets to represent the full DTO.

It is often the case that a web service endpoint expects complex DTOs. The Excel data set makes it very simple to create objects to use as parameters to the web service. When the web service is first referenced, it is given a name and a URL for the WSDL. Java DTO classes in the form **com.lisa.wsgen.SERVICENAME.OBJECTNAME** are automatically generated and made available on the classpath. You can browse to that generated class in the DTO class browser (see the following example), generate an Excel file and simply fill in the template.

Building the data set is a two step process. First, let LISA build the template in Excel, then open the Excel spreadsheet and fill in the data fields in all the sheets that are produced.



Name: ds_CustomerObject

□ Local □ Random Max Records To Fetch: 100

At end of data, ○ Start over ● Execute Add User

File: {{LISA_PROJ_ROOT}}/Data/customers.xls

DTO Class Name: com.itko.examples.dto.Customer

☑ Use flattened child property notation during generation

☑ Use new empty cell semantics for flattened properties

1. To build the template, enter the following in the Data Set Editor:

- **Name**: The name of the data set. This becomes the property used to store the current DTO object.
- **At End Of Data**: Choose whether you want to start over and read values from the start of the data set, or execute the step you select in the pull-down menu.
- **Local**: Whether this should act as a global or local data set. Global is the default. Local data sets are created one-per-simulator; global data sets are created once and shared by all simulators.
- **File**: The fully qualified path name, or browse to the Excel file using the browse pull-down menu.
- **DTO Class Name**: The full package name, or browse to, the DTO object. The class file must be on LISA's Test Manager. Your class can be copied to the hotDeploy directory to put it on the Test Manager.

2. Advanced Settings let you specify:

- **Use flattened child property notation during generation**: Override child property flattening during Excel DTO data set generation. If cleared, child properties will be generated as references with their own worksheets.
- **Use new empty cell semantics for flattened properties**: Empty cell semantics for flattened properties means how empty cells are interpreted in a DTO spreadsheets. For example, given the flattened properties { "prop1.subprop1", "prop1.subprop2" }, if both subprop1 and subprop2 have empty cell values, then under the new semantics the reference to "prop1" will be set to null. Under the old semantics, prop1 would be non-null, but references to subprop1 and subprop2 would both be null. New semantics should be used by default, especially by web services with WSDLs that use non-nillable types. If not used in the case of non-nillable types, the intermediate non-null references for containing properties that are automatically created when reading a DTO spreadsheet may result in the generation of invalid XML according to the schema, because cell values are empty.

3. Click the Generate Template button. The template will be built for you. In the system messages you will see a message that the file has been built.

4. Click the Open XLS File button.

5. The spreadsheet contains everything needed to construct the object. We will show how to add data in the next section. Close the XLS file.

6. Click the Test and Keep button to test and load the data. You will see a window that confirms that the data set can be read, and shows the first set of data.

Building the Excel spreadsheet

To facilitate this explanation, we will use an actual DTO object: **com.itko.example.dto.Customer**. This class is included with the LISA examples and can be found by browsing the Test Manager using the Browse button.

The Customer DTO has the following properties:

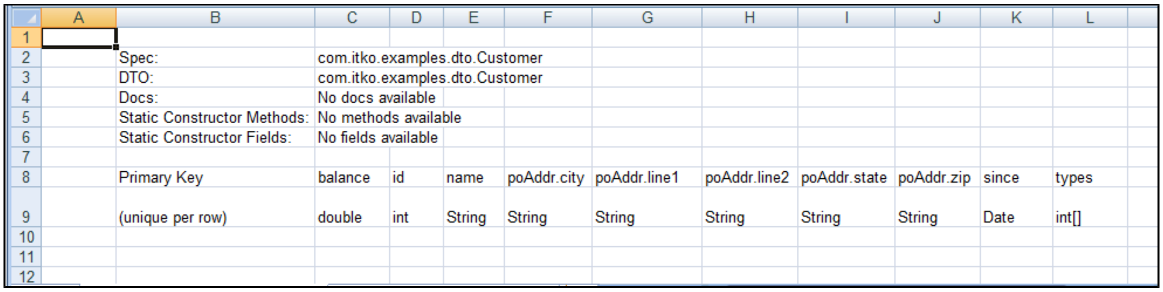
Property Name	**Type**
balance	Double
id	int
name	String
poAddr	Address
since	Date
types	int[]
locations	Address[]

The Address DTO has the following properties:

Property Name	**Type**
city	String
line1	String
line2	String
state	String
zip	String

The first six Customer DTO properties can appear on one Excel spreadsheet. However, the locations property, an array of Address objects, requires a second Excel spreadsheet.

Looking at the first spreadsheet, at the top, LISA lists the DTO spec (Customer) and the current DTO object (Customer). It would also list the Java doc location, if available. The actual data sheet appears as follows, with a row specifying property names, followed by a row specifying the data types. The first field (column) is not a DTO property, but a special field (Primary key), that holds a unique value for each row.

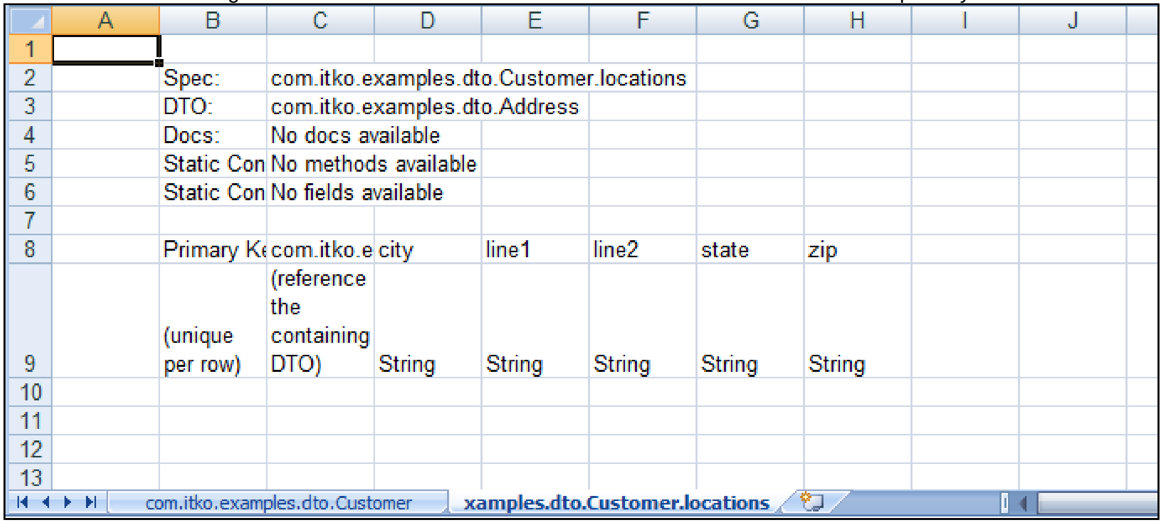


	A	B	C	D	E	F	G	H	I	J	K	L
1												
2		Spec:	com.itko.examples.dto.Customer									
3		DTO:	com.itko.examples.dto.Customer									
4		Docs:	No docs available									
5		Static Constructor Methods:	No methods available									
6		Static Constructor Fields:	No fields available									
7												
8		Primary Key	balance	id	name	poAddr.city	poAddr.line1	poAddr.line2	poAddr.state	poAddr.zip	since	types
9		(unique per row)	double	int	String	String	String	String	String	String	Date	int[]
10												
11												
12												

Looking at the data sheet we can see the following:

- Each row contains the data for a single Customer DTO.
- The poAddr property, of type Address, has been "flattened" and its properties are listed on this sheet. These properties are prefixed with poAddr and then the property name in the address object (i.e. poAddr.city).
- The since property, of type date, is prefilled with today's date. This is to show you the required format for the date mm/dd/yyyy. All dates must have this format in the Excel template.
- The types property, of type int[], is a single cell that can contain the array elements as a comma separated list. This is only possible for arrays of primitives or strings.

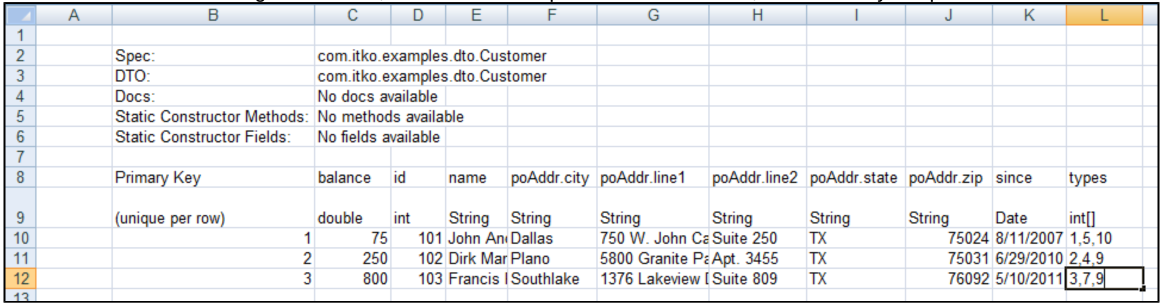
The location property, of type Address[], does not appear on this sheet. It is on the second sheet in the Excel file. This is because it is an array of objects. This sheet contains the data for an Address object in each row. There are two special fields in this sheet: "Primary Key", and the "reference the containing DTO" field that is used to link the rows in this sheet to the rows in the primary sheet.



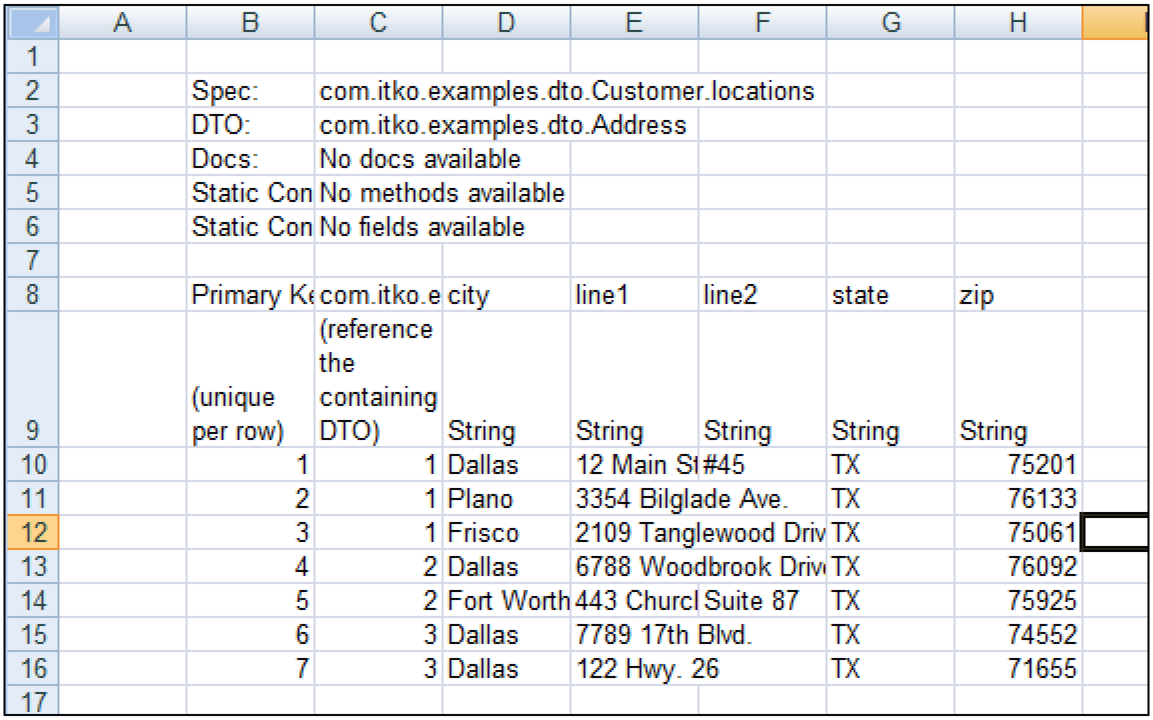
	A	B	C	D	E	F	G	H	I	J
1										
2		Spec:	com.itko.examples.dto.Customer.locations							
3		DTO:	com.itko.examples.dto.Address							
4		Docs:	No docs available							
5		Static Con	No methods available							
6		Static Con	No fields available							
7										
8		Primary Ke	com.itko.e	city	line1	line2	state	zip		
9		(unique per row)	(reference the containing DTO)	String	String	String	String	String		
10										
11										
12										
13										

com.itko.examples.dto.Customer
xamples.dto.Customer.locations

Because each Customer object can have several locations, several rows in the locations sheet belong to an object specified in a single row of the Customer sheet. This is manifested in the second sheet by listing the primary key of the parent Customer object in the "reference the containing DTO" field of each location that belongs to the Customer. You should see a similarity here to primary/foreign key relationships in databases. An examination of the following illustrations, which have the spreadsheets filled out, should clarify the procedure.

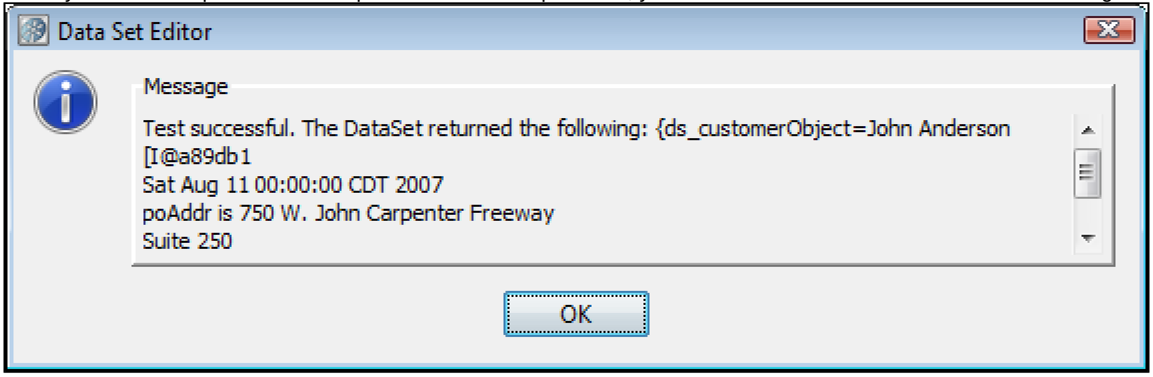


	A	B	C	D	E	F	G	H	I	J	K	L
1												
2		Spec:	com.itko.examples.dto.Customer									
3		DTO:	com.itko.examples.dto.Customer									
4		Docs:	No docs available									
5		Static Constructor Methods:	No methods available									
6		Static Constructor Fields:	No fields available									
7												
8		Primary Key	balance	id	name	poAddr.city	poAddr.line1	poAddr.line2	poAddr.state	poAddr.zip	since	types
9		(unique per row)	double	int	String	String	String	String	String	String	Date	int[]
10		1	75	101	John An	Dallas	750 W. John Ca	Suite 250	TX	75024	8/11/2007	1,5,10
11		2	250	102	Dirk Mar	Plano	5800 Granite Pa	Apt. 3455	TX	75031	6/29/2010	2,4,9
12		3	800	103	Francis I	Southlake	1376 Lakeview [	Suite 809	TX	76092	5/10/2011	3,7,9



	A	B	C	D	E	F	G	H
1								
2		Spec:	com.itko.examples.dto.Customer.locations					
3		DTO:	com.itko.examples.dto.Address					
4		Docs:	No docs available					
5		Static Con	No methods available					
6		Static Con	No fields available					
7								
8		Primary Ke	com.itko.e	city	line1	line2	state	zip
9		(unique per row)	(reference the containing DTO)	String	String	String	String	String
10		1	1	Dallas	12 Main St	#45	TX	75201
11		2	1	Plano	3354 Bilglade Ave.		TX	76133
12		3	1	Frisco	2109 Tanglewood Driv		TX	75061
13		4	2	Dallas	6788 Woodbrook Driv		TX	76092
14		5	2	Fort Worth	443 Churcl	Suite 87	TX	75925
15		6	3	Dallas	7789 17th Blvd.		TX	74552
16		7	3	Dallas	122 Hwy. 26		TX	71655
17								

When you save the spreadsheet and press Test and Keep in LISA, you will see the first Customer DTO in the message.



Message

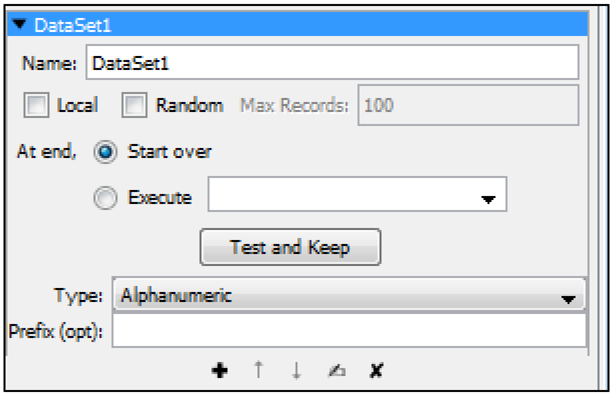
Test successful. The DataSet returned the following: {ds_customerObject=John Anderson
[I@a89db1
Sat Aug 11 00:00:00 CDT 2007
poAddr is 750 W. John Carpenter Freeway
Suite 250

Depending on the complexity of your DTO, you could have several more Excel sheets in the Excel workbook. However, the process is the same as the previous example.

To see how you might use this Customer DTO as a property, see the sections on testing Java objects in Test Steps.

Unique Code Generator Data Set

The **Unique Code Generator** data set provides a unique token (or code) each time it is called. The token can be numeric or alphanumeric, and can have a user-defined prefix pre-pended to it. This is commonly used in testing to create new users, accounts, and so on, to help ensure they do not use a value that is already inside a system.



DataSet1

Name: DataSet1

□ Local □ Random Max Records: 100

At end, ● Start over

○ Execute

Type: Alphanumeric

Prefix (opt):

In the Data Set Editor, enter the following:

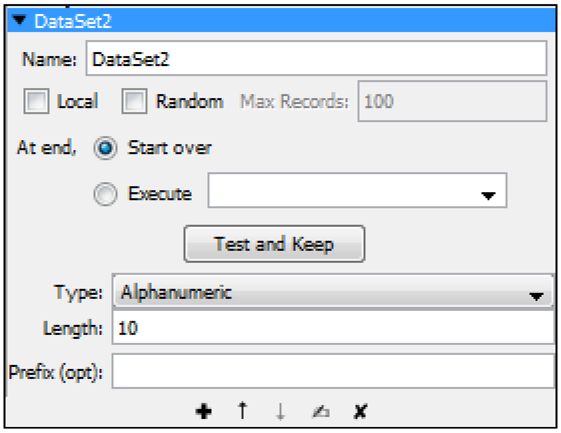
- **Name**: The property that is assigned the value. This is also the name of the data set.
- **Local**: Whether this should act as a global or local data set. Global is the default. Local data sets are created one-per-simulator; global data sets are created once and shared by all simulators.
- **At end**: Select whether you want to start over and read values from the start of the data set, or execute the step you select in the pull-down menu.
- **Type**: The type of token to return. Choices are number or alphanumeric.
- **Prefix (opt)**: Prefix to be pre-pended (optional)

Click the Test and Keep button to test and load the data. You will get a dialog that confirms that the data set can be read, and shows the first set of data.

Because this data set will always return a token, the **At End of Data** section of the Data Set Editor has no meaning for this data set type.

Random Code Generator Data Set

The **Random Code Generator** data set generates numeric or alphanumeric data randomly for use in a test case. This data set is similar to the Unique Code Generator Data Set data set, but it lets you set a length for the result. This data set can be used to create a particular type of unique value such as a telephone number, Social Security number, and so on.



DataSet2

Name: DataSet2

□ Local □ Random Max Records: 100

At end, ● Start over

○ Execute

Type: Alphanumeric

Length: 10

Prefix (opt):

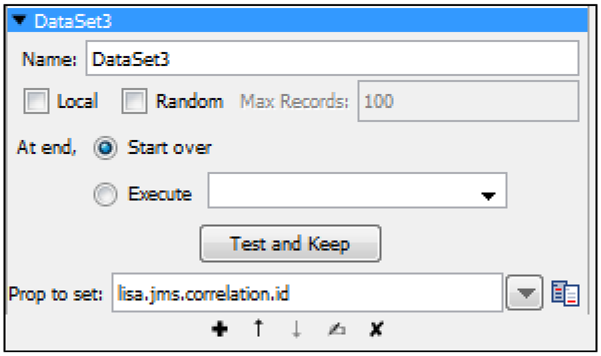
In the Data Set Editor, enter the following:

- **Name**: The name of the data set.
- **Local:** Whether this should act as a global or local data set. Global is the default. Local data sets are created one-per-simulator; global data sets are created once and shared by all simulators.
- **Random:** Whether the record after the current record (sequential access) will be read, or a random record will be read. Sequential reading is the default.
- **Max Records:** The upper bound on the number of records to fetch for random access. This text field is disabled if the Random check box is not selected.
- **At end**: Select whether you want to start over and read values from the start of the data set, or execute the step you select in the pull-down menu.
- **Prefix (opt)**: Adds prefix to the generated data. This field is optional.
- **Type**: This field allows either alphanumeric or number type of data set to be generated.
- **Length**: This field restricts the length of the random data generated to the value set here.

Click Test and Keep to test and load the data. You will get a message that confirms that the data set can be read, and shows the first set of data.

Message_Correlation ID Generator Data Set

The **Message/Correlation ID Generator** data set is a specialized unique code generator useful for messaging. It generates a 24 byte unique code - designed specifically for IBM MQ Series correlation ID, but can also be used for any JMS provider. This data set creates or updates two special LISA properties: **lisa.jms.correlation.id** and **lisa.mq.correlation.id**. The messaging steps recognize these properties and set the message correlation ID appropriately.



Name: DataSet3

□ Local □ Random Max Records: 100

At end, ● Start over

○ Execute

Prop to set: lisa.jms.correlation.id

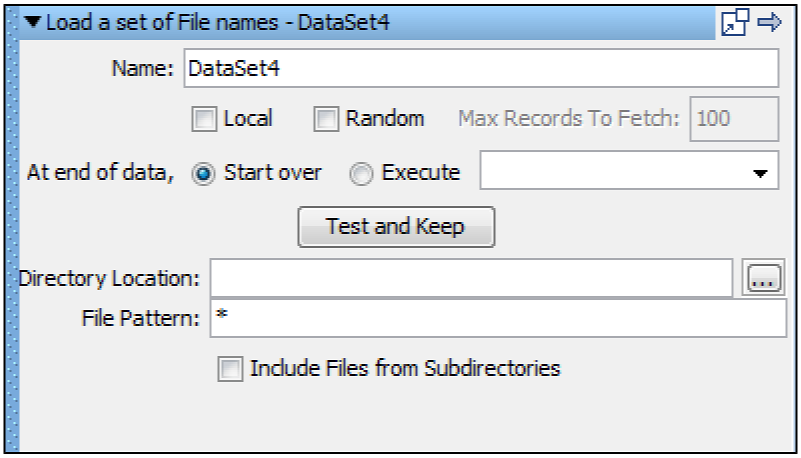
In the Data Set Editor enter the following:

- **Name**: The name of the data set.
- **Local:** Whether this should act as a global or local data set. Global is the default. Local data sets are created one-per-simulator; global data sets are created once and shared by all simulators.
- **Random:** Whether the record after the current record (sequential access) will be read, or a random record will be read. Sequential reading is the default.
- **Max Records:** The upper bound on the number of records to fetch for random access. This text field is disabled if the **Random** check box is not selected.
- **At End**: Select whether you want to start over and read values from the start of the data set, or execute the step you select in the pull-down menu.
- **Prop to set**: LISA property to set. The default is **lisa.jms.correlation.id.**

Load a Set of File Names Data Set

The **Load a Set of File Names** data set assigns a value to a property based on a filtered set of file names from the file system.

The set of file names can include all the files in a given directory, or a set filtered by a "file pattern". There is also an option to recursively include subdirectories in the set. The files are returned in a case-sensitive, alphabetical order, top directories first, followed by subdirectories (using depth first ordering). Each time the data set is used, it returns the next file name (full path name) in the data set. This file name is stored in a property whose name is the same as that of the data set.



Name: DataSet4

□ Local □ Random Max Records To Fetch: 100

At end of data, ● Start over ○ Execute

Directory Location:

File Pattern: *

□ Include Files from Subdirectories

In the Data Set Editor enter the following parameters:

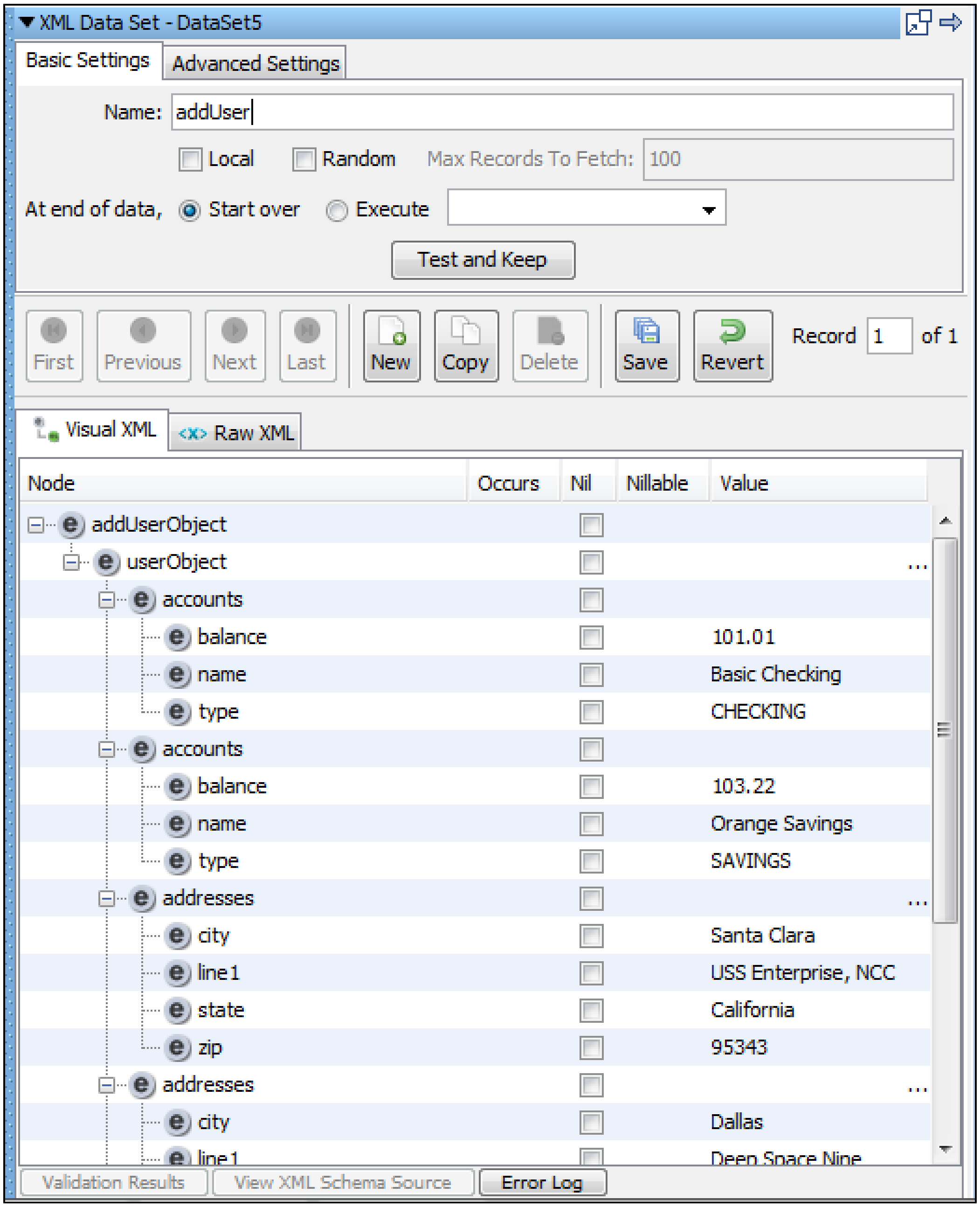
- **Name**: The name of the data set.
- **At End Of Data**: Select whether you want to start over and read values from the start of the data set, or execute the step you select in the pull-down menu.
- **Local**: Whether this should act as a global or local data set. Global is the default. Local data sets are created one-per-simulator; global data sets are created once and shared by all simulators.
- **Directory Location**: The full path name of the directory to scan, or you can browse using the Browse button.
- **File Pattern**: A filter pattern string, using an asterisk (*) as a wild card if wanted.
- **Include Files from Subdirectories**: Check if files in subdirectories are to be listed.

Click Test and Keep to test and load the data. You will get a dialog that confirms that the data set can be read, and shows the first set of data.

⚠ A large amount of data may take longer than you want; use the Cancel button to stop the operation.

XML Data Set

Like other data sets in LISA, the **XML data set** allows user-specified content to be added to distinct records. However, the XML data set specializes in handling XML content.



Name: addUser

☐ Local ☐ Random Max Records To Fetch: 100

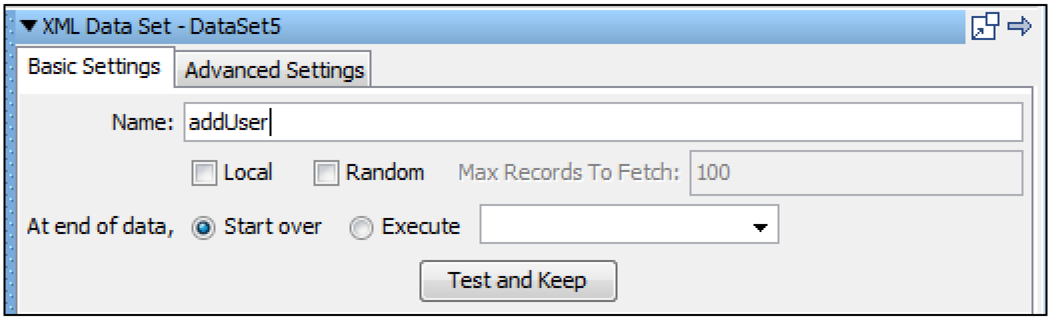
At end of data, ● Start over ○ Execute

Record 1 of 1

Node	Occurs	Nil	Nillable	Value
addUserObject		☐		
userObject		☐		...
accounts		☐		
balance		☐		101.01
name		☐		Basic Checking
type		☐		CHECKING
accounts		☐		
balance		☐		103.22
name		☐		Orange Savings
type		☐		SAVINGS
addresses		☐		...
city		☐		Santa Clara
line1		☐		USS Enterprise, NCC
state		☐		California
zip		☐		95343
addresses		☐		...
city		☐		Dallas
line1		☐		Deep Space Nine

At design-time, the first record of an XML Data Set populates the value for a property in LISA test state by clicking the Test and Keep button. The LISA property name is the same as the name of the data set. For example, if the data set is named "`DataSet1`", then the property populated in the test case is "{{`DataSet1`}}". At run-time, all of the records can be accessed sequentially to create a data-driven test case.

Basic Settings Tab



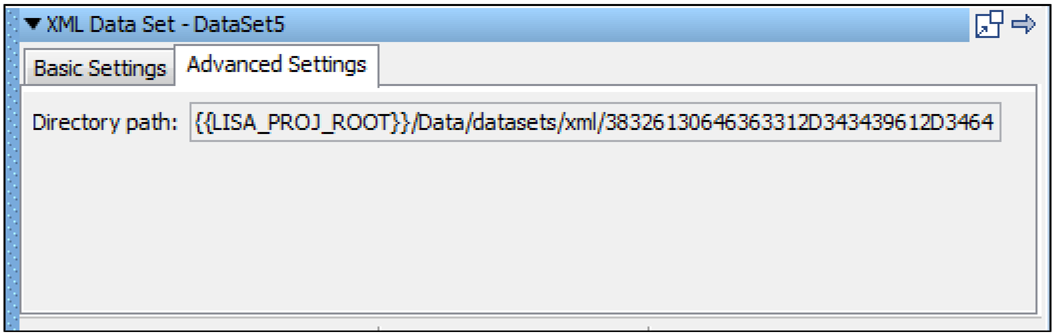
Name: addUser

□ Local □ Random Max Records To Fetch: 100

At end of data, ● Start over ○ Execute

- **Name:** The name of this data set. This value will be used as the name of the property in test step. For example, an XML Data Set named **`DataSet1`** will populate the property {{`DataSet1`}}.
- **Local:** Whether this should act as a global or local data set. Global is the default. Local data sets are created one-per-simulator; global data sets are created once and shared by all simulators.
- **Random:** Whether the record after the current record (sequential access) will be read, or a random record will be read. Sequential reading is the default.
- **Max Records to Fetch:** The upper bound on the number of records to fetch for random access. This text field is disabled if the Random check box is not selected.
- **At end of data, Start over:** After the last record is read, loop back to the very first record. This is the default behavior.
- **At end of data, Execute:** After the last record is read, branch to the specified test step.
- **Test and Keep:** At design-time, populate the LISA property associated with the data set in test state with the value of the first record.

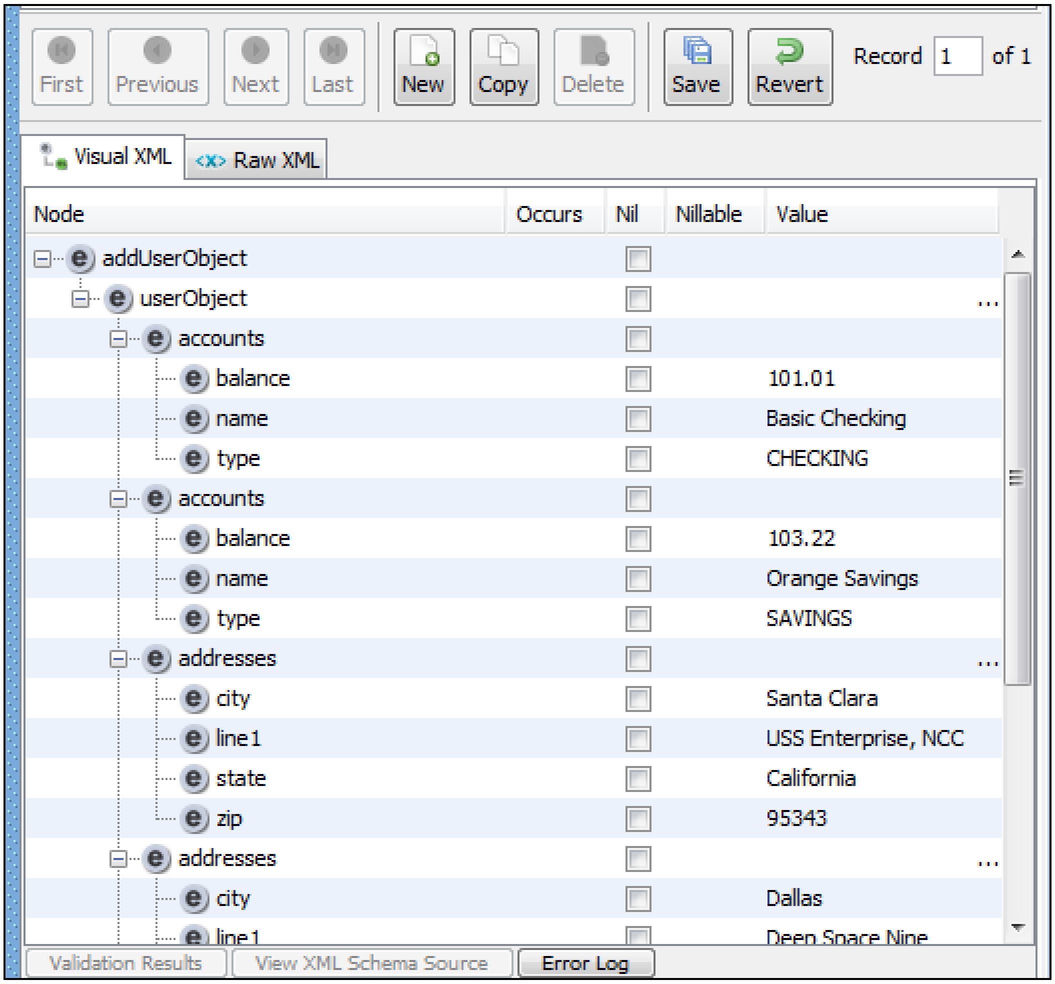
Advanced Settings Tab



Directory path: {{LISA_PROJ_ROOT}}/Data/datasets/xml/38326130646363312D343439612D3464

- **Directory path:** This read-only value represents the directory on the file system where records are saved. There is a one-to-one mapping between a record and a file in the directory path. If an XML data set is created within a project, the directory path will start with "{{ `LISA_PROJ_ROOT`}}`/Data/datasets/xml/`". If no project is used, the directory path will start with "{{`LISA_HOME`}}`/datasets/xml/`".

Record Editing Panel



Record 1 of 1

Node	Occurs	Nil	Nillable	Value
addUserObject		□		
userObject		□		...
accounts		□		
balance		□		101.01
name		□		Basic Checking
type		□		CHECKING
accounts		□		
balance		□		103.22
name		□		Orange Savings
type		□		SAVINGS
addresses		□		...
city		□		Santa Clara
line1		□		USS Enterprise, NCC
state		□		California
zip		□		95343
addresses		□		...
city		□		Dallas
line1		□		Deep Space Nine

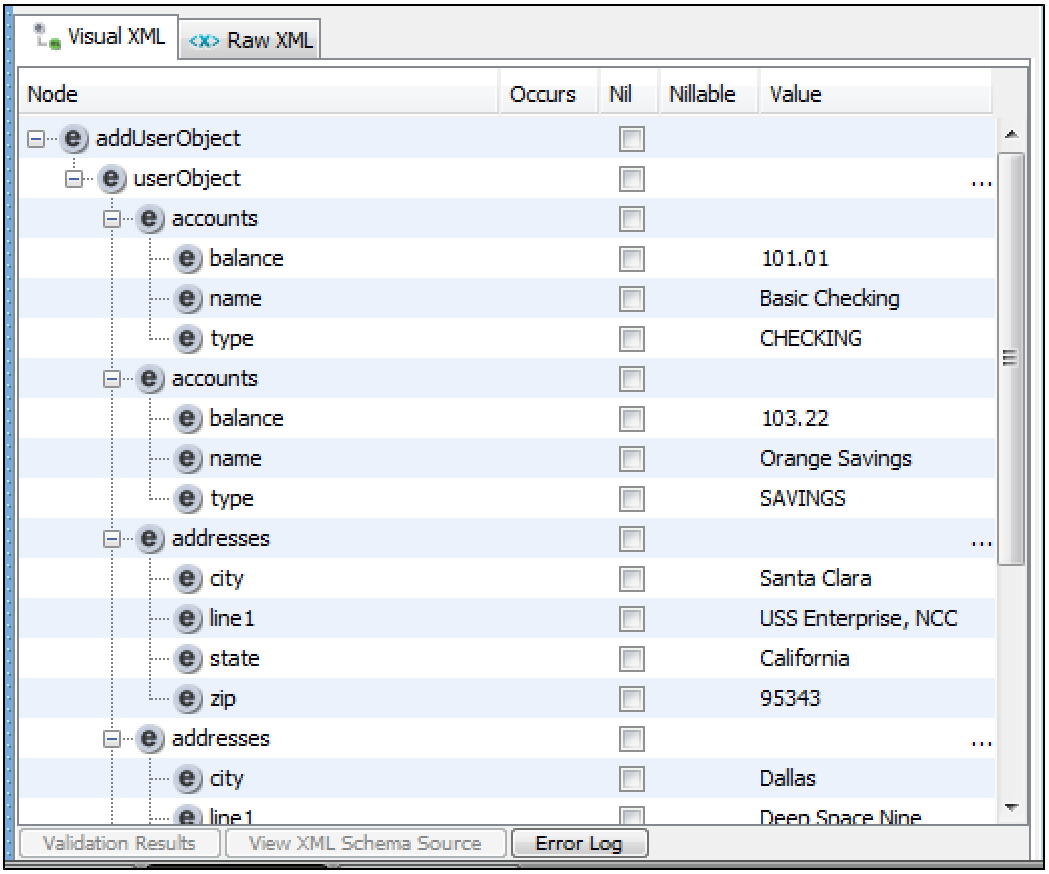
Action Buttons

- **First:** Move to the first record in the XML data set.
- **Previous:** Move to the previous record.
- **Next:** Move to the next record.
- **Last:** Move to the last record.
- **New:** Create a new record.
- **Copy:** Create a copy of the current record at the end of the data set and move to it.
- **Delete:** Delete the current record.
- **Save:** Save all pending changes.
- **Revert:** Revert all pending changes.

Record Number Selector

- **Record X of X:** Enter the record number to jump to a specific record.

Visual XML Tab

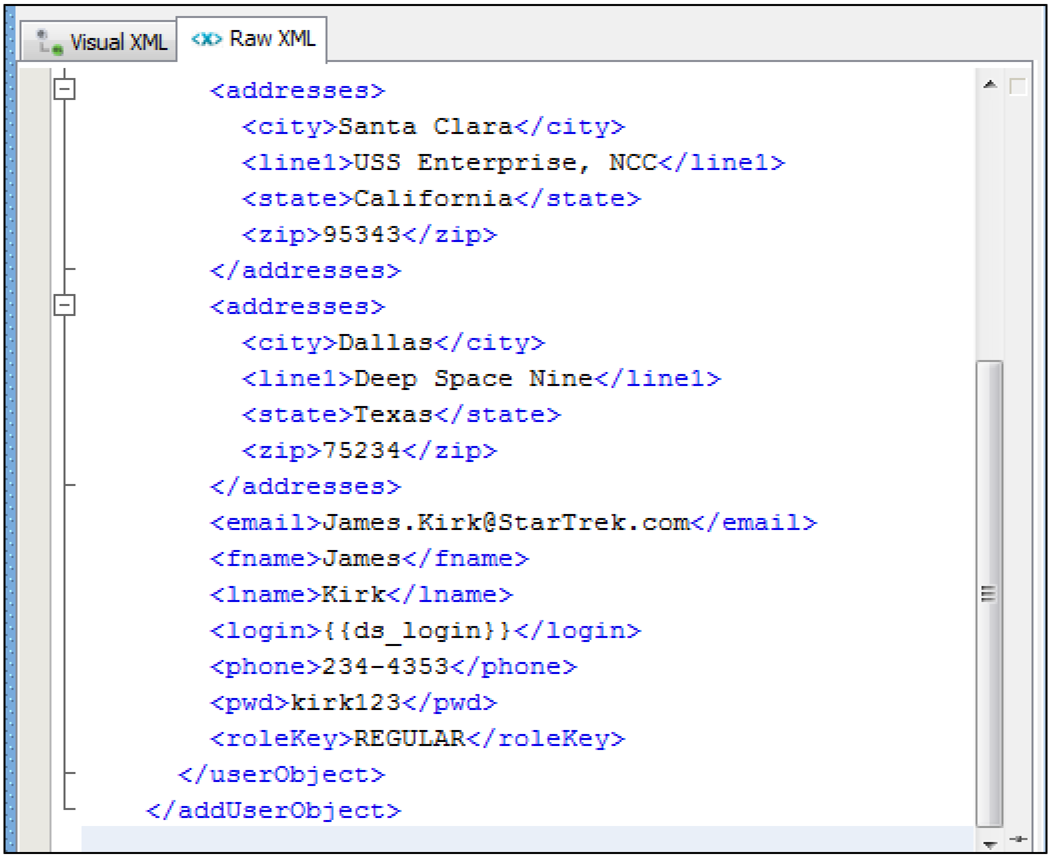


Node	Occurs	Nil	Nillable	Value
addUserObject		□		
userObject		□		...
accounts		□		
balance		□		101.01
name		□		Basic Checking
type		□		CHECKING
accounts		□		
balance		□		103.22
name		□		Orange Savings
type		□		SAVINGS
addresses		□		...
city		□		Santa Clara
line1		□		USS Enterprise, NCC
state		□		California
zip		□		95343
addresses		□		...
city		□		Dallas
line1		□		Deep Space Nine

Visual XML Editor

Moving the mouse over the Node and Value columns will display a tooltip. This can help if the value in the table is too long to be fully displayed.

Raw XML Tab



```
        <addresses>
          <city>Santa Clara</city>
          <line1>USS Enterprise, NCC</line1>
          <state>California</state>
          <zip>95343</zip>
        </addresses>
        <addresses>
          <city>Dallas</city>
          <line1>Deep Space Nine</line1>
          <state>Texas</state>
          <zip>75234</zip>
        </addresses>
        <email>James.Kirk@StarTrek.com</email>
        <fname>James</fname>
        <lname>Kirk</lname>
        <login>{{ds_login}}</login>
        <phone>234-4353</phone>
        <pwd>kirk123</pwd>
        <roleKey>REGULAR</roleKey>
      </userObject>
    </addUserObject>
```

This tab contains a raw text view of the XML contained in a record.

Double-clicking the left border will toggle the visibility of a top toolbar, line numbers bar, and bottom editing info bar in the editor.

Companions

A **companion** is a LISA element that runs before and/or after every test case execution. Companions are used to configure behavior that is global to the test case. These behaviors include simulating browser bandwidth and browser type, setting synchronization points in load tests and reading properties from an external file. In a way, companions set some kind of a context for the test case execution.

Companions work as helpers for the test case, before any of the steps are executed.

The following topics are available in this section.

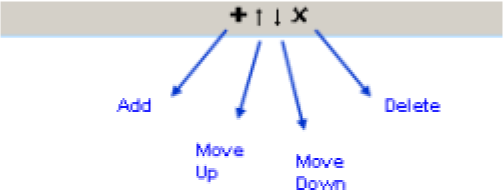
Adding a Companion

To add a companion

1. Open a test case and click the Companion element in the right panel.
2. Click the Add icon ✚. This will open the Companion main menu.
3. Each companion has a different editor. Select the required companion to open the appropriate editor. Each companion type, with all its parameters, is described in detail in Types of Companions.

Companion Toolbar

At the bottom of every element, there is a toolbar that has icons to add/delete/reorder.



Deleting a Companion

To delete a companion

1. Select the companion from the Companion tab and click the Delete ✘ icon in the toolbar. Alternately, you can right-click the companion and select Delete.
   A delete confirmation box appears.
2. Click Yes.

Reordering a Companion

To reorder a companion

- Open the Companion tab and select the companion to be moved.
- Click the Move up ↑ or Down ↓ arrows in the toolbar to set the selected companion's position.

Types of Companions

This section describes each of the companions that are available.

The following types of companions are described in this section:

Common Companions

Web Browser Simulation Companion
Browser Bandwidth Simulation Companion

HTTP Connection Pool Companion
Configure LISA to Use a Web Proxy Companion
Set Up a Synchronization Point Companion
Set Up an Aggregate Step Companion

VSE Companions

Java Protocol Companion

Observed System VSE Companion

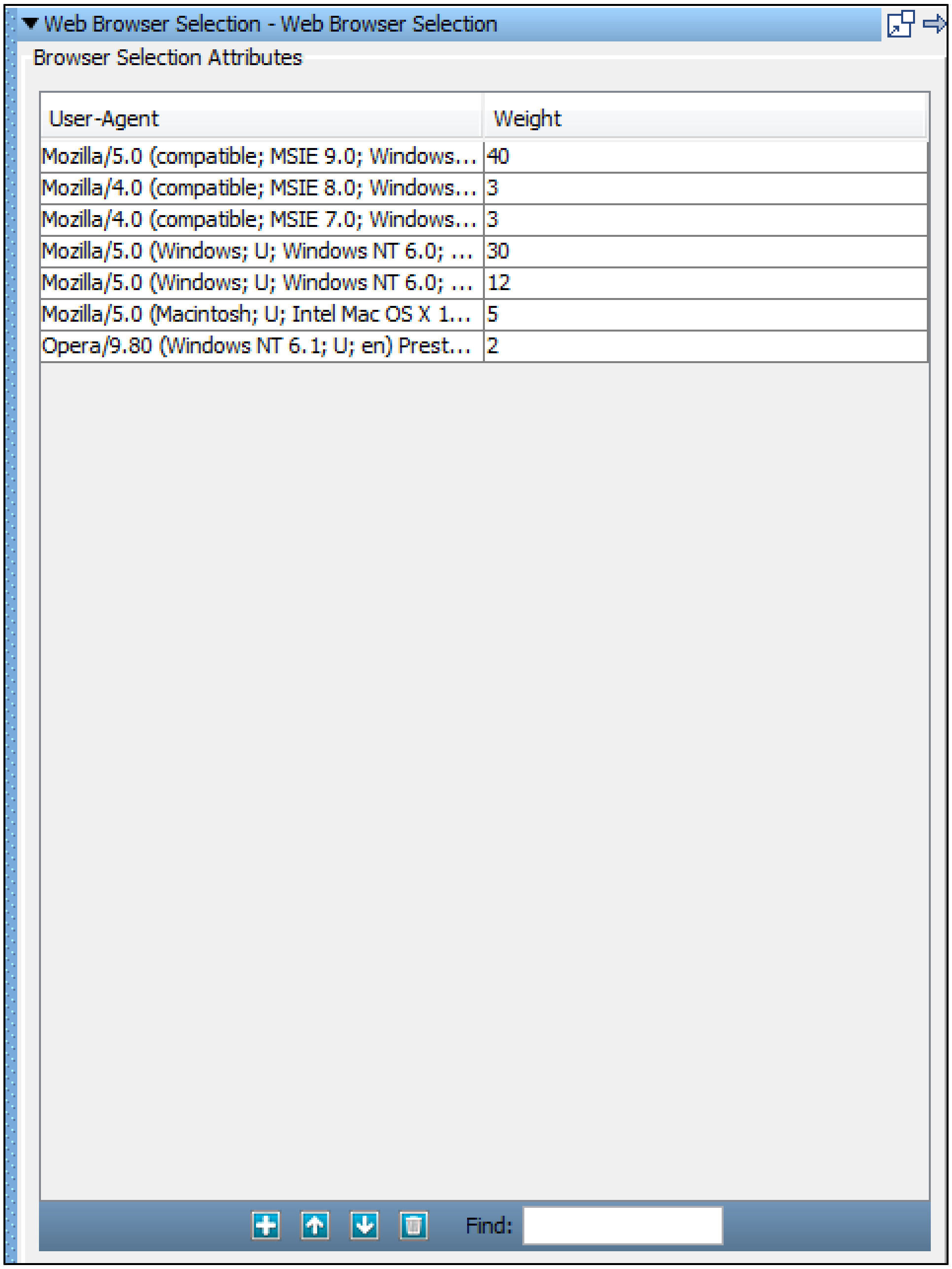
VSE Think Scale Companion

Other Companions
Create a Sandbox Class Loader for Each Step Companion
Set Final Step to Execute Companion
Negative Testing Companion
Fail Test Case Companion
XML Diff Ignored Nodes Companion

Web Browser Simulation Companion

The **Web Browser Simulation** companion lets you simulate a variety of web browsers. Web browsers identify themselves to a web server using the User-Agent HTTP header. You configure LISA to simulate several user-agents when you are running several virtual users in a staged test. Each user-agent string is assigned a relative weight, allowing one browser to appear more often than others.

Use the default Browser Selection Companion Editor to specify the weights.



Browser Selection Attributes

User-Agent	Weight
Mozilla/5.0 (compatible; MSIE 9.0; Windows…	40
Mozilla/4.0 (compatible; MSIE 8.0; Windows…	3
Mozilla/4.0 (compatible; MSIE 7.0; Windows…	3
Mozilla/5.0 (Windows; U; Windows NT 6.0; …	30
Mozilla/5.0 (Windows; U; Windows NT 6.0; …	12
Mozilla/5.0 (Macintosh; U; Intel Mac OS X 1…	5
Opera/9.80 (Windows NT 6.1; U; en) Prest…	2

To configure the Web Browser Selection companion, enter or edit the list of browser agents and the weights:

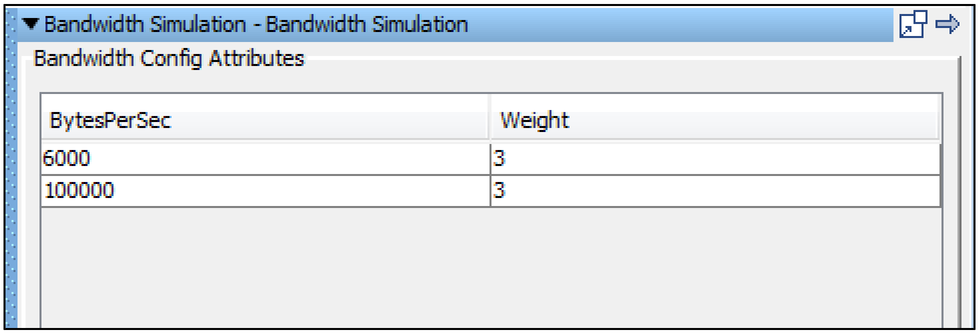
- **User-Agent**: The browser to simulate.
- **Weight**: The weight for this browser. For example, to assign weights of 25%, 25% and 50% for three browsers, enter weights of 1, 1, and 2 for the three rows, and zero for the others (or delete the extra rows).

To add an additional user-agent, use the Add ➕ button.

To delete a line, use the Delete 🗑 button.

Browser Bandwidth Simulation Companion

The **Browser Bandwidth Simulation** companion lets you simulate varied bandwidths for the virtual users. Some testing scenarios call for the simulations of different types of internet connections.



Bandwidth Config Attributes

BytesPerSec	Weight
6000	3
100000	3

To configure the Browser Bandwidth Simulation companion

Enter the following parameters in the Browser Bandwidth Simulation Attributes editor:

- **BytesPerSec**: The connection speed. For example, to simulate a connection speed of 56K, enter a BytesPerSec of 7000 (56000 bits / 8 bits per byte = 7000 bytes per second).
- **Weight**: The weight given to this row. For example, to assign weights of 25%, 25% and 50% to three rows, enter values of 1, 1 and 2 on the **Weight** column of the three rows.

To add a line, use the Add button.

To delete a line, use the Delete button.

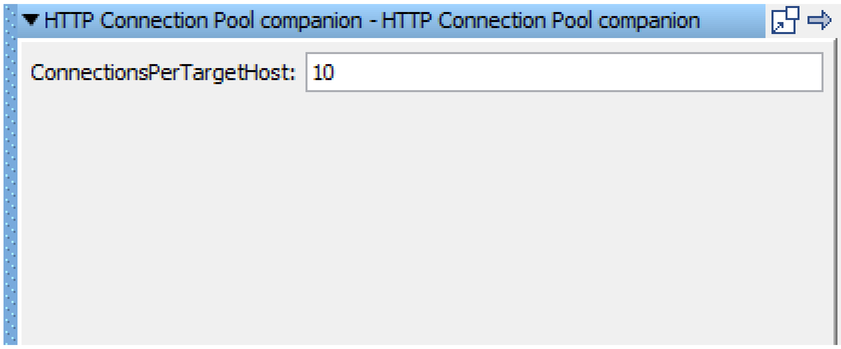
HTTP Connection Pool Companion

The **HTTP Connection Pool** companion enables you to limit the number of HTTP connections per target server. This companion applies only to the HTTP/HTML Request, REST, and Raw SOAP Request test steps.

LISA normally uses one HTTP connection for each virtual user. For example, if you run a test with 100 virtual users, then you would have 100 sockets open on the client and 100 sockets open on the server.

If you run a load test with thousands of virtual users per simulator, the underlying operating system might run out of available sockets. In this scenario, consider using the HTTP Connection Pool companion.

The following image shows the editor for this companion.


HTTP Connection Pool companion - HTTP Connection Pool companion

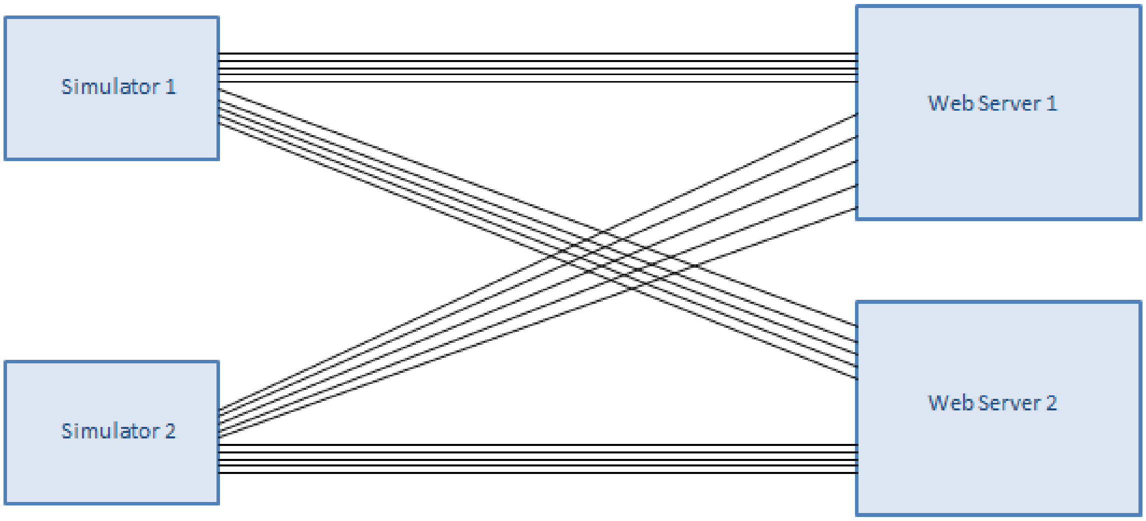
ConnectionsPerTargetHost: 10

The **ConnectionsPerTargetHost** parameter specifies the number of connections to allocate to each unique endpoint.

Assume that a test case has two steps: an http step to hit web server 1, and a second http step to hit web server 2. The test case has the HTTP Connection Pool companion with a setting of 5 connections per target host. The staging document is configured to run 100 virtual users. There are two simulator servers, so by default they will get 50 virtual users each.

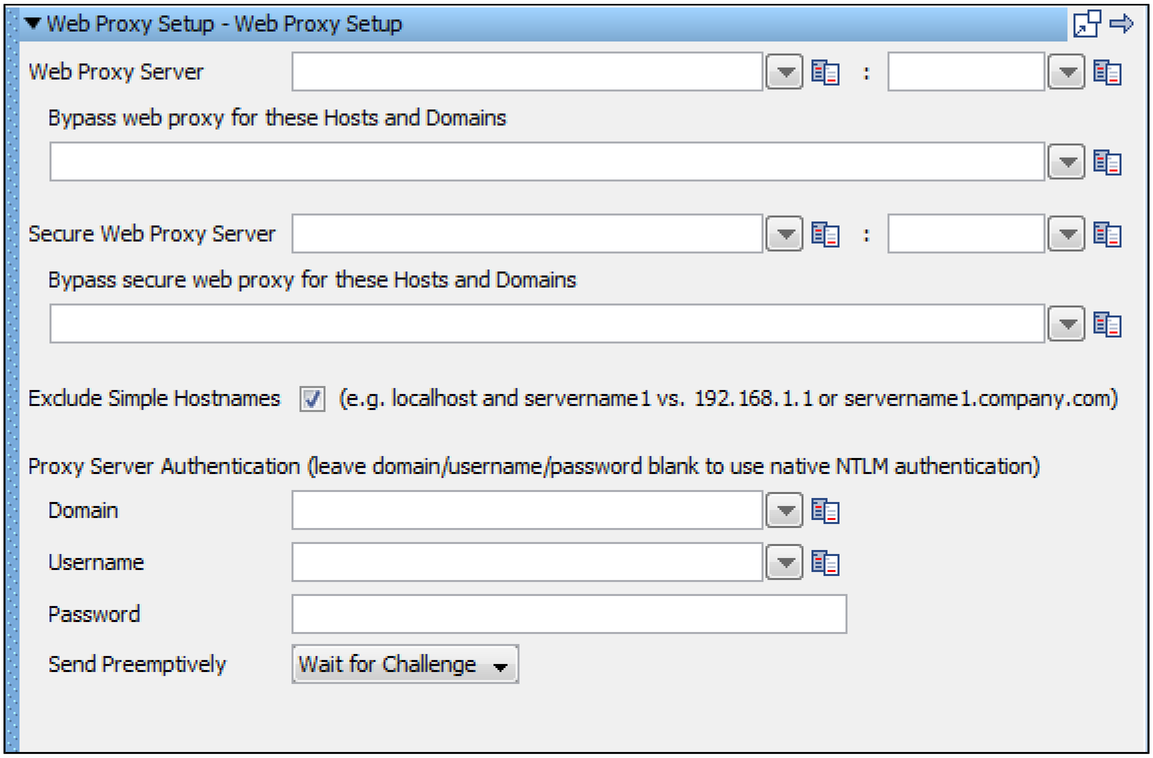
Simulator 1 creates five connections to web server 1, and five connections to web server 2. Simulator 2 does the same thing. Each web server now has 10 client connections. When a virtual user gets to the first http step, it must wait for one of the 5 connections to web server 1 to become available. The virtual user uses the connection to make the HTTP call, and the connection goes back into the pool.

The following diagram illustrates this scenario. Simulator 1 has five connections to web server 1 and five connections to web server 2. Simulator 2 has five connections to web server 1 and five connections to web server 2.



Configure LISA to Use a Web Proxy Companion

The **Configure LISA to use a Web Proxy** companion lets you set up a proxy for all web testing steps. If there are times when your environment dictates the use of a proxy, use this companion. Proxy information is specific to your organization. Consult your operations team for your company's proxy settings.



Web Proxy Setup - Web Proxy Setup

Web Proxy Server :

Bypass web proxy for these Hosts and Domains

Secure Web Proxy Server :

Bypass secure web proxy for these Hosts and Domains

Exclude Simple Hostnames ☑ (e.g. localhost and servername1 vs. 192.168.1.1 or servername1.company.com)

Proxy Server Authentication (leave domain/username/password blank to use native NTLM authentication)

Domain

Username

Password

Send Preemptively: Wait for Challenge

To configure the Web Proxy Setup companion enter the following parameters in the Web Proxy companion editor.

- **Web Proxy Server (Host & Port)**: The name or IP address of the proxy server in the first field, and the port number in the second field.
- **Bypass web proxy for these Hosts & Domains**: Names of the domains/hosts for which you want proxy to be bypassed.
- **Secure Web Proxy Server (SSL Proxy Host & Port)**: The name or IP address of the SSL proxy server in the first field, and the port number in the second field.
- **Bypass secure web proxy for these Hosts & Domains**: Names of the domains/hosts for which you want secure proxy to be bypassed.
- **Exclude Simple Hostnames**: Select to exclude hostnames like **localhost** or **servername.company.com**.
- **Proxy Server Authentication**: Domain name with username and password if required for authenticating to proxy server.

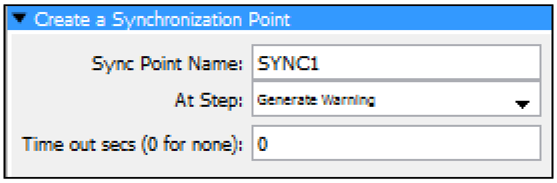
LISA can also use its **local.properties** file to assign a web proxy for all test cases. This file is in the LISA home directory. Update the **lisa.http.webProxy.host** and **lisa.http.webProxy.port** properties as appropriate and restart LISA. If you do not already have a **local.properties** file in the LISA home directory, rename the existing **_local.properties** to **local.properties**, and use it.

Set Up a Synchronization Point Companion

The **Create a Synchronization Point** companion lets you select a test step that will be used as a synchronization point in a test case or a test suite. At a synchronization point, virtual users will pause and wait until each one has reached this step. Then all virtual users will be released to execute the step at the same time. This is useful when setting up load tests for concurrent testing or peak resource utilization.

For example, you could set up a 100-user test to have all users log in to your application and then all order the same seat in a theater session at the same time.

Synchronization points apply to a single test or you can apply them to all tests within a test suite. In test suites, the Synchronization point name must be the same across the suite, but the At Step can be different. Multiple test scenarios (and test suites) must be set to run in parallel, because serial tests will not be able to hit the Synchronization point at the same time by definition. Multiple synchronization points can be defined in a test case or test suite.



Create a Synchronization Point

Sync Point Name: SYNC1

At Step: Generate Warning

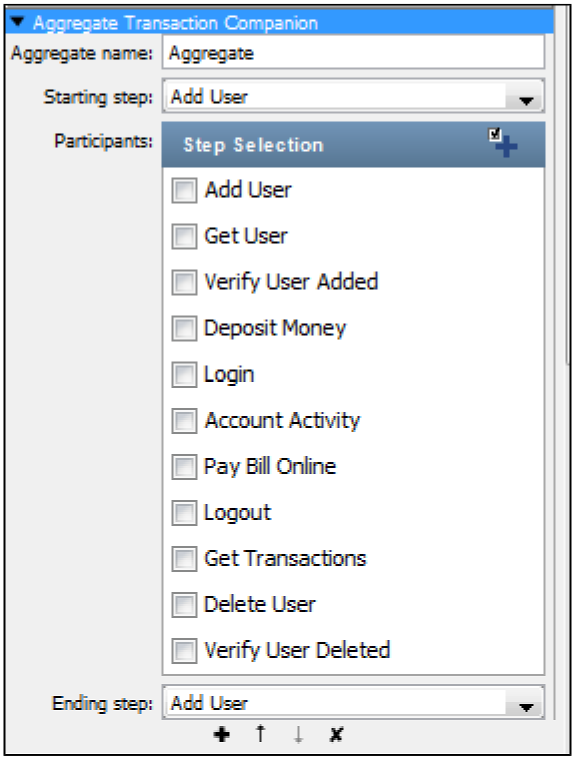
Time out secs (0 for none): 0

To configure the **Create a Synchronization Point** companion, enter the following parameters:

- **Sync Point Name**: The name you specify for the synchronization point.
- **At Step**: Select the step for the synchronization point from the drop-down list. Virtual users will pause before executing this step.
- **Time out secs (0 for none)**: The number of seconds to wait for synchronization to occur. All virtual users must reach the At Step before the time out period elapses.

Set Up an Aggregate Step Companion

The **Set Up an Aggregate Step** companion lets you aggregate and report several physical test steps as one logical step for metrics collections and reporting purposes. Unless you select the Quiet check box on a step, LISA also collects metrics and report events for the individual steps in an aggregate. You can set multiple aggregation points.



Aggregate Transaction Companion

Aggregate name: Aggregate

Starting step: Add User

Participants: Step Selection

□ Add User

□ Get User

□ Verify User Added

□ Deposit Money

□ Login

□ Account Activity

□ Pay Bill Online

□ Logout

□ Get Transactions

□ Delete User

□ Verify User Deleted

Ending step: Add User

Enter the following parameters in the Aggregate Transaction Companion Editor:

- **Aggregate name**: The name of the aggregation step. Spaces are allowed.
- **Starting step**: Select the start (first) step of the aggregation from the pull-down menu.
- **Participants**: Check the steps to include in the aggregate (exclude the Starting and Ending steps).
- **Ending step**: Select the end (last) step of the aggregation from the pull-down menu.

You can use the Add ⊞ icon to select or clear all test steps.

All reports show the aggregate step and any individual steps that are not set to Quiet.

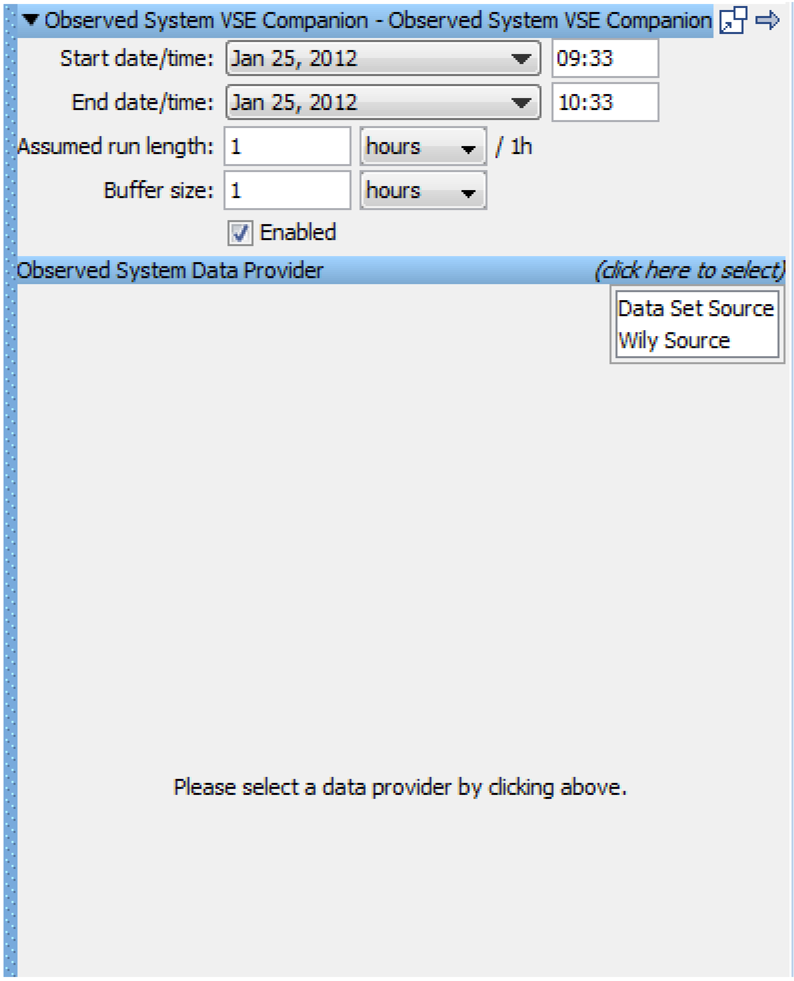
Java Protocol Companion

The only purpose of the Java protocol companion is to support Java virtual services. It is used only internally.

Observed System VSE Companion

Before LISA 6.0.6, the response time VSE would try to help ensure for an inbound request would come from the think time specification on the particular response that was determined for that request. There are times, primarily during load and performance testing scenarios, where the response times should be modeled after those from a live system. So, for example, a live system might show a drop in performance equivalent to twice its nominal response times during peak load and it is helpful to have VSE emulate this response time curve. Further, it is helpful to allow VSE to cover (or play back) this curve over an arbitrary time interval thus allowing for, for example, a 12 hour period of observed response time metrics to be played out over a 3 hour testing window.

The VSE Observed System companion supports these requirements. If a virtual service is to provide this behavior, you must add and configure this companion.



▼ Observed System VSE Companion - Observed System VSE Companion

Start date/time: Jan 25, 2012 09:33

End date/time: Jan 25, 2012 10:33

Assumed run length: 1 hours / 1h

Buffer size: 1 hours

☑ Enabled

Observed System Data Provider (click here to select)

Please select a data provider by clicking above.

Configuration Information

The Observed System companion requires the following parameters:

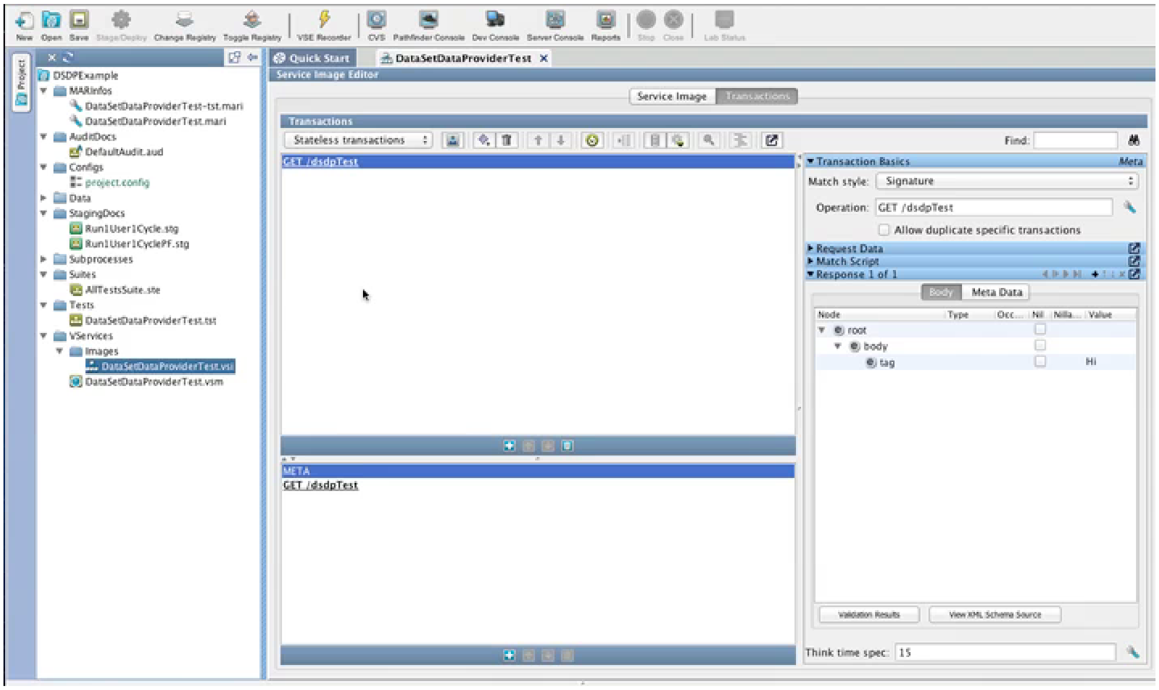
- **Start date/time** and **End date/time:** A time "window" to read observed response times between. These time stamps are inclusive. If your data set or data provider contains timestamp data that is outside this window, it will be ignored.
- **Assumed run length**: This is specified as a time duration and represents the amount of time over which the virtual service should "fit" the response time curve to. In the example mentioned earlier, this would be set to 3 hours.
- **Buffer size**: This is also specified as a time duration and may be used to control how much of the response time data is acquired at one time. This defaults to 1 hour of data at a time.
- **Observed System data provider**: This tells the companion where to get the data from. We currently provide a LISA data set data provider and a CA Application Performance Management (Wily) data provider.

Any data provider must provide three pieces of information: an ID (which is a string), a timestamp and the response time at that timestamp. The LISA data set data provider requires that the data set must provide these in fields labeled "id", "timestamp" and "responseTime", respectively. The timestamp value, if not an actual Date object, must be a string in the form "yyyy-MM-dd HH:mm:ss.SSS". How the ID maps to any specified inbound request is data-provider specific. For the data set provider it must match the request's operation. The CA Application Performance Management (Wily) provider uses a regular expression based approach.

Data Set Source Example

This example illustrates using a data set to provide the input for the Observed System VSE companion. This companion allows you to change the response time for a transaction, or for multiple transactions over time, based on the definition of the data provided by this companion.

The service image shown contains one transaction, and the Think time spec is set to 15 milliseconds.

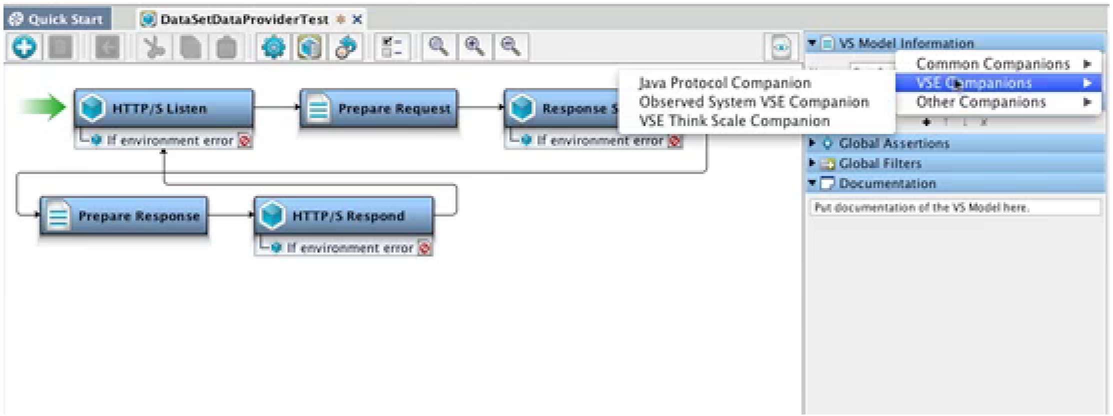


Match style: Signature

Operation: GET /dsdpTest

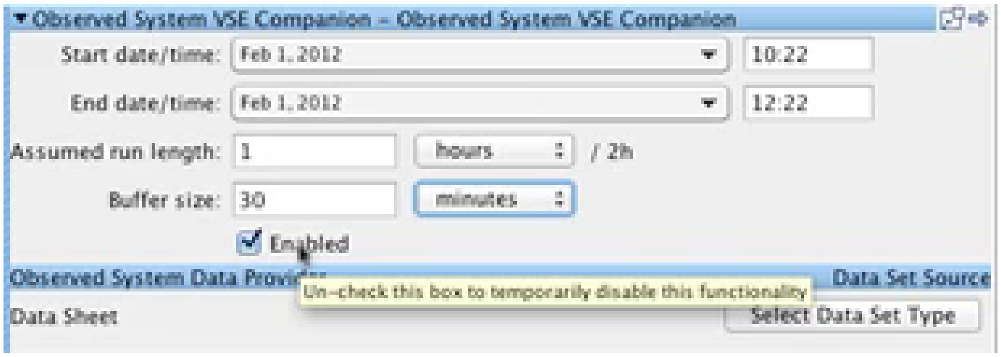
Think time spec: 15

The virtual service model associated with this service image has a few simple steps. To add a companion, click the + Add icon under the Companions panel, and select VSE Companions > Observed System VSE Companion.



The top part of the panel provides general information about the parameters for the companion.

In this example, the Start date/time and End date/time define a two-hour window, but the Assumed run length is set to one hour, which means for every one hour of VSE runtime, it will go through two hours of data from the data set data provider. The buffer size of 30 minutes means that VSE will go to the data set every 30 minutes to retrieve data. That buffer size is before any scaling is done, so with a Buffer size of 30 minutes in this example, VSE will go to the data set and retrieve data between 10:22 and 10:52, and VSE will do the scaling of, in this case, one half, so it can do the calculations correctly. The Enabled check box can be unselected to temporarily disable the companion.



▼ Observed System VSE Companion - Observed System VSE Companion

Start date/time: Feb 1, 2012 10:22

End date/time: Feb 1, 2012 12:22

Assumed run length: 1 hours / 2h

Buffer size: 30 minutes

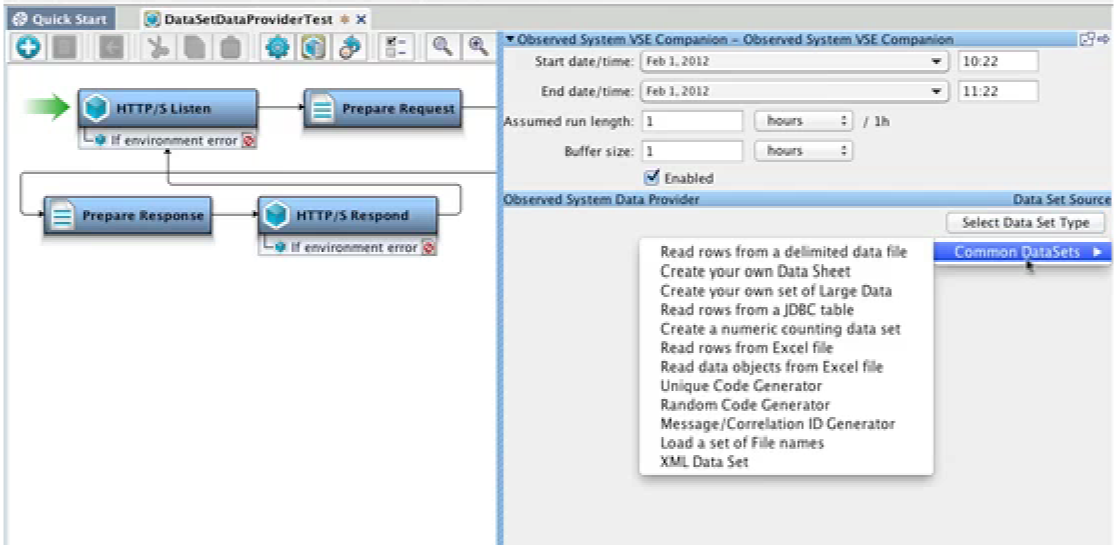
☑ Enabled

Observed System Data Provider — Data Set Source

Un-check this box to temporarily disable this functionality

Data Sheet

The Observed System Companion can be backed by any kind of data providers, including LISA data sets and the Wily Observed System Data Provider. Click **(click here to select)**, select Data Set Source, and the Select Data Set Type button will appear. Click that button and select Common DataSets, then select Create your own Data Sheet.



▼ Observed System VSE Companion - Observed System VSE Companion

Start date/time: Feb 1, 2012 10:22

End date/time: Feb 1, 2012 11:22

Assumed run length: 1 hours / 1h

Buffer size: 1 hours

☑ Enabled

Observed System Data Provider — Data Set Source

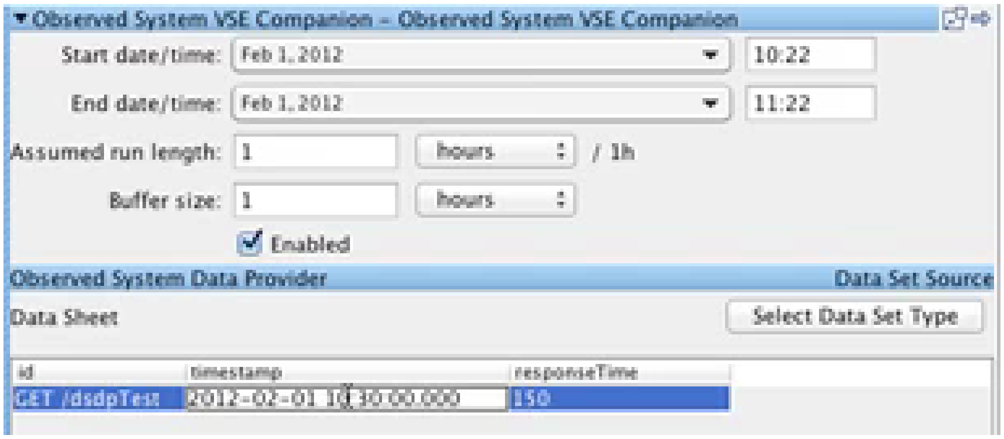
Select Data Set Type
Common DataSets ▶
Read rows from a delimited data file
Create your own Data Sheet
Create your own set of Large Data
Read rows from a JDBC table
Create a numeric counting data set
Read rows from Excel file
Read data objects from Excel file
Unique Code Generator
Random Code Generator
Message/Correlation ID Generator
Load a set of File names
XML Data Set

When the table for Create your own Data Sheet is opened, the columns are pre-populated with:

- **id**: Matched against the Operation Name when the request is being processed.
- **timestamp**:
- **responseTime**: The response time to use during playback. This overrides the Think time spec of 15 milliseconds that was set in the service image.

These are the columns you need to have for any type of data set provider you use to provide information for this companion.

In this example, copy the Operation Name of **GET / dsdpTest** from the service image and enter it in the id field. In the responseTime field, enter **150**.



▼ Observed System VSE Companion - Observed System VSE Companion

Start date/time: Feb 1, 2012 11:22

End date/time: Feb 1, 2012 11:22

Assumed run length: 1 hours / 1h

Buffer size: 1 hours

☑ Enabled

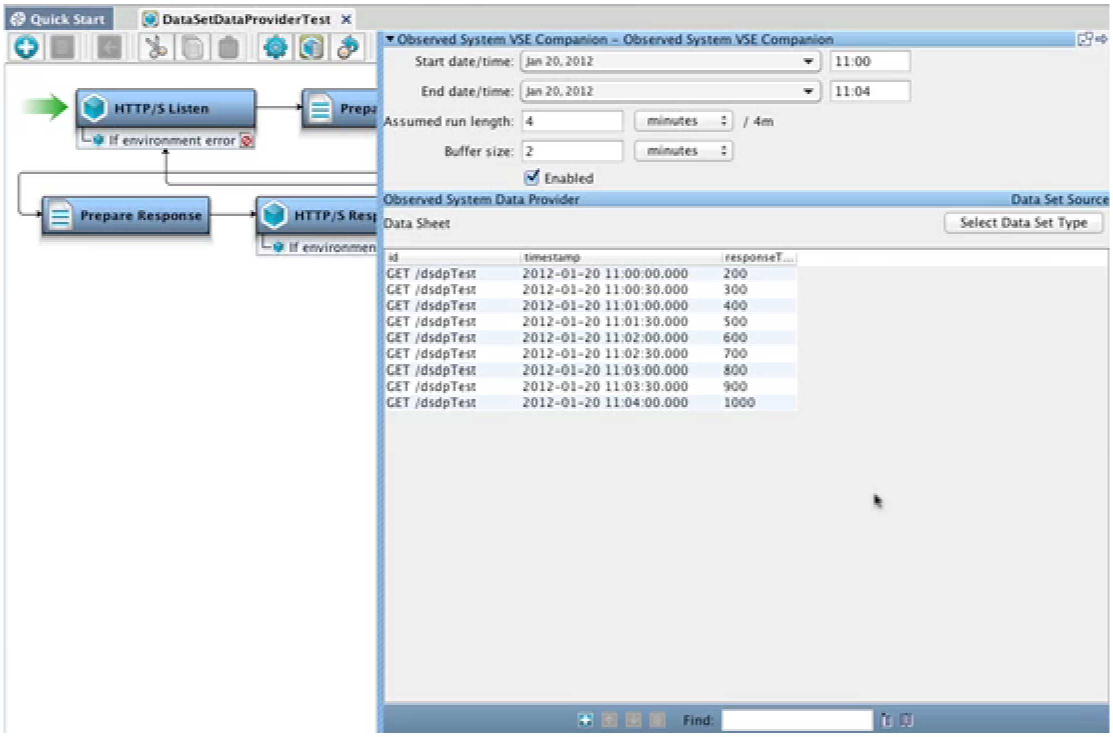
Observed System Data Provider — Data Set Source

Data Sheet

id	timestamp	responseTime
GET /dsdpTest	2012-02-01 10:30:00.000	150

In the timestamp field, enter a date and time that falls into the time window defined by Start date/time and End date/time.

To continue the example, we will use an existing project with a pre-populated data set.



Start date/time: Jan 20, 2012 11:00

End date/time: Jan 20, 2012 11:04

Assumed run length: 4 minutes / 4m

Buffer size: 2 minutes

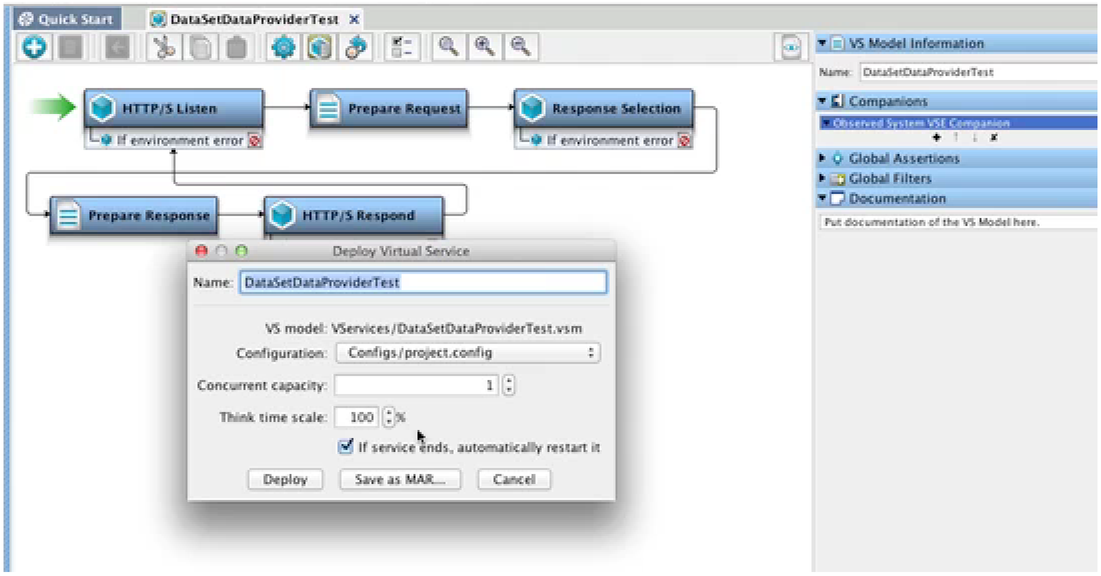
☑ Enabled

Data Sheet

id	timestamp	responseT...
GET /dsdpTest	2012-01-20 11:00:00.000	200
GET /dsdpTest	2012-01-20 11:00:30.000	300
GET /dsdpTest	2012-01-20 11:01:00.000	400
GET /dsdpTest	2012-01-20 11:01:30.000	500
GET /dsdpTest	2012-01-20 11:02:00.000	600
GET /dsdpTest	2012-01-20 11:02:30.000	700
GET /dsdpTest	2012-01-20 11:03:00.000	800
GET /dsdpTest	2012-01-20 11:03:30.000	900
GET /dsdpTest	2012-01-20 11:04:00.000	1000

The response times in the data set define an increase of 100 milliseconds every 30 seconds, so the Think time spec set in the virtual service model of 15 milliseconds should always be overridden by this companion.

When you deploy the virtual service, do not enter a Think time scale of 0%.



Name: DataSetDataProviderTest

Put documentation of the VS Model here.

Name: DataSetDataProviderTest

VS model: VServices/DataSetDataProviderTest.vsm

Configuration: Configs/project.config

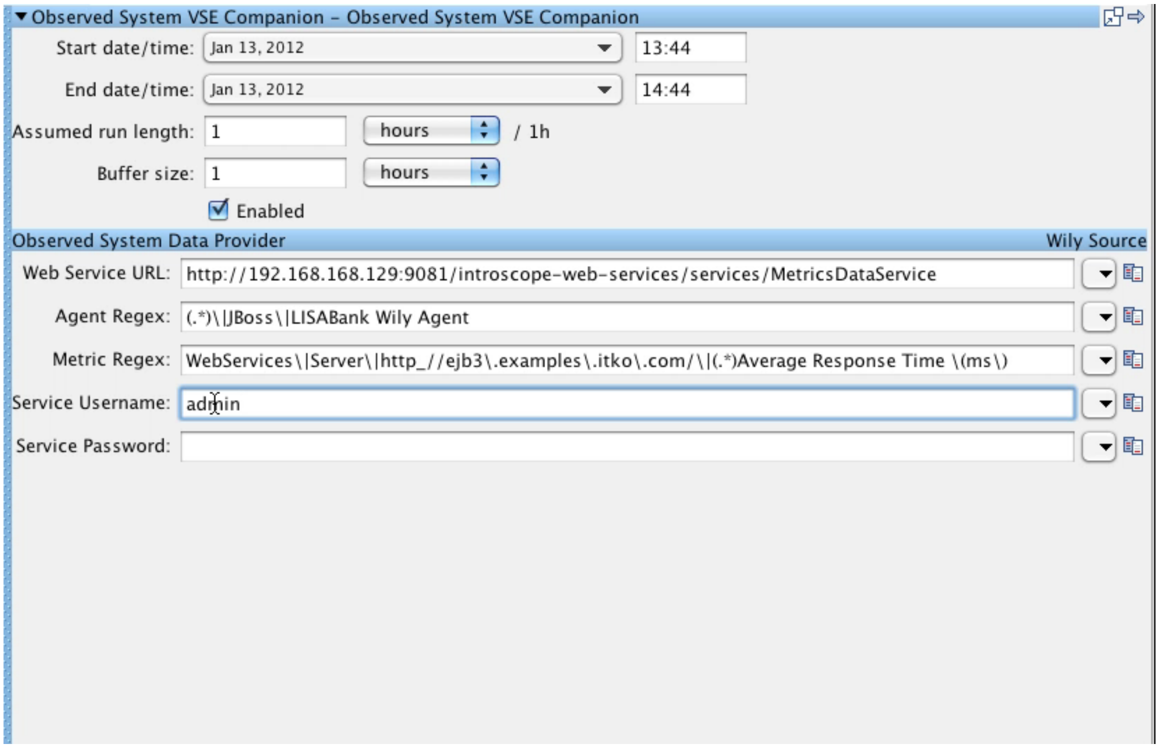
Concurrent capacity: 1

Think time scale: 100 %

☑ If service ends, automatically restart it

Observed System Data Provider (Wily) Source

The Observed System VSE Companion can also receive its input from the CA Application Performance Management (Wily) application. To configure the companion to work with Wily, enter the configuration information in the top part of the panel, then click **(click here to select)**, select Data Set Source of Wily Source.



Observed System VSE Companion - Observed System VSE Companion

Start date/time: Jan 13, 2012 13:44

End date/time: Jan 13, 2012 14:44

Assumed run length: 1 hours / 1h

Buffer size: 1 hours

☑ Enabled

Observed System Data Provider Wily Source

Web Service URL: http://192.168.168.129:9081/introscope-web-services/services/MetricsDataService

Agent Regex: (.*)\|JBoss\|LISABank Wily Agent

Metric Regex: WebServices\|Server\|http_//ejb3\.examples\.itko\.com/\|(.*)Average Response Time \(ms\)

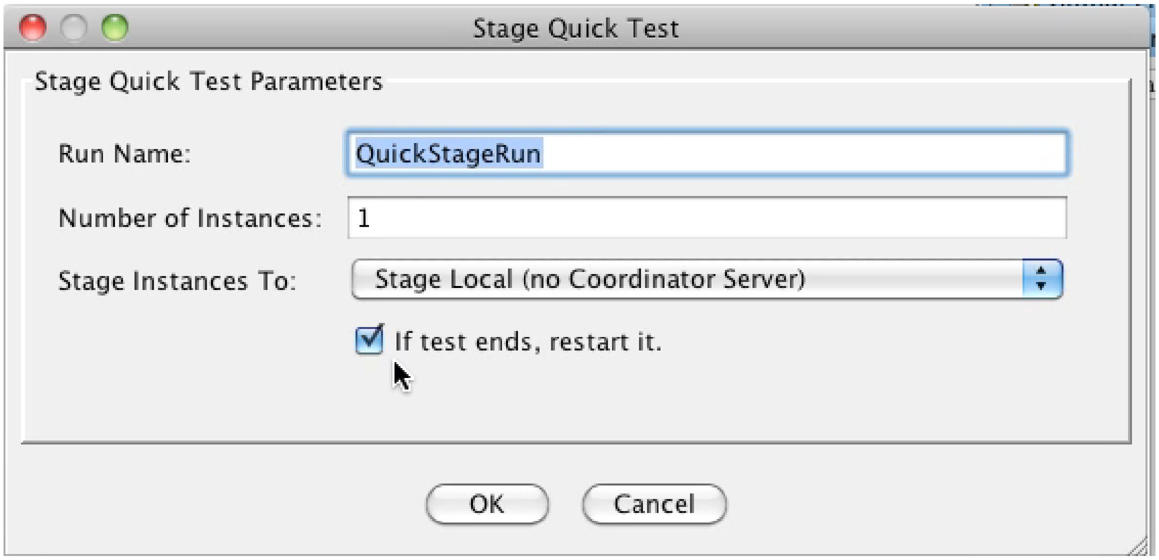
Service Username: admin

Service Password:

There are five parameters for the Wily Observed System companion:

- **Web Service URL**: The URL for the Wily web service.
- **Agent Regex**: A regular expression identifying the agent.
- **Metric Regex**: A regular expression identifying the metric.
- **Service Username**: The user name to access the Wily service.
- **Service Password**: The password for the Wily service, if required.

To test the behavior of the companion, stage a quick test.



Stage Quick Test

Stage Quick Test Parameters

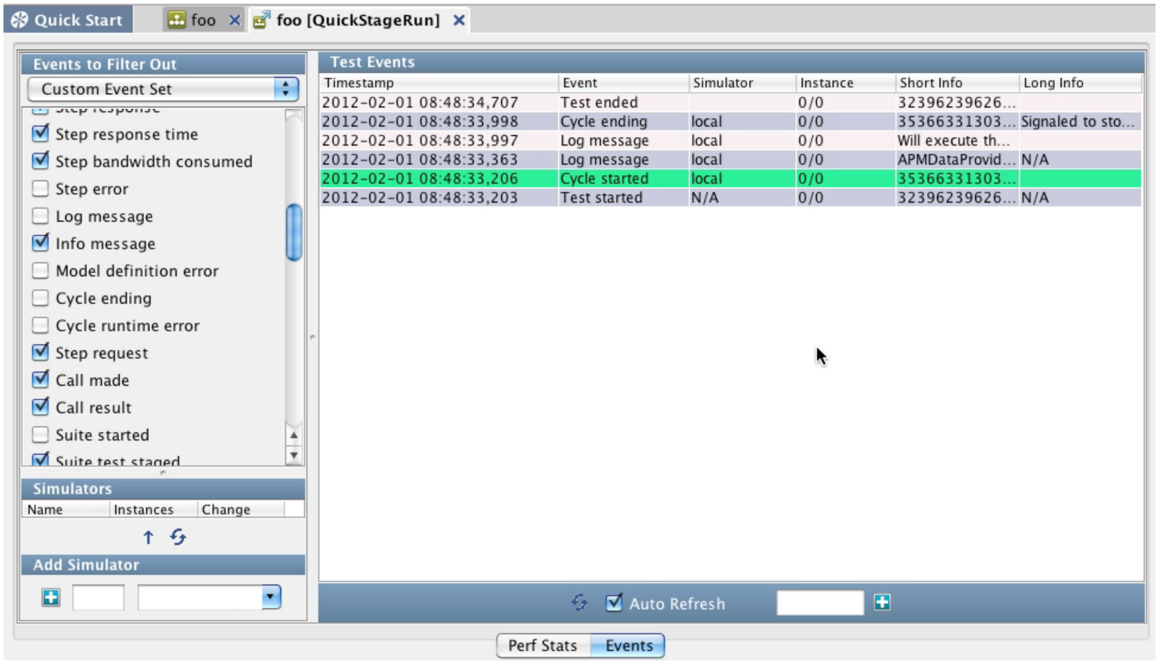
Run Name: QuickStageRun

Number of Instances: 1

Stage Instances To: Stage Local (no Coordinator Server)

☑ If test ends, restart it.

The test case in this example had one step, an output log message step. You can see the output in the Test Events tab of the Quick Stage Run window.



Events to Filter Out: Custom Event Set

- ☑ Step response time
- ☑ Step bandwidth consumed
- □ Step error
- □ Log message
- ☑ Info message
- □ Model definition error
- □ Cycle ending
- □ Cycle runtime error
- ☑ Step request
- ☑ Call made
- ☑ Call result
- □ Suite started
- ☑ Suite test staged

Test Events

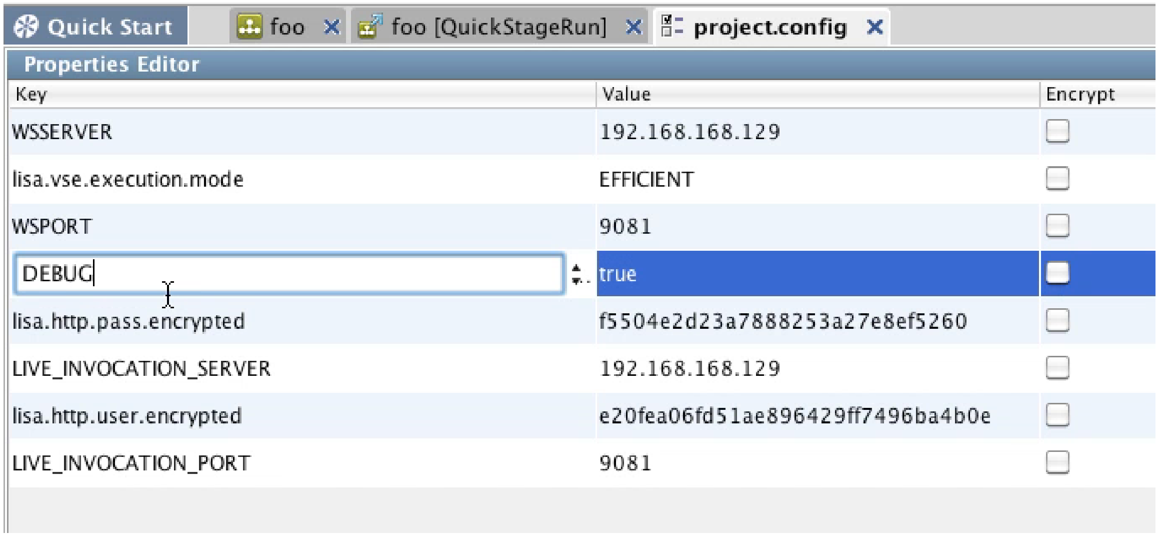
Timestamp	Event	Simulator	Instance	Short Info	Long Info
2012-02-01 08:48:34,707	Test ended		0/0	32396239626...	
2012-02-01 08:48:33,998	Cycle ending	local	0/0	35366331303...	Signaled to sto...
2012-02-01 08:48:33,997	Log message	local	0/0	Will execute th...	
2012-02-01 08:48:33,363	Log message	local	0/0	APMDataProvid...	N/A
2012-02-01 08:48:33,206	Cycle started	local	0/0	35366331303...	
2012-02-01 08:48:33,203	Test started	N/A	0/0	32396239626...	N/A

Simulators

Name	Instances	Change

☑ Auto Refresh

When you look at the project config file used for this test case, you can see that having the property debug set to true means you will get the log message in the output



Properties Editor

Key	Value	Encrypt
WSSERVER	192.168.168.129	□
lisa.vse.execution.mode	EFFICIENT	□
WSPORT	9081	□
DEBUG	true	□
lisa.http.pass.encrypted	f5504e2d23a7888253a27e8ef5260	□
LIVE_INVOCATION_SERVER	192.168.168.129	□
lisa.http.user.encrypted	e20fea06fd51ae896429ff7496ba4b0e	□
LIVE_INVOCATION_PORT	9081	□

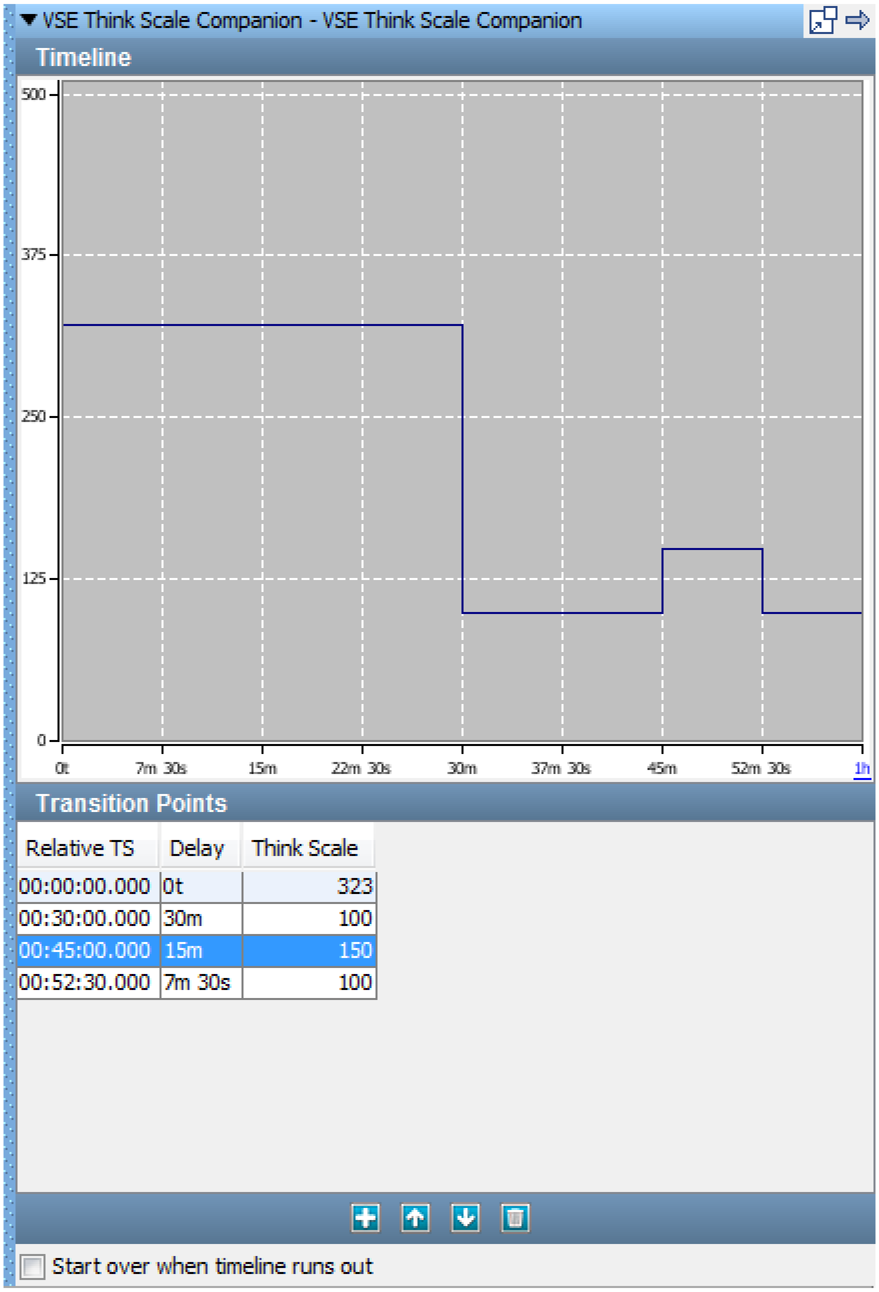
Runtime Priorities

The process that is followed during runtime is:

1. The think time specification from the VSE response will be examined at the point when the delay factor is being determined.
2. If the think time specification contains state (that is, a LISA double-brace expression), it will be evaluated directing and the result used as the response time for the response.
3. If it does not contain state and the virtual service contains the Observed System companion, the companion will be asked to determine the response time for the response.
4. If the companion is not present or not able to determine a response time, the think time specification will be used as-is as the response time.

VSE Think Scale Companion

The VSE Think Scale Companion can be added to a virtual service model to allow the think scale percent for the service to change over time by specifying the graph of think scale changes over time.



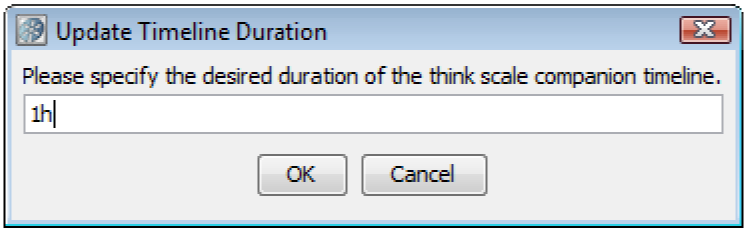
Relative TS	Delay	Think Scale
00:00:00.000	0t	323
00:30:00.000	30m	100
00:45:00.000	15m	150
00:52:30.000	7m 30s	100

□ Start over when timeline runs out

You can use the Add, Delete and Move buttons to add, delete and reorder Transition Points. You can directly edit the Delay and Think Scale entries, or you can click and drag the lines in the Timeline display to indicate the Delay and Scale you want, and the Transition Points table will be updated to correspond.

- **Relative TS**: Calculated based on the Delay you enter, in format hh:mm:ss.ms.
- **Delay**: The amount of time, in minutes and seconds, to wait before the specified Think Scale is applied.
- **Think Scale**: The think scale is a percentage that is applied to the think times in the responses.
- **Start over when timeline runs out**: If the end of the timeline is reached, start from the first Relative TS.

If you click the label for the right-most tick mark on the horizontal axis (in the window shown previously, the "1h" label) you can adjust the total timeline of the companion. You will see the Update Timeline Duration dialog. If you update the timeline duration to be shorter than your transition points, you will receive a warning that some transition points will be removed.



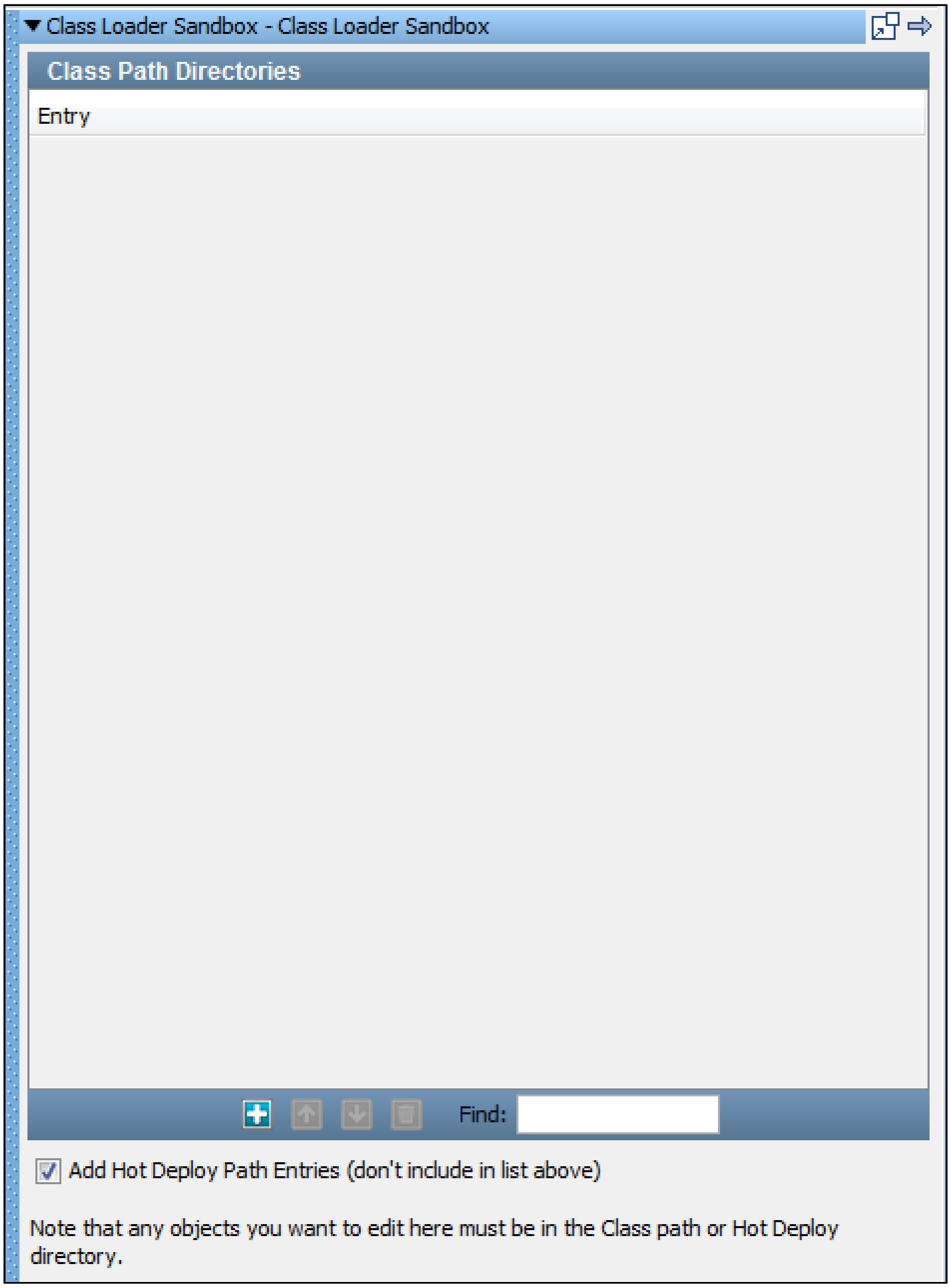
Please specify the desired duration of the think scale companion timeline.

1h

As you move your cursor over the graph, you will see that tooltips are displayed showing time on vertical lines and think scale percentages on horizontal ones.

Create a Sandbox Class Loader for Each Step Companion

The **Create a Sandbox Class Loader** companion lets you verify that every test run executes in its own Java Class loader (JVM). This is valuable when testing local Java objects that are not designed for multi-threaded or multi-user access. Most testing will not require this companion; it is usually only necessary when testing local Java objects with the Dynamic Java Execution step.



☑ Add Hot Deploy Path Entries (don't include in list above)

Note that any objects you want to edit here must be in the Class path or Hot Deploy directory.

To configure the Class Loader Sandbox companion, use the Class Path Sandbox companion editor.

- If the class you want to run is already in the **hotDeploy** directory, check the **Add Hot Deploy Path Entries** check box.
- If the class you want to run is not in the **hotDeploy** directory, use the Add ⊞ button to add a line in the Class Path Directories list and add the appropriate class path for the class.

> ⚠ Any Java objects you want to edit or run:
>
> - Must be in the class path or the **hotDeploy** directory.
> - Must not be in the LISA **lib** or **bin** directories. The Class Loader Sandbox will not work because of the way the Java VM loads classes.

Set Final Step to Execute Companion

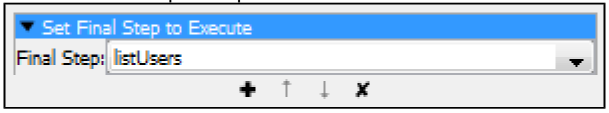
The **Set Final Step to Execute** companion lets you verify that the system under test is left in a consistent state regardless of the outcome of the test. You specify which steps are always run last when the normal test flow is circumvented.

A common use is to make sure that resources are released at the end of the test. Specifying the first step is not commonly needed, but there are many scenarios where specifying the final step is important.

The final step will be executed even if the test case reaches an End step or the test case is instructed to end abruptly.

The final step is not shown in the **Execution History** list in the ITR, but the results can be seen in the **Events** tab for the last step executed.

The Set Final Steps companion Editor is used to set the final step.



▼ Set Final Step to Execute

Final Step: listUsers

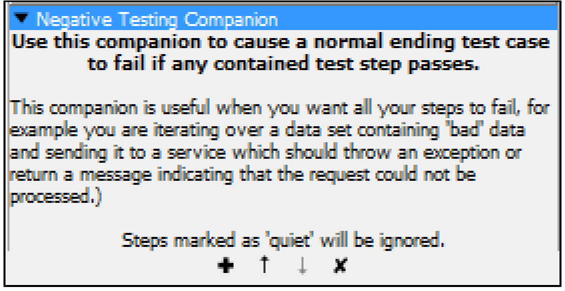
To configure the Set Final Steps companion, enter the following parameter:

- **Finalize Step**: Select the final step to execute from the drop-down menu.

Negative Testing Companion

the **Negative Testing** companion is useful when you want all your steps to fail. Use this companion to cause a normal-ending test case to fail if any contained test step passes.

Example:



▼ Negative Testing Companion

Use this companion to cause a normal ending test case to fail if any contained test step passes.

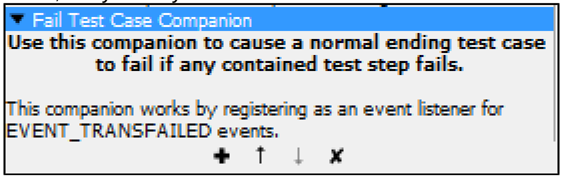
This companion is useful when you want all your steps to fail, for example you are iterating over a data set containing 'bad' data and sending it to a service which should throw an exception or return a message indicating that the request could not be processed.)

Steps marked as 'quiet' will be ignored.

There are no configuration parameters for this companion.

Fail Test Case Companion

The **Fail Test Case** companion will mark a test case that ended normally as failed if any of the test steps failed. This would typically happen when a failed test redirected to a step other than the fail step. An event listener for the EVENT_TRANSFAILED event is registered in this companion. An example of this would be when you have a WSDL validation assertion on a web service step. You want to be informed that the validation failed, but you may also want to continue with the test case.



▼ Fail Test Case Companion

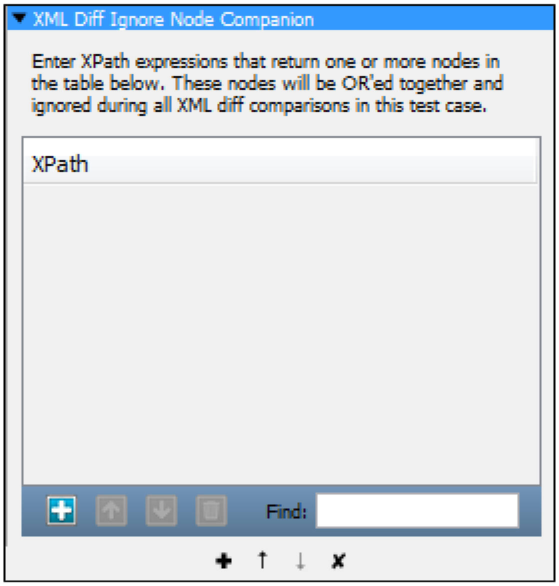
Use this companion to cause a normal ending test case to fail if any contained test step fails.

This companion works by registering as an event listener for EVENT_TRANSFAILED events.

There are no configuration parameters for this companion.

XML Diff Ignored Nodes Companion

The **XML Diff Ignored Nodes** companion lets you enter XPath expressions that return one or more nodes in the following table. These nodes will be 'OR'ed together and ignored during all XML diff comparisons during this test case.



Enter XPath expressions that return one or more nodes in the table below. These nodes will be OR'ed together and ignored during all XML diff comparisons in this test case.

LISA Hooks

LISA hooks are an automatic global means to execute logic at start/end of a test case and applied to all test cases in the environment where the hook is deployed.

Hooks work similarly to companions; they run before a test starts and after a test finishes.

The difference is, hooks are defined at the *LISA application level,* and every test case in LISA runs the hooks. There is no option not to run a hook that is added to LISA, for any test case.

A hook is a mechanism in LISA that allows for the automatic inclusion of test setup and/or teardown logic for all the tests running in LISA. An alternate definition of a hook is a system-wide companion. Anything that a hook can perform can be modeled as a companion in LISA.

Hooks are used to configure test environments, prevent tests from executing that are not properly configured or do not follow defined best practices, or simply to provide common operations.

LISA hooks are Java classes that are on the LISA Class path. LISA hooks are NOT defined in .lisaextensions file, but in local.properties or lisa.properties as

```
lisa.hooks=com.mypackage1.MyHook1, com.mypackage2.MyHook2
```

A hook is executed/invoked at the start/end of all subprocesses in a test case in addition to the test case itself. So if a test case has three subprocesses, the hook logic will be executed four times (once for the main test and once for each of three subprocesses).

To prevent startHook and/or endHook to be executed by a subprocess, perform this check:

```
String marker = (String)testExec.getStateValue(TestExec.FROM_PARENT_TEST_MARKER_KEY)


"marker" is set to "true" only when it is a subprocess.


if (!"true".equals(marker))


all start/end Hook logic here that should not be executed when in sub process
```

Differences between Hooks and Companions

- Hooks are global in scope. Testers do not specifically include a hook in their test case as is the required practice for companions. Hooks are registered at the system level. If you need every test to include the logic and do not want users to accidentally omit it, a hook is a better mechanism.
- Hooks are deployed at the LISA install level, not at the test case level. If a test is run on two computers, one computer has a hook registered and the other does not, then the hook will only run when the test is staged on the computer where it is explicitly deployed. Companions defined in the test case execute regardless of any install-level configuration.
- Hooks are practically invisible to the user and therefore cannot request any custom parameters from the user. They get their parameters from properties in the configuration or from the system. Companions can have custom parameters because they are rendered in the Model Editor.

For more details on hooks, see LISA Advanced Features.

Complex Object Editor (COE)

The **Complex Object Editor (COE)** lets you interact with Java objects without having to write additional Java code.

You can change the current value of an input parameter, make method calls, and examine return values. You can also do simple in-line filtering and add simple assertions on the return value.

Many test steps involve the manipulation of Java objects. Whether you are working directly with Java objects, as in a Dynamic Java Execution step, or an Enterprise JavaBean (EJB) step, or indirectly - such as input parameters to a web service, or return messages from an Enterprise Service Bus (ESB), you will be working in the Complex Object Editor.

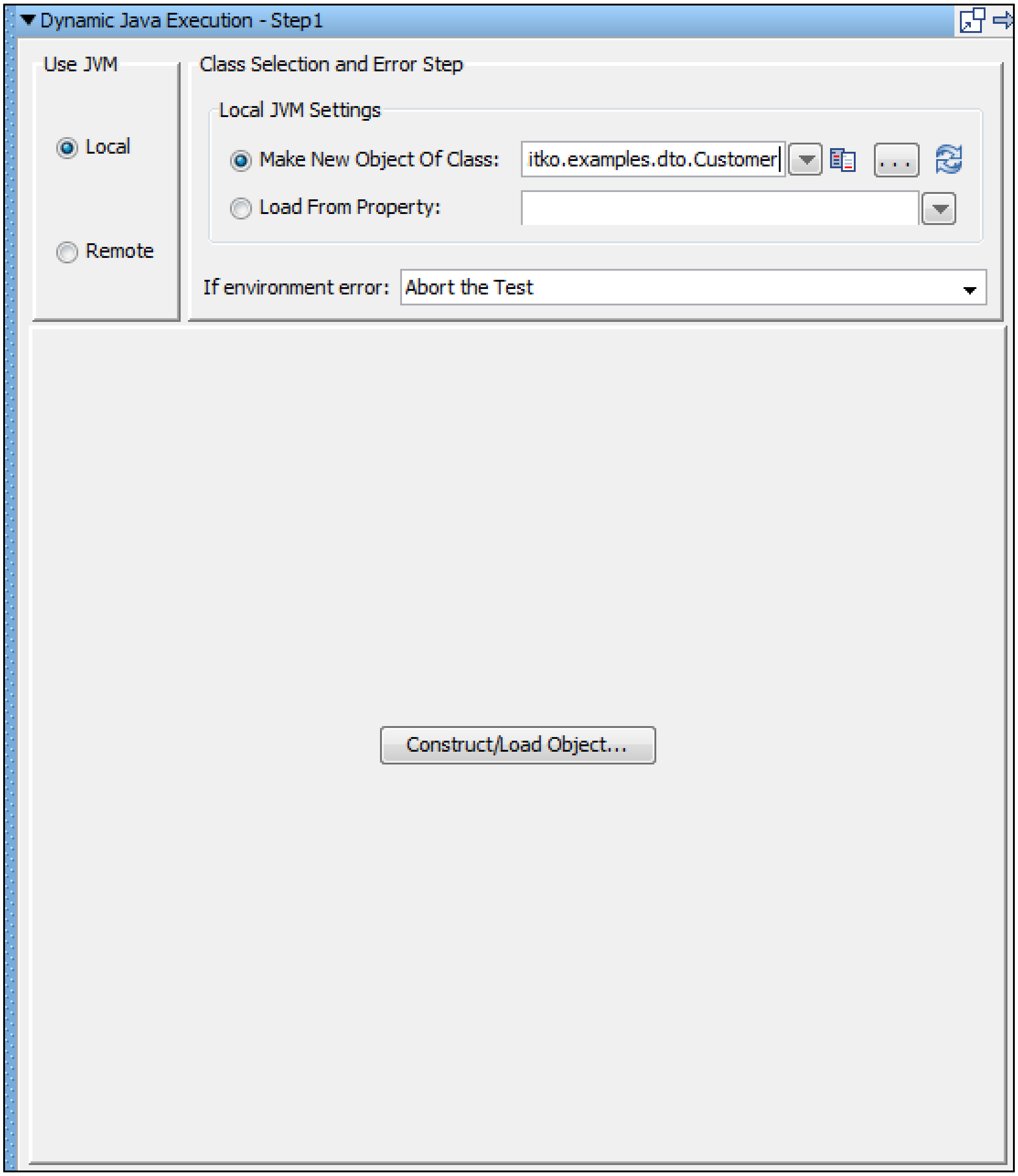
This chapter starts by describing the user interface. Then it looks at several usage scenarios that become progressively more complex.

The following topics are available in this chapter.

Invoking the COE

Regardless of where it appears, the Complex Object Editor (COE) has the same look and feel.

For an example, we will show the Dynamic Java Execution step, using the **Customer** class. When you invoke this step, this is what you see.



Dynamic Java Execution - Step1

Use JVM

● Local

○ Remote

Class Selection and Error Step

Local JVM Settings

● Make New Object Of Class: itko.examples.dto.Customer

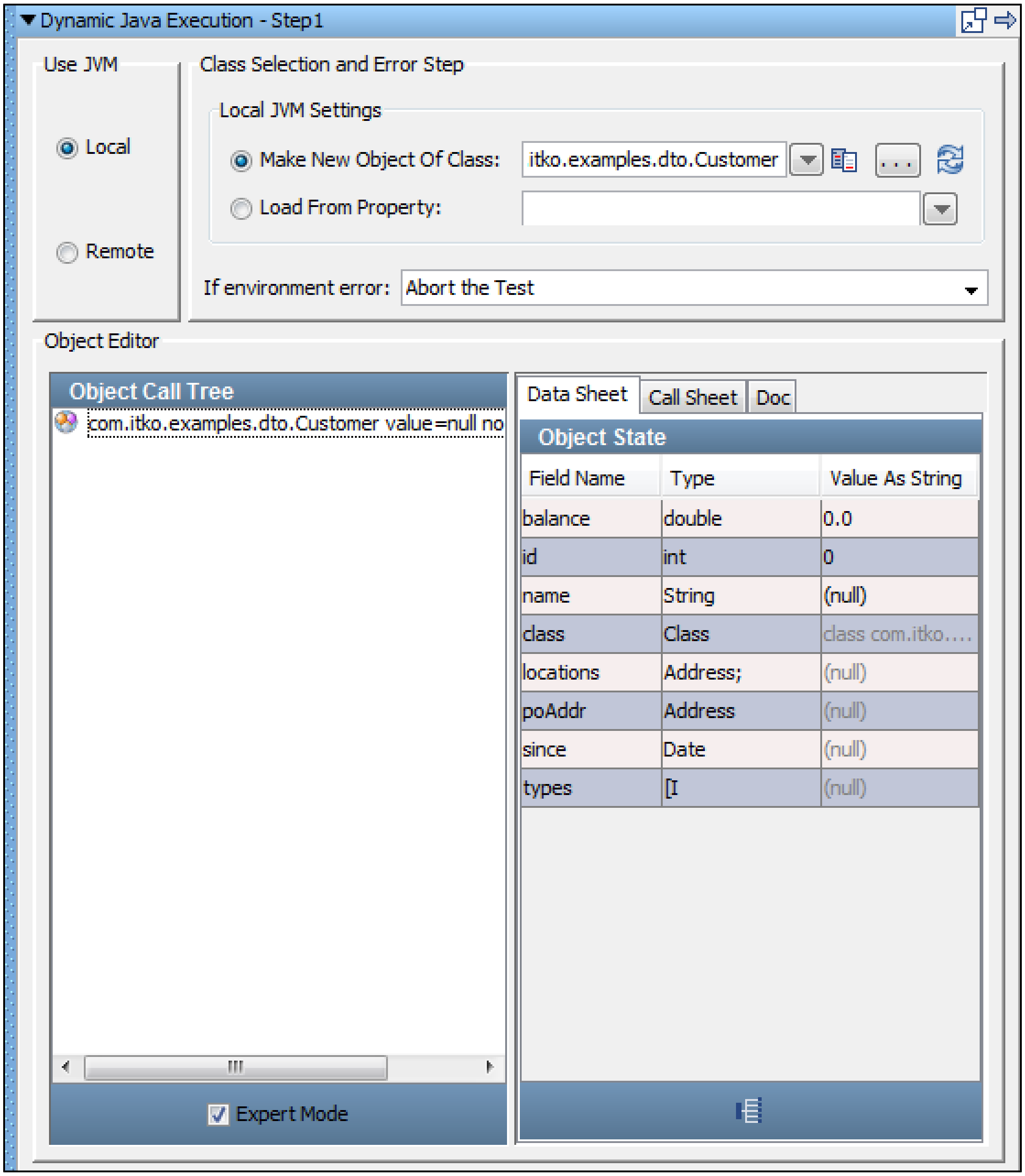
○ Load From Property:

If environment error: Abort the Test

1. Select to use the Local JVM and enter **com.itko.examples.dto.Customer** in the Make new object of class field.
2. Click Construct/Load Object to load the object into the object editor.

Complex Object Editor

After the object is loaded, the Complex Object Editor is invoked.



Use JVM

● Local

○ Remote

Class Selection and Error Step

Local JVM Settings

● Make New Object Of Class: itko.examples.dto.Customer

○ Load From Property:

If environment error: Abort the Test

com.itko.examples.dto.Customer value=null no

Field Name	Type	Value As String
balance	double	0.0
id	int	0
name	String	(null)
class	Class	class com.itko....
locations	Address;	(null)
poAddr	Address	(null)
since	Date	(null)
types	[I	(null)

☑ Expert Mode

The object editor is divided into two panels. The left panel, the Object Call Tree, keeps track of method invocations, and their input parameters and return values. The Object Call Tree Panel is described in detail in the next topic.

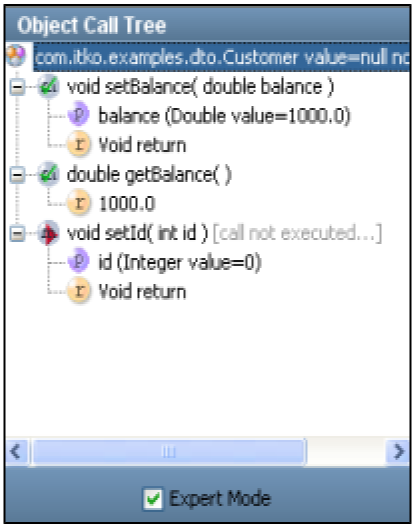
The right panel is the Object State, which has a set of dynamic tabs (Data Sheet Panel, Call Sheet Panel, Doc Panel) that will show you available options at any given time.

The previous screen shows a Java object of type Customer loaded in the editor. This particular object was loaded using the Dynamic Java Execution test step, but it could have been loaded as the result of any number of operations. No calls have been invoked on the object.

Object Call Tree Panel

As you manipulate your Java object (**Customer**), the Object Call Tree will expand to keep track of the calls invoked and the associated parameter values.

As an example, an Object Call Tree after several methods have been invoked follows:



Object Call Tree

```
com.itko.examples.dto.Customer value=null nc
⊟ void setBalance( double balance )
    balance (Double value=1000.0)
    Void return
⊟ double getBalance( )
    1000.0
⊟ void setId( int id ) [call not executed...]
    id (Integer value=0)
    Void return
```

☑ Expert Mode

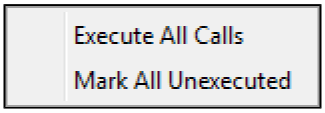
Object Call Tree icons

The following icons are used to identify the branches in the object call tree:

Icon	Description
●	The type (class) of the object currently loaded, followed by response from calling **toString** method of object.
✓	The constructor that was called. This is shown if multiple constructors exist.
▶	A method call that has not been executed.
✓	A method call that has been executed.
p	The input parameters (type and current value) for the enclosing method.
r	The return value (current value if call has been executed) for the enclosing method.

Clicking an item in the Object Call Tree will display the appropriate set of tabs in the Data Sheet and Call Sheet in the right panel.

Right-clicking an item in the Object Call Tree displays the following menu.

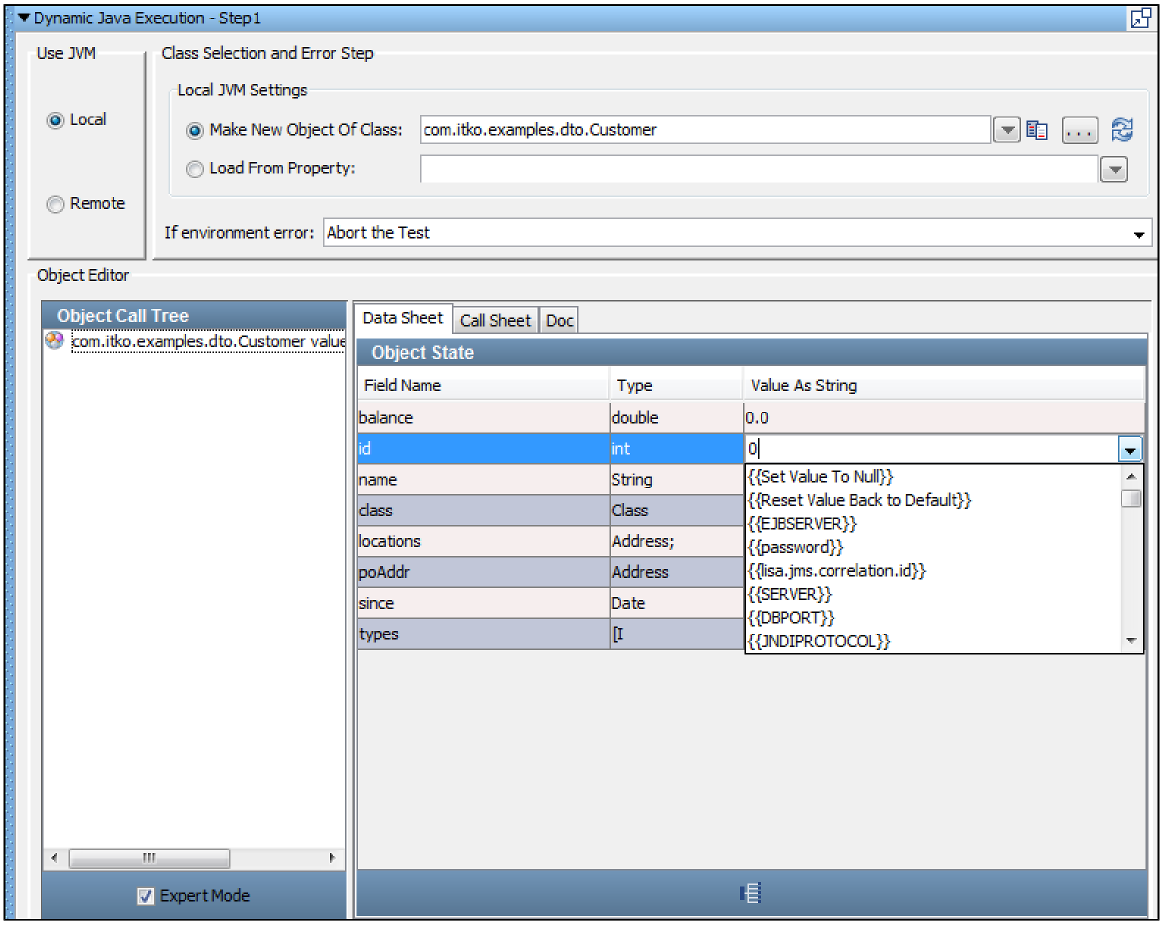


You can execute all calls or mark all calls as unexecuted in the Object Call Tree.

Data Sheet and Call Sheet Panels

Data Sheet Panel

The right panel shows three tabs, with the Data Sheet tab active by default.



Dynamic Java Execution - Step1

Use JVM

● Local

○ Remote

Class Selection and Error Step

Local JVM Settings

● Make New Object Of Class: com.itko.examples.dto.Customer

○ Load From Property:

If environment error: Abort the Test

Object Editor

Object Call Tree

com.itko.examples.dto.Customer value

Data Sheet | Call Sheet | Doc

Object State

Field Name	Type	Value As String
balance	double	0.0
id	int	0
name	String	
class	Class	
locations	Address;	
poAddr	Address	
since	Date	
types	[I	

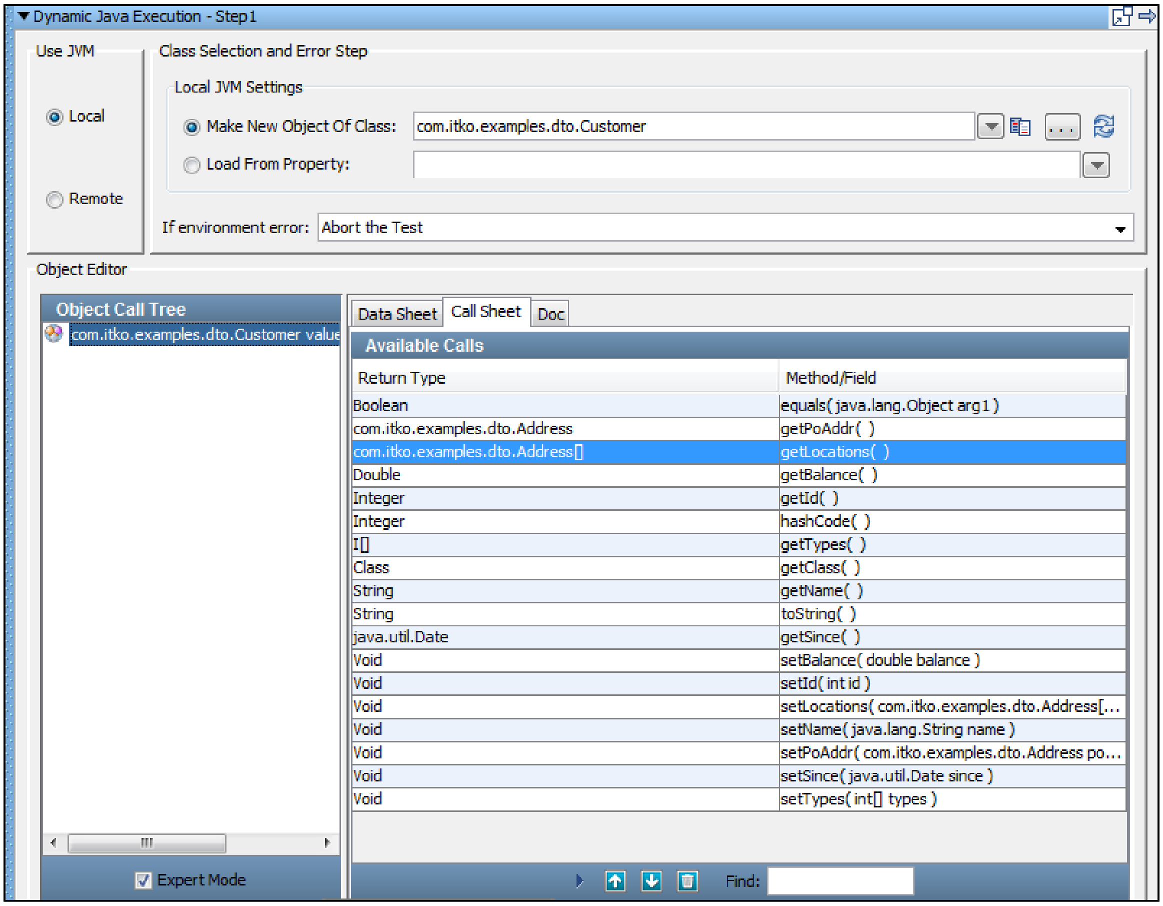
{{Set Value To Null}}
{{Reset Value Back to Default}}
{{EJBSERVER}}
{{password}}
{{lisa.jms.correlation.id}}
{{SERVER}}
{{DBPORT}}
{{JNDIPROTOCOL}}

☑ Expert Mode

Data shown in black can be edited in this tab. The data values are edited in the Value as String column. These will always be primitives or strings. Values dimmed cannot be edited here, but there will be other screens where these objects can be edited. The address field, for example, is an object of type Address that cannot be edited in this tab.

Call Sheet Panel

In the Call Sheet tab, the COE shows the methods/fields calls that are available, and their return types.



Dynamic Java Execution - Step1

Use JVM

● Local

○ Remote

Class Selection and Error Step

Local JVM Settings

● Make New Object Of Class: com.itko.examples.dto.Customer

○ Load From Property:

If environment error: Abort the Test

Object Editor

Object Call Tree

com.itko.examples.dto.Customer value

☑ Expert Mode

Available Calls

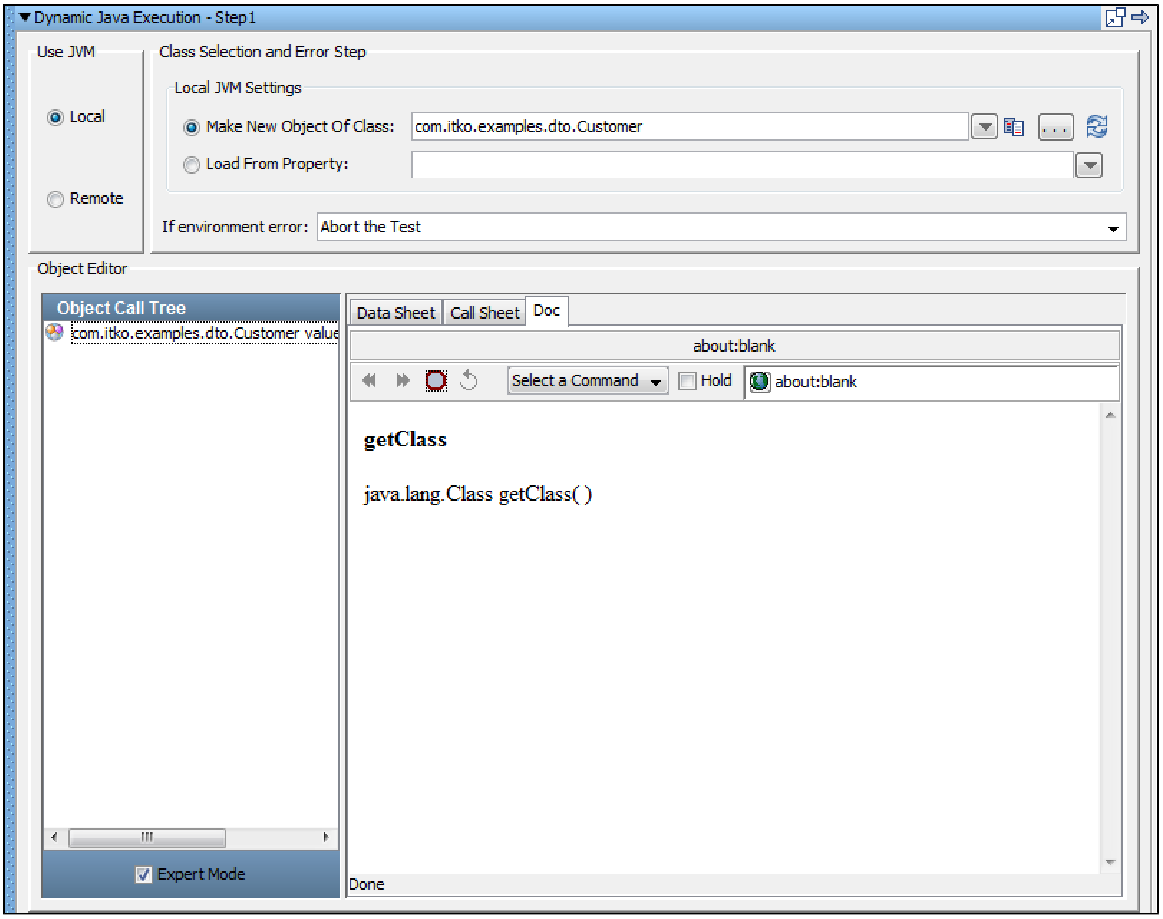
Return Type	Method/Field
Boolean	equals(java.lang.Object arg1)
com.itko.examples.dto.Address	getPoAddr( )
com.itko.examples.dto.Address[]	getLocations( )
Double	getBalance( )
Integer	getId( )
Integer	hashCode( )
I[]	getTypes( )
Class	getClass( )
String	getName( )
String	toString( )
java.util.Date	getSince( )
Void	setBalance(double balance)
Void	setId(int id)
Void	setLocations(com.itko.examples.dto.Address[...
Void	setName(java.lang.String name)
Void	setPoAddr(com.itko.examples.dto.Address po...
Void	setSince(java.util.Date since)
Void	setTypes(int[] types)

To add a method, select it in the Methods/Fields list and click the Add Method ▶ icon at the bottom of the tab. The selected method now appears in the Object Call Tree (in the left panel).

When you are in the object call tree you can provide input parameters and invoke the method.

Doc Panel

The Doc tab displays any Java API documentation that has been made available for this class.



Use JVM: ● Local ○ Remote

● Make New Object Of Class: com.itko.examples.dto.Customer

○ Load From Property:

If environment error: Abort the Test

com.itko.examples.dto.Customer value

□ Hold

getClass

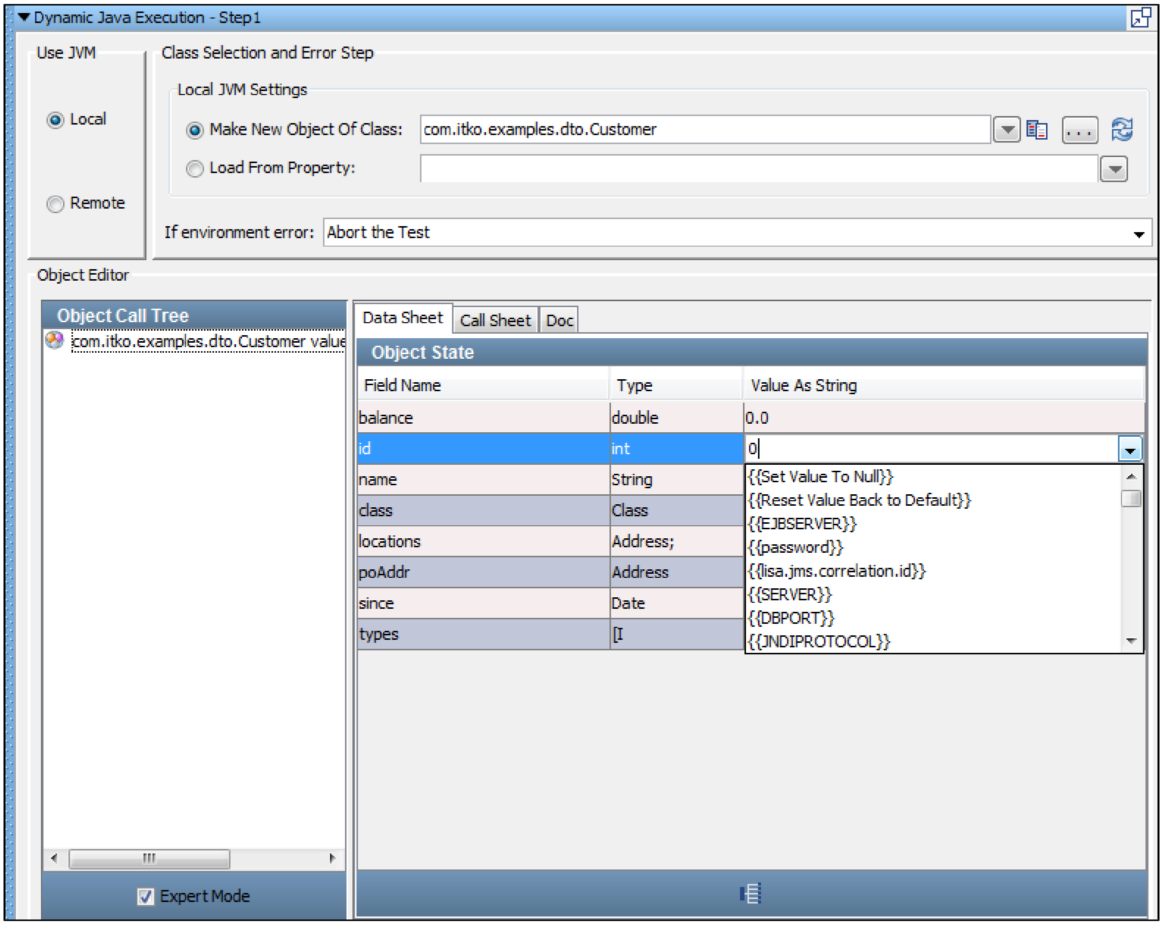
java.lang.Class getClass( )

☑ Expert Mode

Object Interaction Panels

There is a different interface for each action that is be displayed in the Data Sheet and Call Sheet.

The tab and its interface will change depending on what you have selected in the Object Call Tree in the left panel.



Dynamic Java Execution - Step1

Use JVM: ● Local ○ Remote

Class Selection and Error Step

Local JVM Settings

● Make New Object Of Class: com.itko.examples.dto.Customer

○ Load From Property:

If environment error: Abort the Test

Object Editor

Object Call Tree: com.itko.examples.dto.Customer value

Data Sheet | Call Sheet | Doc

Object State

Field Name	Type	Value As String
balance	double	0.0
id	int	0
name	String	
class	Class	
locations	Address;	
poAddr	Address	
since	Date	
types	[I	

☑ Expert Mode

Object is selected

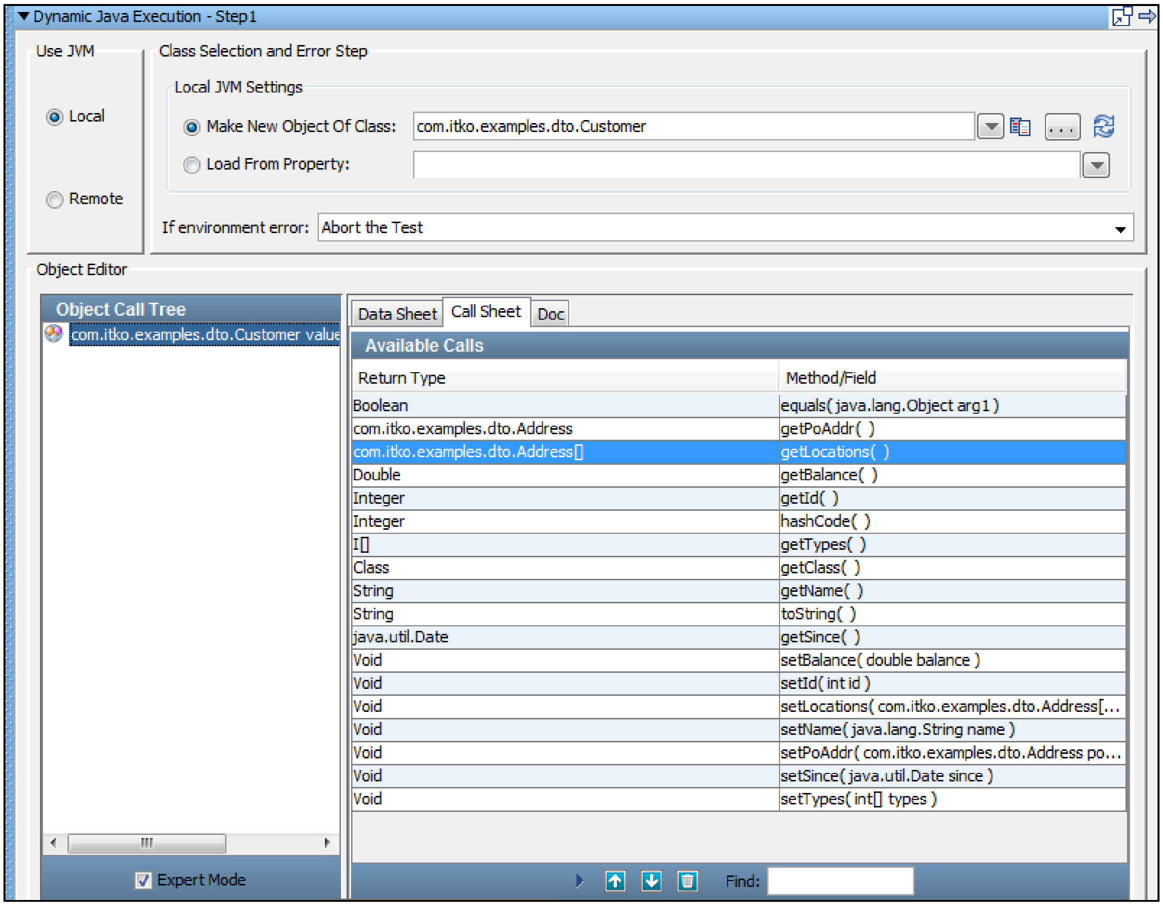
Method is selected

Input Parameter is selected

Return Value is selected

Object Panels

In the last section, you have seen that the Data Sheet, Call Sheet, and Doc tabs are available when an object is selected in the Object Call Tree.



Dynamic Java Execution - Step1

Use JVM

● Local

○ Remote

Class Selection and Error Step

Local JVM Settings

● Make New Object Of Class: com.itko.examples.dto.Customer

○ Load From Property:

If environment error: Abort the Test

Object Editor

Object Call Tree

com.itko.examples.dto.Customer value

Data Sheet | Call Sheet | Doc

Available Calls

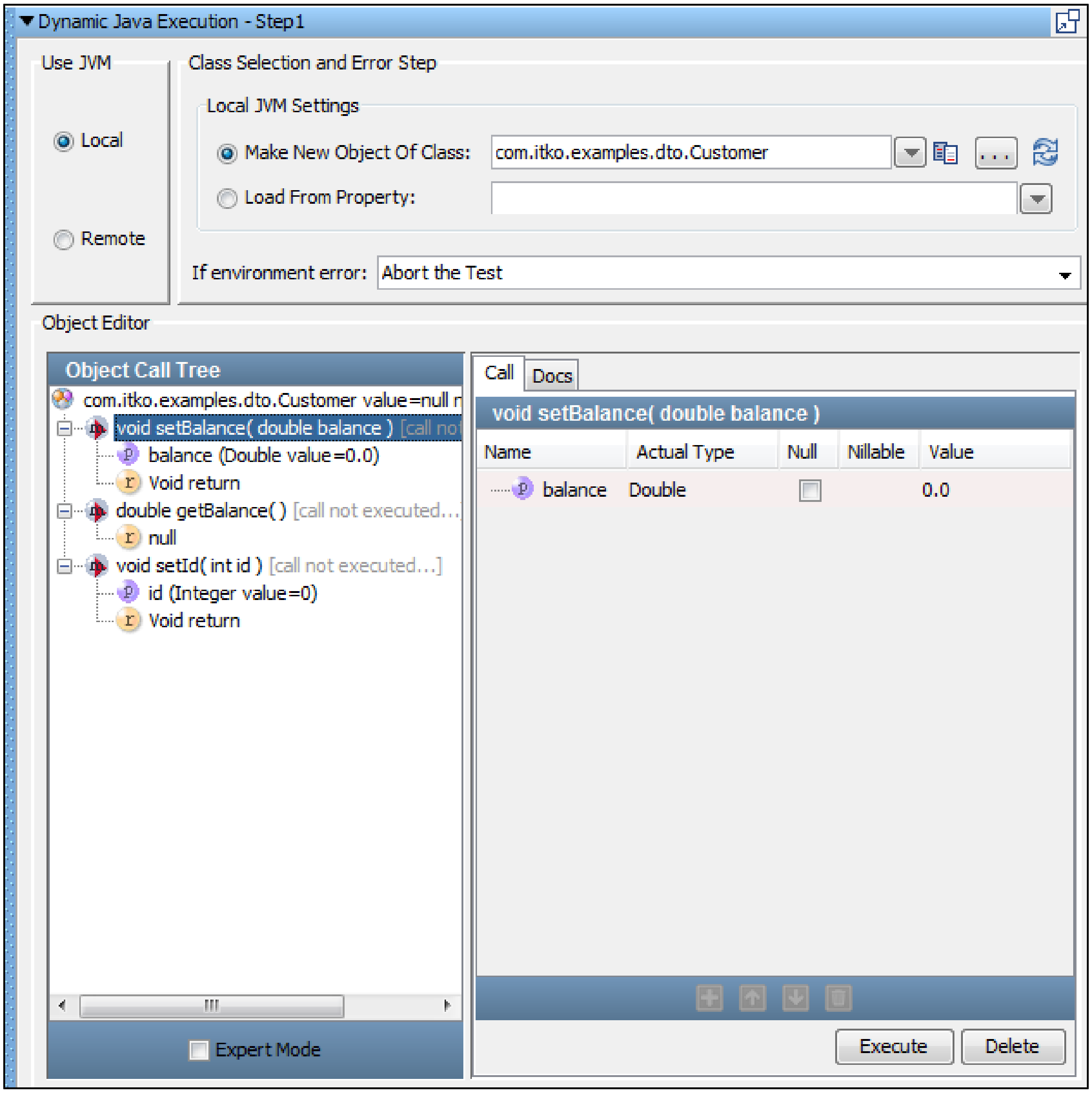
Return Type	Method/Field
Boolean	equals(java.lang.Object arg1)
com.itko.examples.dto.Address	getPoAddr( )
com.itko.examples.dto.Address[]	getLocations( )
Double	getBalance( )
Integer	getId( )
Integer	hashCode( )
I[]	getTypes( )
Class	getClass( )
String	getName( )
String	toString( )
java.util.Date	getSince( )
Void	setBalance(double balance)
Void	setId(int id)
Void	setLocations(com.itko.examples.dto.Address[...
Void	setName(java.lang.String name)
Void	setPoAddr(com.itko.examples.dto.Address po...
Void	setSince(java.util.Date since)
Void	setTypes(int[] types)

☑ Expert Mode

Find:

Method Call Panels

If you select a method call in the Object Call Tree, the Call and Doc tabs are available in the right panel.



Dynamic Java Execution - Step1

Use JVM

● Local

○ Remote

Class Selection and Error Step

Local JVM Settings

● Make New Object Of Class: com.itko.examples.dto.Customer

○ Load From Property:

If environment error: Abort the Test

Object Editor

Object Call Tree

```
com.itko.examples.dto.Customer value=null r
  void setBalance( double balance ) [call not
    balance (Double value=0.0)
    Void return
  double getBalance( ) [call not executed...]
    null
  void setId( int id ) [call not executed...]
    id (Integer value=0)
    Void return
```

□ Expert Mode

void setBalance(double balance)

Name	Actual Type	Null	Nillable	Value
balance	Double	□		0.0

This is where you provide values for the input parameters.

The information about each parameter is provided, so you only need to supply the value. In this example, it is straightforward; the single parameter is of type "double," so we can type in a value, or a property name in the Value column.

The pull-down menu maintains a list of the current properties. If you are required to enter an object as an input parameter this requires more work. We will describe several approaches to providing objects in the subsequent sections.

Notice that the Expert Mode at the bottom of the left panel is not checked. This shows we are in the Simple mode.

Simple and Expert Mode

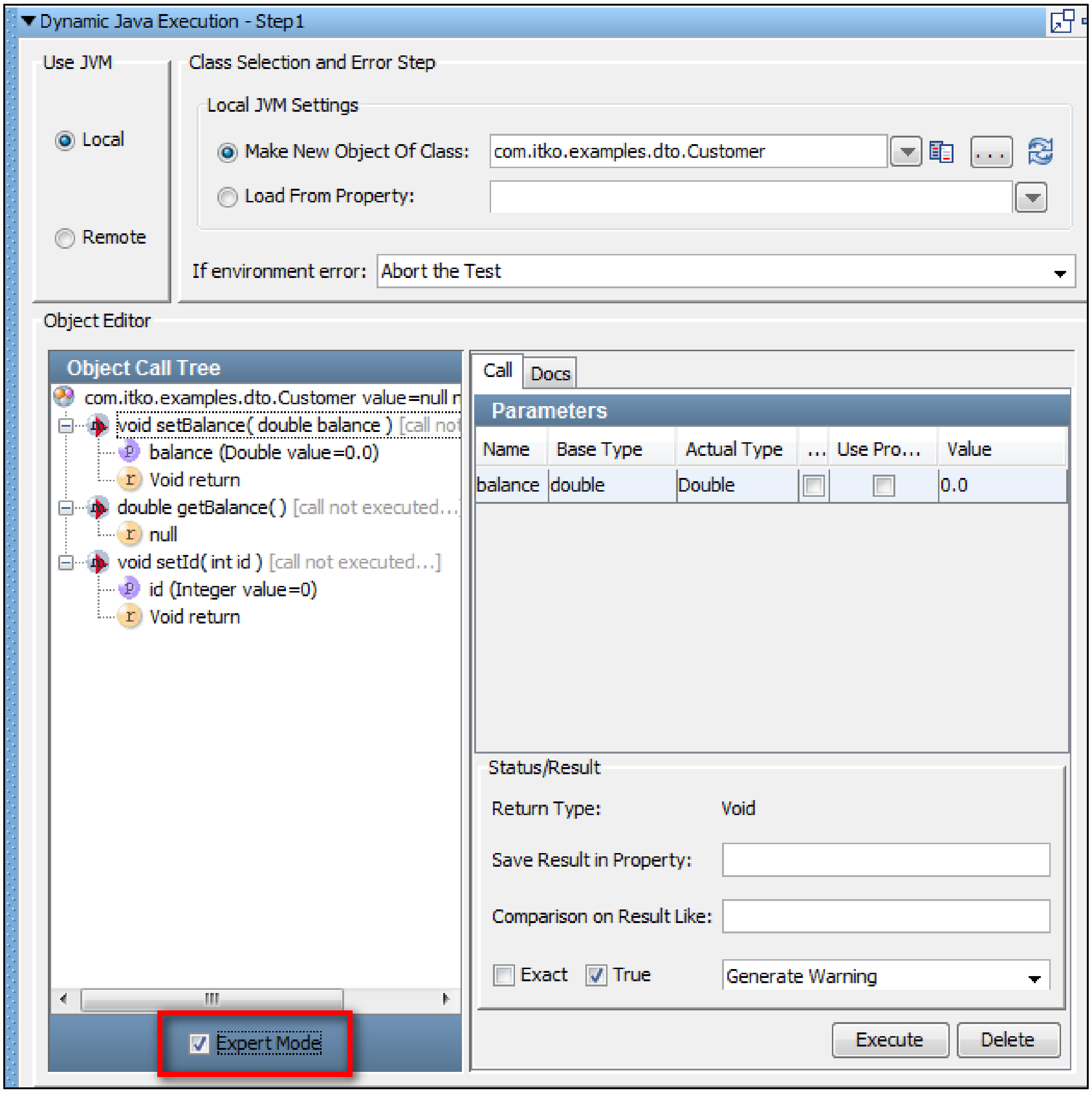
There are two editing modes available: simple and expert.

Simple Mode is useful when your object is a simple object, such as a Java Bean with just a default constructor and several setter/getter methods. This is the classic Data Transfer Object (DTO). These are very common as inputs to web service calls. With objects of this type you can toggle back and forth between simple and expert mode. A DTO that contains a DTO as a property can be manipulated in simple mode. We will see examples of this later.

You must use Expert Mode for more complex objects, such as objects that have multiple constructors. Some composite objects that contain other complex objects cannot use the simple mode, and in fact the simple mode option is disabled if the current object requires expert mode. We shall see examples of using the expert mode later.

All the illustrations shown previously have used simple mode.

The following illustration shows the example shown in the previous illustration, but with expert mode selected.



Dynamic Java Execution - Step1

Use JVM: ● Local ○ Remote

Class Selection and Error Step

Local JVM Settings

● Make New Object Of Class: com.itko.examples.dto.Customer

○ Load From Property:

If environment error: Abort the Test

Object Editor

Object Call Tree

```
com.itko.examples.dto.Customer value=null r
  void setBalance( double balance ) [call not
    balance (Double value=0.0)
    Void return
  double getBalance( ) [call not executed...]
    null
  void setId( int id ) [call not executed...]
    id (Integer value=0)
    Void return
```

☑ Expert Mode

Call | Docs

Parameters

Name	Base Type	Actual Type	...	Use Pro...	Value
balance	double	Double	□	□	0.0

Status/Result

Return Type: Void

Save Result in Property:

Comparison on Result Like:

□ Exact ☑ True — Generate Warning

A new Status/Result panel opens up as shown.

You can now add Inline Filters (Save Result Property In) and Assertions (Comparison On Result Like) in the object editor.

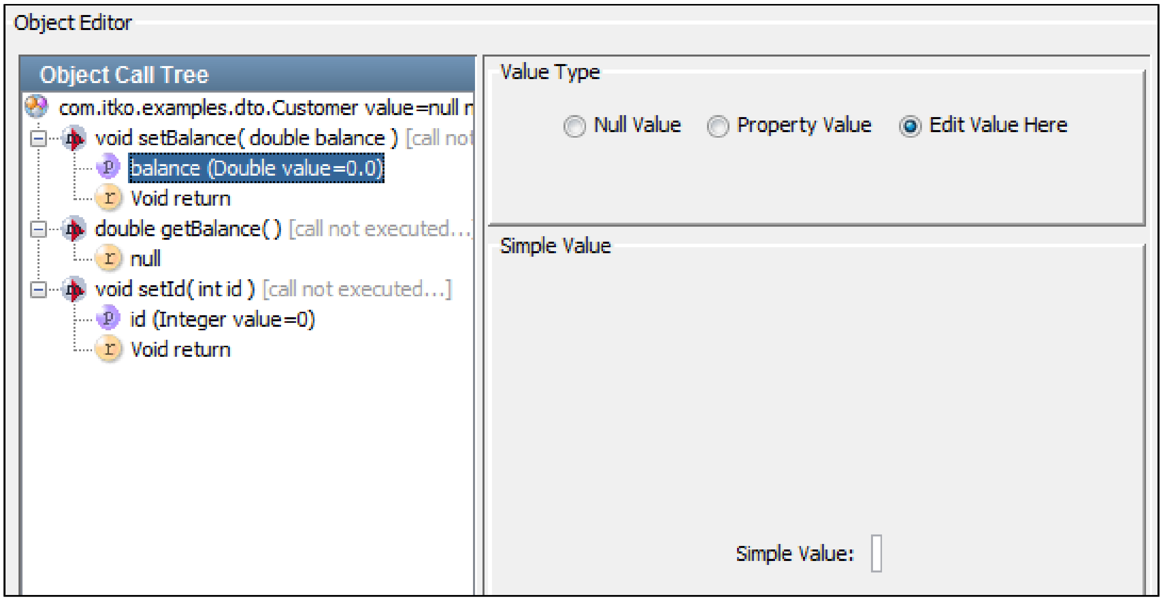
⚠ The in-line filters and assertions that are applied are not seen in the filters or assertions list.

There are several other differences that will become apparent in our later examples.

Input Parameter Panels

If you select an input parameter in the Object Call Tree, the tabs available in the right panel will vary depending on the input parameter type: Primitives/Strings or Objects.

For input parameters that are primitives and strings, the following panel will be displayed.



Object Editor

Object Call Tree

- com.itko.examples.dto.Customer value=null r
  - void setBalance(double balance) [call not
    - balance (Double value=0.0)
    - Void return
  - double getBalance( ) [call not executed...]
    - null
  - void setId(int id) [call not executed...]
    - id (Integer value=0)
    - Void return

Value Type

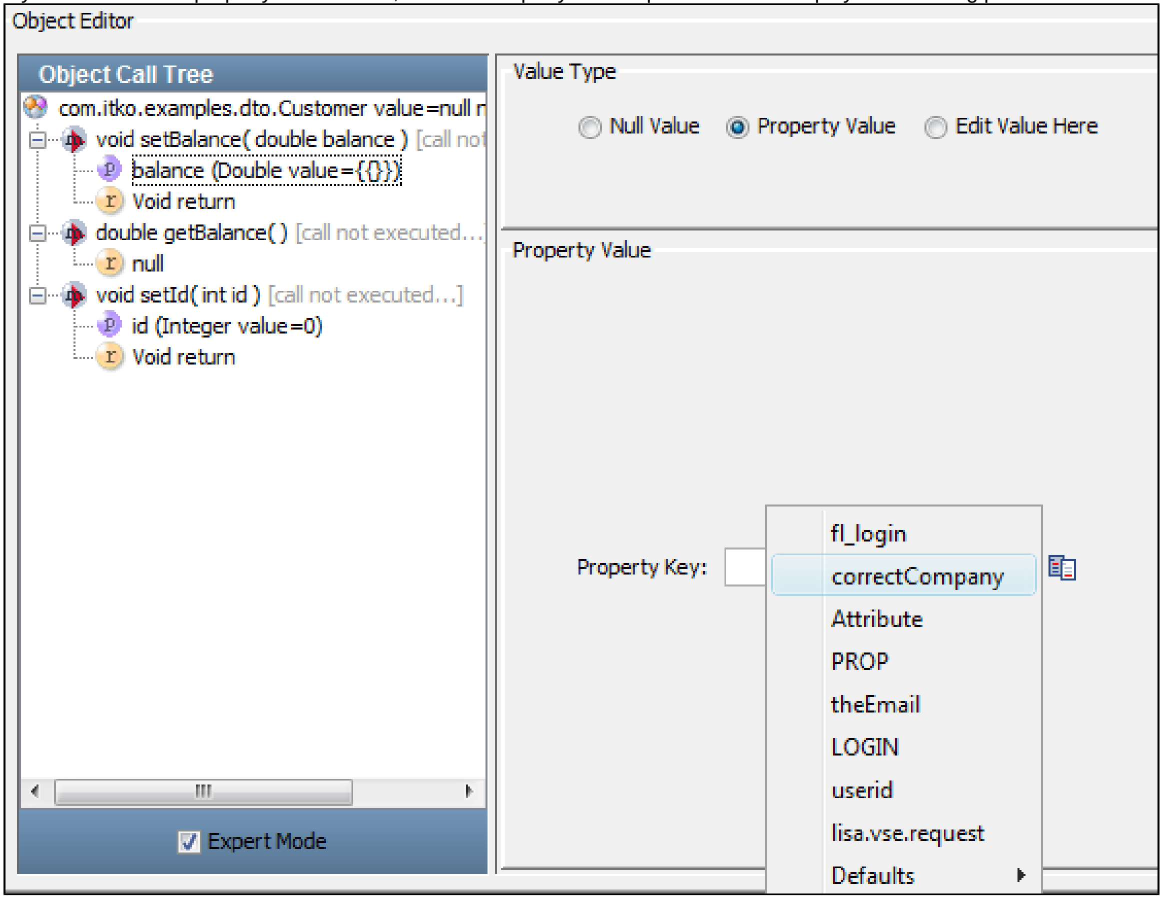
○ Null Value ○ Property Value ● Edit Value Here

Simple Value

Simple Value:

You can edit the value in this panel in Simple Value field.

If you want to use a property as the value, click the Property Value option button to display the following panel.



Object Editor

Object Call Tree

- com.itko.examples.dto.Customer value=null r
  - void setBalance(double balance) [call not
    - balance (Double value={{}})
    - Void return
  - double getBalance( ) [call not executed...
    - null
  - void setId(int id) [call not executed...]
    - id (Integer value=0)
    - Void return

☑ Expert Mode

Value Type

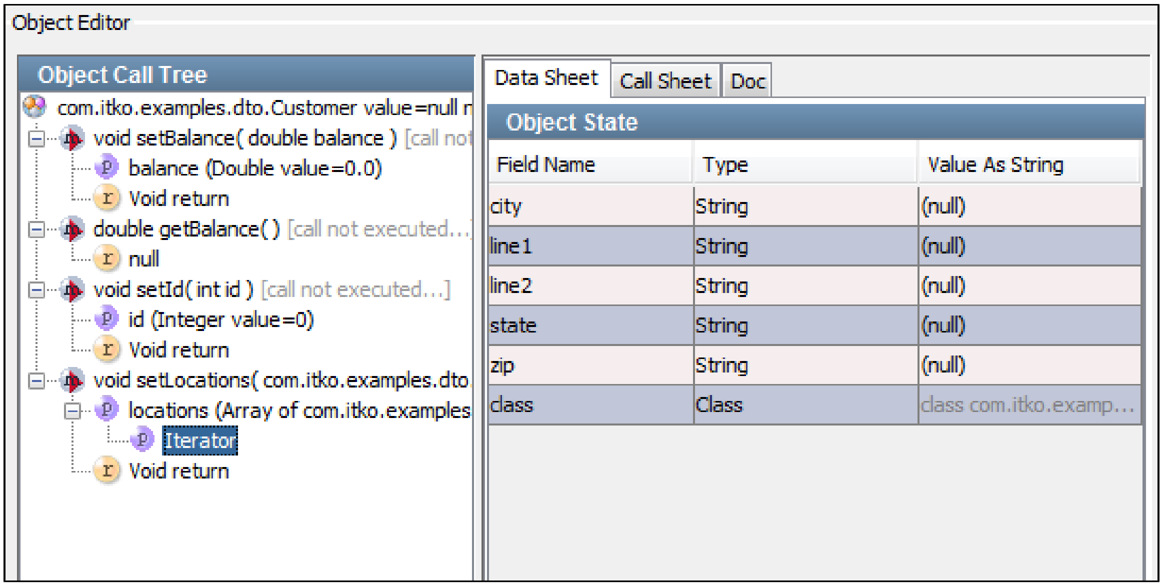
○ Null Value ● Property Value ○ Edit Value Here

Property Value

Property Key:

You can type the property name or use the pull-down menu or click the List icon to open the available property keys.

For input parameters that are objects, the following panel will be displayed.



Object Editor

Object Call Tree

com.itko.examples.dto.Customer value=null r
void setBalance(double balance) [call not
balance (Double value=0.0)
Void return
double getBalance( ) [call not executed...]
null
void setId(int id) [call not executed...]
id (Integer value=0)
Void return
void setLocations(com.itko.examples.dto
locations (Array of com.itko.examples
Iterator
Void return

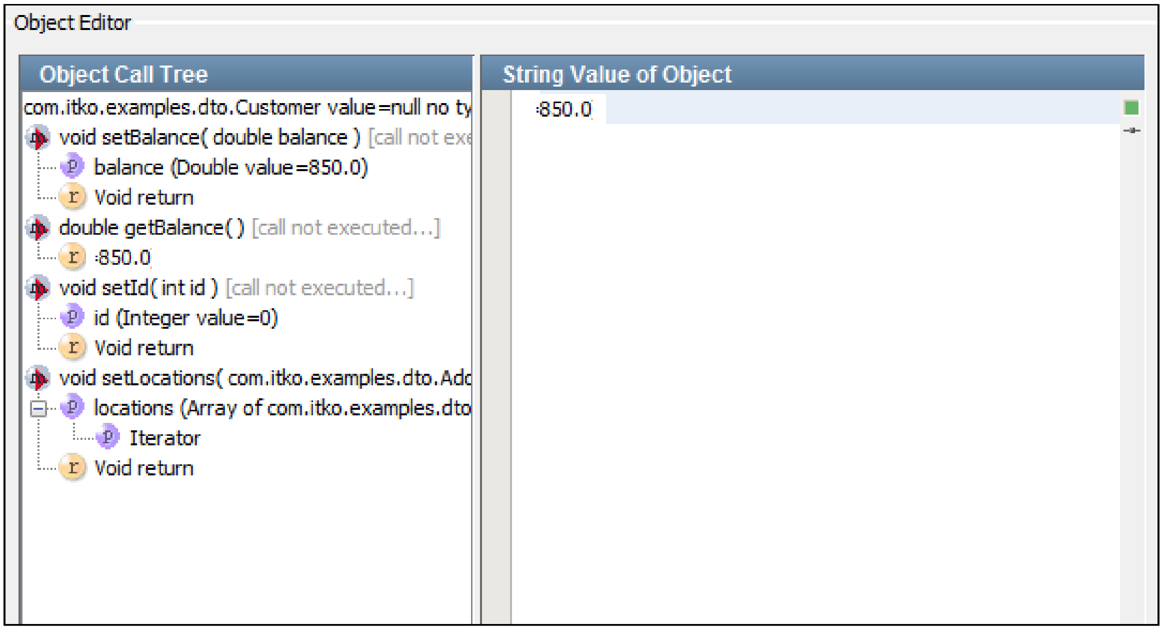
Object State

Field Name	Type	Value As String
city	String	(null)
line1	String	(null)
line2	String	(null)
state	String	(null)
zip	String	(null)
class	Class	class com.itko.examp...

Return Value Panels

If you select a return value in the Object Call Tree, the tabs available in the right panel will vary depending on the input parameter type.

For input parameters that are primitives and strings, the following panel will be displayed.



Object Editor

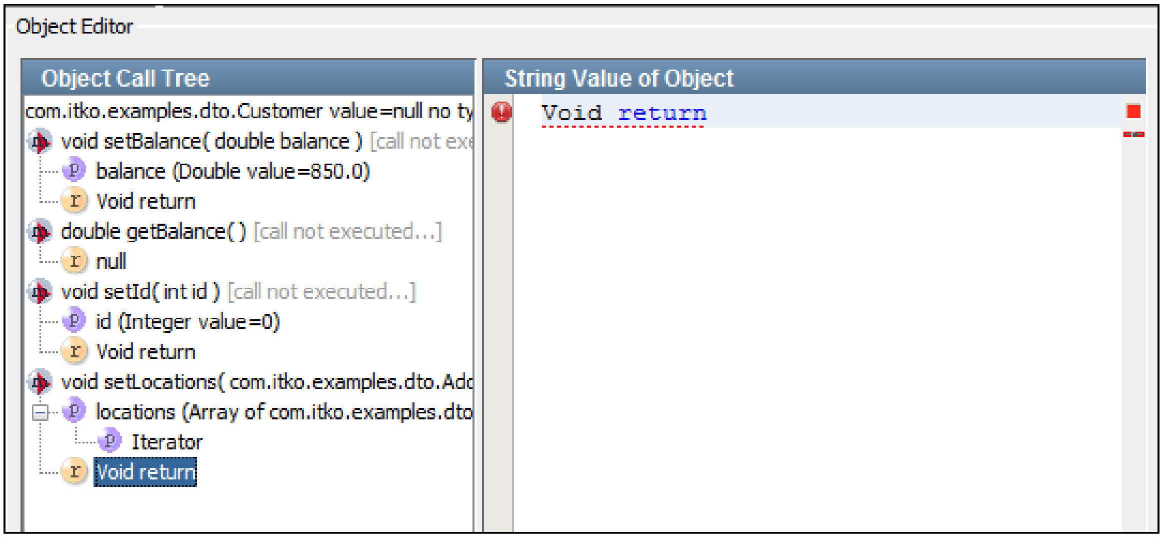
Object Call Tree

com.itko.examples.dto.Customer value=null no ty
void setBalance(double balance) [call not exe
balance (Double value=850.0)
Void return
double getBalance( ) [call not executed...]
850.0
void setId(int id) [call not executed...]
id (Integer value=0)
Void return
void setLocations(com.itko.examples.dto.Add
locations (Array of com.itko.examples.dto
Iterator
Void return

String Value of Object

850.0

For input parameters that are objects, the following panel is displayed.



com.itko.examples.dto.Customer value=null no ty
void setBalance(double balance) [call not exe
balance (Double value=850.0)
Void return
double getBalance( ) [call not executed...]
null
void setId(int id) [call not executed...]
id (Integer value=0)
Void return
void setLocations(com.itko.examples.dto.Add
locations (Array of com.itko.examples.dto
Iterator
Void return

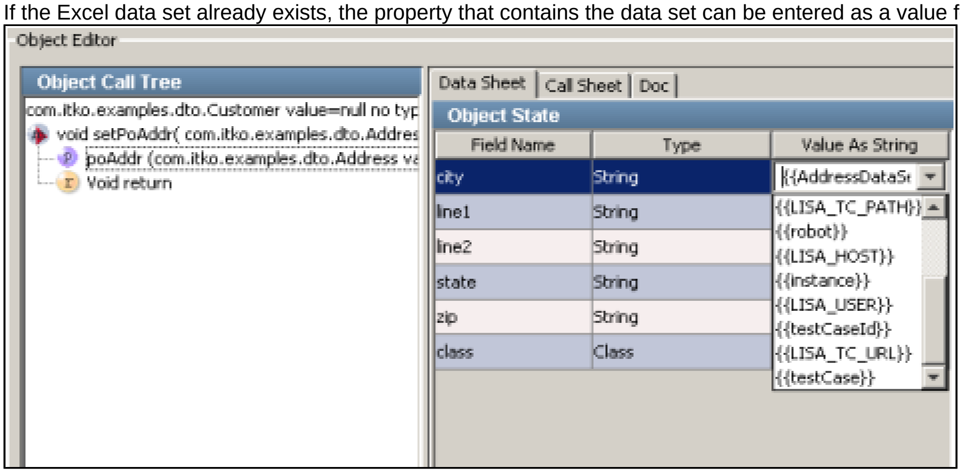
```
Void return
```

Using Data Sets in the COE

A common way to provide data for a Java DTO object is a data set.

A data set, Read DTOs from Excel File, is provided for this purpose.

If the Excel data set already exists, the property that contains the data set can be entered as a value for the DTO object.

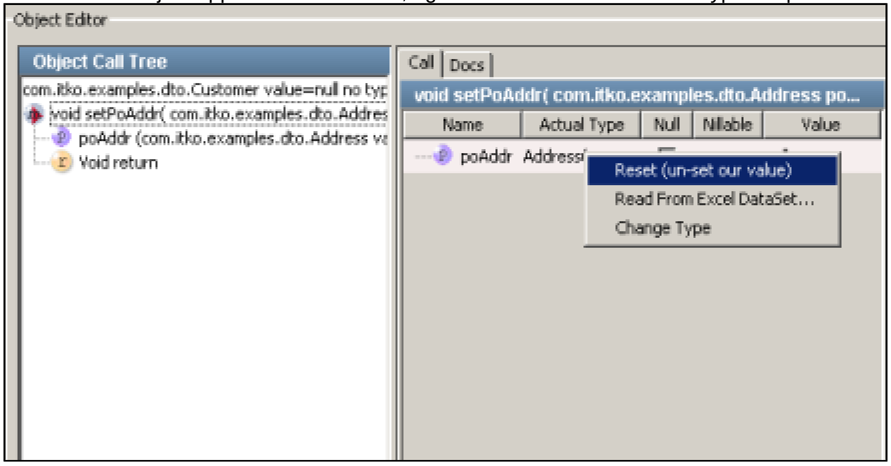


com.itko.examples.dto.Customer value=null no typ
void setPoAddr(com.itko.examples.dto.Addres
poAddr (com.itko.examples.dto.Address va
Void return

Field Name	Type	Value As String
city	String	{{AddressDataSe
line1	String	
line2	String	
state	String	
zip	String	
class	Class	

You can also initiate the creation of a new DTO data set from within the object editor.

1. When a DTO object appears in the call list, right-click the Name or Actual Type to open a menu.

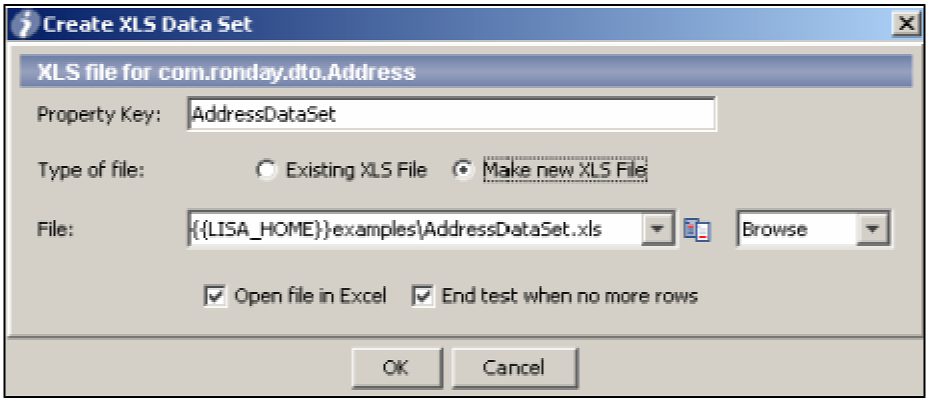


com.itko.examples.dto.Customer value=null no typ
void setPoAddr(com.itko.examples.dto.Addres
poAddr (com.itko.examples.dto.Address va
Void return

void setPoAddr(com.itko.examples.dto.Address po...

Name	Actual Type	Null	Nillable	Value
poAddr	Address			

2. Select the data set Read From Excel Data set to display the following screen:


Create XLS Data Set

XLS file for com.ronday.dto.Address

Property Key: AddressDataSet

Type of file: ○ Existing XLS File ● Make new XLS File

File: {{LISA_HOME}}examples\AddressDataSet.xls

☑ Open file in Excel ☑ End test when no more rows

3. The parameters on this screen are required to initiate the creation of this data set. Enter the following parameters:

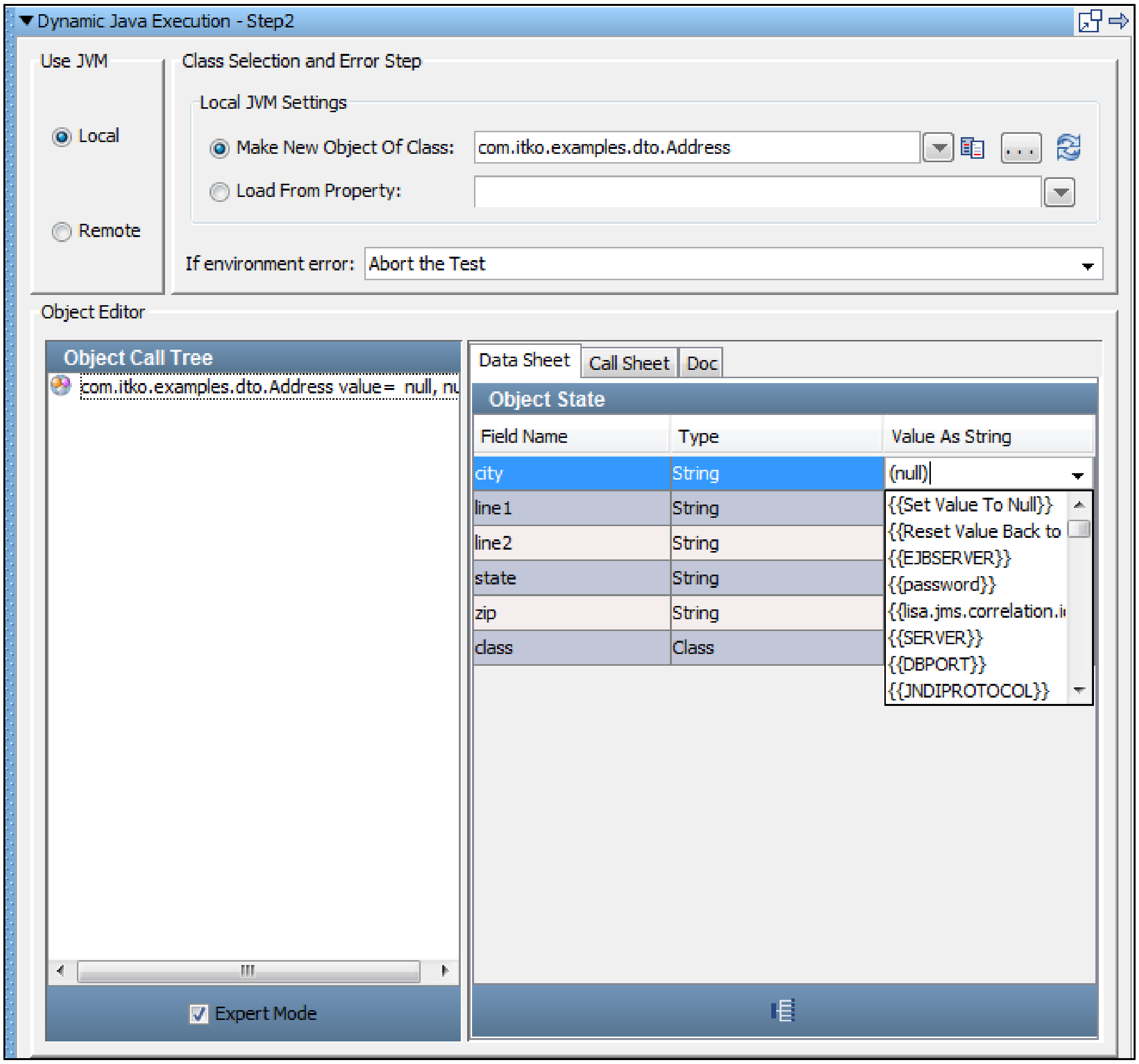
- **Property Key**: The property that stores the current values obtained from the data set.
- **Type of File**: Select if it is an existing XLS file or make a new XLS file.
- **File**: The name of the Excel file that will be the template for the DTO data.
- **Open file in Excel**: Opens the spreadsheet in Excel.
- **End test when no more rows**: Ends the test after all rows have been read.

Usage Scenarios for Simple Objects

The following examples are based on standard Java classes for simple objects. We have used classes from the Demo Server included with LISA. They should be easy to reproduce in your environment.

Simple DTO Object Scenario 1

A simple DTO com.itko.examples.dto.Address has been loaded in the COE using a Dynamic Java Execution step. The Address class has simple properties only.



Dynamic Java Execution - Step2

Use JVM

● Local

○ Remote

Class Selection and Error Step

Local JVM Settings

● Make New Object Of Class: com.itko.examples.dto.Address

○ Load From Property:

If environment error: Abort the Test

Object Editor

Object Call Tree

com.itko.examples.dto.Address value= null, nu

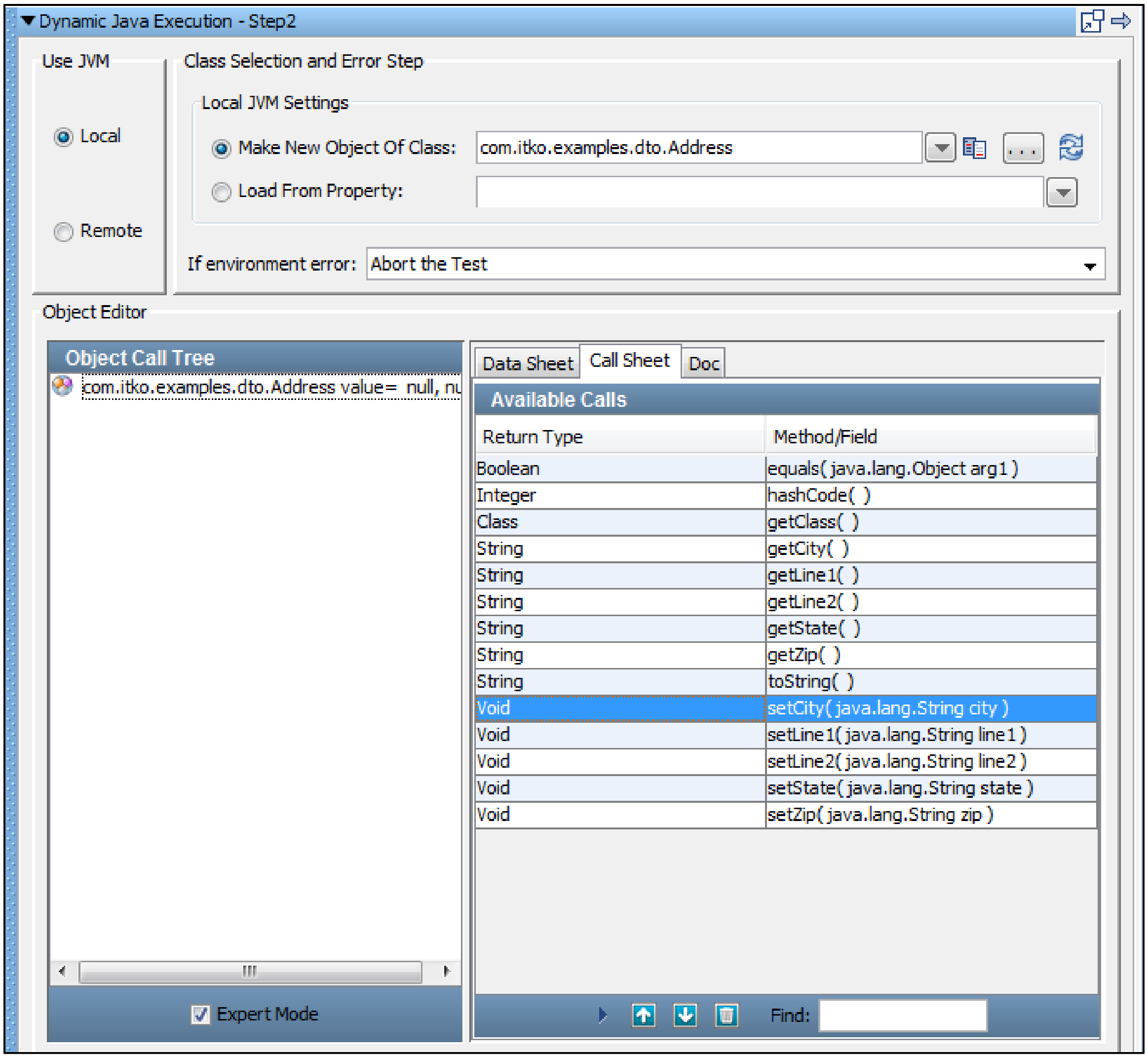
Object State

Field Name	Type	Value As String
city	String	(null)
line1	String	
line2	String	
state	String	
zip	String	
class	Class	

☑ Expert Mode

For a simple DTO, parameter values can be entered into the Data Sheet panel as fixed values or properties. In the previous example, *city* could be set equal to the property `currentCity`.

Parameters can also be entered using the DTO setters in the Call sheet panel.



Use JVM

● Local

○ Remote

Class Selection and Error Step

Local JVM Settings

● Make New Object Of Class: com.itko.examples.dto.Address

○ Load From Property:

If environment error: Abort the Test

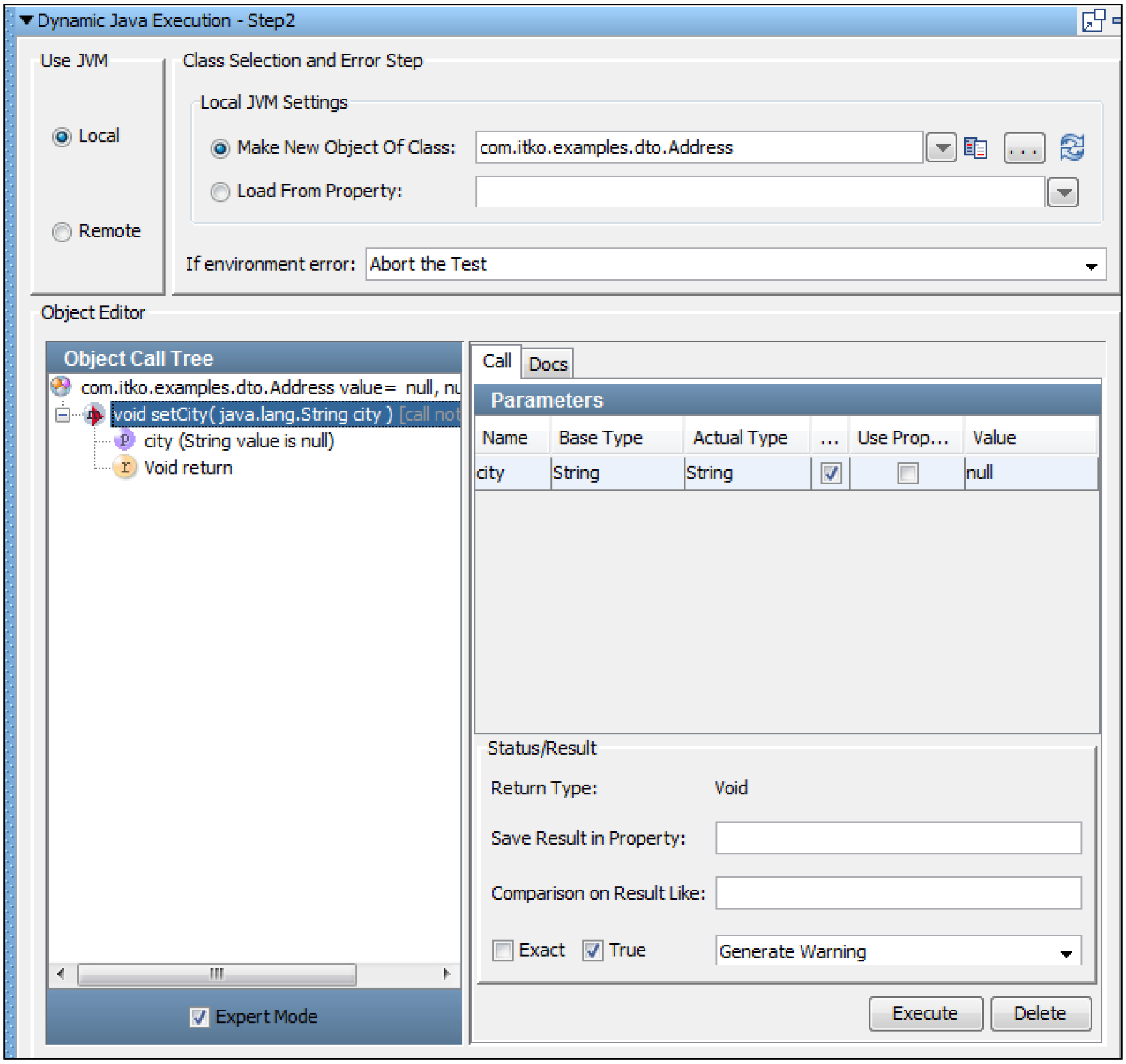
Object Editor

com.itko.examples.dto.Address value= null, nu

☑ Expert Mode

Return Type	Method/Field
Boolean	equals(java.lang.Object arg1)
Integer	hashCode( )
Class	getClass( )
String	getCity( )
String	getLine1( )
String	getLine2( )
String	getState( )
String	getZip( )
String	toString( )
Void	setCity(java.lang.String city)
Void	setLine1(java.lang.String line1)
Void	setLine2(java.lang.String line2)
Void	setState(java.lang.String state)
Void	setZip(java.lang.String zip)

In the Call Sheet tab, select a getter, such as setCity(java.lang.String city) and click the Add Method ▶ icon. The method will run and COE will open in the Call tab.



Dynamic Java Execution - Step2

Use JVM

● Local

○ Remote

Class Selection and Error Step

Local JVM Settings

● Make New Object Of Class: com.itko.examples.dto.Address

○ Load From Property:

If environment error: Abort the Test

Object Editor

Object Call Tree

com.itko.examples.dto.Address value= null, nu

void setCity(java.lang.String city) [call not

city (String value is null)

Void return

Parameters

Name	Base Type	Actual Type	...	Use Prop...	Value
city	String	String	☑	□	null

Status/Result

Return Type: Void

Save Result in Property:

Comparison on Result Like:

□ Exact ☑ True Generate Warning

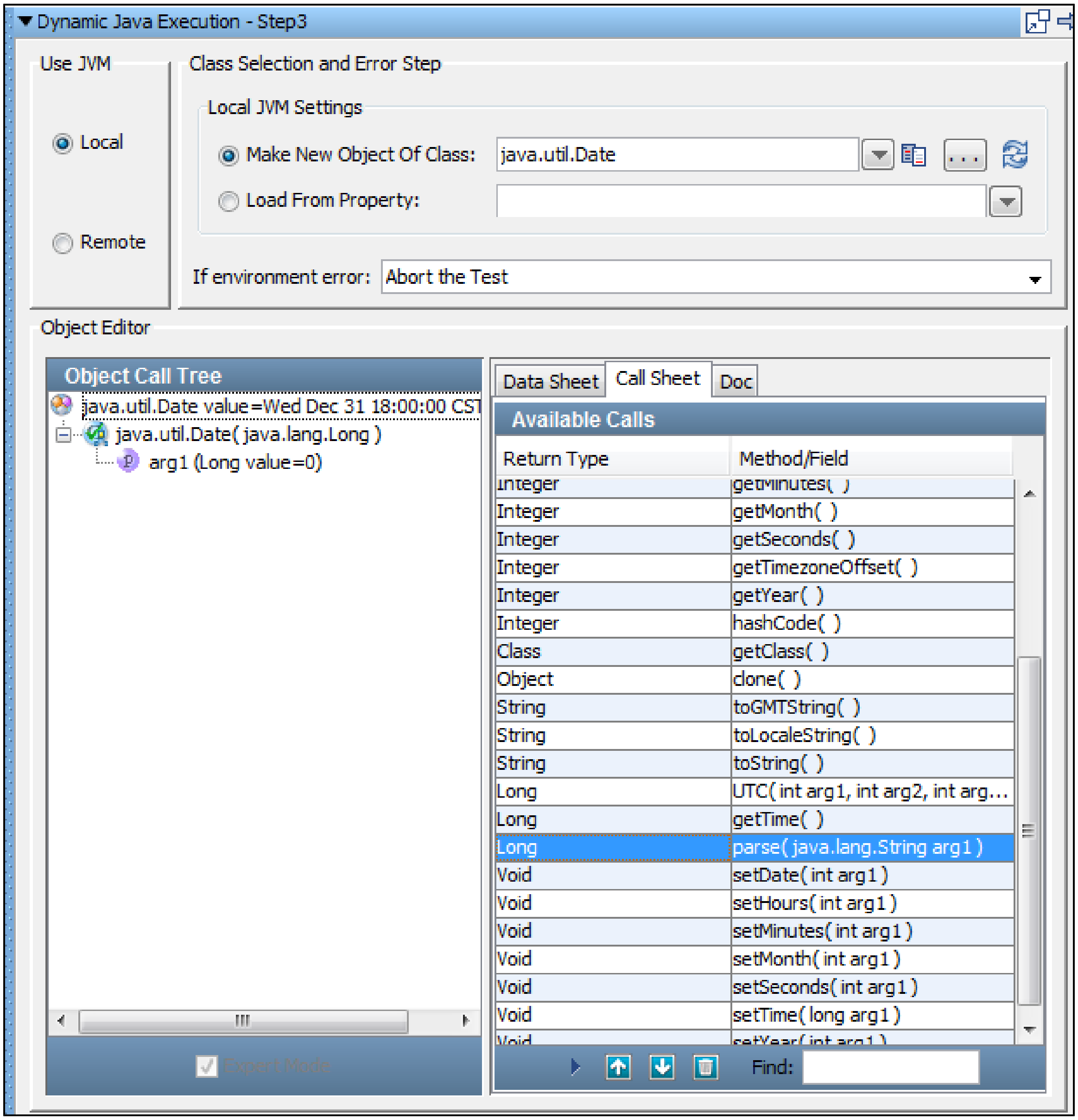
☑ Expert Mode

- Enter the parameter value as a fixed value or a property. You must use the LISA property syntax here, **`propname`**.
- Click the Execute button to invoke the method.

Repeat this procedure to set other DTO properties.

Simple Java Object Scenario 2

A simple Java object, java.util.Date, has been loaded in the COE using a Dynamic Java Execution step.



▼ Dynamic Java Execution - Step3

Use JVM

● Local

○ Remote

Class Selection and Error Step

Local JVM Settings

● Make New Object Of Class: java.util.Date

○ Load From Property:

If environment error: Abort the Test

Object Editor

Object Call Tree

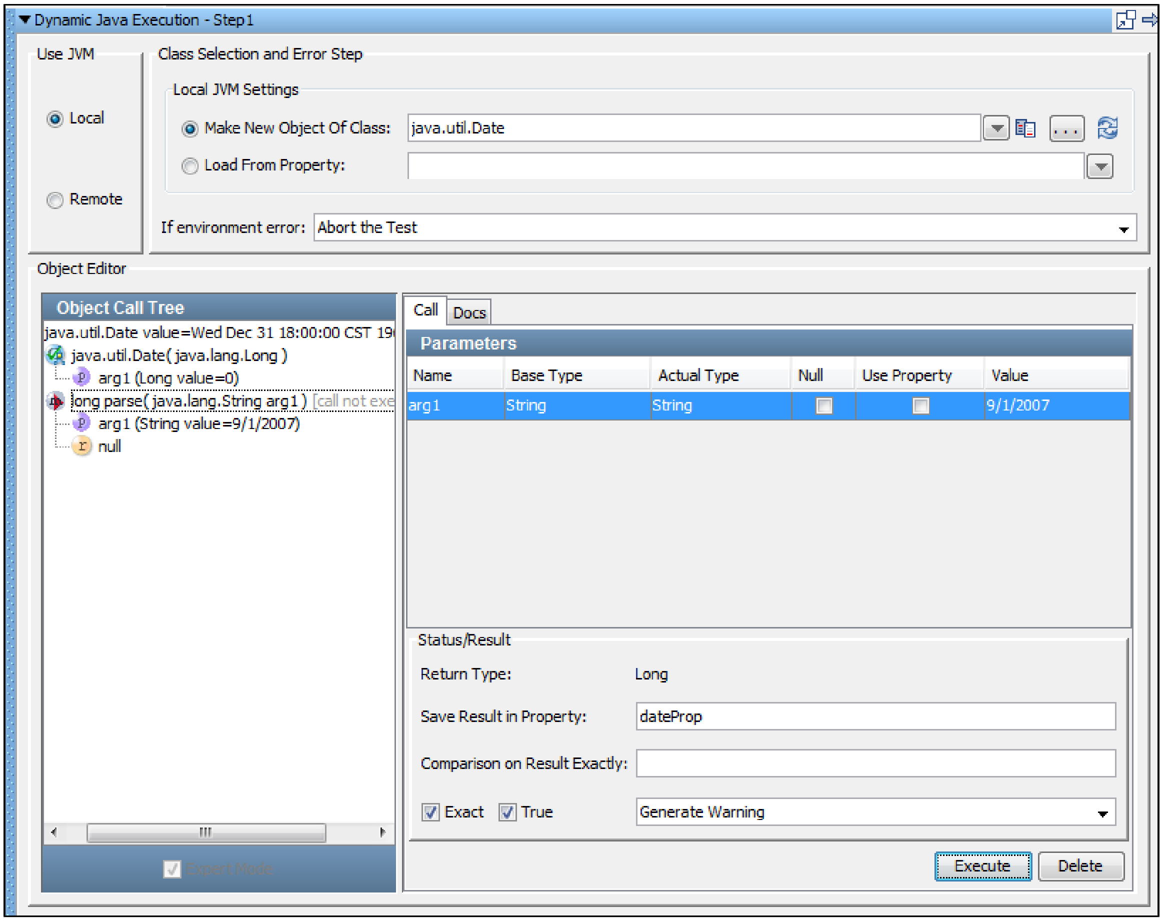
- java.util.Date value=Wed Dec 31 18:00:00 CST
  - java.util.Date(java.lang.Long)
    - arg1 (Long value=0)

Data Sheet | Call Sheet | Doc

Available Calls

Return Type	Method/Field
Integer	getMinutes( )
Integer	getMonth( )
Integer	getSeconds( )
Integer	getTimezoneOffset( )
Integer	getYear( )
Integer	hashCode( )
Class	getClass( )
Object	clone( )
String	toGMTString( )
String	toLocaleString( )
String	toString( )
Long	UTC(int arg1, int arg2, int arg...
Long	getTime( )
Long	parse(java.lang.String arg1)
Void	setDate(int arg1)
Void	setHours(int arg1)
Void	setMinutes(int arg1)
Void	setMonth(int arg1)
Void	setSeconds(int arg1)
Void	setTime(long arg1)
Void	setYear(int arg1)

In this case we must stay in Expert Mode, because the Date type is not a DTO. In fact the COE forces us into Expert mode.



Dynamic Java Execution - Step1

Use JVM: ● Local ○ Remote

Class Selection and Error Step

Local JVM Settings

● Make New Object Of Class: java.util.Date

○ Load From Property:

If environment error: Abort the Test

Object Editor

Object Call Tree

```
java.util.Date value=Wed Dec 31 18:00:00 CST 19
  java.util.Date( java.lang.Long )
    arg1 (Long value=0)
  long parse( java.lang.String arg1 ) [call not exe
    arg1 (String value=9/1/2007)
    null
```

☑ Expert Mode

Call | Docs

Parameters

Name	Base Type	Actual Type	Null	Use Property	Value
arg1	String	String	□	□	9/1/2007

Status/Result

Return Type: Long

Save Result in Property: dateProp

Comparison on Result Exactly:

☑ Exact ☑ True Generate Warning

Execute Delete

But we can still enter parameter values and invoke methods. In the previous example we execute the parse method, which requires one input parameter. We have entered it as a string value, 9/1/2007.

> ⚠ In expert mode we use the Null or Use Property check box to denote the parameter type. If neither is checked will enter a fixed value. We would not use the {{ }} property syntax here. Even if we were entering a property rather than a string value, we would enter only the property name.

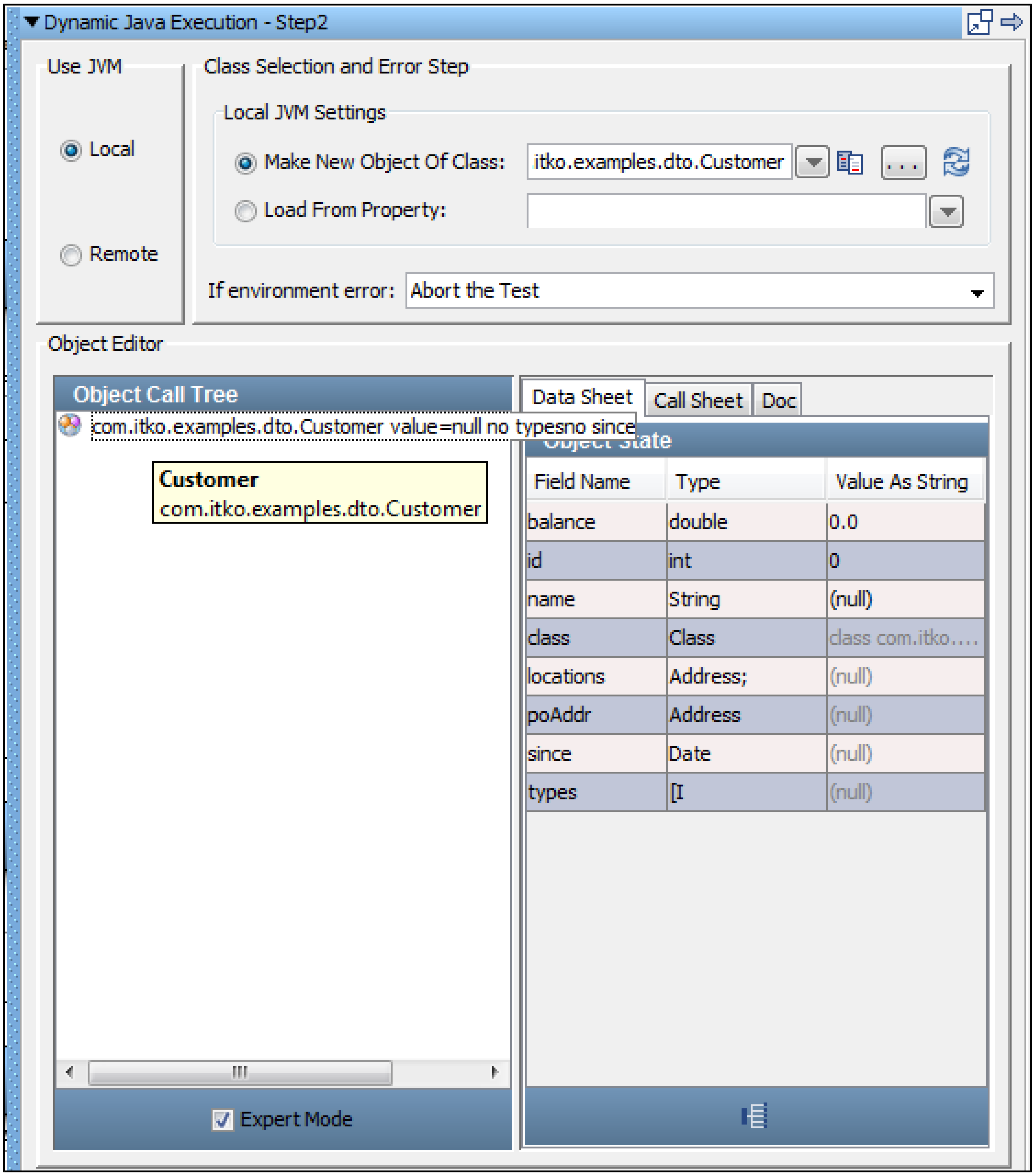
Because we are in Expert Mode we have the opportunity here to add inline filters and assertions.

Usage Scenarios for Complex Objects

The following examples are based on standard Java classes for complex objects. We have used classes from the Demo Server included with LISA. You should have no difficulty reproducing them in your environment.

Complex DTO Object Scenario 1

A complex DTO Object, **com.itko.examples.dto.Customer**, has been loaded in the Complex Object Editor using a **Dynamic Java Execution** step.



● Local

○ Remote

● Make New Object Of Class: itko.examples.dto.Customer

○ Load From Property:

If environment error: Abort the Test

com.itko.examples.dto.Customer value=null no typesno since

Customer
com.itko.examples.dto.Customer

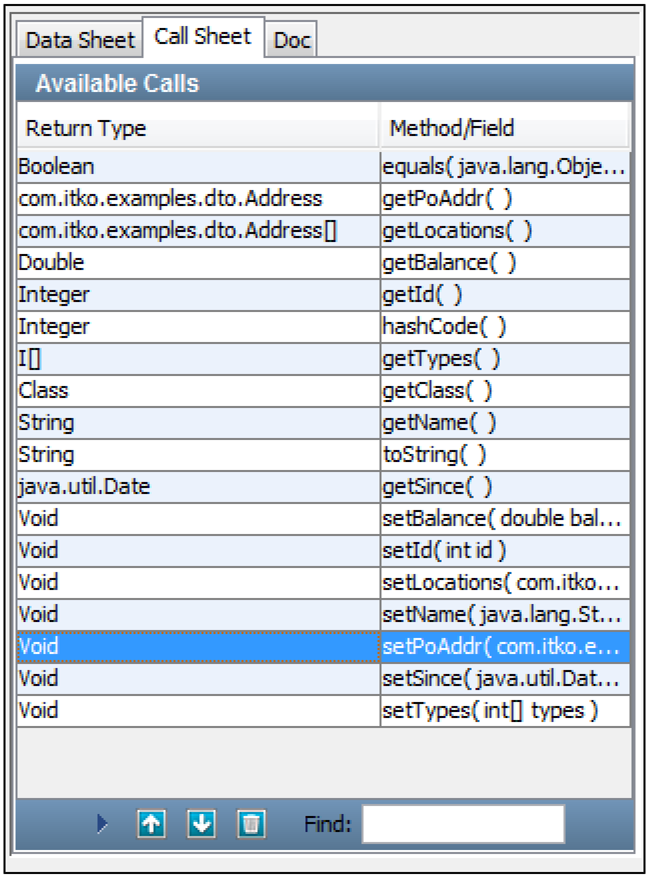
Field Name	Type	Value As String
balance	double	0.0
id	int	0
name	String	(null)
class	Class	class com.itko....
locations	Address;	(null)
poAddr	Address	(null)
since	Date	(null)
types	[I	(null)

This DTO is complex because its properties are not all simple values such as primitives or strings. However, because of its DTO structure we can still use **Simple Mode** here if we want.

Each of the properties must be given values before the Customer object can be used.

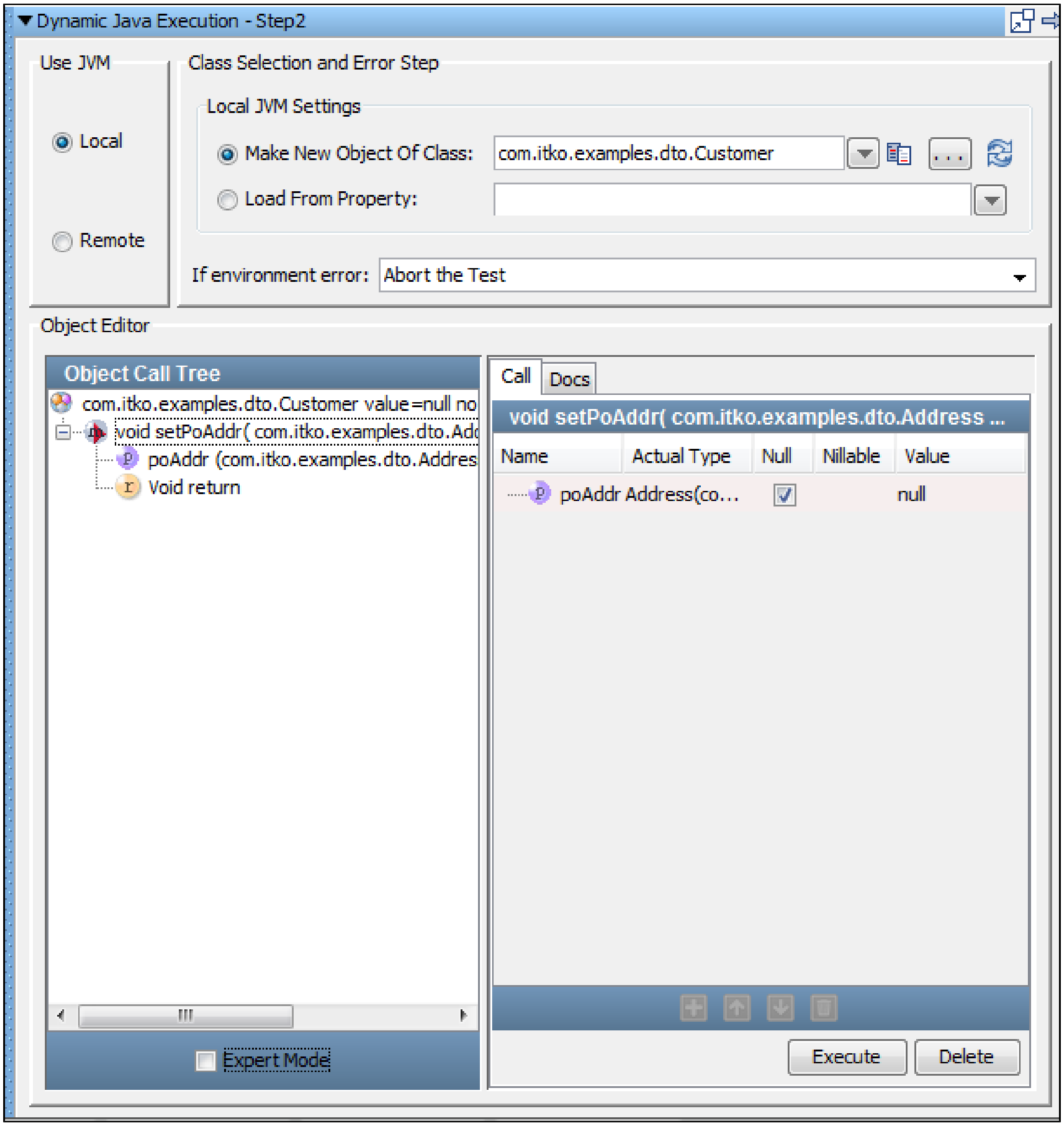
- **locations**: An array of Address objects
- **poAddr**: An Address object
- **since**: A Java Date object
- **types**: An array of integers

1. Starting with the **poAddr** object, identify the **setPoAddr** method in the **Call Sheet**.



Return Type	Method/Field
Boolean	equals(java.lang.Obje…
com.itko.examples.dto.Address	getPoAddr( )
com.itko.examples.dto.Address[]	getLocations( )
Double	getBalance( )
Integer	getId( )
Integer	hashCode( )
I[]	getTypes( )
Class	getClass( )
String	getName( )
String	toString( )
java.util.Date	getSince( )
Void	setBalance(double bal…
Void	setId(int id)
Void	setLocations(com.itko…
Void	setName(java.lang.St…
Void	setPoAddr(com.itko.e…
Void	setSince(java.util.Dat…
Void	setTypes(int[] types)

2. Select the **setPoAddr** method and double-click or click the **Add Method**. ▶ icon to invoke/run this method.



Dynamic Java Execution - Step2

Use JVM

● Local

○ Remote

Class Selection and Error Step

Local JVM Settings

● Make New Object Of Class: com.itko.examples.dto.Customer

○ Load From Property:

If environment error: Abort the Test

Object Editor

Object Call Tree

com.itko.examples.dto.Customer value=null no
void setPoAddr(com.itko.examples.dto.Add
poAddr (com.itko.examples.dto.Addres
Void return

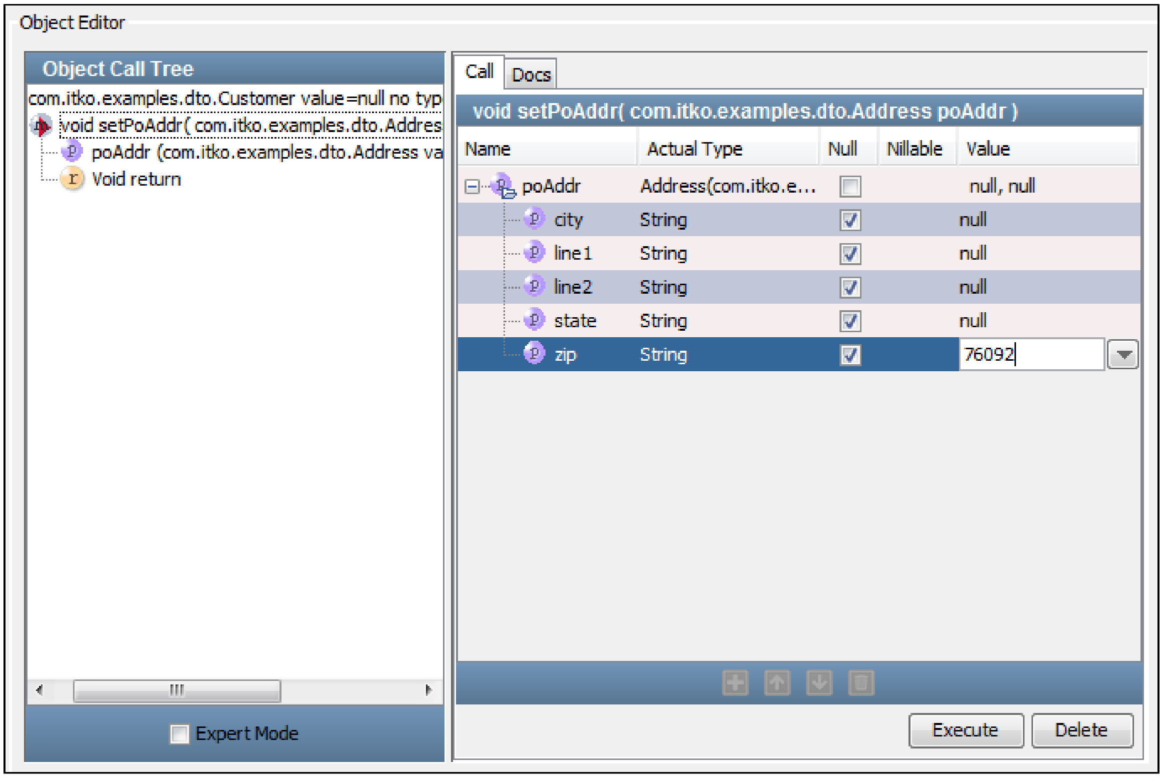
□ Expert Mode

Call | Docs

void setPoAddr(com.itko.examples.dto.Address ...

Name	Actual Type	Null	Nillable	Value
poAddr	Address(co...	☑		null

3. This is a DTO that enables the use of **Simple** mode, so do not select the **Expert Mode** check box. The **poAddress** property is identified as type Address.
4. In a **Simple mode**, when you clear the **Null** parameter, the Address object is expanded to expose its properties. You know, from the previously illustrated Simple Data Object Scenario, that Address has simple properties, so you can enter them as values or LISA properties in the **Value** column.



com.itko.examples.dto.Customer value=null no typ
void setPoAddr(com.itko.examples.dto.Addres
poAddr (com.itko.examples.dto.Address va
Void return

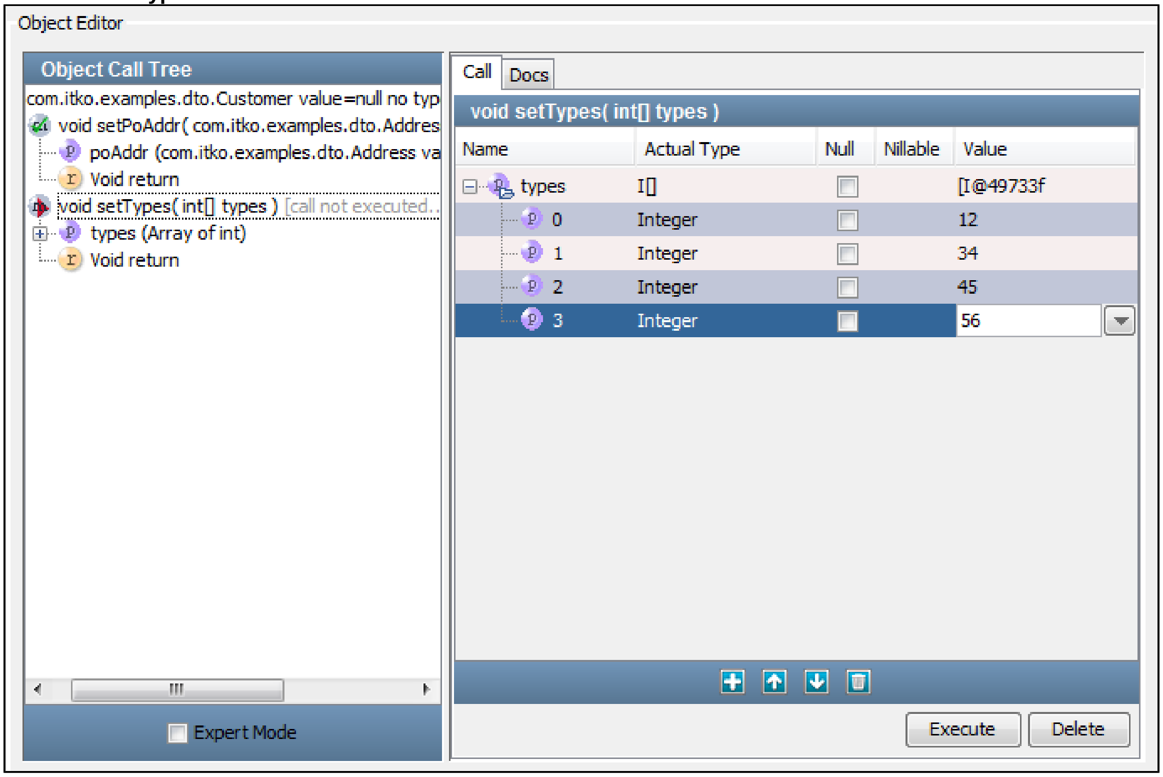
☐ Expert Mode

void setPoAddr(com.itko.examples.dto.Address poAddr)

Name	Actual Type	Null	Nillable	Value
poAddr	Address(com.itko.e...	☐		null, null
city	String	☑		null
line1	String	☑		null
line2	String	☑		null
state	String	☑		null
zip	String	☑		76092

5. Click the **Execute** button to invoke the **setPoAddr** method.

6. Select the **setTypes** method on the **Call Sheet** and click the **Invoke Method** ▶ icon.



com.itko.examples.dto.Customer value=null no typ
void setPoAddr(com.itko.examples.dto.Addres
poAddr (com.itko.examples.dto.Address va
Void return
void setTypes(int[] types) [call not executed..
types (Array of int)
Void return

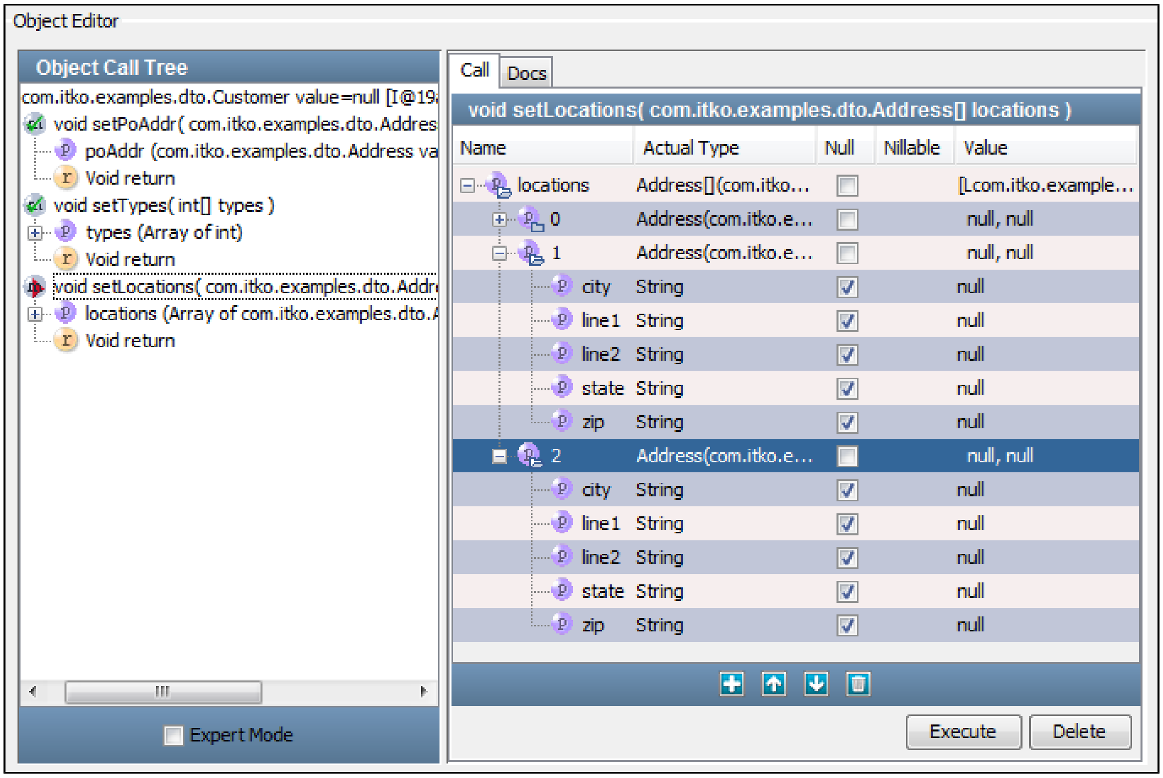
☐ Expert Mode

void setTypes(int[] types)

Name	Actual Type	Null	Nillable	Value
types	I[]	☐		[I@49733f
0	Integer	☐		12
1	Integer	☐		34
2	Integer	☐		45
3	Integer	☐		56

7. "Types" is an array of integers, so it is required that you click the **Add** ⊞ icon at the bottom, to add as many elements in the array as needed. In the previous example we added four elements, and entered values for each.
8. Click the **Execute** button to invoke the **setTypes** method.

9. Select the **setLocations** method on the **Call Sheet** and click the **Invoke Method** ▶ icon.



Object Editor

Object Call Tree

```
com.itko.examples.dto.Customer value=null [I@19
  void setPoAddr( com.itko.examples.dto.Addres
    poAddr (com.itko.examples.dto.Address va
    Void return
  void setTypes( int[] types )
    types (Array of int)
    Void return
  void setLocations( com.itko.examples.dto.Addr
    locations (Array of com.itko.examples.dto.A
    Void return
```

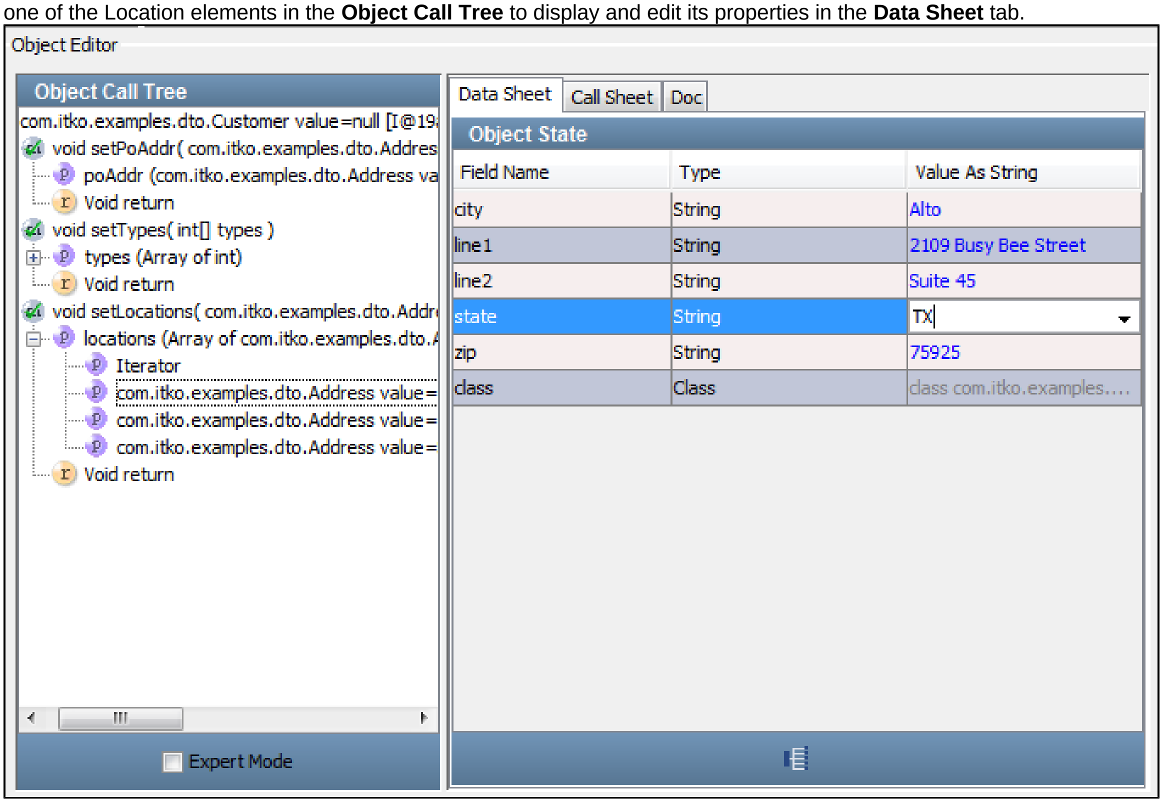
□ Expert Mode

Call | Docs

void setLocations(com.itko.examples.dto.Address[] locations)

Name	Actual Type	Null	Nillable	Value
locations	Address[](com.itko…	□		[Lcom.itko.example…
0	Address(com.itko.e…	□		null, null
1	Address(com.itko.e…	□		null, null
city	String	☑		null
line1	String	☑		null
line2	String	☑		null
state	String	☑		null
zip	String	☑		null
2	Address(com.itko.e…	□		null, null
city	String	☑		null
line1	String	☑		null
line2	String	☑		null
state	String	☑		null
zip	String	☑		null

10. "Locations" is an array of Address objects, so click the **Add** ⊞ icon to add as many elements (of type Address) as needed.
11. In the previous example we added three Address objects. Two have been complete; we are about to expand the third Address object to enter values for the properties. When complete, click the **Execute** button to invoke the **setLocations** method. Notice that you can click one of the Location elements in the **Object Call Tree** to display and edit its properties in the **Data Sheet** tab.



Object Editor

Object Call Tree

```
com.itko.examples.dto.Customer value=null [I@19
  void setPoAddr( com.itko.examples.dto.Addres
    poAddr (com.itko.examples.dto.Address va
    Void return
  void setTypes( int[] types )
    types (Array of int)
    Void return
  void setLocations( com.itko.examples.dto.Addr
    locations (Array of com.itko.examples.dto.A
      Iterator
      com.itko.examples.dto.Address value=
      com.itko.examples.dto.Address value=
      com.itko.examples.dto.Address value=
    Void return
```

□ Expert Mode

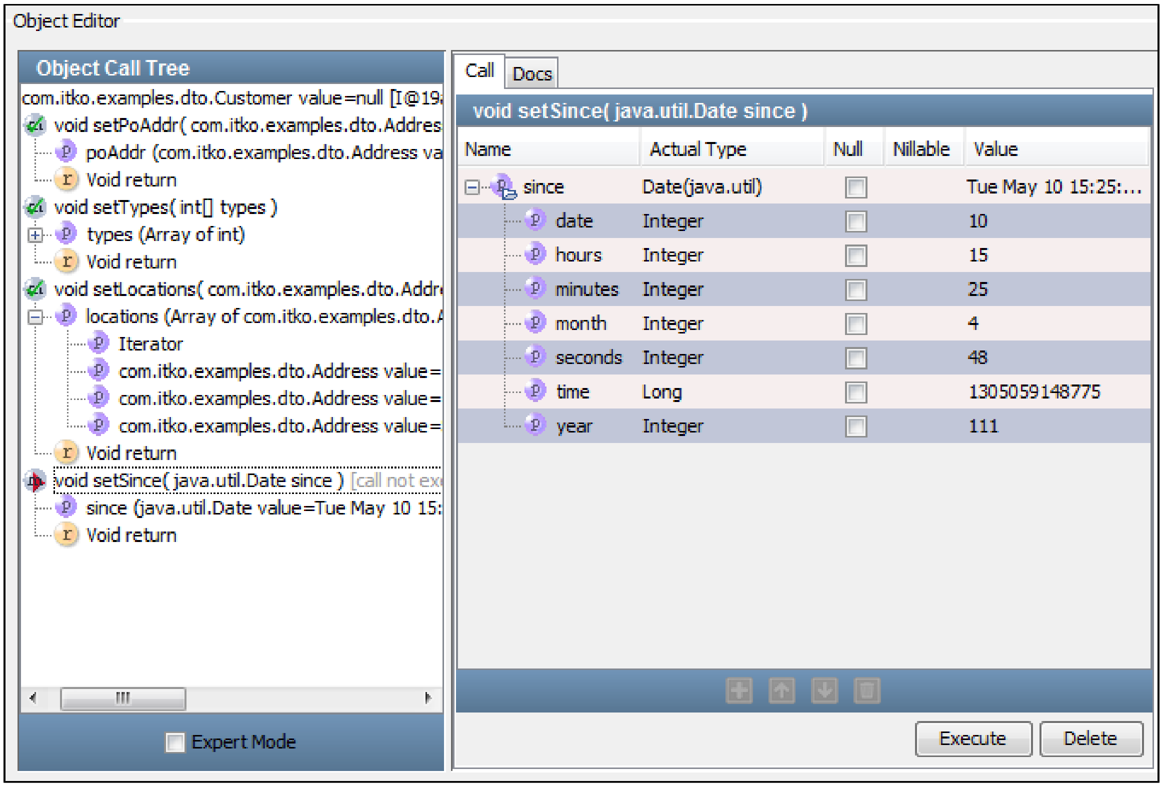
Data Sheet | Call Sheet | Doc

Object State

Field Name	Type	Value As String
city	String	Alto
line1	String	2109 Busy Bee Street
line2	String	Suite 45
state	String	TX
zip	String	75925
class	Class	class com.itko.examples.…

This holds true for all the properties listed in the **Object Call** tree.

12. Select the **getSince** method on the **Call Sheet** and click the **Invoke Method** ▶ icon.



com.itko.examples.dto.Customer value=null [I@19
- void setPoAddr(com.itko.examples.dto.Addres
  - poAddr (com.itko.examples.dto.Address va
  - Void return
- void setTypes(int[] types)
  - types (Array of int)
  - Void return
- void setLocations(com.itko.examples.dto.Addr
  - locations (Array of com.itko.examples.dto.A
    - Iterator
    - com.itko.examples.dto.Address value=
    - com.itko.examples.dto.Address value=
    - com.itko.examples.dto.Address value=
  - Void return
- void setSince(java.util.Date since) [call not ex
  - since (java.util.Date value=Tue May 10 15:
  - Void return

void setSince(java.util.Date since)

Name	Actual Type	Null	Nillable	Value
since	Date(java.util)	□		Tue May 10 15:25:...
date	Integer	□		10
hours	Integer	□		15
minutes	Integer	□		25
month	Integer	□		4
seconds	Integer	□		48
time	Long	□		1305059148775
year	Integer	□		111

13. The input parameters for the Data object are displayed and can be given values. Click the **Execute** button.

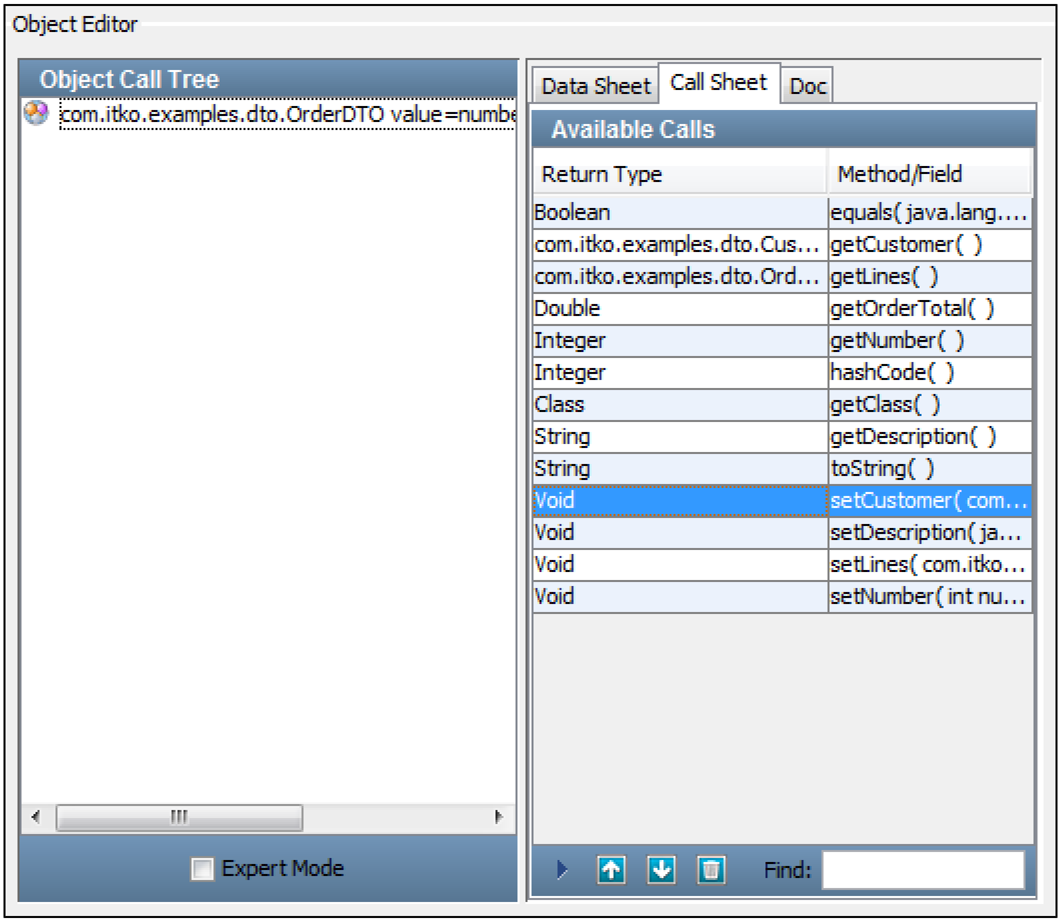
The Customer object is now fully specified and can be used in your test case.

Complex DTO Object Scenario 2

The last scenario shows an example that builds on the last three scenarios.

This DTO, **com.itko.examples.dto.OrderDTO**, has a Customer object as one of its properties. This scenario shows how easy it is to build a Customer object in simple mode without calling any setter methods.

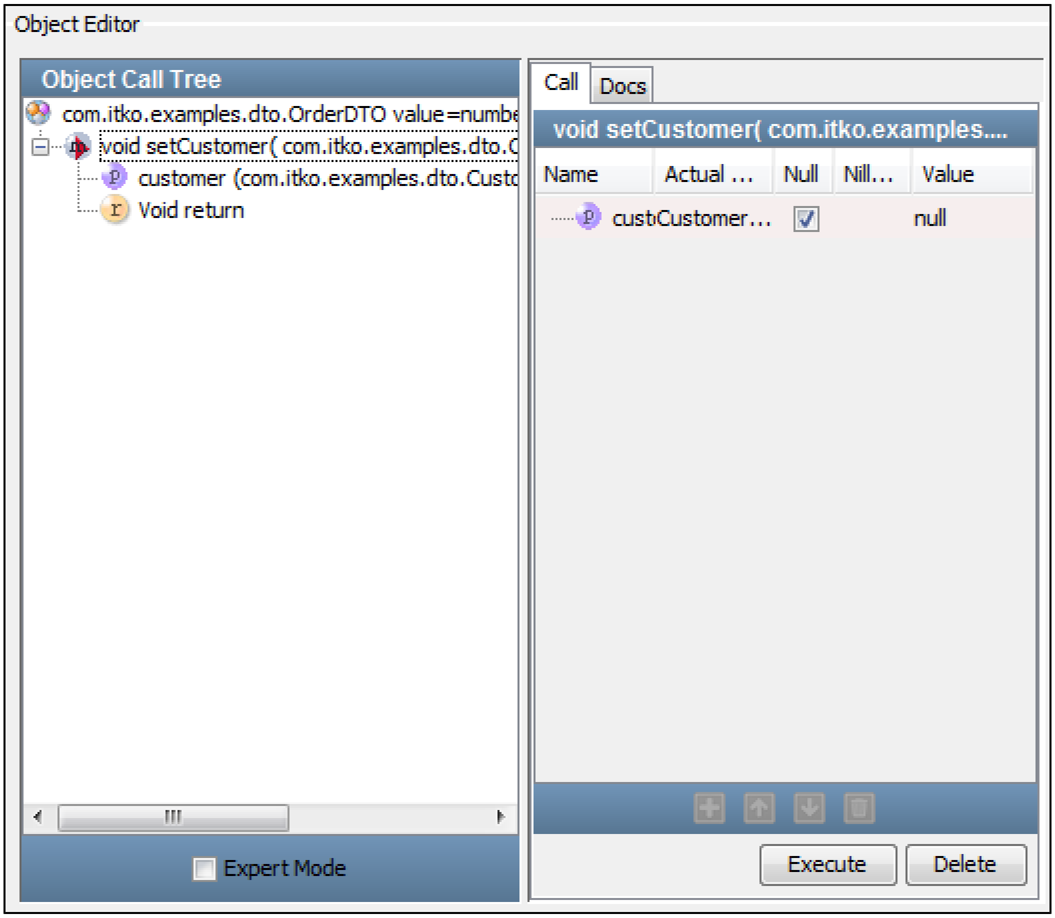
The **OrderDTO** object has been loaded in COE using a **Dynamic Java Execution** step.



Return Type	Method/Field
Boolean	equals(java.lang.…
com.itko.examples.dto.Cus…	getCustomer( )
com.itko.examples.dto.Ord…	getLines( )
Double	getOrderTotal( )
Integer	getNumber( )
Integer	hashCode( )
Class	getClass( )
String	getDescription( )
String	toString( )
Void	setCustomer(com…
Void	setDescription(ja…
Void	setLines(com.itko…
Void	setNumber(int nu…

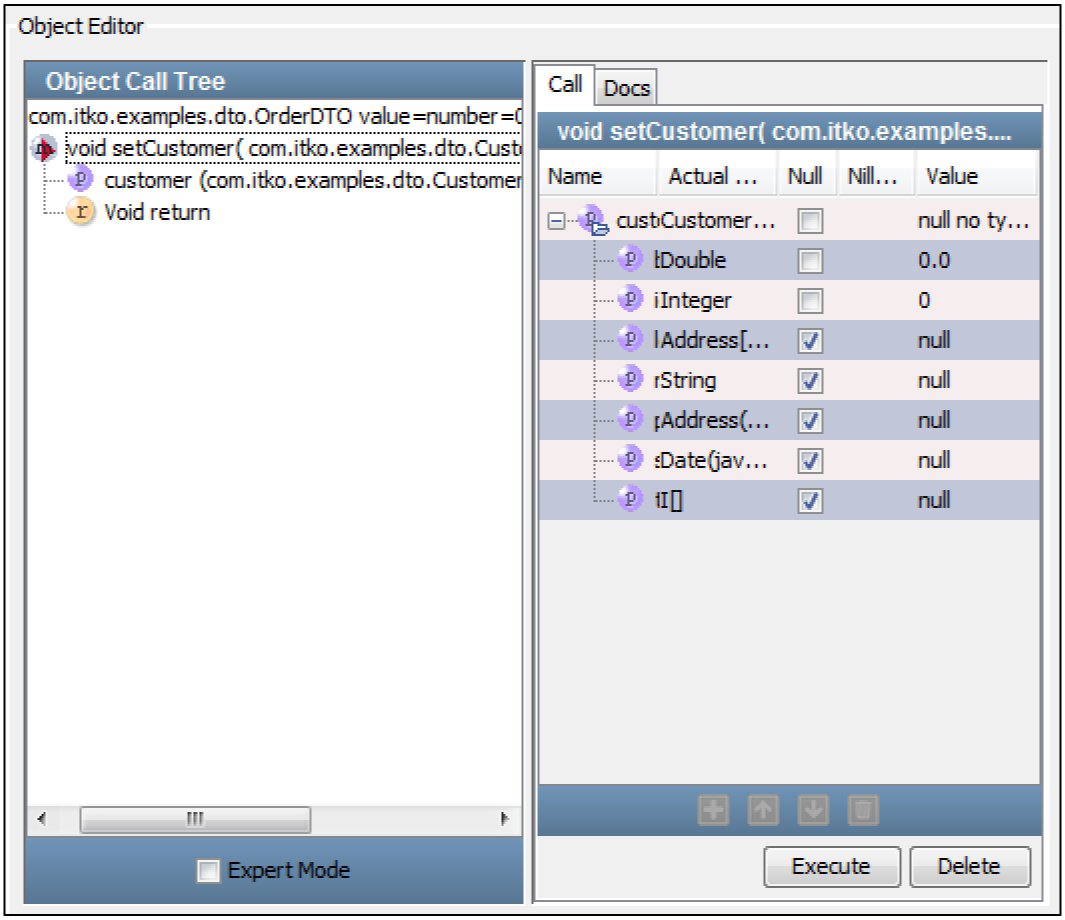
Again we can use **Simple Mode** for the **OrderDTO** object.

1. Select the **setCustomer** method in the **Call Sheet** and click the **Invoke Method** ▶ icon. As expected, the input parameter is a Customer object.


Object Editor
Object Call Tree
com.itko.examples.dto.OrderDTO value=numbe
void setCustomer(com.itko.examples.dto.C
customer (com.itko.examples.dto.Custo
Void return
Expert Mode
Call
Docs
void setCustomer(com.itko.examples....
Execute
Delete

Name	Actual ...	Null	Nill...	Value
custCustomer...		☑		null

2. Clear the check box in the **Null** column. The Customer row expands to expose its properties.



Object Editor

Object Call Tree

com.itko.examples.dto.OrderDTO value=number=(
void setCustomer(com.itko.examples.dto.Custo
customer (com.itko.examples.dto.Customer
Void return

□ Expert Mode

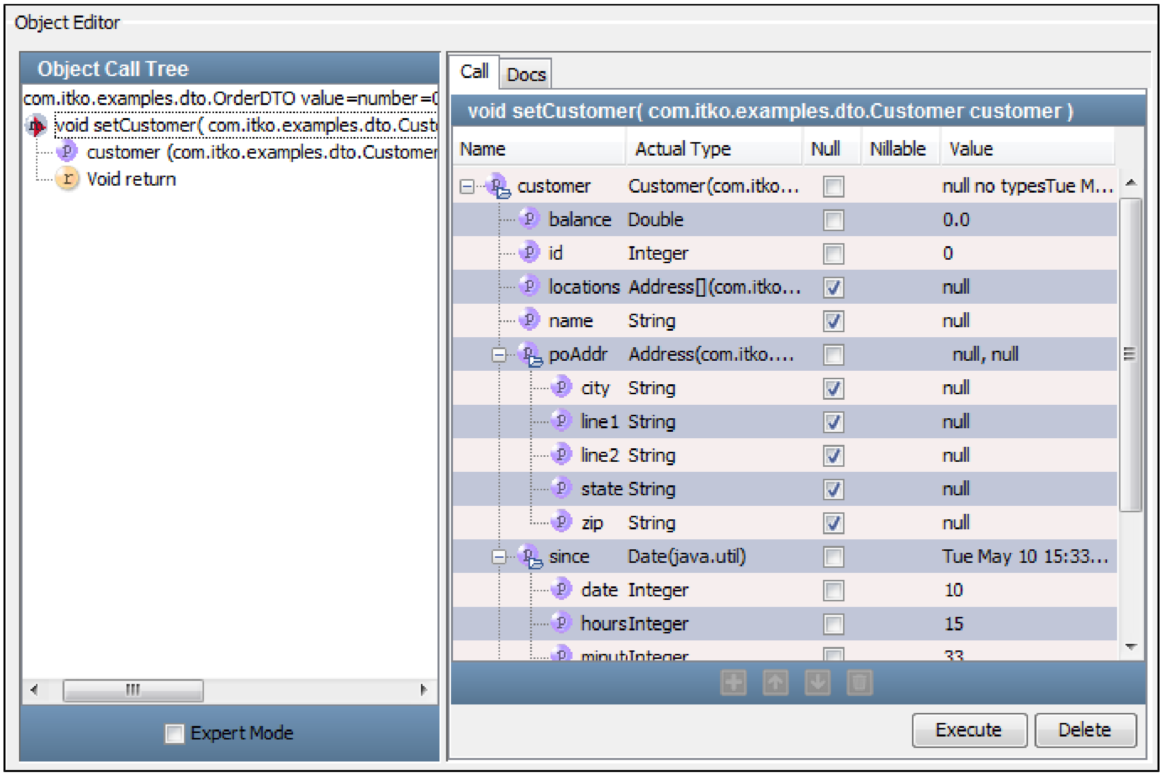
Call | Docs

void setCustomer(com.itko.examples....

Name	Actual ...	Null	Nill...	Value
cust	Customer...	□		null no ty...
l	Double	□		0.0
i	Integer	□		0
l	Address[...	☑		null
r	String	☑		null
p	Address(...	☑		null
s	Date(jav...	☑		null
i	I[]	☑		null

Execute | Delete

3. All the properties can be edited on this screen. The Integer and String properties can be added in the Value column. The remaining properties will expand to show their properties when you clear the **Null** box for the property. If the property is a single object it will expand to expose its properties. If it is an array or collection, you can add the appropriate number of elements using the **Add** ⊞ icon. This illustration shows a snapshot of the editing process.



com.itko.examples.dto.OrderDTO value=number=(
void setCustomer(com.itko.examples.dto.Cust
customer (com.itko.examples.dto.Customer
Void return

□ Expert Mode

void setCustomer(com.itko.examples.dto.Customer customer)

Name	Actual Type	Null	Nillable	Value
customer	Customer(com.itko...	□		null no typesTue M...
balance	Double	□		0.0
id	Integer	□		0
locations	Address[](com.itko...	☑		null
name	String	☑		null
poAddr	Address(com.itko....	□		null, null
city	String	☑		null
line1	String	☑		null
line2	String	☑		null
state	String	☑		null
zip	String	☑		null
since	Date(java.util)	□		Tue May 10 15:33...
date	Integer	□		10
hours	Integer	□		15
minut	Integer	□		33

The **locations** property has two elements; the **types** property has three elements. The **poAddr** object is of type Address, and is expanded exposing simple string properties.

Building Test Steps

A test step is an element in the LISA test case workflow that represents a single test action that is to be performed. There are two major categories of test steps.

Most test steps perform an action on the system under test, and evaluate the response. Some common examples are testing an Enterprise JavaBean (EJB) method, a web service, or a message through messaging service provider.

A second category of test steps perform utility functions, such as data conversion, data manipulation (such as encoding), logging, and writing information to files, and so on.

Both categories of steps go into the building of a test case.

The following topics are available in this section.

Adding a Test Step
Configuring Test Steps
Adding Filters, Assertions, and Data Sets to a Step
Common Test Step Actions
Configuring Next Step
Generating Warnings and Errors
Types of Steps

Adding a Test Step

To add a new test step

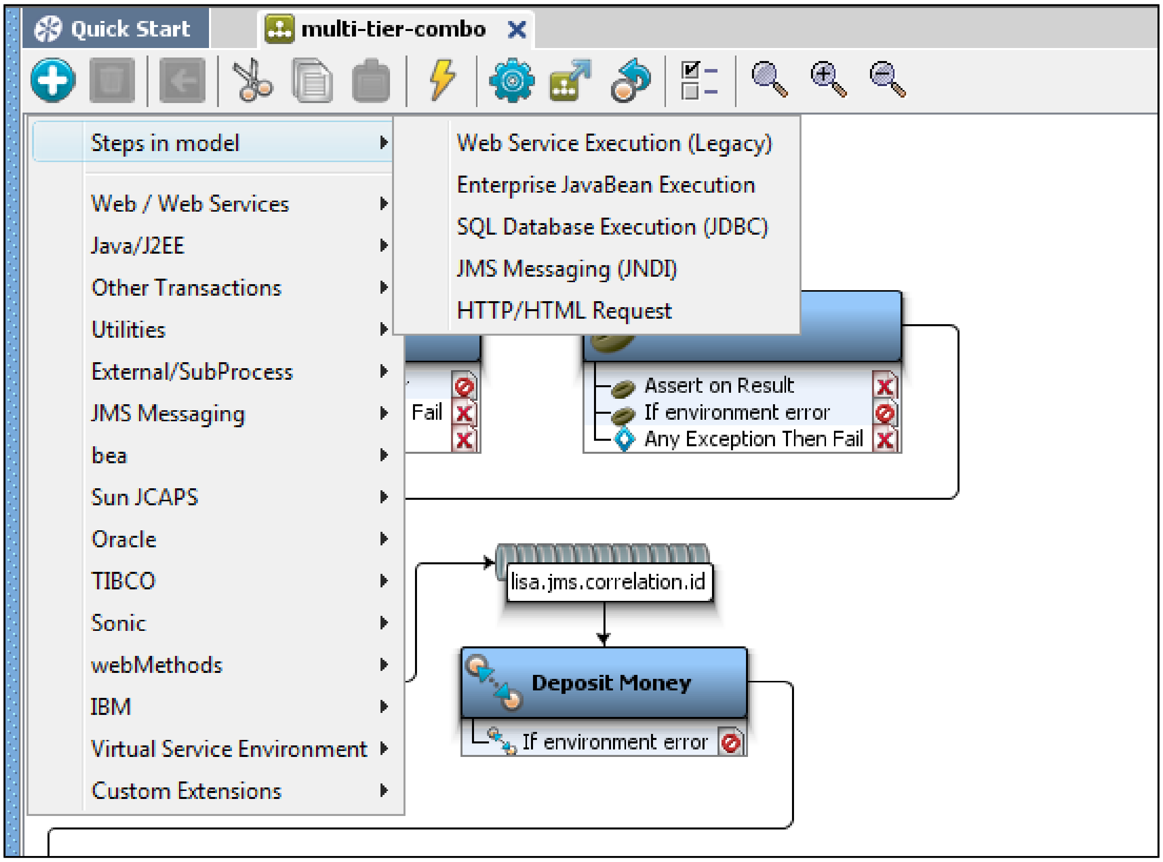
- Click Add Step on the toolbar or
- From the main menu select Commands > Create a New Step.

You can also add a step in a specific place in the workflow by right-clicking the step in the workflow to open a menu. Click Add Step After and select the required test step.

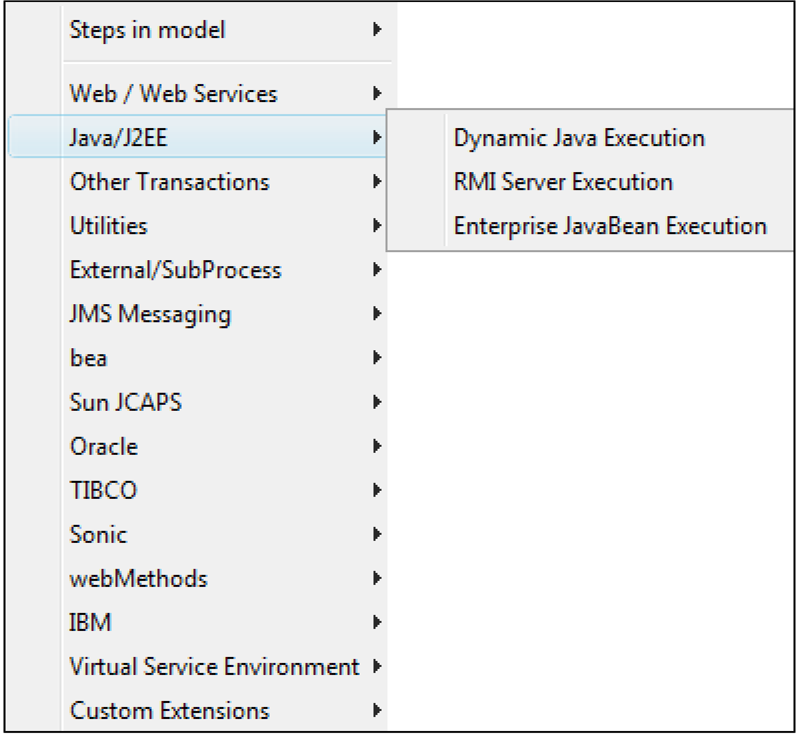
Adding a Test Step (example)

This example adds a new step to the multi-tier-combo test case in the examples directory (multi-tier-combo.tst). A new Dynamic Java Execution step will be added to the multi-tier-combo test case, after the Get User step.

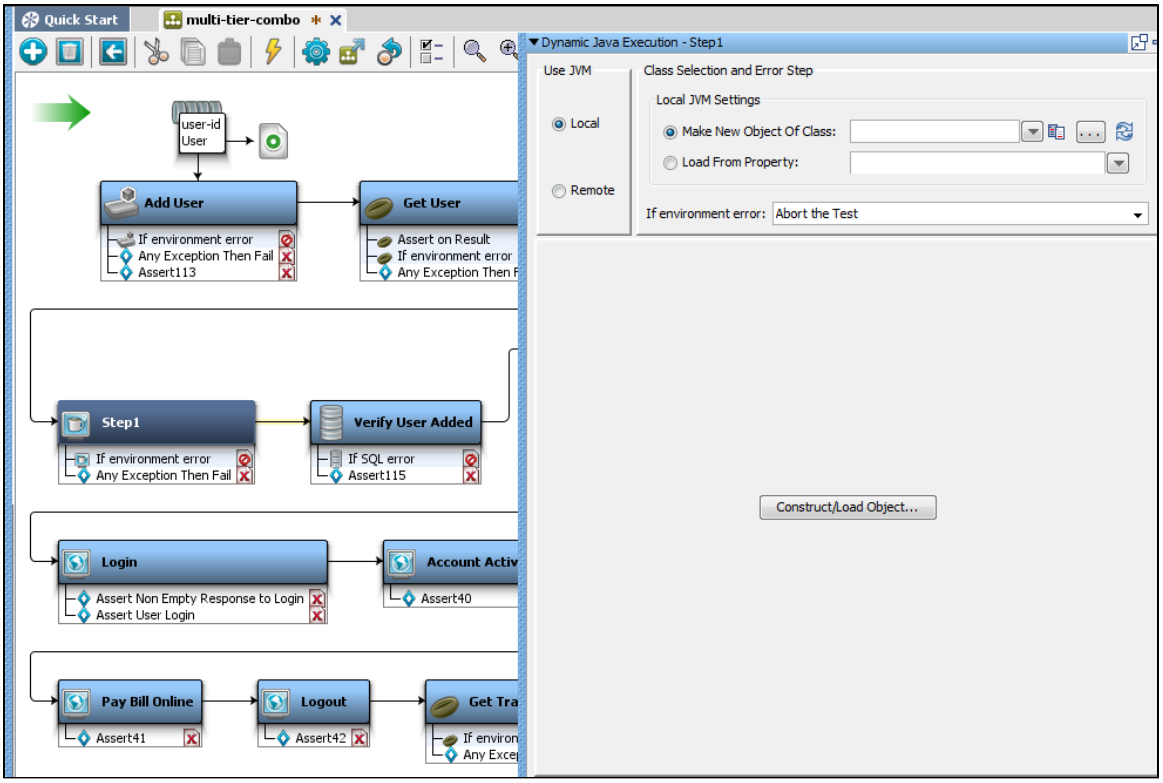
1. Click the Add Step button to open a panel that has the listing of the common test steps. If steps have been already created in the test case, as in this test case, the panel will show Steps in Model on top, which will list all the steps present in the test case. For a new test case, this field is empty. When you open the multi-tier-combo test case, the steps that are in this test case are seen in the Steps in Model box.



2. Select the main category of the step to be configured (for example, Web/Web Services, Java/J2EE, Utilities, and so on), which will open the sub category.



3. Click the step to add it to the test case. This will open its step editor. Each step type has a different step editor.
4. To add a new Dynamic Java Execution step after the Get User step, right-click the Get User step to open the menu and select the Dynamic Java Execution step.
5. The step is added and the Step editor for the Dynamic Java Execution step opens.



Use JVM: ● Local ○ Remote

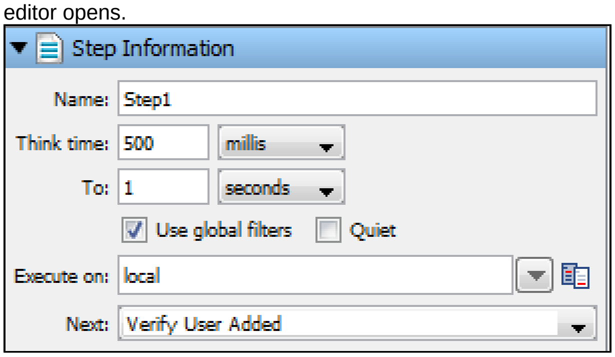
● Make New Object Of Class:

○ Load From Property:

If environment error: Abort the Test

Adding Step Information

1. To add the basic step information, open and expand the Step Information tab in the Elements tree in the right panel. The Step Information editor opens.



Step Information

Name: Step1

Think time: 500 millis

To: 1 seconds

☑ Use global filters □ Quiet

Execute on: local

Next: Verify User Added

2. Enter the following parameters:

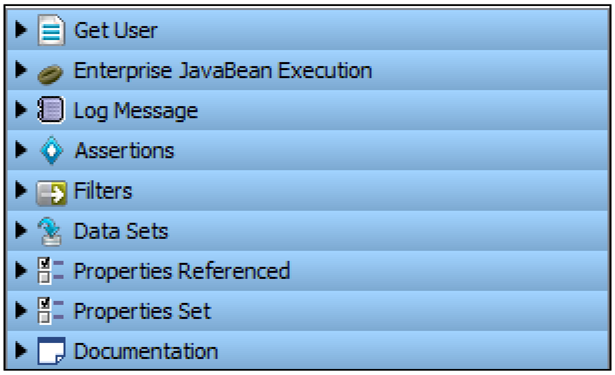
- **Name**: The name of the step. By default it appears as StepX (X being a number). You can rename the step in this text box.
- **Think Time**: The amount of time the test case should wait before executing this step. This provides the ability to simulate the amount of time it takes a user to decide what to do before taking action. To specify how the time is calculated, select the time unit by clicking the appropriate drop-down (millis, seconds, minutes), then enter a value in the Think Time field for the starting value. In the To field, enter an end value and a time indicator. LISA will pick a random think time within this range. For example, to simulate a user think time for a random amount of time between 500 milliseconds and 1 second, enter "500 millis" and "1 seconds."
- **Use Global Filters**: Select this box if the step should be instructed to use global filters. For more information on filters, see Adding a Filter.
- **Quiet**: Select this box if you want LISA to ignore this step for response time events, and performance calculations.
- **Execute On**: Specifies the simulator that the step is to be run on. Specific simulators should be specified for steps that must run on a specific computer. For example, when reading a log file the step should be run on the computer where the log resides.
- **Next**: The next step to execute in the test. If the step in question ends the test case execution, you will not specify a next step. An assertion that fires in this step overrides this value.

Configuring Test Steps

To configure a test step

1. Add the step in the LISA Model Editor. After the step is added, a different test step panel appears in the Element tree.
2. Configure the test step by setting the parameters in the configuration elements (assertions, filters, data sets, and so forth).

Details of the each test case/step element can be seen by clicking the element arrow ▶ next to the configuration elements to expand it.



- **Step Information**: This is the first element in the element tree and carries the name of the step as its label (Get User in the preceding example). After expanding it, you may be able to change the name of the test step, and set the next step to be executed after the current step is completed. The other options are explained in Adding a Test Step.

- **Step Type Information:** This appears next and has the title of the selected step type (Enterprise JavaBean Execution in our example). Each step is different and has its own specific configuration requirement, and thus has a custom editor to provide the information needed to run the step and test it (and possibly to get a response also). This custom editor opens up when the element is expanded.

- **Log Message**: This appears after any step is added in LISA and let you set the log message that appears after the step execution is complete.

- **Assertions**: The addition and configuration of one or more assertions. In an assertion configuration, you also need to set the next step to be executed when the assertion fires.

- **Filters**: The addition and configuration of filters. Filters are added under the filter element of each test step.

- **Data Sets**: The addition and configuration of one or more data sets that apply to the test step. Data sets are fired before executing a test step. Any property that the data set sets, is available to the test step.

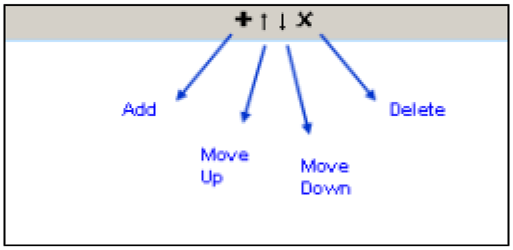
- **Properties Referenced**: A read-only list of properties that the step references (reads).

- **Properties Set**: A read-only list of properties the step sets (assigns a value).

- **Documentation**: Notes accompanying the test step.

Step Element Toolbar

All elements (assertions, filters and data sets) have their own toolbar at the bottom of the Element tab.



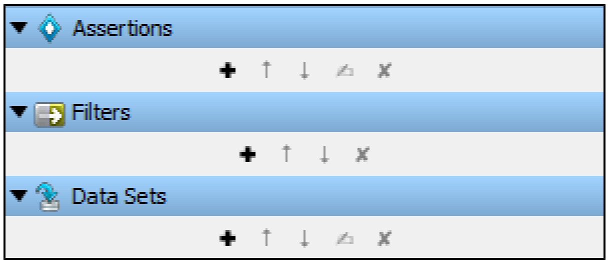
You can add/delete an element to a step by clicking the icons at the bottom of the individual elements.

For detailed information on adding filters, see Adding Filters.

For detailed information on adding assertions, see Adding Assertions.

Adding Filters, Assertions, and Data Sets to a Step

Filters, assertions, and data sets are added under the corresponding element of each step in the right panel.



Click the Add ✚ icon on the toolbar to add a filter, assertion, or data set.

For detailed information about adding and configuring filters see Adding Filters. For detailed information about adding and configuring assertions see Adding Assertions. For detailed information about adding and configuring data sets see Adding Data Sets.

Common Test Step Actions

Following are some of the common test step actions:

- Editing/Modifying a step
- Deleting a step
- Reordering a step
- Renaming a step
- Copying a step
- Cutting a step
- Pasting a step

Editing/Modifying a Step

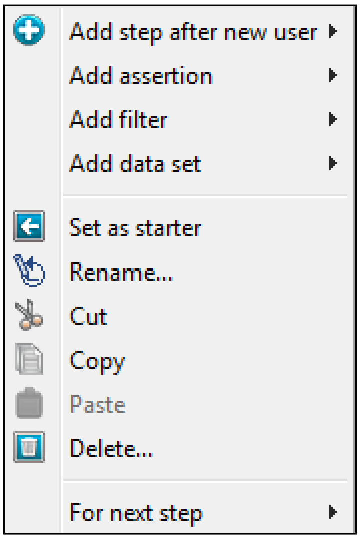
To modify a step

1. Double-click the step to be modified. This will open its editor.
2. Modify the step as required and save the test.

Deleting a Step

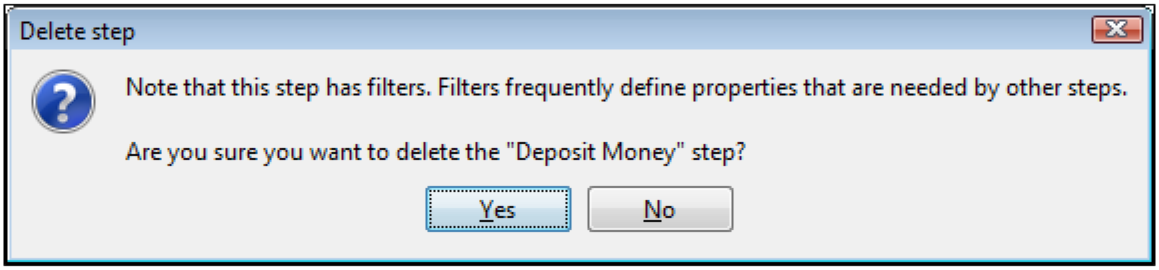
To delete a step

1. Select the step in the workflow and right-click to open a menu.
2. Click Delete to delete the step.



Be careful when you want to delete a test step that has a filter associated to it. *Do not* delete a test step that has filters associated to it. Filters define properties that could be needed by other test steps, and deleting a step also means deleting the filter.

If you attempt to delete such a test step, you get the following message.



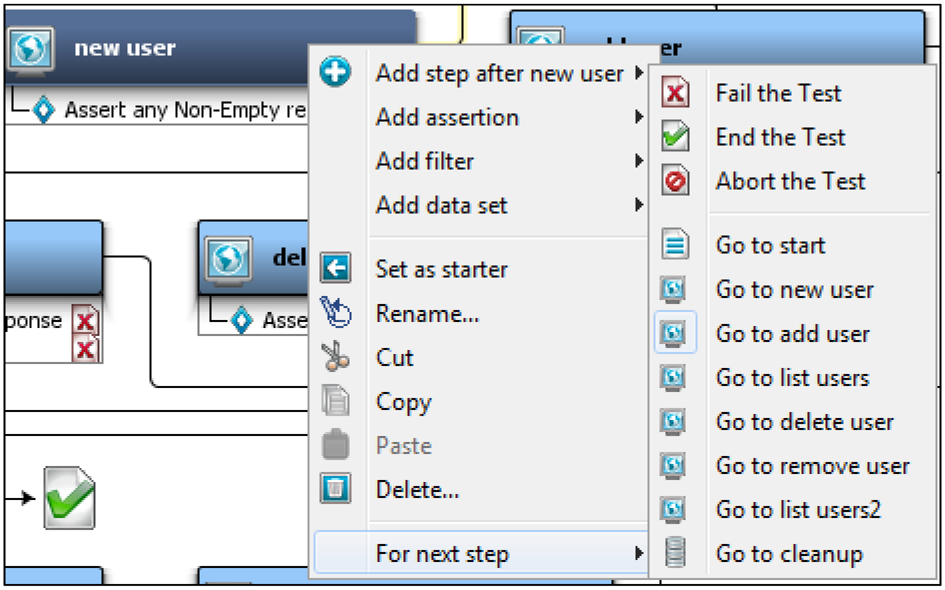
Note that this step has filters. Filters frequently define properties that are needed by other steps.

Are you sure you want to delete the "Deposit Money" step?

Reordering a Step

To reorder a step

- Use the drag and drop facility in the model editor. Click on the step and drag it to the new location to rearrange the workflow.
- Or select the step in the workflow and right-click to open a menu. Click For Next Step and select the step to reorder.



Renaming a Step

To rename a step

1. Right-click a step to open a menu and click Rename.
2. Open the Step Information panel and rename the step in the Name field.

After you rename the step in the workflow, the new name will be reflected in the Step Information tab in the right panel. Any step information pointing to this step will be updated.

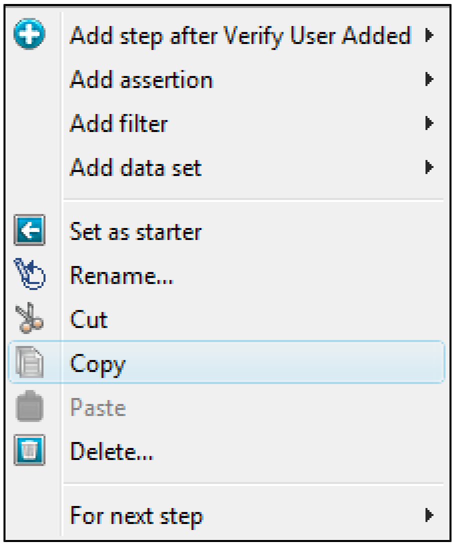
The next step is updated for linear and non-linear workflows in this case.

Copying a Step

You can copy a step and paste it anywhere within the model editor.

To copy a step

1. Select the step to be copied.
2. Right-click the step and click Copy.



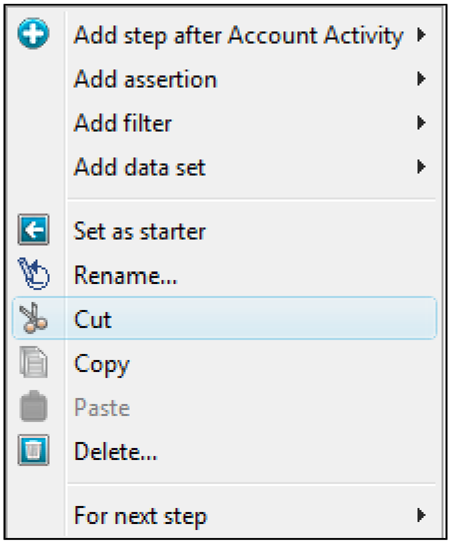
This will copy the selected test step.

Cutting a Step

You can cut a step and paste it anywhere within the model editor.

To cut a step

1. Select the step to be cut.
2. Right-click the step and click Cut.



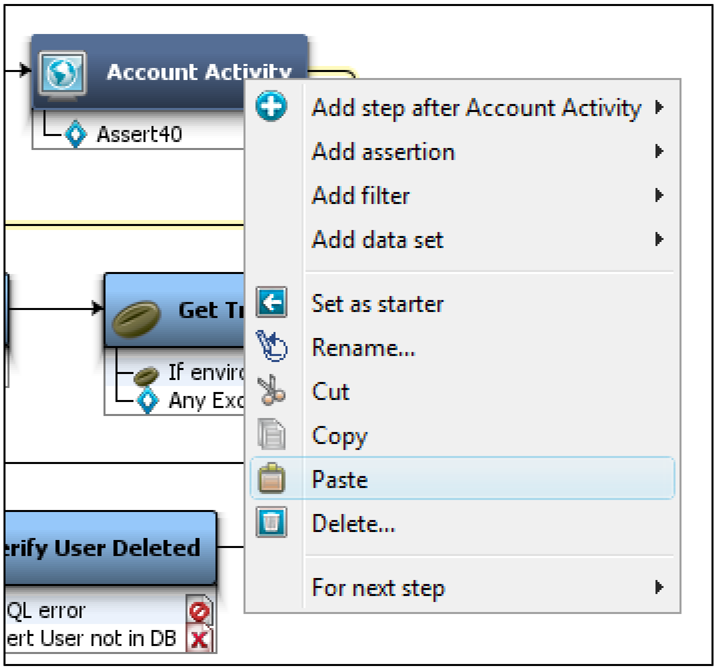
This will cut the step from the model editor.

Pasting a Step

You can paste the step on any target step in the workflow. After you perform the paste operation, the step will be added after the selected step within the workflow.

To paste a step

1. Select the step after which you need to paste the cut step.
2. Right-click and select Paste.



This will paste the step after the selected step.

Configuring Next Step

Assigning the Next Step

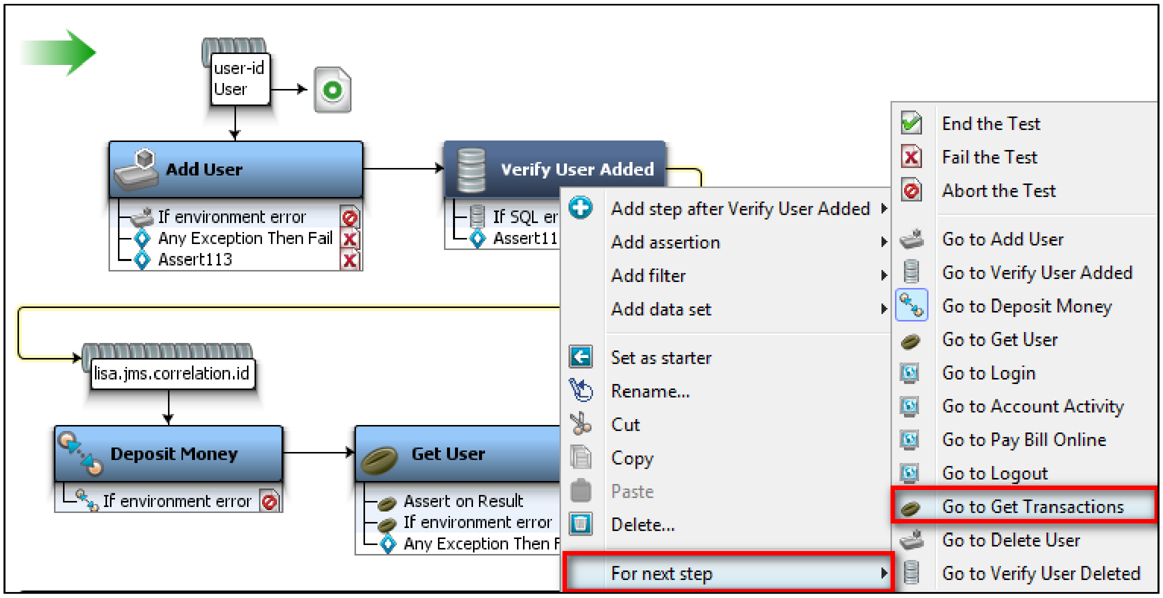
Within a test case workflow, you can assign the "next step" to a selected test step.

After executing the selected step in the workflow, it will then go to the defined next step for execution.

You can configure the "next step" to either go to the other steps in the workflow or direct to either end the test, fail the test or abort the test.

To assign the next step

1. Click the step for which you want to decide the next step. (In the example following, Verify User Added).
2. Right-click and select "For next Step" and click on the targeted next step (Get Transactions).



The workflow in the model editor will change. This will also change the information in the Next field in the step editor.

You can also End the test, Fail the test or Abort the test.

End Step

The End step is to bring an end to a workflow, and is run when a workflow completes successfully. The entire test case is deemed to be successful if the execution reaches this step.

Fail Step

The Fail step is the end of a workflow, and is run when a workflow fails due to an error event. The entire test case is deemed to have failed if the execution reaches this step. The fail step is the default for many exceptions internal to LISA (for example, an exception in an EJB), but it can be set as the next step by assertions to fail a test case.

Abort Step

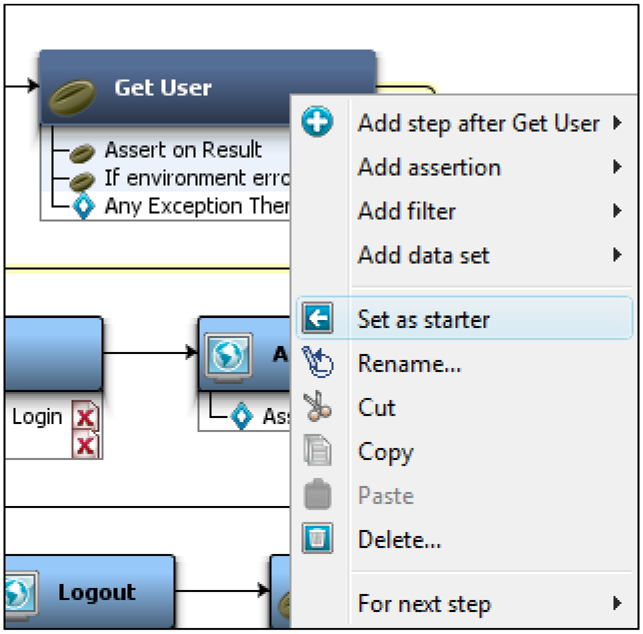
The Abort step is also the end of a workflow, and is run when a workflow is abruptly aborted. The test case is deemed to be aborted (without completion) if this step is reached.

Setting a Starter Step

You can set any test step to be a **starter step** within the workflow.

This test step will then start the test case workflow.

- Click the step and right-click to select Set as Starter.



This will set the selected step as the first step in the workflow. This option is not available to the first step in the test case.

Generating Warnings and Errors

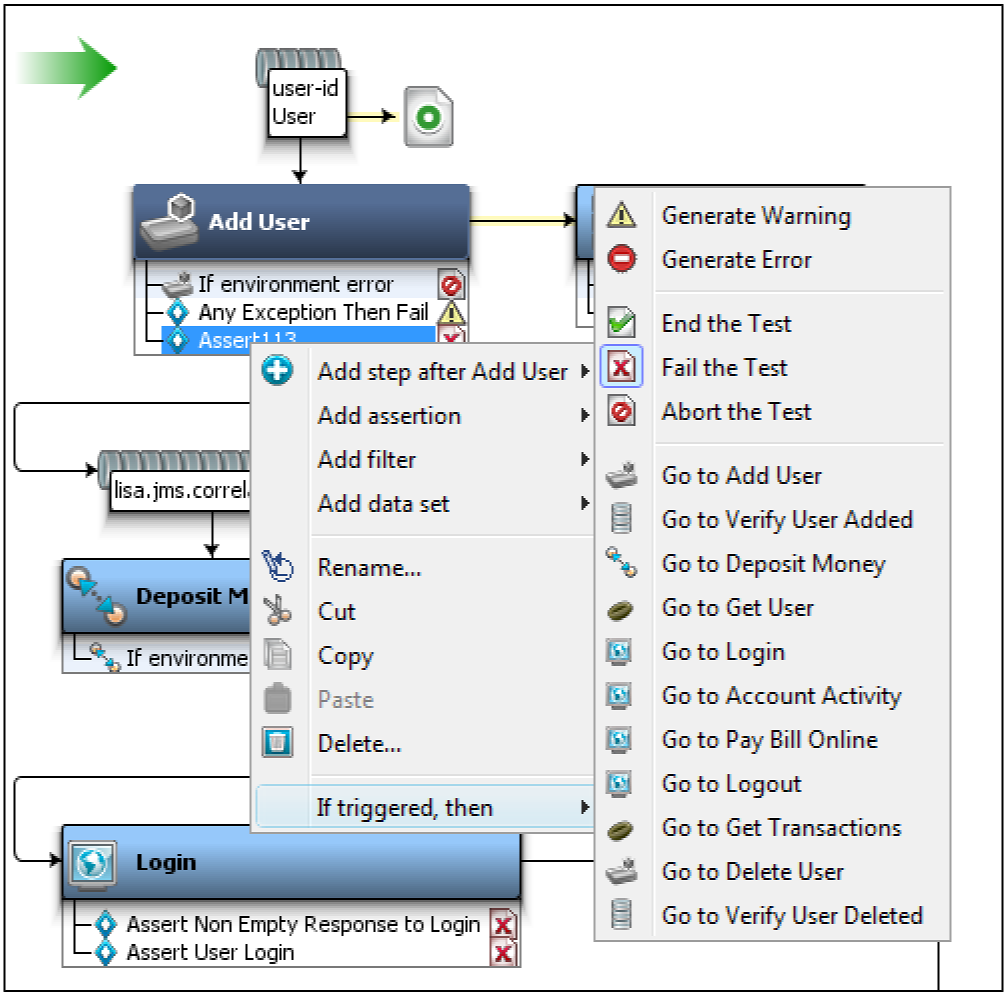
There are two other types of tests that you can configure as next steps.

- Generate Warning
- Generate Error

For an example, we have selected the assertion in the Get User step.

1. Select a step and right-click on the assertion of a step to open a menu.
2. Select the Generate Warning option.

The test case will go to this next step (Generate Warning), only when the assertion is triggered.



Generate Warning Step

When the test step fails, there is also an "Ignore" type of a step logic that will not raise an alarm or event. Hence this will not change the test case workflow. This is the Generate Warning Step. This is the "Continue Quiet" step in previous versions of LISA.

Generate Error Step

Test steps can either pass or fail. When they fail they do not actually fail the test.

To fail the test, the test step sets the test case workflow to execute the "fail" step; this implicitly makes the step considered as failing.

If they explicitly fail, they generate an error and raise a NODEFAILED event and continue the test step. This is the Continue step in earlier versions of LISA.

Types of Steps

The following test step types are available in LISA.

Test Step Information

The following are some standard test steps.

Abort the Test

To abort the test

1. Right-click the test step after which you want to abort the test case.
2. Select For Next Step > Abort the Test.

The Abort the Test step will quit the test case and mark the step as having aborted.

End the Test

To end the test

1. Right-click the test step after which you want to end the test case.
2. Select For Next Step > End the Test.

The step will complete the test and mark the test as having ended successfully.

Fail the Test

To fail the test

1. Right-click the test step after which you want to fail the test case.
2. Select For Next Step > Fail the Test.

The Fail the Test step will fail the test case and mark the test as having failed.

Web_Web Services Steps

The following steps are available in this chapter.

- HTTP_HTML Request
- REST Step
- Web Service Execution (XML) Step
- WSDL Validation
- Web_ Raw SOAP Request
- Base64 Encoder
- Multipart MIME (Multipurpose Internet Mail Extensions) Step
- SAML Assertion Query
- Web_Web Service Execution (Legacy)
- Start or Stop Web Server (Legacy)

HTTP_HTML Request

This step is used while testing a traditional web application to send and receive HTTP(S) requests, including GET and POST parameters and optionally, embedded images as a response. You can also record HTTP steps using the Website Proxy Recorder.

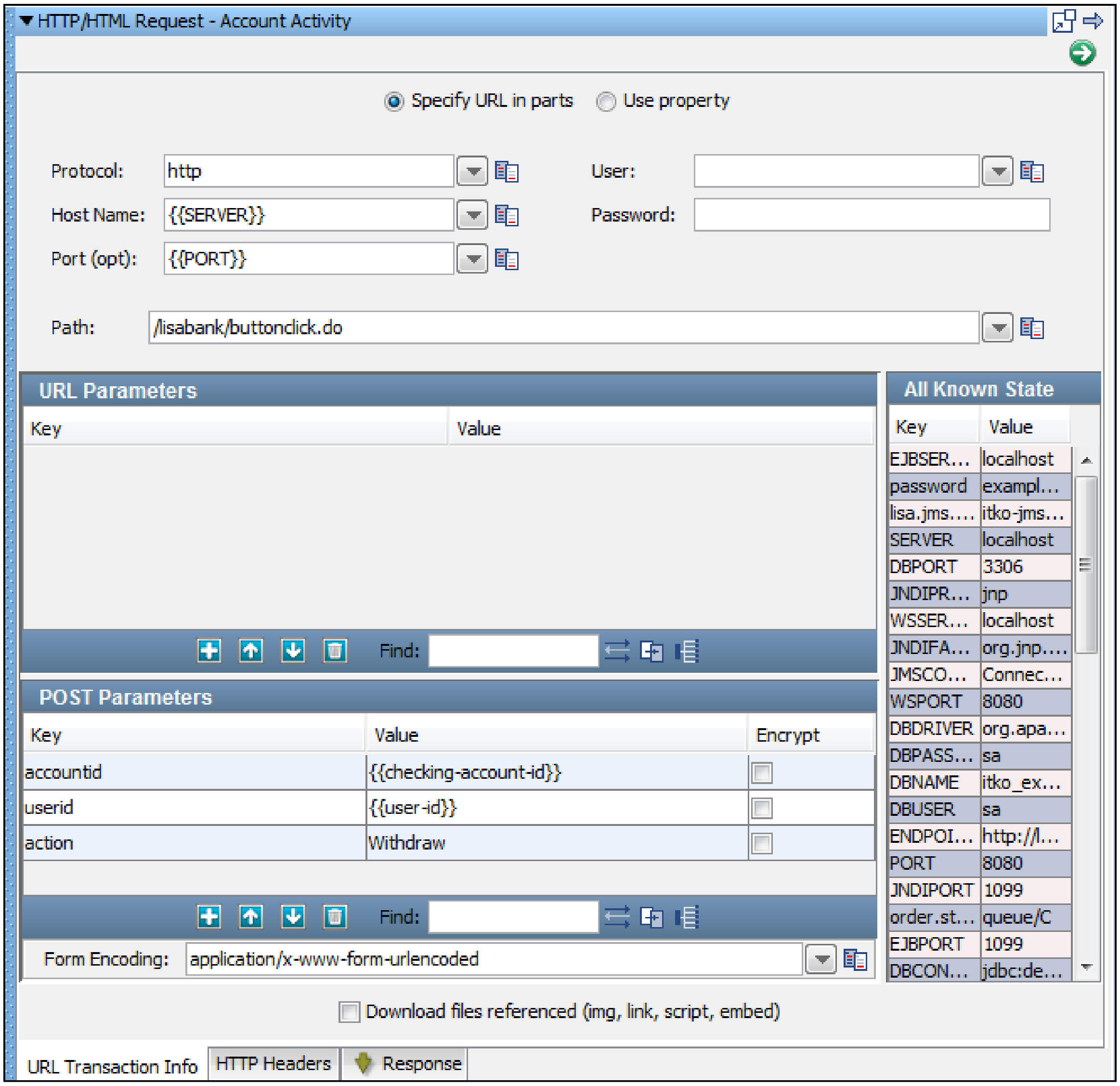
You can manually execute the HTTP/HTML step at design time (similar to the WS step) and the Replay To functionality will save the step responses so the step editors can display the response values.

When you add this step to a test case, the step editor includes the following tabs:

- URL Transaction Info Tab
- HTTP Headers Tab
- Response Tab

URL Transaction Info Tab

Use the URL Transaction Info tab to specify the information used to construct the URL.



HTTP/HTML Request - Account Activity

● Specify URL in parts ○ Use property

Protocol: http

Host Name: {{SERVER}}

Port (opt): {{PORT}}

User:

Password:

Path: /lisabank/buttonclick.do

URL Parameters

Key	Value

Find:

POST Parameters

Key	Value	Encrypt
accountid	{{checking-account-id}}	□
userid	{{user-id}}	□
action	Withdraw	□

Find:

Form Encoding: application/x-www-form-urlencoded

All Known State

Key	Value
EJBSER...	localhost
password	exampl...
lisa.jms....	itko-jms...
SERVER	localhost
DBPORT	3306
JNDIPR...	jnp
WSSER...	localhost
JNDIFA...	org.jnp....
JMSCO...	Connec...
WSPORT	8080
DBDRIVER	org.apa...
DBPASS...	sa
DBNAME	itko_ex...
DBUSER	sa
ENDPOI...	http://l...
PORT	8080
JNDIPORT	1099
order.st...	queue/C
EJBPORT	1099
DBCON...	jdbc:de...

□ Download files referenced (img, link, script, embed)

You can set up the URL transaction information with either of these options:

- Specify URL in parts
- Use property

Specify URL in parts

Select the Specify URL in parts option (default) to specify the URL in its essential pieces.

- **Protocol**: The protocol that is used to communicate with the web server. The default is **http**.
- **Host Name**: The host name of the web server. Use the LISA property SERVER or enter hostname or IP address of your application server. This can be a domain name, such as www.mycompany.com or an IP address, such as 123.4.5.6. For a local web server, use the host name **localhost** or the IP address **127.0.0.1**.
- **Port**: Optional. Use the LISA property PORT or the port on the web server used to access the web server, if necessary. For example, the port required to access the Apache Tomcat web server by default is 8080.
- **Path**: The path to the file to access. For example, if the URL to access is http://localhost:8080/mysite/index.jsp, enter mysite/index.jsp in the Path field.
- **User**: Enter if a user ID is required for the application server.
- **Password**: Enter if a password is required for the application server.
- **URL Parameters**: GET (or URL) Request Parameters: these request parameters are passed as part of the URL, and so they are exposed to the user in address bar of the web browser.
- **POST Parameters**: POST Request Parameters: the request parameters are passed as part of the body of the page request, and so they are not exposed to the user in the address bar of the web browser.
- **Form Encoding**: During a step execution, parameters are URL encoded as they are sent. The MIME type used is application/form-urlencoded.
- **All Known State:** All known properties, such as test case properties, data sets, and filters, are listed.
- **Download images referenced (test bandwidth)**: If this element is selected, the step will download web page images into the test

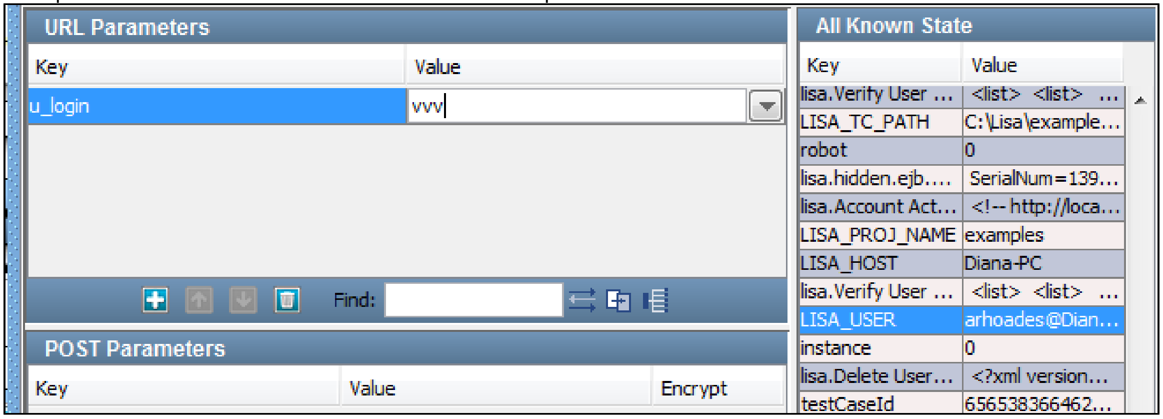
environment. If you do not select this box, there will be no images downloaded.

Other functions that are part of the toolbar available on the URL Parameters and POST Parameters sections are:

Field	Icon	Description
Add	➕	Add a request parameter
Up	↑	Move an existing parameter up in the list of parameters
Down	↓	Move an existing parameter down in the list of parameters
Delete	🗑	Delete an existing parameter
Find		Find text
Auto-Generate a Filter	⇄	From the referring step to make this parameter dynamic. Create a new filter to auto-populate this property at run time. For more information on filters, see the Filters chapter.
Apply selected All Known State property	⧉	Apply state to the parameter. For more information on applying state, see the All Known State section that follows.
Auto Apply all Known State properties	☰	Apply all state to all properties possible by patterns. For more information on applying state, see the All Known State section that follows.

All Known State

All known properties, such as test case properties, data sets and filters, are displayed in the All Known State panel, as seen in the following example in this URL Transaction element of an HTML step:



URL Parameters

Key	Value
u_login	vvv

Find:

POST Parameters

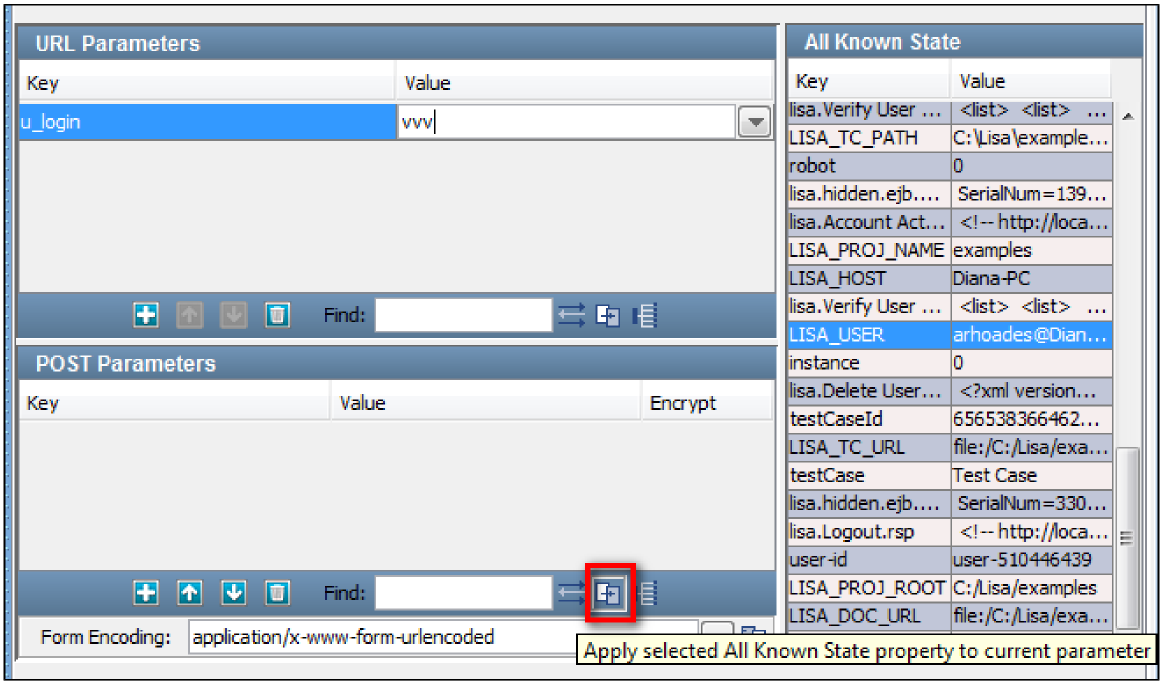
Key	Value	Encrypt

All Known State

Key	Value
lisa.Verify User ...	<list> <list> ...
LISA_TC_PATH	C:\Lisa\example...
robot	0
lisa.hidden.ejb....	SerialNum=139...
lisa.Account Act...	<!-- http://loca...
LISA_PROJ_NAME	examples
LISA_HOST	Diana-PC
lisa.Verify User ...	<list> <list> ...
LISA_USER	arhoades@Dian...
instance	0
lisa.Delete User...	<?xml version...
testCaseId	656538366462...

You can assign the values of properties to URL request parameters. For example, to assign the value of the LISA_USER data set key to the u_login request parameter in the previous example, select the u_login key in the URL Parameters pane, and then the LISA_USER key in the All Known State panel.

Then click Apply selected All Known State property to current parameter:



URL Parameters

Key	Value
u_login	vvv

POST Parameters

Key	Value	Encrypt

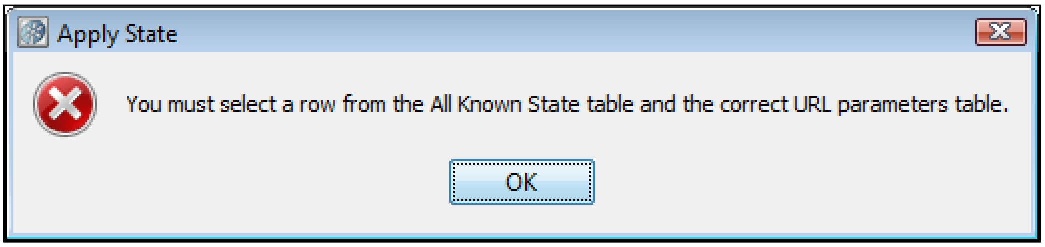
Form Encoding: application/x-www-form-urlencoded

All Known State

Key	Value
lisa.Verify User ...	<list> <list> ...
LISA_TC_PATH	C:\Lisa\example...
robot	0
lisa.hidden.ejb....	SerialNum=139...
lisa.Account Act...	<!-- http://loca...
LISA_PROJ_NAME	examples
LISA_HOST	Diana-PC
lisa.Verify User ...	<list> <list> ...
LISA_USER	arhoades@Dian...
instance	0
lisa.Delete User...	<?xml version...
testCaseId	656538366462...
LISA_TC_URL	file:/C:/Lisa/exa...
testCase	Test Case
lisa.hidden.ejb....	SerialNum=330...
lisa.Logout.rsp	<!-- http://loca...
user-id	user-510446439
LISA_PROJ_ROOT	C:/Lisa/examples
LISA_DOC_URL	file:/C:/Lisa/exa...

Find:
Apply selected All Known State property to current parameter

You will be warned of the impending change:

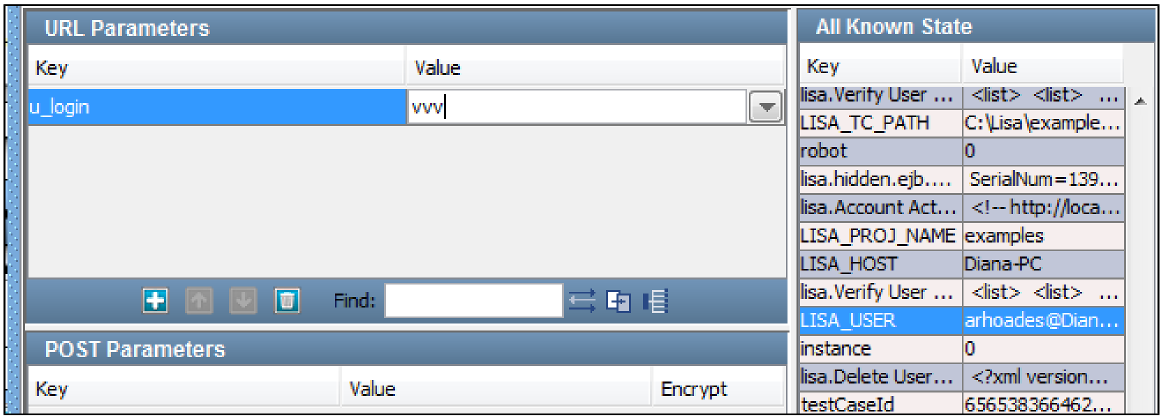


Apply State

You must select a row from the All Known State table and the correct URL parameters table.

Click OK.

The new property is displayed in the URL Parameters pane:



URL Parameters

Key	Value
u_login	vvv

POST Parameters

Key	Value	Encrypt

All Known State

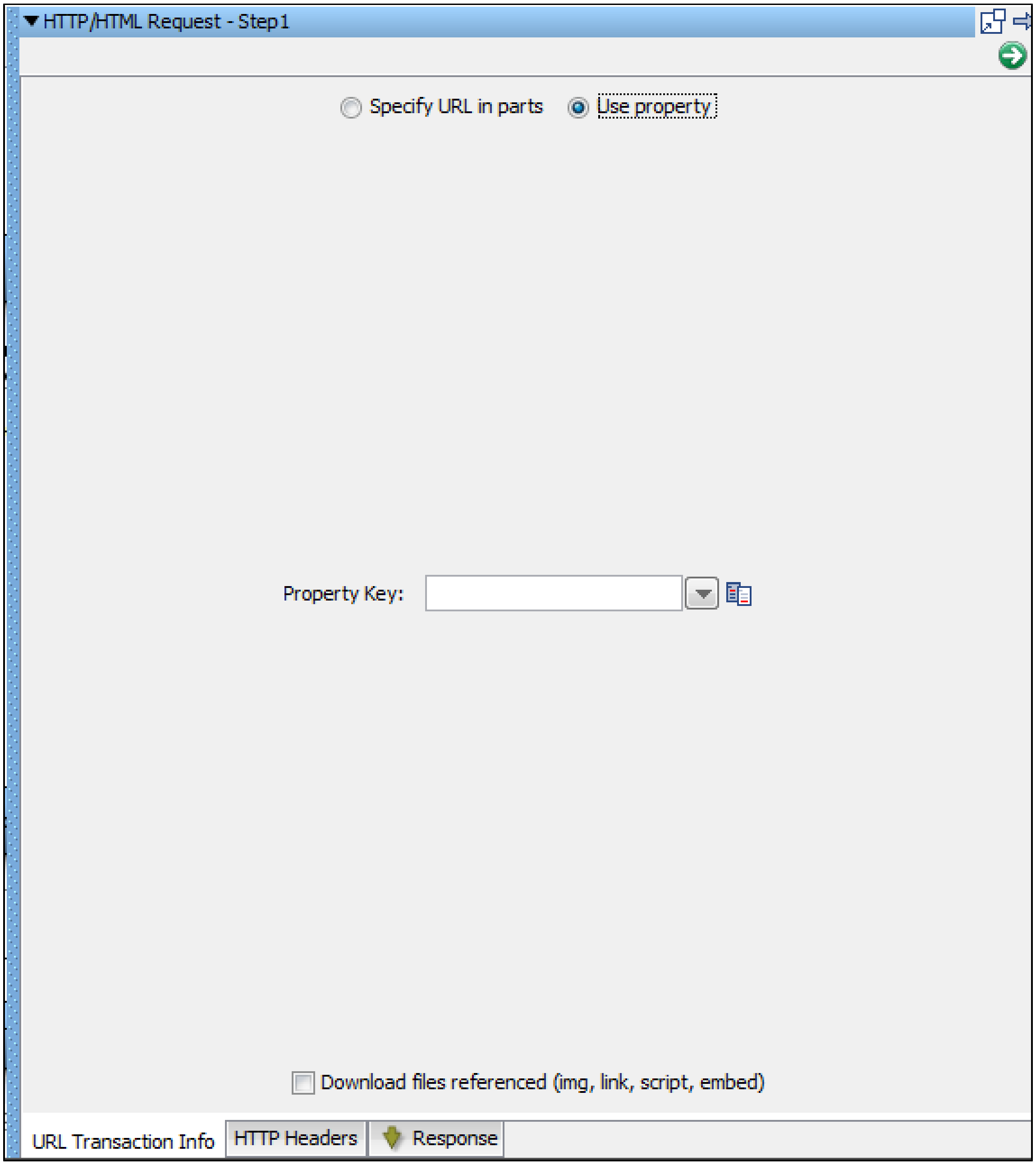
Key	Value
lisa.Verify User ...	<list> <list> ...
LISA_TC_PATH	C:\Lisa\example...
robot	0
lisa.hidden.ejb....	SerialNum=139...
lisa.Account Act...	<!-- http://loca...
LISA_PROJ_NAME	examples
LISA_HOST	Diana-PC
lisa.Verify User ...	<list> <list> ...
LISA_USER	arhoades@Dian...
instance	0
lisa.Delete User...	<?xml version...
testCaseId	656538366462...

If all the names of the URL Parameter keys are the same as the names of the All Known State keys, you can quickly assign all the properties to the associated parameters by clicking the Apply to All icon.

Use property option

If the Use property option button is selected the following parameters can be specified:

- **Property Key**: Specify a property that contains the connection information.
- **Download images referenced (test bandwidth)**: If this element is selected, the step will download web page images into the test environment. If you do not select this box, there will be no images downloaded.



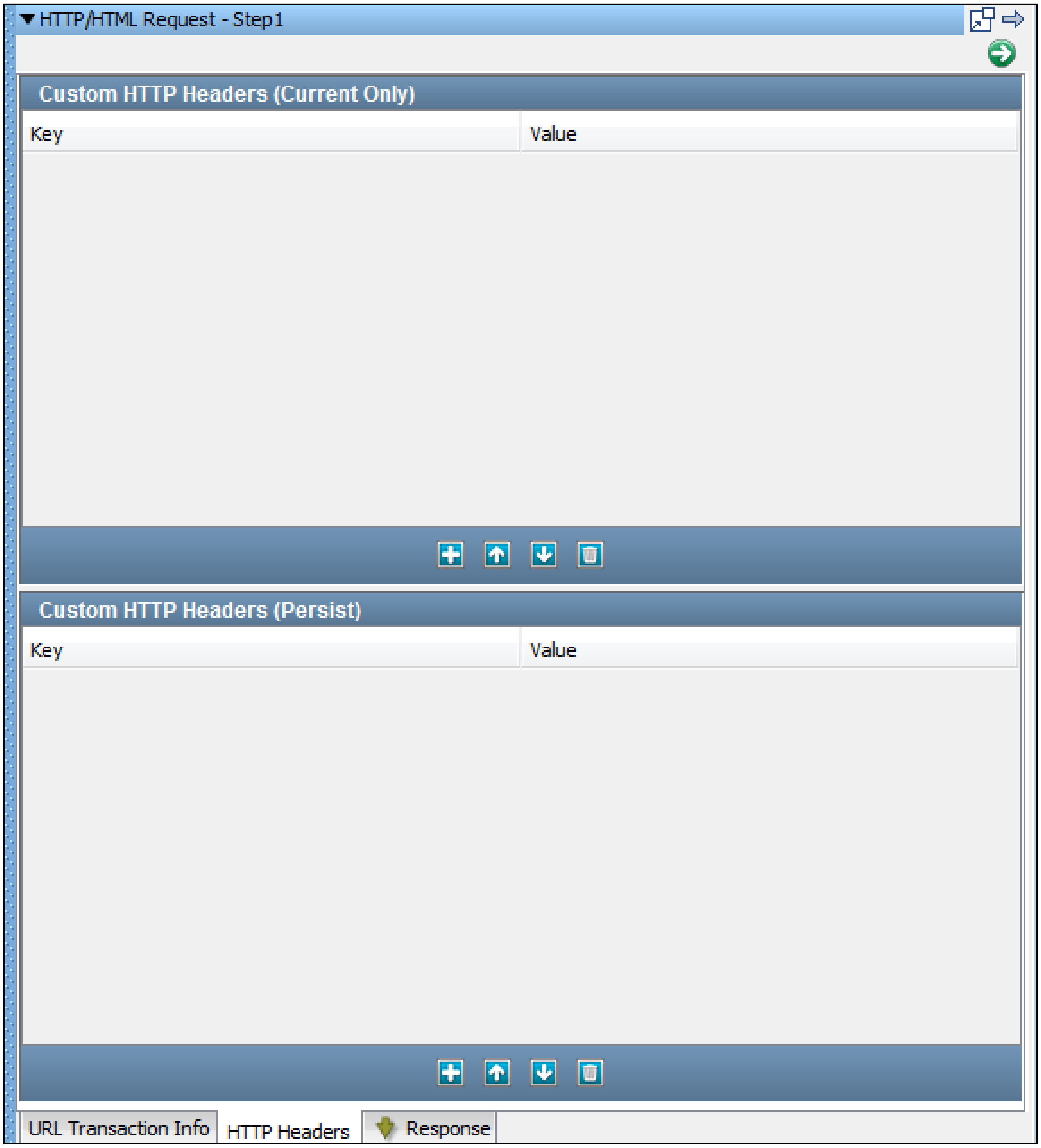
○ Specify URL in parts ● Use property

Property Key:

□ Download files referenced (img, link, script, embed)

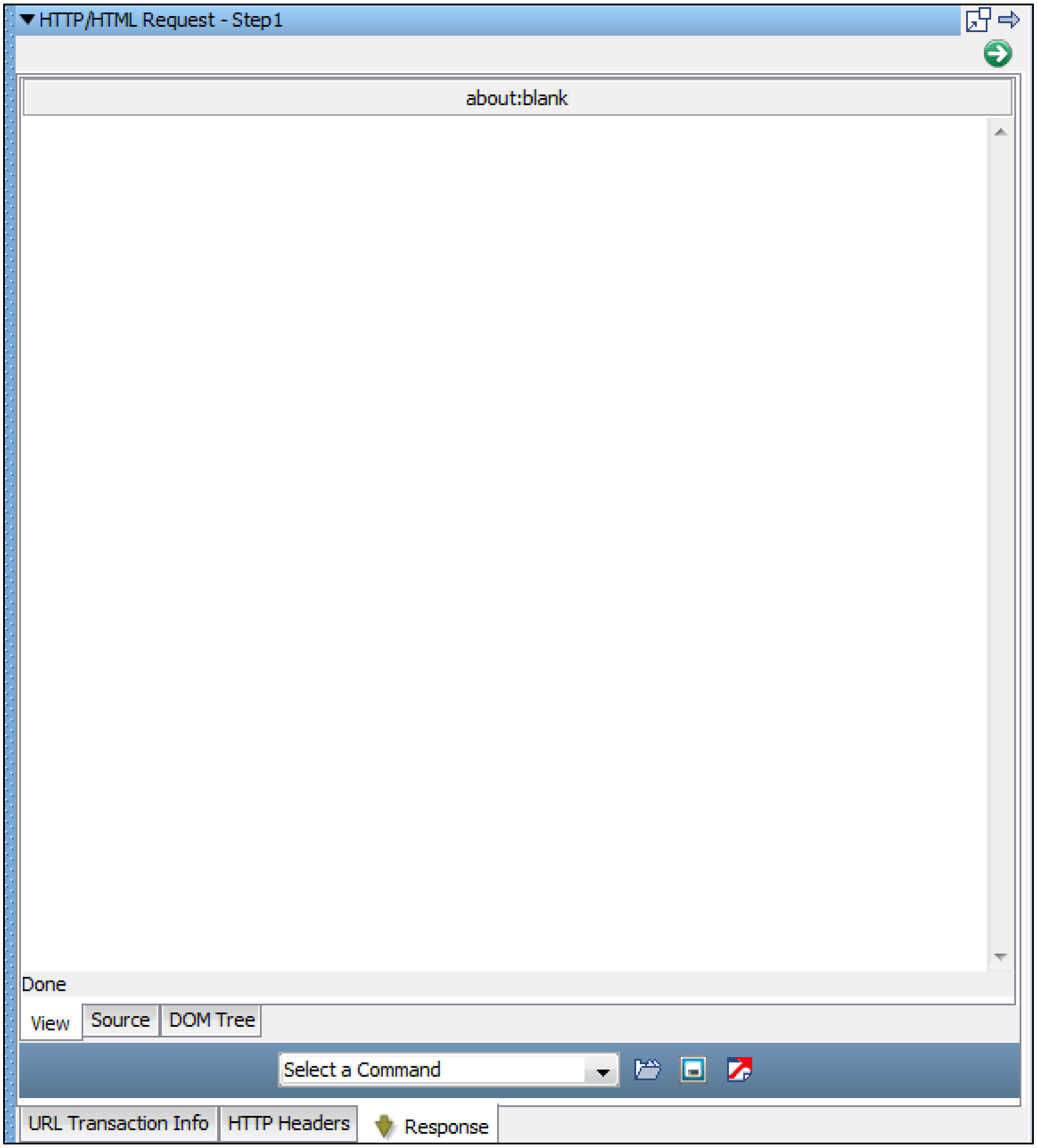
HTTP Headers Tab

On the HTTP Headers tab, create any custom HTTP headers. The top section Custom HTTP Headers (Current Only) is for headers that are only sent to the server for this request. The bottom section Custom HTTP Headers (Persist) is for headers that are sent on this transaction and every other transaction in the test. To create a request parameter in either section, click the Add ➕ icon and change the key and value to the target values.

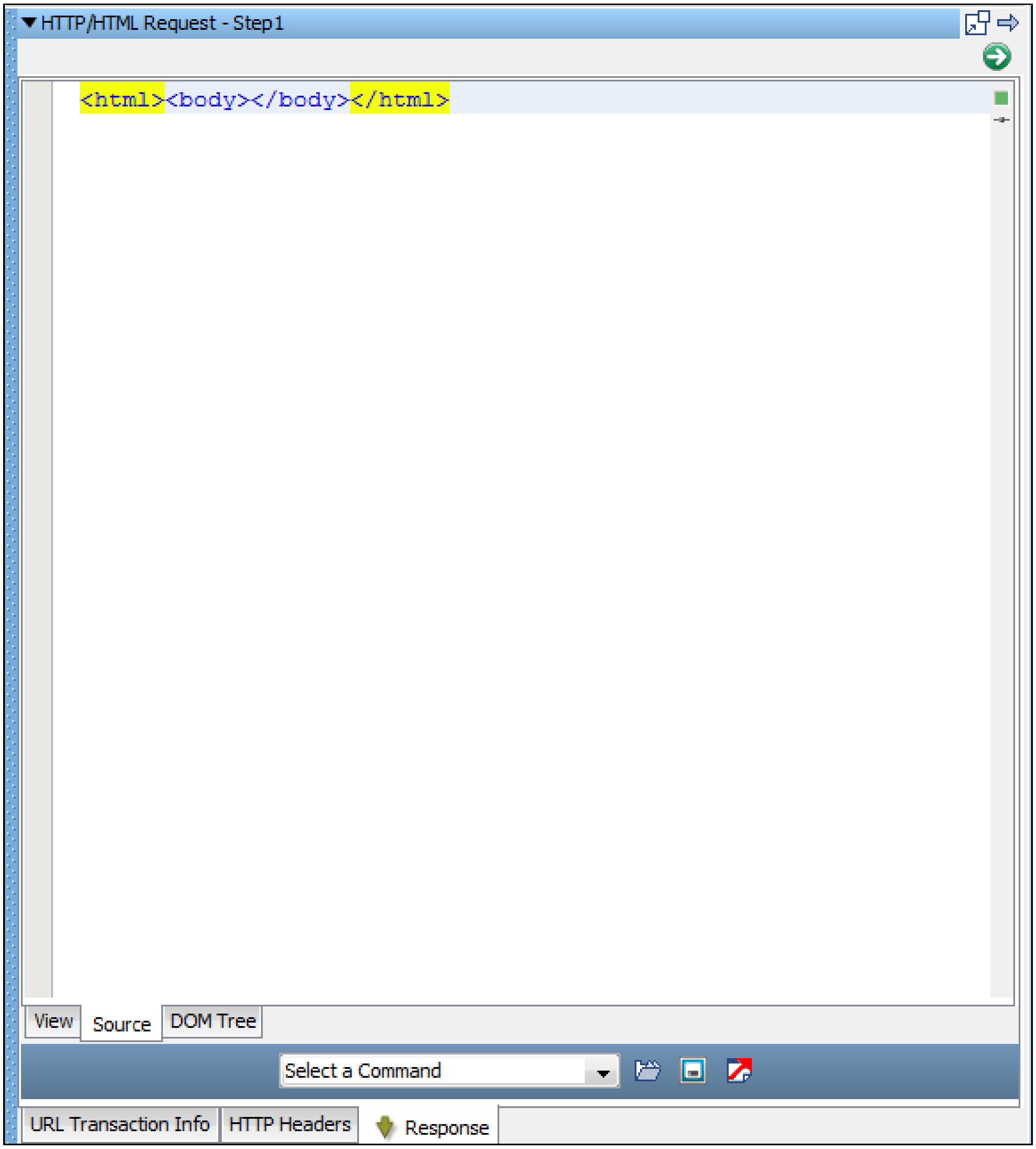

HTTP/HTML Request - Step1
Custom HTTP Headers (Current Only)
Key
Value
Custom HTTP Headers (Persist)
Key
Value
URL Transaction Info
HTTP Headers
Response

Response Tab

On the Response tab, view the HTTP response returned by the server when this test was recorded.

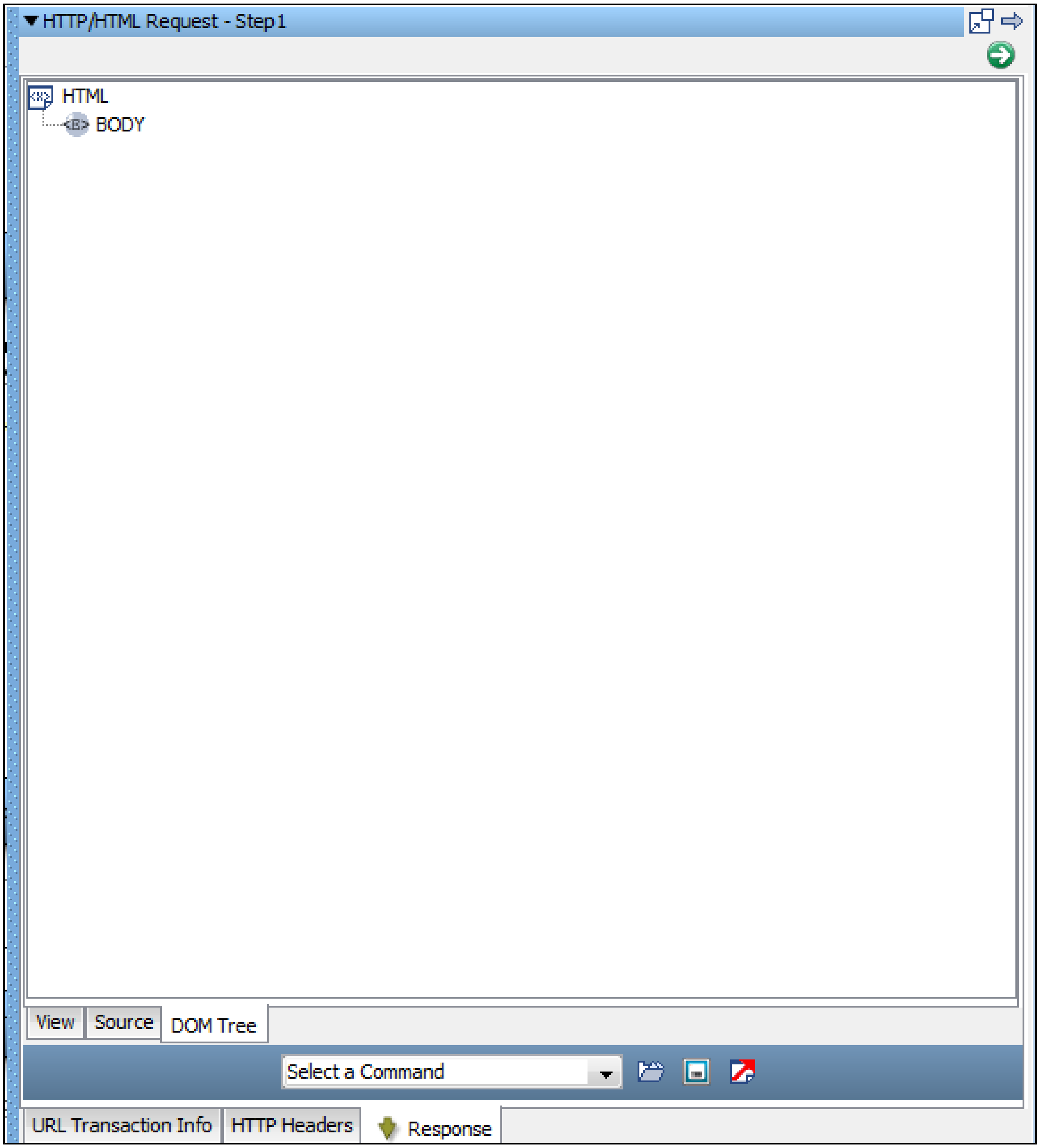


You can view the source of the response.



```
<html><body></body></html>
```

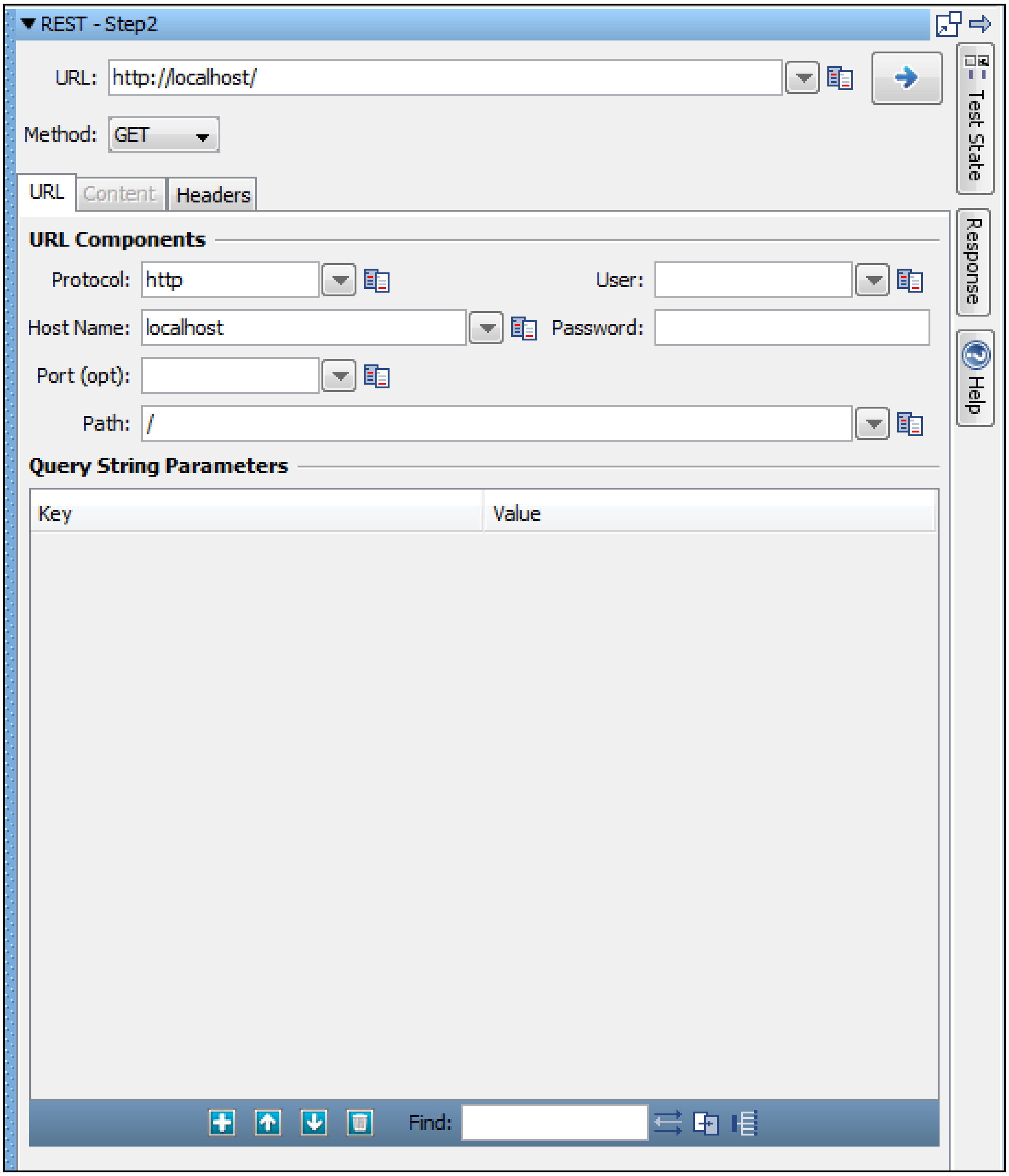
You can view the DOM Tree of the response.



REST Step

Use the REST step when testing REST applications.

This step is used to send and receive HTTP(S) requests, including GET and POST parameters.



URL: http://localhost/

Method: GET

URL Components

Protocol: http

User:

Host Name: localhost

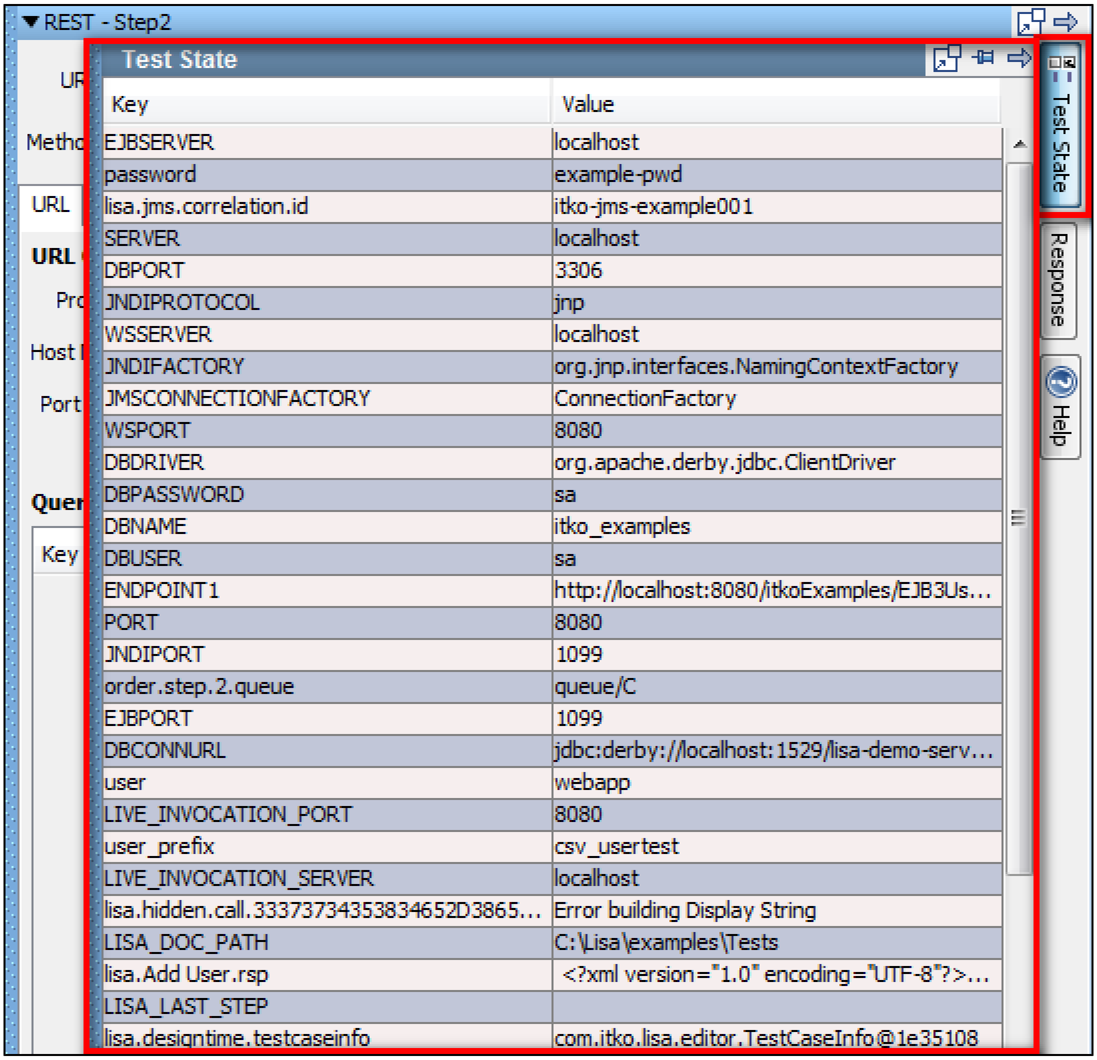
Password:

Port (opt):

Path: /

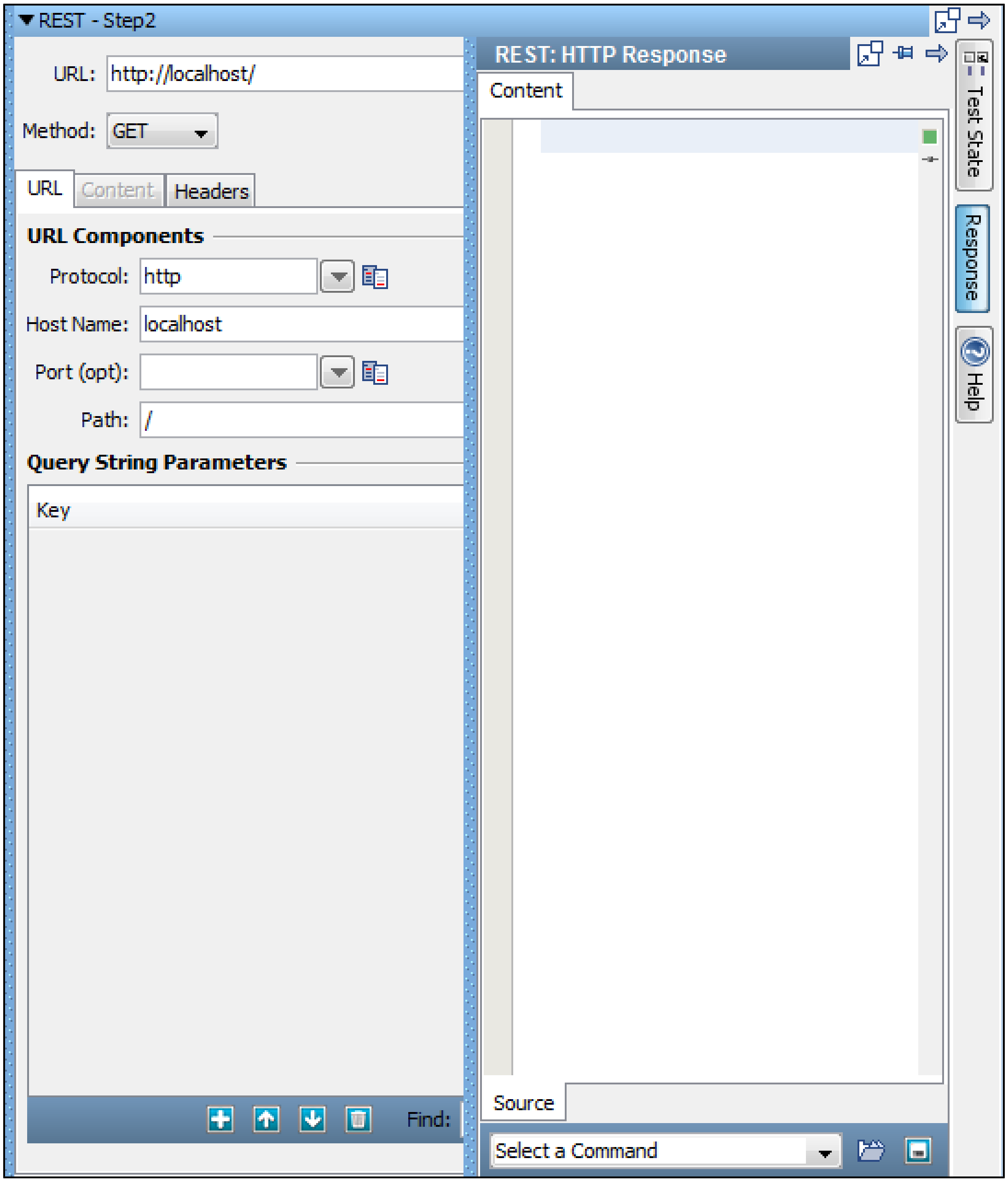
Query String Parameters

Click the Test State button on the right vertical bar to view parameters. The Test State area slides open. You can dock, pin, and hide the list.



Key	Value
EJBSERVER	localhost
password	example-pwd
lisa.jms.correlation.id	itko-jms-example001
SERVER	localhost
DBPORT	3306
JNDIPROTOCOL	jnp
WSSERVER	localhost
JNDIFACTORY	org.jnp.interfaces.NamingContextFactory
JMSCONNECTIONFACTORY	ConnectionFactory
WSPORT	8080
DBDRIVER	org.apache.derby.jdbc.ClientDriver
DBPASSWORD	sa
DBNAME	itko_examples
DBUSER	sa
ENDPOINT1	http://localhost:8080/itkoExamples/EJB3Us...
PORT	8080
JNDIPORT	1099
order.step.2.queue	queue/C
EJBPORT	1099
DBCONNURL	jdbc:derby://localhost:1529/lisa-demo-serv...
user	webapp
LIVE_INVOCATION_PORT	8080
user_prefix	csv_usertest
LIVE_INVOCATION_SERVER	localhost
lisa.hidden.call.33373734353834652D3865...	Error building Display String
LISA_DOC_PATH	C:\Lisa\examples\Tests
lisa.Add User.rsp	<?xml version="1.0" encoding="UTF-8"?>...
LISA_LAST_STEP	
lisa.designtime.testcaseinfo	com.itko.lisa.editor.TestCaseInfo@1e35108

Click the Response button to view the HTTP response.



URL: http://localhost/

Method: GET

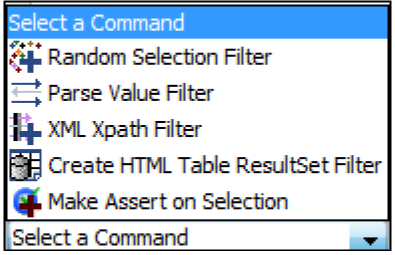
Protocol: http

Host Name: localhost

Port (opt):

Path: /

At the bottom of the Response panel is a selection of filters and an assertion that you can add to the response.



The LISA demo server contains an example of invoking a JSON service to retrieve information from the LISA Bank user database. The URL is http://localhost:8080/rest-example/.

An example test case for the REST step is in the examples project, named rest-example.tst.

Web Service Execution (XML) Step

The Web Service Execution (XML) step is designed to execute an operation on a SOAP-based Web Service using an HTTP POST or JMS message.

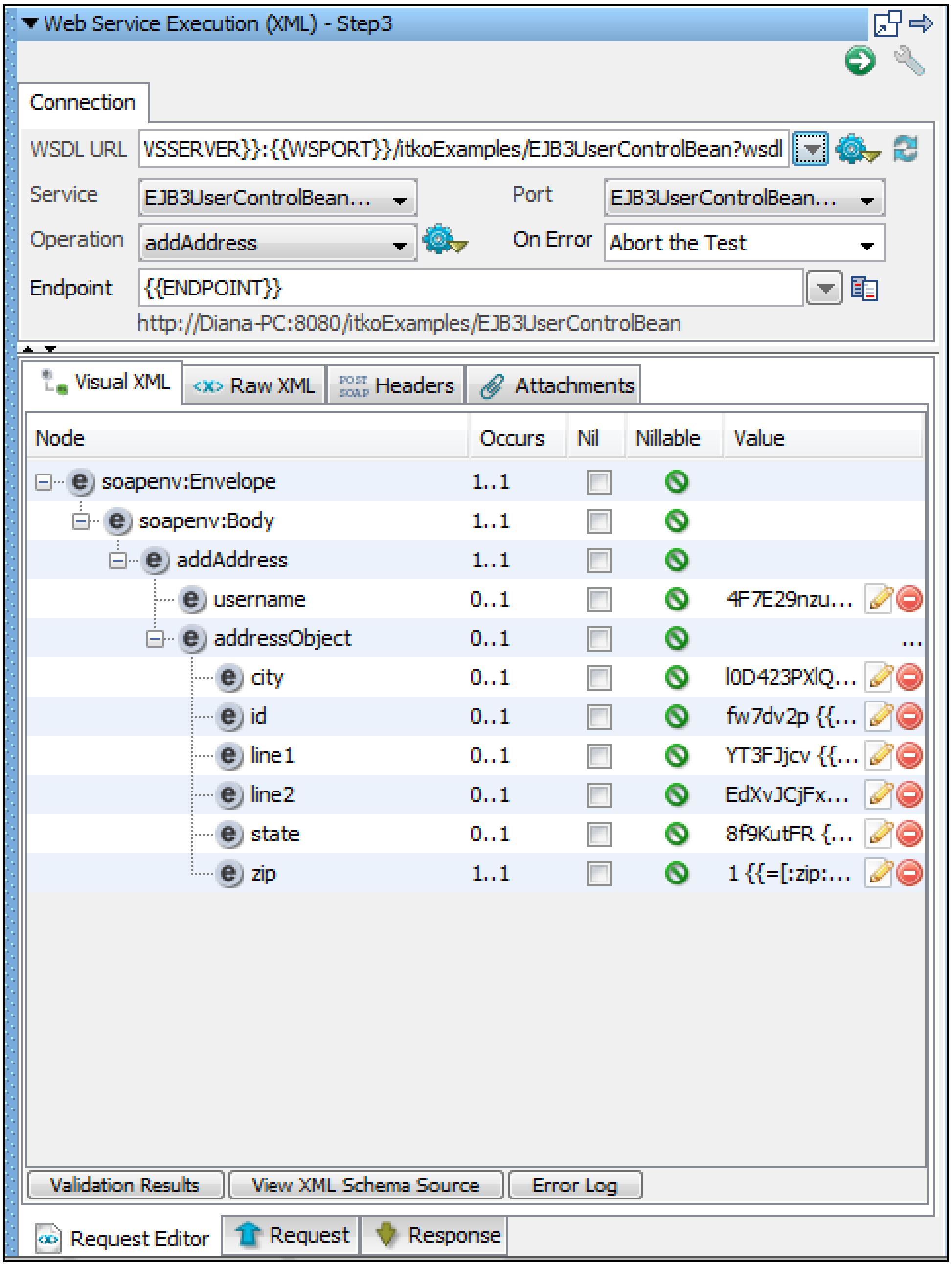
Access to a WSDL is not required; rather, it is a recommended but optional piece of configuration information. If a WSDL is configured, it helps in the process of building a SOAP message to be sent to the service. This step lets you manipulate the raw SOAP message (XML) directly. This gives you a great deal of flexibility and power, but does expose you to the details of how web services work.

In general, the top portion of the editor is dedicated to how and where to send the SOAP message, and the bottom portion is dedicated to what the message will say.

After the test step opens, it has two tabs, with each tab having multiple subtabs within.

Connection Tab

The Connection tab has fields for connection. It has subtabs on the top bar and bottom bar.



WSDL URL: VSSERVER}}:{{WSPORT}}/itkoExamples/EJB3UserControlBean?wsdl

Service: EJB3UserControlBean...

Port: EJB3UserControlBean...

Operation: addAddress

On Error: Abort the Test

Endpoint: {{ENDPOINT}}

http://Diana-PC:8080/itkoExamples/EJB3UserControlBean

Node	Occurs	Nil	Nillable	Value
soapenv:Envelope	1..1	□	⊘	
soapenv:Body	1..1	□	⊘	
addAddress	1..1	□	⊘	
username	0..1	□	⊘	4F7E29nzu...
addressObject	0..1	□	⊘	...
city	0..1	□	⊘	l0D423PXlQ...
id	0..1	□	⊘	fw7dv2p {{...
line1	0..1	□	⊘	YT3FJjcv {{...
line2	0..1	□	⊘	EdXvJCjFx...
state	0..1	□	⊘	8f9KutFR {...
zip	1..1	□	⊘	1 {{=[:zip:...

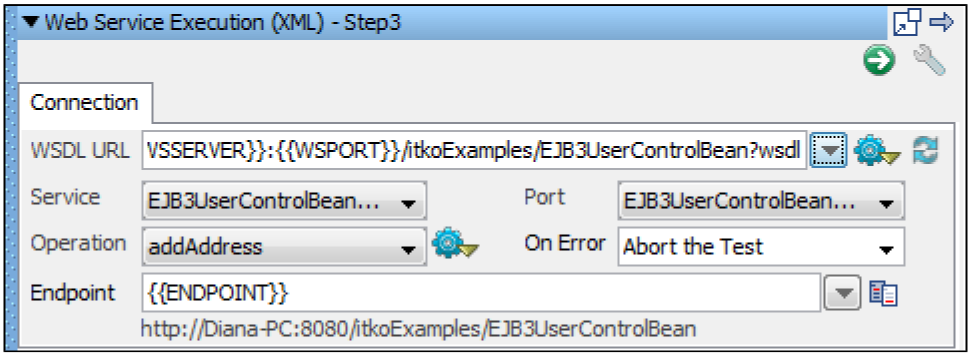
Top bar - for viewing the Visual XML, Raw XML, Headers, Attachments.

Bottom bar - for Request and Response

- Basic Configuration
- Design Time Execution

Basic Configuration

Connection



Web Service Execution (XML) - Step3

Connection

WSDL URL: VSSERVER}}:{{WSPORT}}/itkoExamples/EJB3UserControlBean?wsdl

Service: EJB3UserControlBean...

Port: EJB3UserControlBean...

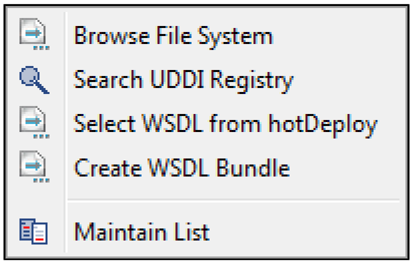
Operation: addAddress

On Error: Abort the Test

Endpoint: {{ENDPOINT}}

http://Diana-PC:8080/itkoExamples/EJB3UserControlBean

- **WSDL URL**: The WSDL URL is an optional but recommended field (denoted by its slightly gray label). It must be a URL (either file:/, http:/, or https:/). From the More Options menu you can choose to:



- Browse the File System for a local WSDL or WSDL Bundle file.
- Search a UDDI Registry (which populates the advanced UDDI access point lookup).
- Select WSDL from hotDeploy to migrate from the legacy WS step.
- Create and use a WSDL Bundle. Creating a WSDL Bundle can also be accessed from the Actions menu, but from here it automatically populates the WSDL URL with the resulting WSDL Bundle's file URL. Or if you already have a WSDL URL populated, it pre-populates the WSDL URL in the WSDL Bundle dialog.

When a WSDL URL is entered, one that is not already a WSDL Bundle, LISA will create a WSDL Bundle and stores it locally in the project's Data/wsdls directory, caching the WSDL locally for quicker access. The WSDL is parsed and its schema is used to build sample SOAP messages and is used by the Visual XML editor to help assist you when manually editing the SOAP Message. It will first try and load a cached WSDL Bundle whenever processing the WSDL. If the external WSDL has changed and you want to force the local WSDL cache to update, use the Refresh WSDL Cache button. At any time you can manually drop a WSDL Bundle into the Data/wsdls and anytime the step tries to process the 'live' WSDL URL it will use the cached bundle instead.

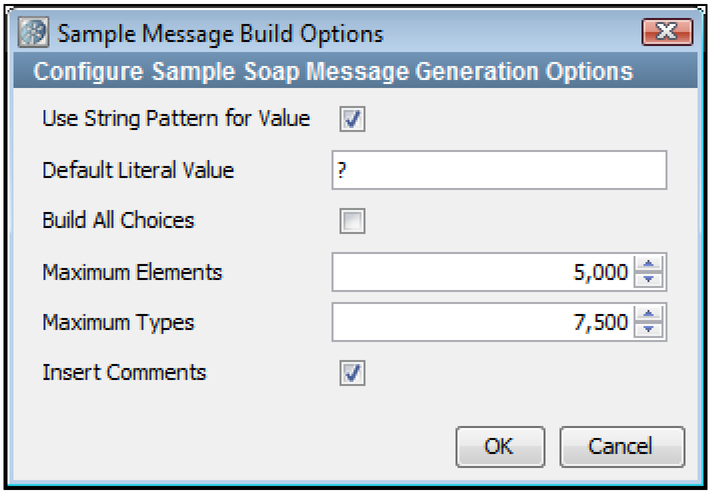
- **Service, Port, Operation**: If the WSDL URL is populated, LISA will process the WSDL and populate the Service, Port, and Operation selection. These optional but recommended fields can be used to build a sample SOAP request message. Selecting a port also updates the Endpoint URL to match the definition in the WSDL. Changing the WSDL URL causes these items to be refreshed, and if the Endpoint URL and SOAP Message are unchanged, they update also to correspond to the new WSDL, service, port, and operation that are selected.

If the Endpoint was changed and no longer matches what is defined in the WSDL, a Warning button appears next to the field. A tooltip on the button indicates what the differences are between the entered value and the WSDL definition. Clicking the button updates the field to match the WSDL definition.

If the SOAP Request Message no longer matches the default, it will not be updated automatically. You can force the SOAP Request Message to be updated by using the Build Message button next to the Operation field.

Operation: Any of the following options can be used here:

- Build empty SOAP request message.
- Build full SOAP request message.
  Build Options: When building sample SOAP messages,various build options are used to determine what to do in various situations.



Sample Message Build Options

Configure Sample Soap Message Generation Options

Use String Pattern for Value ☑

Default Literal Value: ?

Build All Choices □

Maximum Elements: 5,000

Maximum Types: 7,500

Insert Comments ☑

- **Use String Pattern for Value**: When selected, it populates element values using LISA string patterns as opposed to using a hard-coded literal value.
- **Default Literal Value**: When not using string patterns, use this literal value for all string values.
- **Build All Choices**: By default, only the first element in a XML schema choice is generated. Select this option to build all possible choice elements.

> ⚠ It will not be a valid SOAP request if you include more than one choice element, but it will show you a sample for each possible choice, making it easier to build a message when not using the first choice.

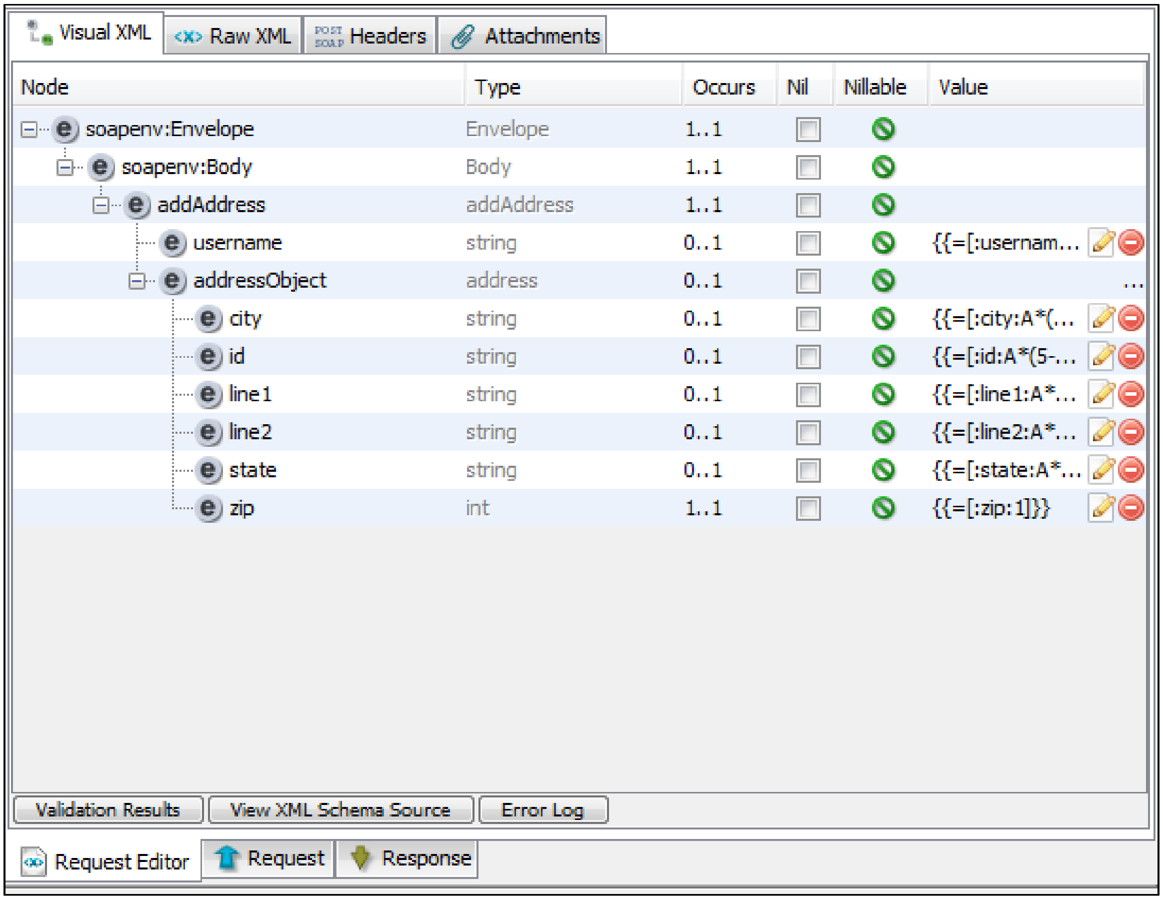
- **Maximum Elements**: This indicates when to give up building the sample message if it would end up creating a enormously large message: how many elements should it stop after.
- **Maximum Type**: This indicates when to give up building the sample message if it would end up creating a enormously large message: how many complex schema types should it stop after.
- **Insert Comments**: By default, comments will be generated related to the schema (for example, when an element is optional, alternative choices, nilable elements, and so forth). You do not see these comments in the Visual XML Editor, but they are visible in the Raw Editor.
- **Port**: This field indicates the server port on which the service is available.
- **On Error**: This field indicates what action to be taken when some error occurs during execution.
- **Endpoint**: The URL to the SAML Query API of the Identity Provider.

Web Service Execution Tabs

Tabs available in the Web Service Execution editor are described here:.

Visual XML Tab

The Visual XML Editor (or VXE) is a graphical editor for any XML document. Because a SOAP message is an XML document that conforms to the SOAP specification (SOAP schema), you can use the editor to build and edit the SOAP message.



Node	Type	Occurs	Nil	Nillable	Value
soapenv:Envelope	Envelope	1..1	☐	⊘	
soapenv:Body	Body	1..1	☐	⊘	
addAddress	addAddress	1..1	☐	⊘	
username	string	0..1	☐	⊘	{{=[:usernam…
addressObject	address	0..1	☐	⊘	…
city	string	0..1	☐	⊘	{{=[:city:A*(…
id	string	0..1	☐	⊘	{{=[:id:A*(5-…
line1	string	0..1	☐	⊘	{{=[:line1:A*…
line2	string	0..1	☐	⊘	{{=[:line2:A*…
state	string	0..1	☐	⊘	{{=[:state:A*…
zip	int	1..1	☐	⊘	{{=[:zip:1]}}

The table shows the XML document including the SOAP Envelope and Body elements.

The Type column shows the type for each element.

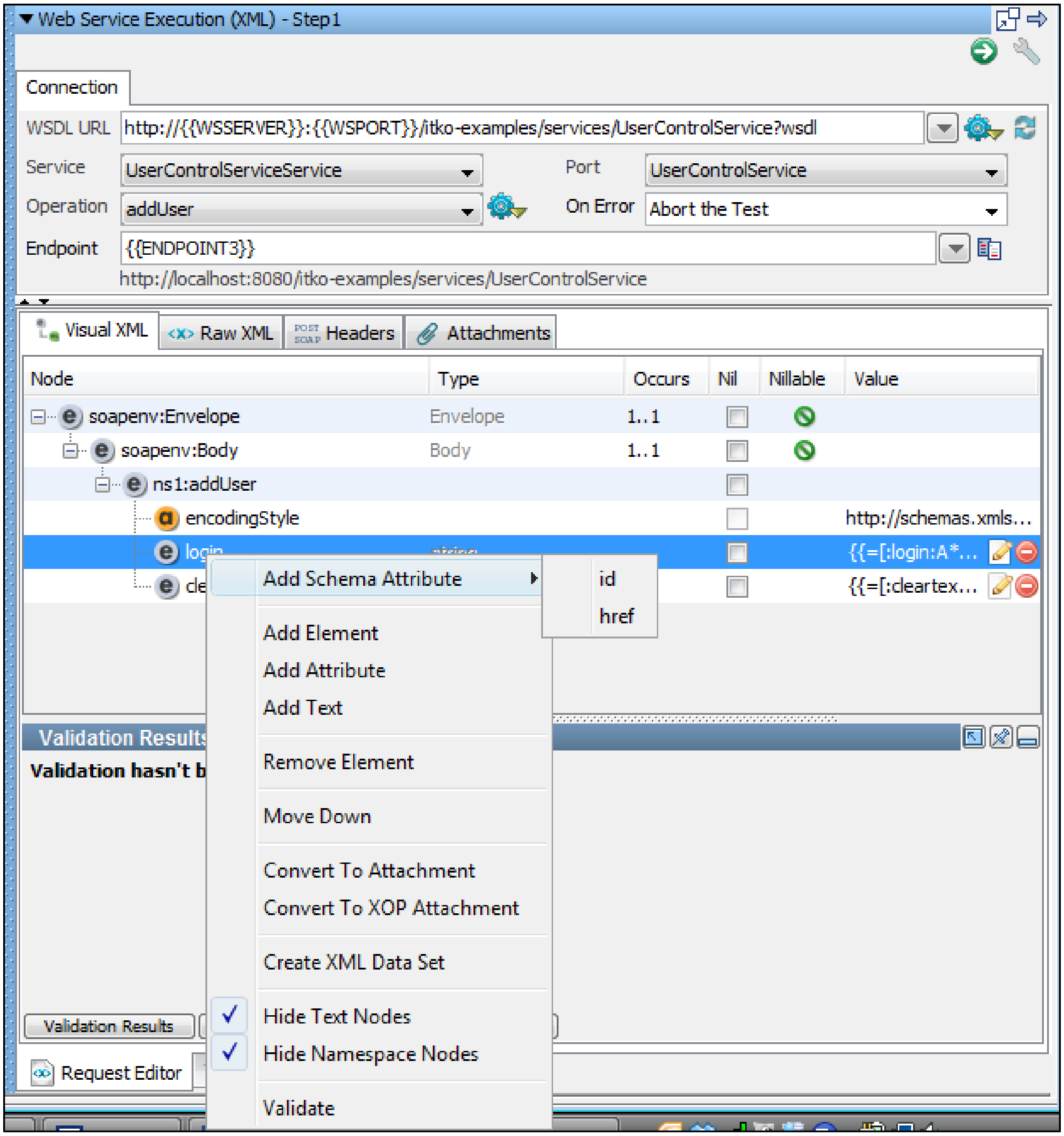
The Occurs column indicates how many elements are expected. The first number indicates the minimum number of times the element can occur. Zero would mean that it is optional and can be removed. The second number indicates the maximum number of times the element can occur. Infinity would mean there can be an unlimited number of elements of that name.

The Nil column lets you nil out or un-nil the element value. Selecting the check box will remove any element children or values but leave all attributes alone. Clearing the check box will populate all expected children and attributes as defined in the WSDL schema.

The Nillable column indicates whether the element can be nil. It will show a red icon if the element is nil, but is non-nillable. It will show a green icon if it is non-nillable and is not nil.

You can type element values directly into the Value column. If the element type is one of a set of known type, specialized edit buttons appear.

When you right-click the editor, a context-sensitive menu provides options for manipulating the document.



Web Service Execution (XML) - Step1

Connection

WSDL URL: http://{{WSSERVER}}:{{WSPORT}}/itko-examples/services/UserControlService?wsdl

Service: UserControlServiceService

Port: UserControlService

Operation: addUser

On Error: Abort the Test

Endpoint: {{ENDPOINT3}}

http://localhost:8080/itko-examples/services/UserControlService

Node	Type	Occurs	Nil	Nillable	Value
soapenv:Envelope	Envelope	1..1	□	⊘	
soapenv:Body	Body	1..1	□	⊘	
ns1:addUser			□		
encodingStyle			□		http://schemas.xmls...
login	string		□		{{=[:login:A*...
cle			□		{{=[:cleartex...

Validation Results

Validation hasn't b

This menu contains the following options:

- **Add Schema Attribute/Element**: Lets you select from a list of valid attributes or elements. If a schema is present, and an element or attribute is selected in the editor, child elements and attributes are populated and can be selected for addition. If more than 20 schema objects are available, a dialog can be used to select the schema object. This dialog contains a search text field, which makes it easier to find a particular schema object when there are dozens of them.
- **Add Element**: Add an element to the document.
- **Add Attribute**: Add an attribute to the document.
- **Add Text**
- **Remove Element/Attribute**: Remove the selected element or attribute.
- **Move Up/Down**: Move the selected node up or down.
- **Convert to Attachment**: A shortcut method for creating a standard referenced attachment. For more information, see Attachments Tab.
- **Convert to XOP Attachment**: A shortcut method for creating a standard referenced XOP Include attachment. For more information, see Attachments Tab.
- **Create XML Data Set**: A shortcut method for generating an XML Data Set. A typical use case would be to build a full SOAP message then select the section of the XML document that you want as the first record of the data set. Creating an XML Data Set will automatically populate the first record with the selected XML element tree and set the new data set LISA property as the value in the editor.
- **Hide Text Nodes**: By default, LISA hides Text Nodes which are typically redundant (for example, white space), but on some occasions it is useful to view them, including when the XML element is of mixed type (element that supports intermixed elements and text).
- **Hide Namespace Nodes**: By default, LISA hides namespace declarations and namespace prefix declarations. You may want to show them to confirm a prefix value or if you want to change prefixes or namespace scoping.

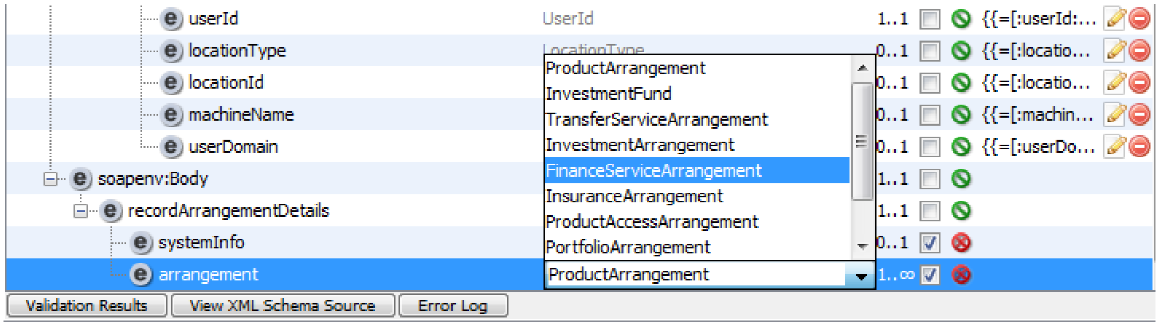
You can also use keyboard shortcuts for certain tasks:

- To remove the selected element or attribute on Windows, press Ctrl+Backspace.
- To move the selected node up on Windows, press Ctrl+up arrow.
- To move the selected node down on Windows, press Ctrl+down arrow.

Editing the Type field

The Type column shows the XML schema type (local name) and has a tooltip to show the fully qualified name (qName) with namespace. This column is editable for derived XSD types. Only base types with derived types can be edited.

When editable, a list of available derived types and the base types are presented in a combo box. You can select a type, and the associated element will now be associated with the selected type.



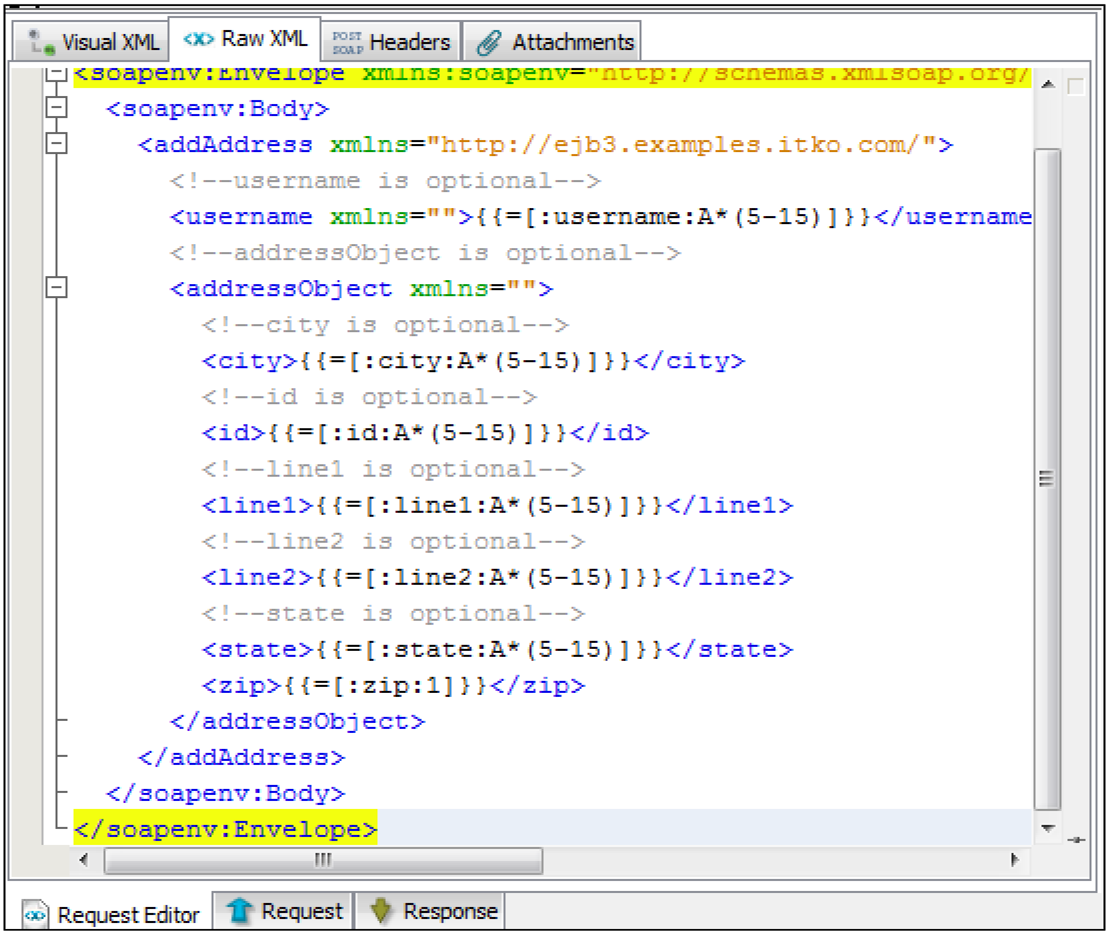
userId	UserId	1..1	□	{{=[:userId:…
locationType	LocationType	0..1	□	{{=[:locatio…
locationId		0..1	□	{{=[:locatio…
machineName		0..1	□	{{=[:machin…
userDomain		0..1	□	{{=[:userDo…
soapenv:Body		1..1	□	
recordArrangementDetails		1..1	□	
systemInfo		0..1	☑	
arrangement	ProductArrangement	1..∞	☑	

- ProductArrangement
- InvestmentFund
- TransferServiceArrangement
- InvestmentArrangement
- FinanceServiceArrangement
- InsuranceArrangement
- ProductAccessArrangement
- PortfolioArrangement

Changing the type will remove all child elements and attributes from an element and set the nil attribute on the element to nil=true.

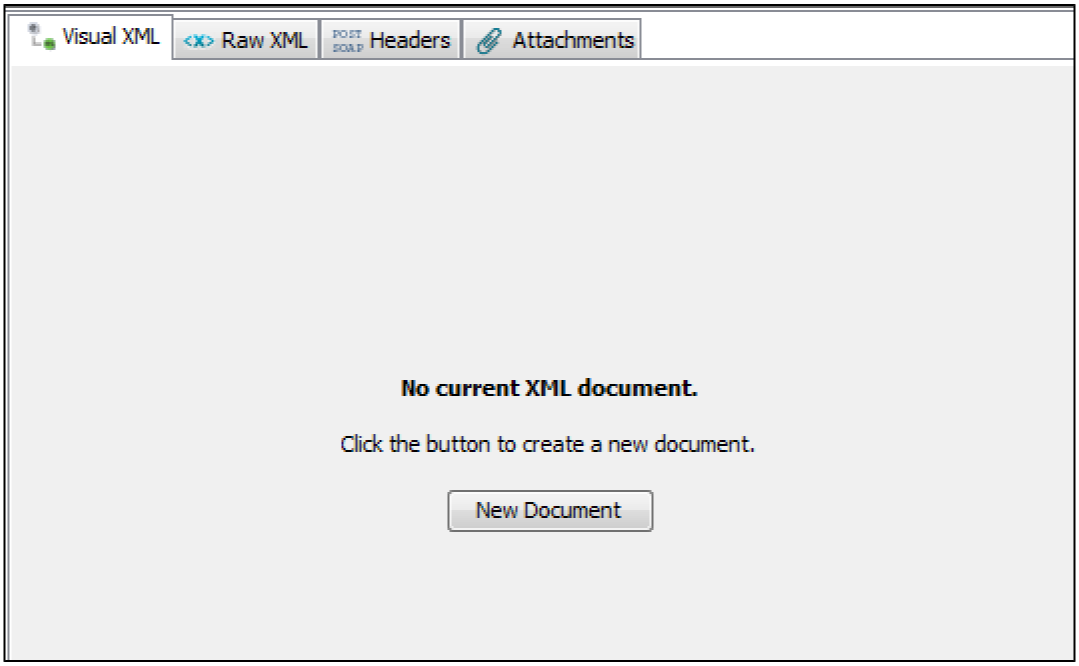
Raw XML Tab

The Raw XML Editor is a text-based editor that is XML aware and lets you manually edit the raw XML SOAP message. Any changes that are made will be seen when you switch back to the Visual XML Editor (and conversely).



```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/
  <soapenv:Body>
    <addAddress xmlns="http://ejb3.examples.itko.com/">
      <!--username is optional-->
      <username xmlns="">{{=[:username:A*(5-15)]}}</username
      <!--addressObject is optional-->
      <addressObject xmlns="">
        <!--city is optional-->
        <city>{{=[:city:A*(5-15)]}}</city>
        <!--id is optional-->
        <id>{{=[:id:A*(5-15)]}}</id>
        <!--line1 is optional-->
        <line1>{{=[:line1:A*(5-15)]}}</line1>
        <!--line2 is optional-->
        <line2>{{=[:line2:A*(5-15)]}}</line2>
        <!--state is optional-->
        <state>{{=[:state:A*(5-15)]}}</state>
        <zip>{{=[:zip:1]}}</zip>
      </addressObject>
    </addAddress>
  </soapenv:Body>
</soapenv:Envelope>
```

If you make an edit that causes the document to no longer be a valid XML document, the Visual XML Editor might show an error message when you switch back to it.



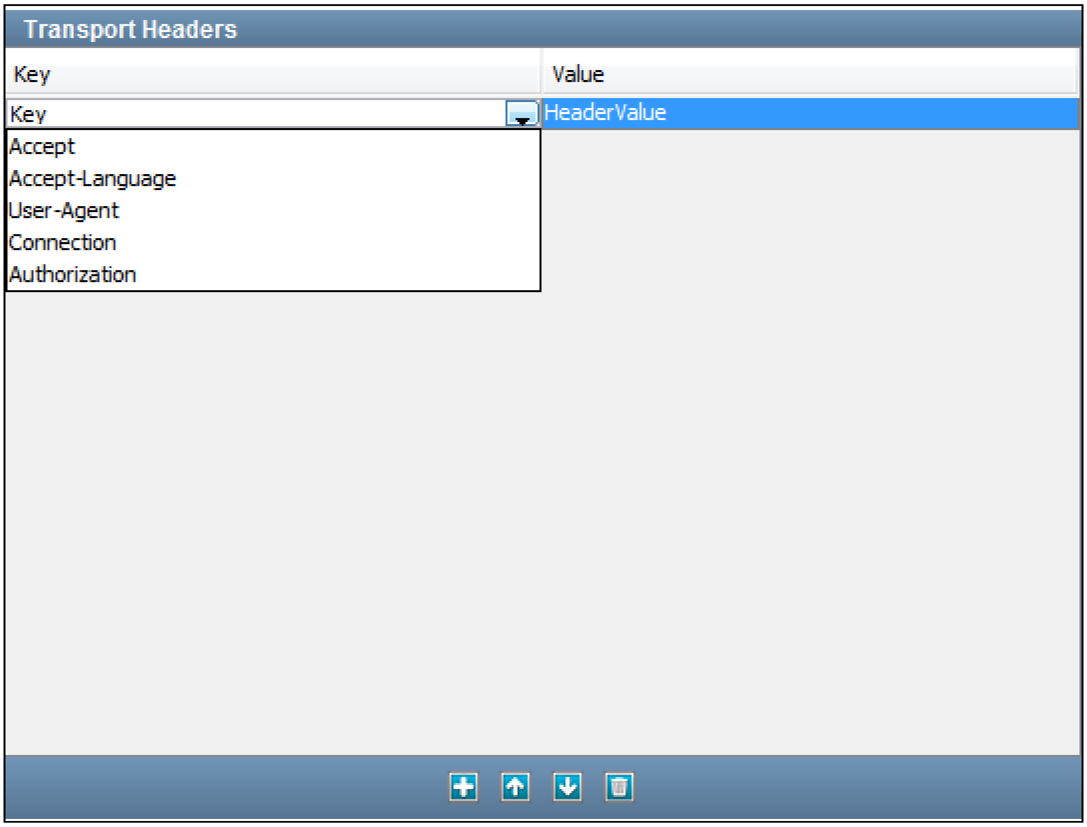
No current XML document.

Click the button to create a new document.

Fix your changes in the Raw XML Editor, and the Visual XML Editor will begin working again.

Headers Tab

The Headers tab lets you insert headers that will be transmitted with the SOAP message (for example, HTTP Headers or JMS properties).



Key	Value
Key	HeaderValue

Click the ➕ sign to add a header row and select a header from the drop-down list.

- Accept
- Accept - Language
- User - Agent
- Connection

- Authorization

Accept

The Accept request-header field can be used to specify certain media types that are acceptable for the response. Accept headers can be used to indicate that the request is specifically limited to a small set of desired types, as in the case of a request for an in-line image.

This field contains a semicolon-separated list of representation schemes that will be accepted in the response to this request.

```
Accept         = "Accept" ":"

                #( media-range [ accept-params ] )
```

If no Accept: field is present, then it is assumed that text/plain and text/html are accepted.

Accept - Language

The Accept - Language header field is similar to Accept, but lists the Language values that are preferable in the response. A response in an unspecified language is not illegal.

```
Accept-Language = "Accept-Language" ":"

 1#( language-range [ ";" "q" "=" qvalue ] )

   language-range  = ( ( 1*8ALPHA *( "-" 1*8ALPHA ) ) | "*" )
```

User - Agent

The User-Agent request-header field contains information about the user agent originating the request.

This is for statistical purposes, the tracing of protocol violations, and automated recognition of user agents for the sake of tailoring responses to avoid particular user agent limitations. User agents *should* include this field with requests.

By convention, the product tokens are listed in order of their significance for identifying the application.

```
User-Agent     = "User-Agent" ":" 1*( product | comment
```

Connection

The Connection general-header field allows the sender to specify options that are desired for that particular connection and *must not* be communicated by proxies over further connections.

```
Connection = "Connection" ":" 1#(connection-token)
connection-token  = token
```

Message headers listed in the Connection header *must not* include end-to-end headers, such as Cache-Control.

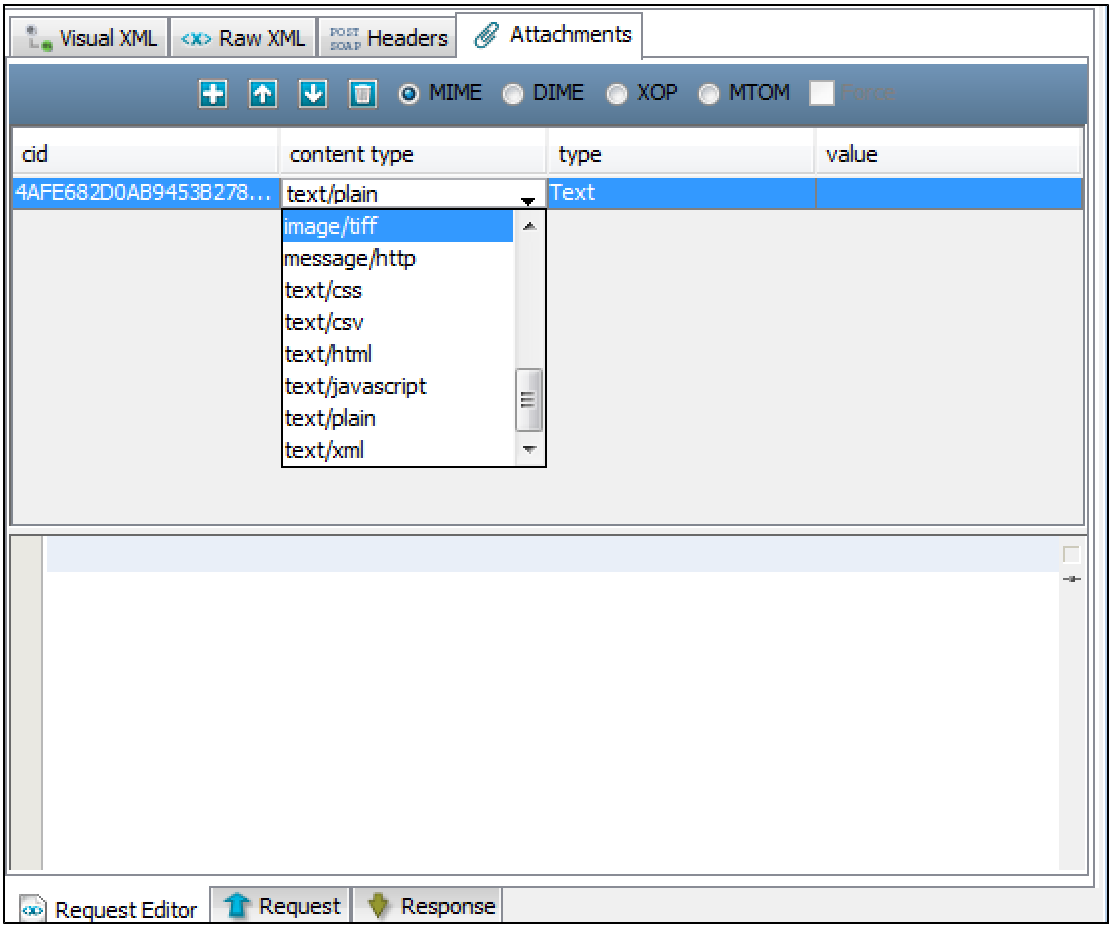
Authorization

A user agent that wishes to authenticate itself with a server usually, but not necessarily, after receiving a 401 response, does so by including an Authorization request-header field with the request. The Authorization field value consists of credentials containing the authentication information of the user agent for the realm of the resource being requested.

```
Authorization  = "Authorization" ":" credential
```

Attachments Tab

Explicitly showing attachments is one of the major usability differences from the legacy Web Service Execution step. In the legacy step, attachments were handled automatically by the generated Java classes, but it could be difficult to configure the necessary Java objects to use them properly. There is now an explicit tab dedicated to the editing of Attachment data (and a tab to show Sent/Received attachments in the Request/Response tabs described in the following sections).



● MIME ○ DIME ○ XOP ○ MTOM □ Force

cid	content type	type	value
4AFE682D0AB9453B278...	text/plain	Text	

Referenced Attachments

If you plan to use referenced attachments (attachments referenced in the SOAP message), in the VXE you can right-click on the element that you want to be the references attachment and select to Convert to Attachment (for MIME and DIME) or Convert to XOP Attachment (for MTOM/XOP Include style). This will automatically create the necessary elements and attributes and configure the content id used to match up the element with the attachment. It will then switch over to the Attachments tab and pre-populate a new attachment with the content id, select a default content type and attachment type, and populate the value with any existing element data from the VXE.

Unreferenced Attachments

If you plan to use unreferenced (that is, anonymous) attachments, switch over the Attachments tab and add an attachment manually.

Use the Add, Up, Down, and Delete icons to add, remove, or rearrange the attachments in the table.

- **MIME DIME XOP MTOM**: This controls how the attachments are sent, either using MIME, DIME, XOP. or MTOM standards. XOP sends different content headers based on the SOAP version.

When MTOM is selected, any base64binary schema types are automatically optimized using the XOP standard. There is no need to manually add attachments. If an element is already configured as an attachment it will be left alone. Any extra attachments added manually will also be sent.

If **Force** is selected, even if there are no base64binary elements in the document, the document will be formatted and sent as an attachment (Microsoft MTOM method).

The limitation of automatically optimizing elements is that the element must be understood by the VXE as a base64binary schema type (or extension/restriction thereof). In the case where a Data Set or Property is used, the expanded Property or first Data Set entry must contain all possible elements that need to be optimized (i.e., if the element that needs to be optimized isn't displayed in the VXE it won't be optimized).

- **cid**: The Content ID that can be used in a **href** attribute in the SOAP message to link the element to the attachment data
- **content type**: The mime encoding type used to assist the server in how to process the attachment data
- **type value**: The LISA attachment type determines how to edit and interpret the value data. Each type has its own editor.

Type Editors

- **XML**: An XML-aware text editor to edit the attachment value.
- **Text**: A text-based editor to edit the attachment value.
- **Base64 Encoded**: A a text-based editor to edit the attachment value and a bytes viewer to view the decoded binary data.
- **Hex Encoded**: A text-based editor to edit the attachment value and a bytes viewer to view the decoded binary data.

- **URL/Text**: A URL field to edit the attachment value and a text data viewer to view the results of loading the data from the URL.
- **URL/XML**: A URL field to edit the attachment value and a XML-aware text data viewer to view the results of loading the data from the URL.
- **URL/Binary**: A URL field to edit the attachment value and a binary data viewer to view the results of loading the data from the URL.
- **Property**: A property field to edit the attachment value. If the resulting property is a string, it will send the attachment as text, otherwise it will send it as binary data.
- **Property/URL**: A property field to edit the attachment value. The property value is assumed to be a URL. The URL content is loaded and sent as the attachment data.

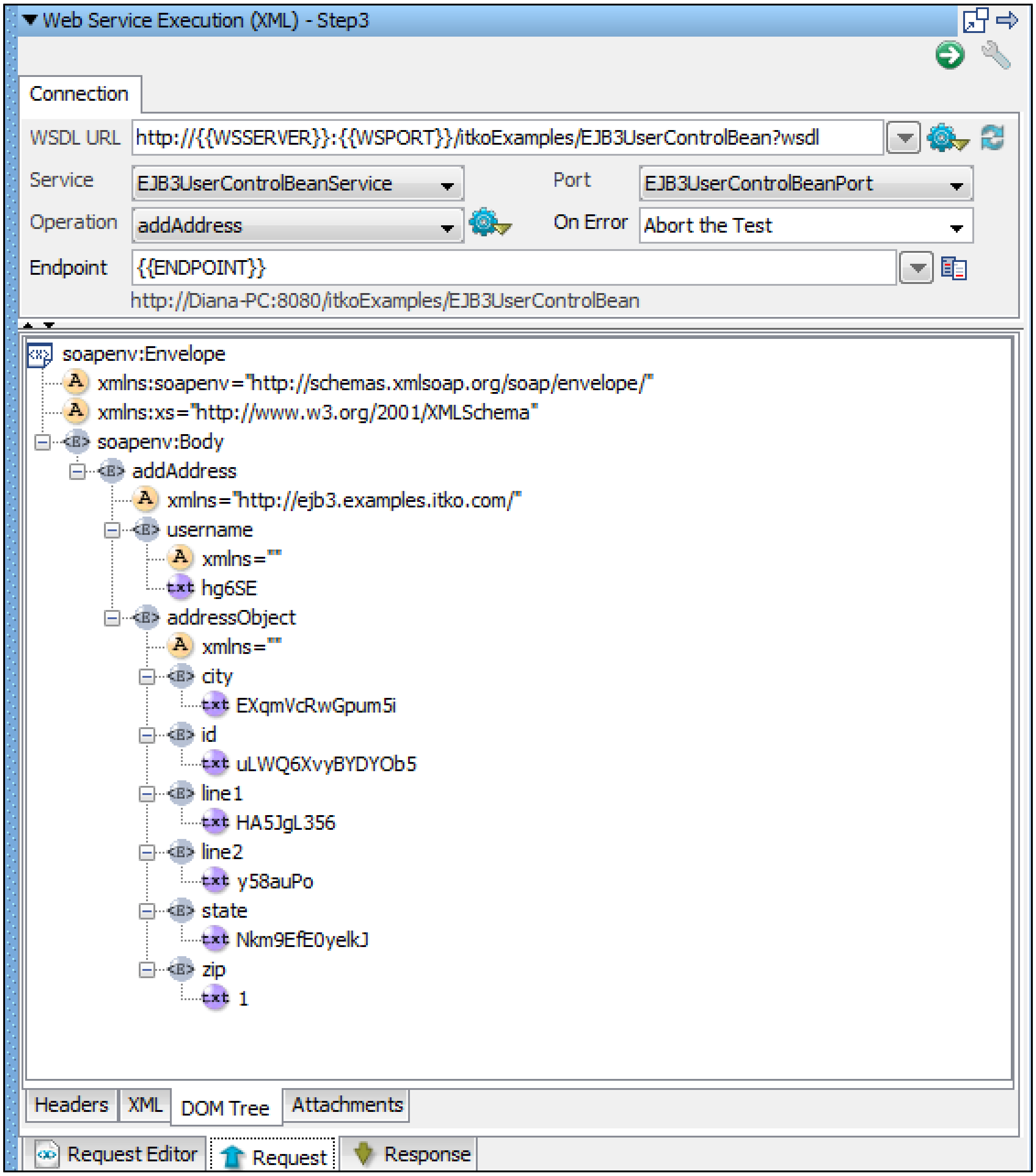
Design Time Execution

Now that you have completed configuring the connection information and building the SOAP Request Message you can now test the step by executing it at design time.

Execute the Web Service operation by clicking the Execute button ➔ in the upper right corner. After it has executed, the Request and Response tabs will be populated and it will switch to the Response tab automatically.

Request Tab

The Request tab will show the resulting request data that was sent after any post processing (for example, substituting LISA properties). If the message contained any attachment, it will not show the raw MIME or DIME encoded message, but rather the processed message and attachments. If you want to see the raw message, use a tool like TCPMon.



Web Service Execution (XML) - Step3

WSDL URL http://{{WSSERVER}}:{{WSPORT}}/itkoExamples/EJB3UserControlBean?wsdl

Service EJB3UserControlBeanService

Port EJB3UserControlBeanPort

Operation addAddress

On Error Abort the Test

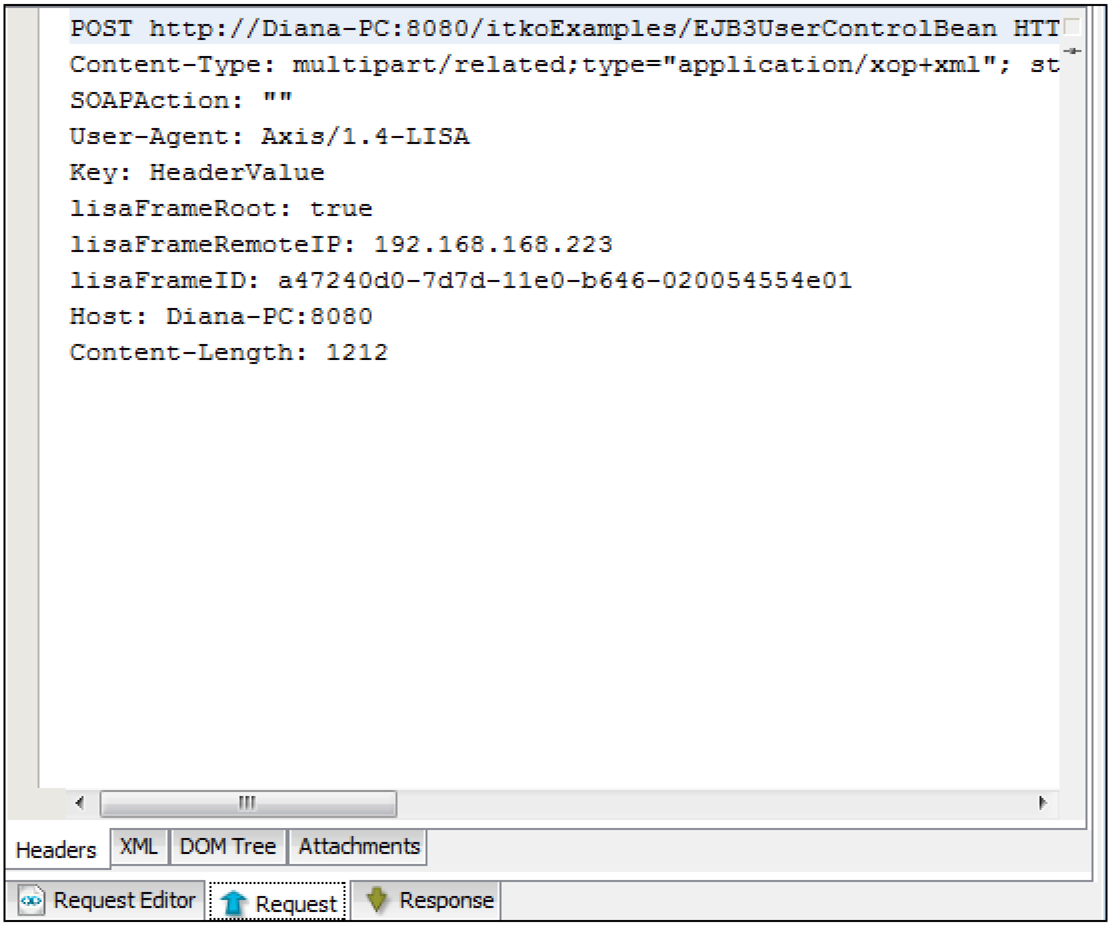
Endpoint {{ENDPOINT}}

http://Diana-PC:8080/itkoExamples/EJB3UserControlBean

```
soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  soapenv:Body
    addAddress
      xmlns="http://ejb3.examples.itko.com/"
      username
        xmlns=""
        hg6SE
      addressObject
        xmlns=""
        city
          EXqmVcRwGpum5i
        id
          uLWQ6XvyBYDYOb5
        line1
          HA5JgL356
        line2
          y58auPo
        state
          Nkm9EfE0yelkJ
        zip
          1
```

Header Tab

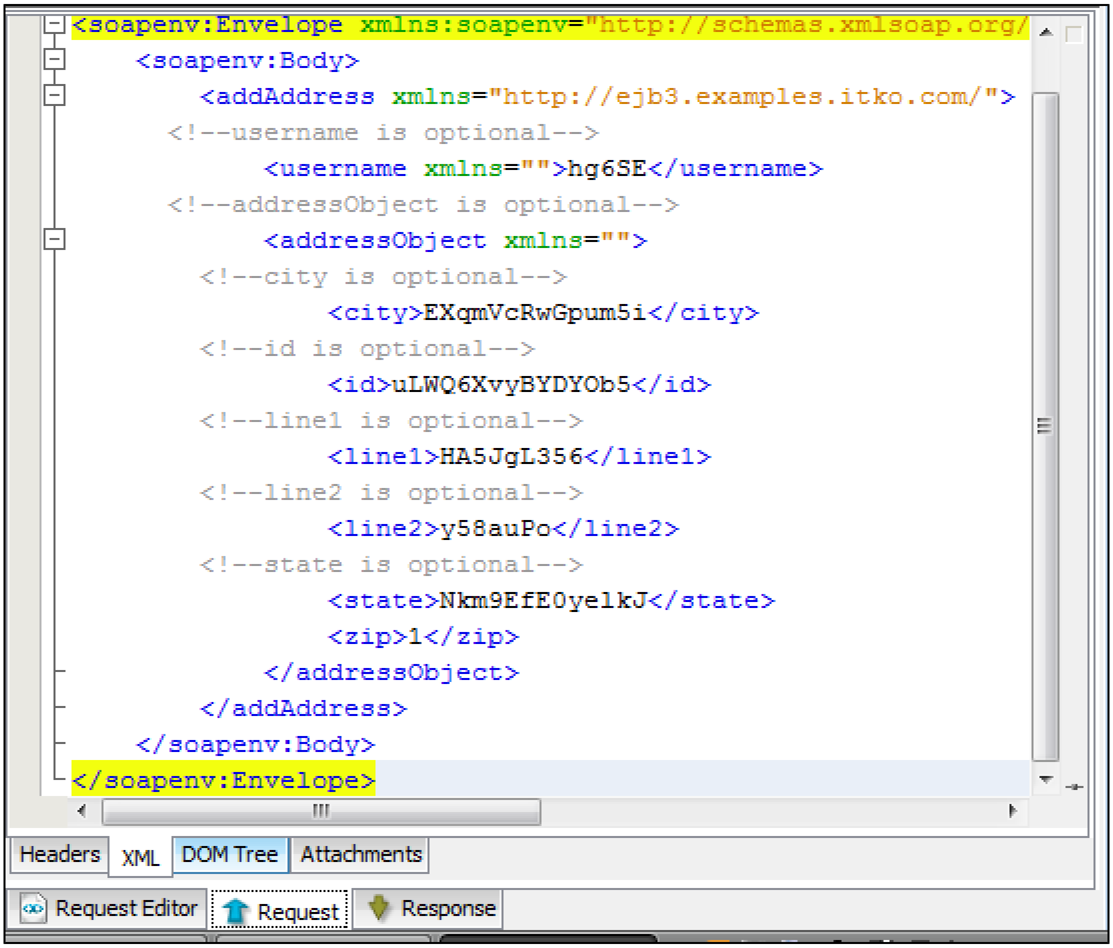
The Header tab shows the Transport Headers that were sent for the request.



```
POST http://Diana-PC:8080/itkoExamples/EJB3UserControlBean HTT
Content-Type: multipart/related;type="application/xop+xml"; st
SOAPAction: ""
User-Agent: Axis/1.4-LISA
Key: HeaderValue
lisaFrameRoot: true
lisaFrameRemoteIP: 192.168.168.223
lisaFrameID: a47240d0-7d7d-11e0-b646-020054554e01
Host: Diana-PC:8080
Content-Length: 1212
```

XML Tab

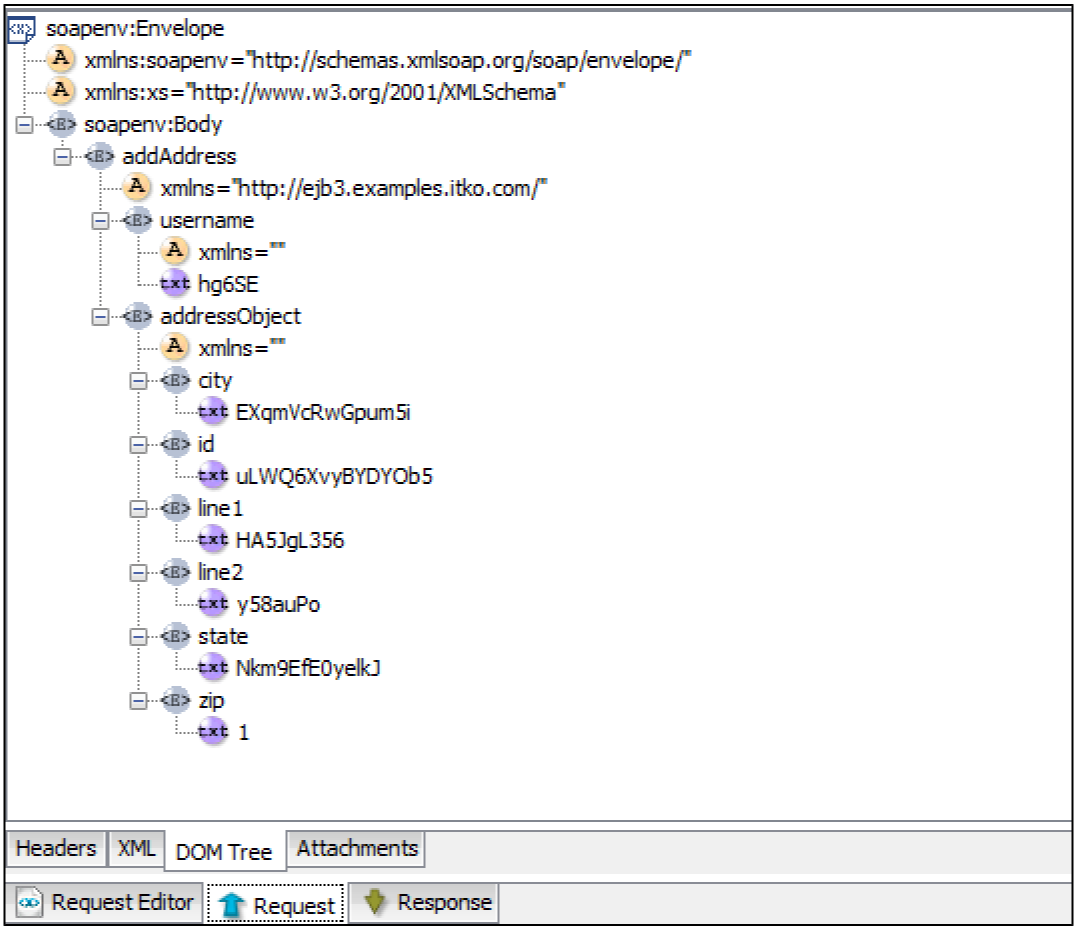
The XML tab shows the raw SOAP message that was sent after any advanced processing.



```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/
    <soapenv:Body>
        <addAddress xmlns="http://ejb3.examples.itko.com/">
      <!--username is optional-->
            <username xmlns="">hg6SE</username>
      <!--addressObject is optional-->
            <addressObject xmlns="">
        <!--city is optional-->
                <city>EXqmVcRwGpum5i</city>
        <!--id is optional-->
                <id>uLWQ6XvyBYDYOb5</id>
        <!--line1 is optional-->
                <line1>HA5JgL356</line1>
        <!--line2 is optional-->
                <line2>y58auPo</line2>
        <!--state is optional-->
                <state>Nkm9EfE0yelkJ</state>
                <zip>1</zip>
            </addressObject>
        </addAddress>
    </soapenv:Body>
</soapenv:Envelope>
```

DOM Tree Tab

The DOM Tree tab shows a DOM tree for the SOAP message.



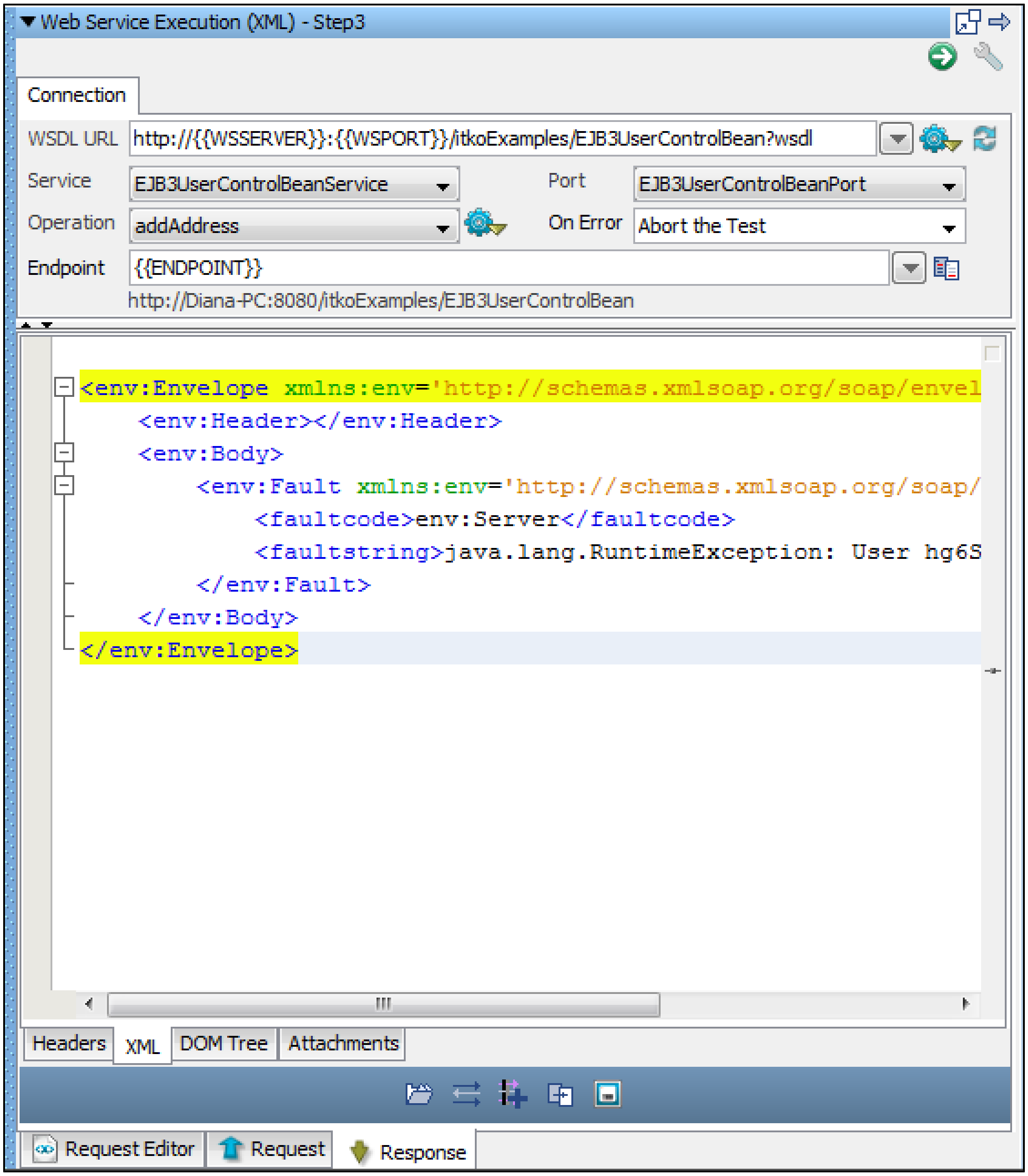
```
soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  soapenv:Body
    addAddress
      xmlns="http://ejb3.examples.itko.com/"
      username
        xmlns=""
        hg6SE
      addressObject
        xmlns=""
        city
          EXqmVcRwGpum5i
        id
          uLWQ6XvyBYDYOb5
        line1
          HA5JgL356
        line2
          y58auPo
        state
          Nkm9EfE0yelkJ
        zip
          1
```

Attachments Tab

The Attachments tab shows any attachments that were sent.

Response Tab

The Response tab show the resulting response data that was received. If the message contained any attachment, it will not show the raw MIME or DIME encoded message, but rather the processed message and attachments. To see the raw message, use a tool like TCPMon. If there are any advanced post-processing options set (see the following window), the SOAP response message will be shown post-processed.



WSDL URL http://{{WSSERVER}}:{{WSPORT}}/itkoExamples/EJB3UserControlBean?wsdl

Service EJB3UserControlBeanService

Port EJB3UserControlBeanPort

Operation addAddress

On Error Abort the Test

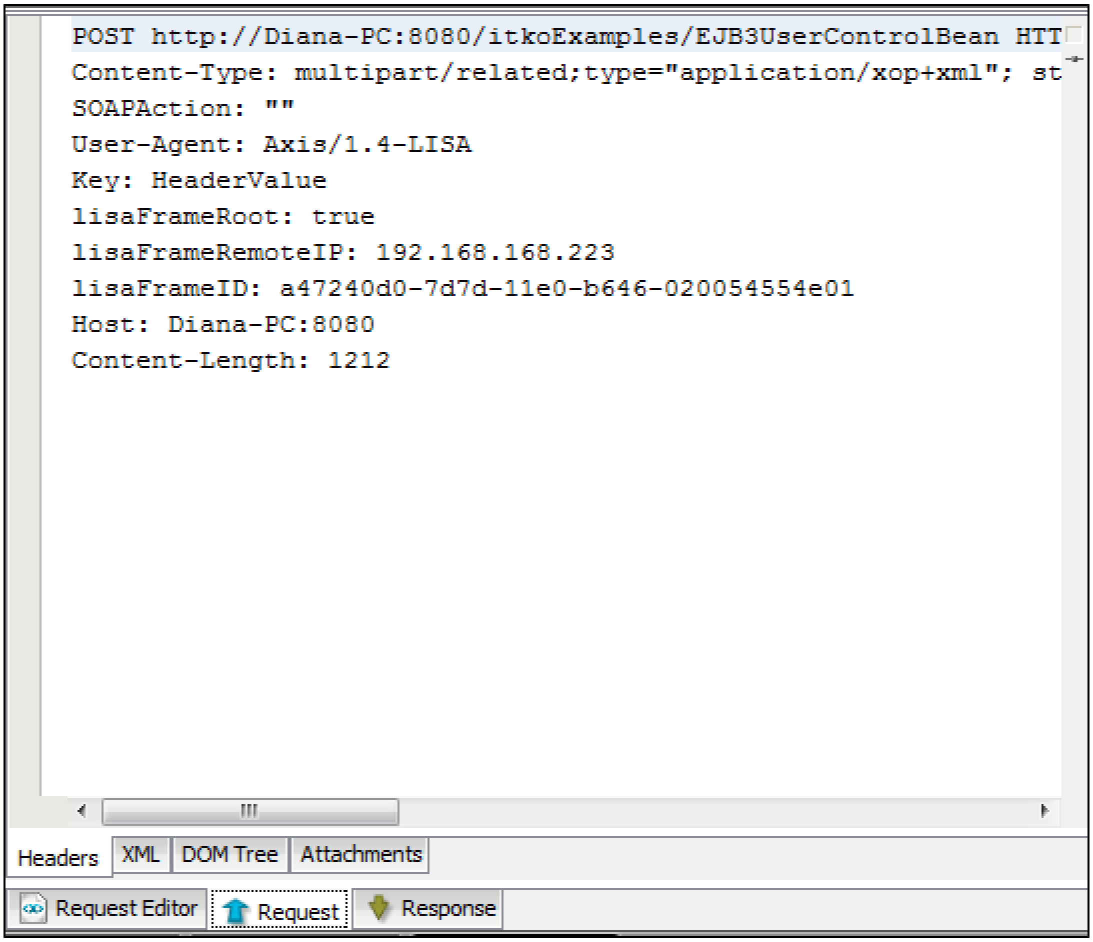
Endpoint {{ENDPOINT}}

http://Diana-PC:8080/itkoExamples/EJB3UserControlBean

```
<env:Envelope xmlns:env='http://schemas.xmlsoap.org/soap/envel
    <env:Header></env:Header>
    <env:Body>
        <env:Fault xmlns:env='http://schemas.xmlsoap.org/soap/
            <faultcode>env:Server</faultcode>
            <faultstring>java.lang.RuntimeException: User hg6S
        </env:Fault>
    </env:Body>
</env:Envelope>
```

Header Tab

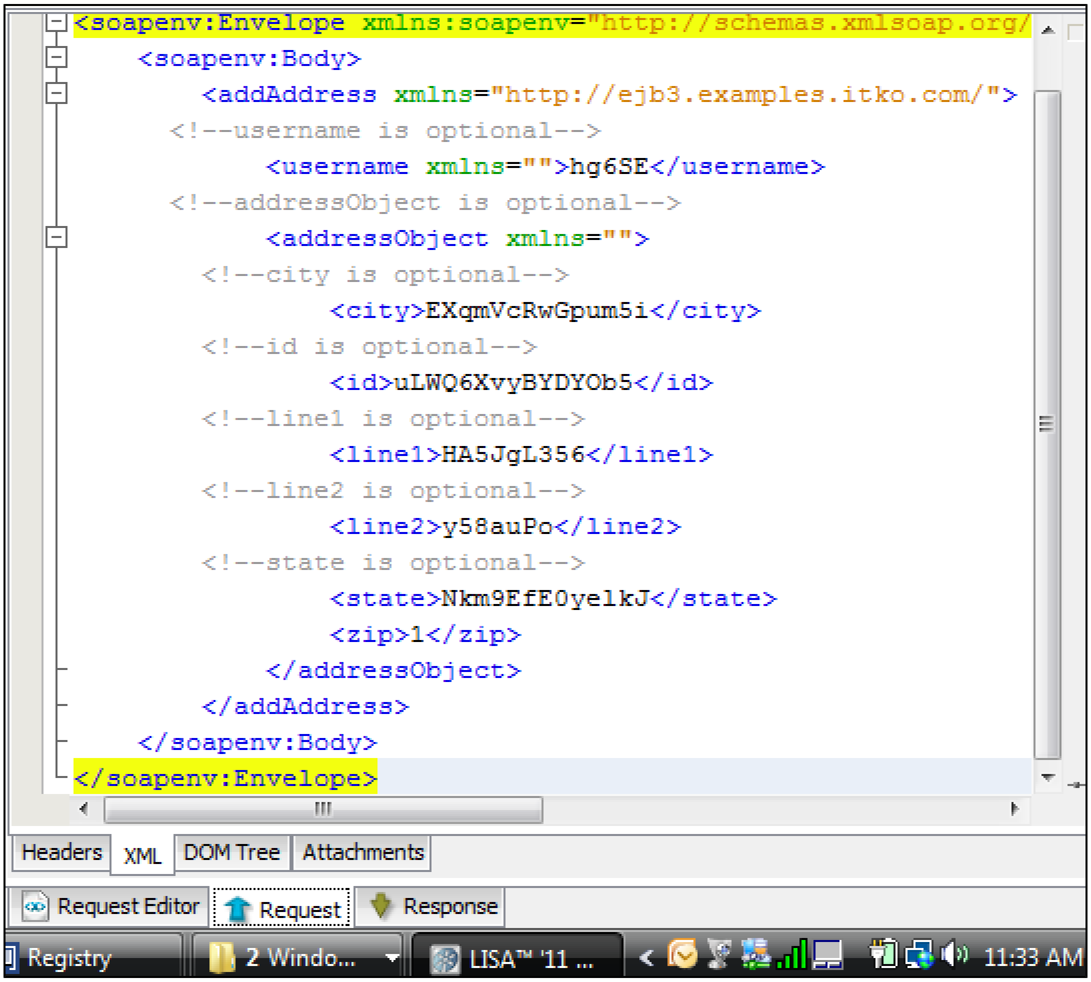
The Header tab shows the Transport Headers that were received from the response.



```
POST http://Diana-PC:8080/itkoExamples/EJB3UserControlBean HTT
Content-Type: multipart/related;type="application/xop+xml"; st
SOAPAction: ""
User-Agent: Axis/1.4-LISA
Key: HeaderValue
lisaFrameRoot: true
lisaFrameRemoteIP: 192.168.168.223
lisaFrameID: a47240d0-7d7d-11e0-b646-020054554e01
Host: Diana-PC:8080
Content-Length: 1212
```

XML Tab

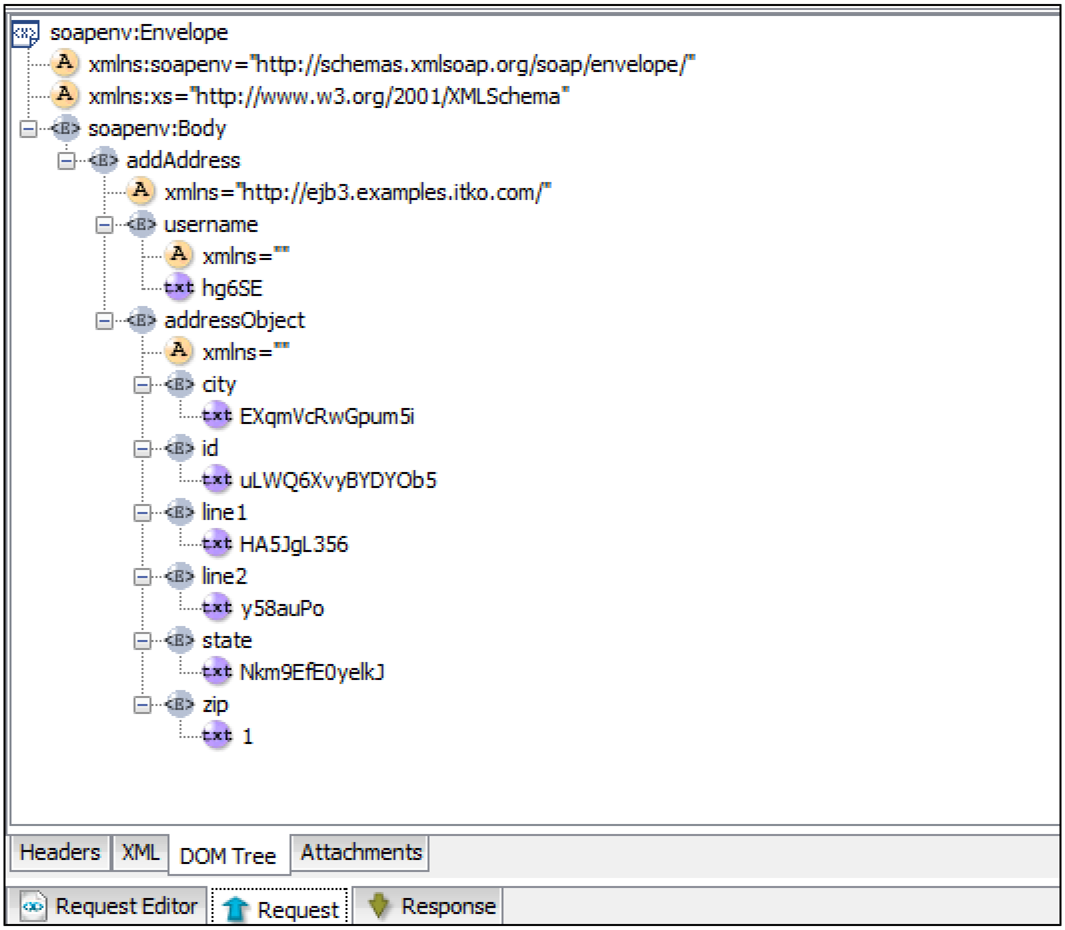
The XML tab shows the raw SOAP message that was received after any advanced processing.



```xml
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/
    <soapenv:Body>
        <addAddress xmlns="http://ejb3.examples.itko.com/">
      <!--username is optional-->
            <username xmlns="">hg6SE</username>
      <!--addressObject is optional-->
            <addressObject xmlns="">
        <!--city is optional-->
                <city>EXqmVcRwGpum5i</city>
        <!--id is optional-->
                <id>uLWQ6XvyBYDYOb5</id>
        <!--line1 is optional-->
                <line1>HA5JgL356</line1>
        <!--line2 is optional-->
                <line2>y58auPo</line2>
        <!--state is optional-->
                <state>Nkm9EfE0yelkJ</state>
                <zip>1</zip>
            </addressObject>
        </addAddress>
    </soapenv:Body>
</soapenv:Envelope>
```

DOM Tree Tab

The DOM Tree tab shows a DOM tree for the SOAP message. From this tab you can quickly add filters and assertions on the resulting SOAP message.



```
soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    soapenv:Body
        addAddress
            xmlns="http://ejb3.examples.itko.com/"
            username
                xmlns=""
                hg6SE
            addressObject
                xmlns=""
                city
                    EXqmVcRwGpum5i
                id
                    uLWQ6XvyBYDYOb5
                line1
                    HA5JgL356
                line2
                    y58auPo
                state
                    Nkm9EfE0yelkJ
                zip
                    1
```

Attachments Tab

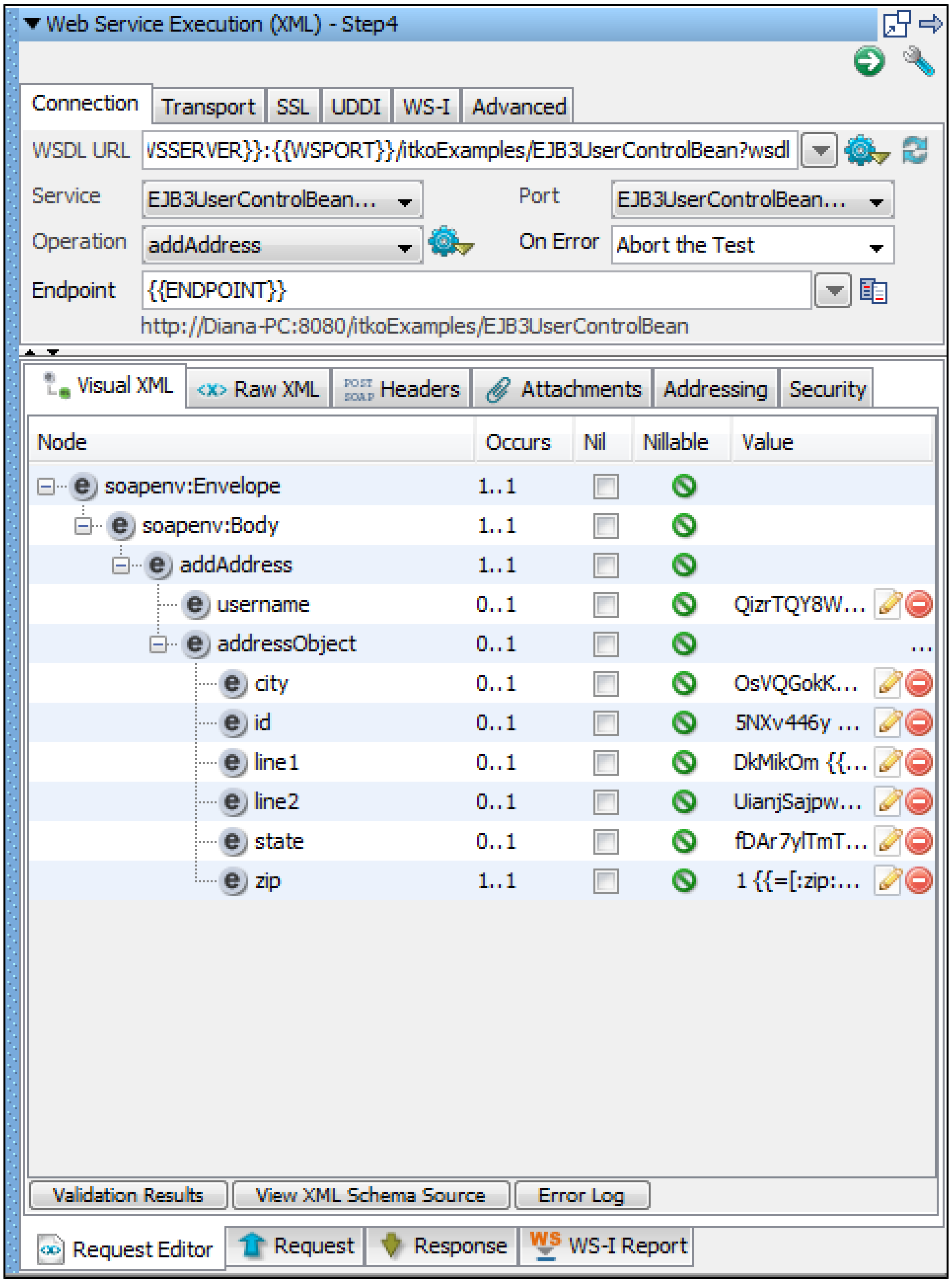
The Attachments tab shows any attachments that were received. Received attachments can be accessed using automatically generated LISA properties (for use in later steps, filters, or assertions). For each attachment the following LISA properties are set.

- **lisa.<step name>.rsp.attachment.<cid>**: Attachment value, byte or String
- **lisa.<step name>.rsp.attachment.contenttype.<cid>**: Content type (mime type, for example, text/plain)
- **lisa.<step name>.rsp.attachment.<index>**: Attachment value, byte or String
- **lisa.<step name>.rsp.attachment.contenttype.<index>**: Content type (mime type, for example, text/plain)
- **lisa.<step name>.rsp.attachment.contentid.<index>**: Content Id (cid)

The parameter **<step name>** is the LISA step name that is being executed.
The parameter **<cid>** is the Content ID that is usually referenced in the SOAP message.
The parameter **<index>** starts at 0 and is increased for each attachment in the response attachments list.

Advanced Settings

Click the Toggle Advanced Options icon to open the Advanced settings tabs. Five new tabs open in the top of the panel, and two new tabs open at the bottom level.



▼ Web Service Execution (XML) - Step4

Connection | Transport | SSL | UDDI | WS-I | Advanced

WSDL URL: VSSERVER}}:{{WSPORT}}/itkoExamples/EJB3UserControlBean?wsdl

Service: EJB3UserControlBean...

Port: EJB3UserControlBean...

Operation: addAddress

On Error: Abort the Test

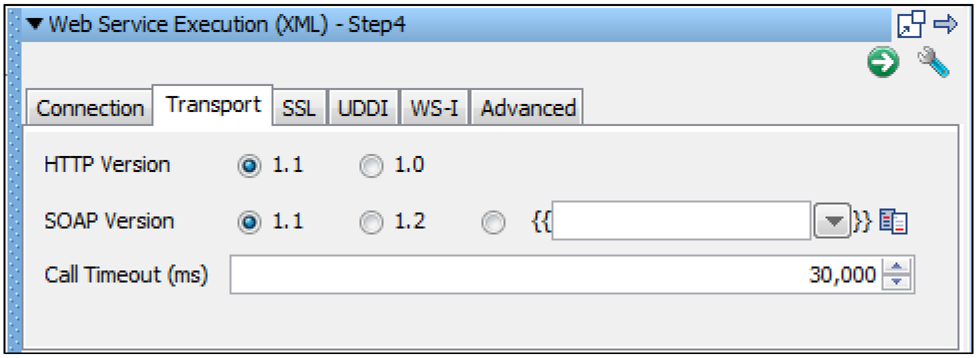
Endpoint: {{ENDPOINT}}

http://Diana-PC:8080/itkoExamples/EJB3UserControlBean

Visual XML | Raw XML | Headers | Attachments | Addressing | Security

Node	Occurs	Nil	Nillable	Value
soapenv:Envelope	1..1	□	⊘	
soapenv:Body	1..1	□	⊘	
addAddress	1..1	□	⊘	
username	0..1	□	⊘	QizrTQY8W...
addressObject	0..1	□	⊘	...
city	0..1	□	⊘	OsVQGokK...
id	0..1	□	⊘	5NXv446y ...
line1	0..1	□	⊘	DkMikOm {{...
line2	0..1	□	⊘	UianjSajpw...
state	0..1	□	⊘	fDAr7ylTmT...
zip	1..1	□	⊘	1 {{=[:zip:...

Transport Tab



▼ Web Service Execution (XML) - Step4

Connection | Transport | SSL | UDDI | WS-I | Advanced

HTTP Version: ● 1.1 ○ 1.0

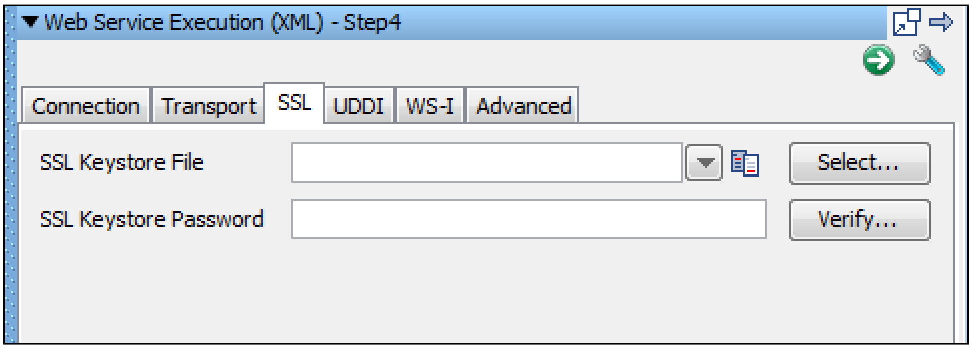
SOAP Version: ● 1.1 ○ 1.2 ○ {{ }}

Call Timeout (ms): 30,000

- **HTTP Version**: This is 1.1 by default and controls which HTTP protocol is used when sending the operation request.
- **SOAP Version**: This is auto-populated based on the WSDL definition. SOAP Version will control the generation of a number of transport headers (for example, SOAPAction and contentType).
- **Call Timeout (ms)**: This defines how to wait while trying to execute the operation. After the timeout is hit, an exception will be thrown and

the On Error handling will occur.

SSL Tab



- **SSL Keystore File**: The name of the keystore file where the client identity certificate is stored. It can be in jks or pkcs format.
- **SSL Keystore Password**: The password for the keystore.

Specify global certificates properties for SSL in your **local.properties** file.

For global certificates (web server, raw SOAP, and web service steps):

- **ssl.client.cert.path**: A full path to the keystore.
- **ssl.client.cert.pass**: Password for the keystore (this password will be automatically encrypted when LISA runs).
- **ssl.client.key.pass**: An optional password for the key entry if you are using the JKS keystore and the key has different password from the keystore. This password will be automatically encrypted using AES (Advanced Encryption Standard) when LISA runs.

> ⚠ This is currently not an available option to be set in the WS Test step and if required must be set in the **local.properties** file.

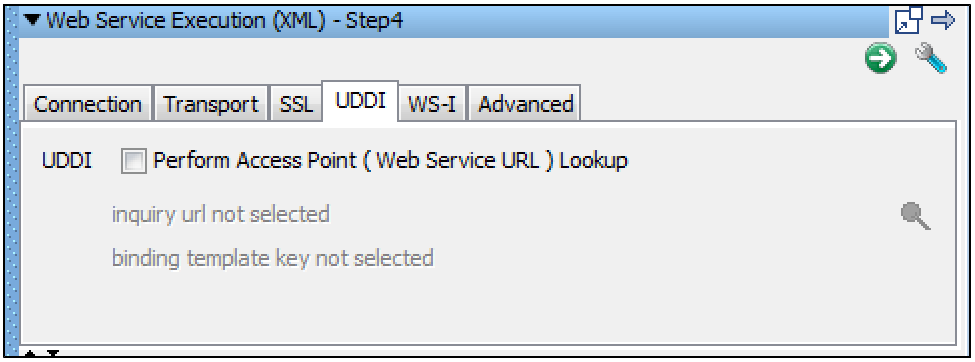
For web service steps only certificates (not raw SOAP steps):

- **ws.ssl.client.cert.path**: A full path to the keystore.
- **ws.ssl.client.cert.pass**: Password for the keystore (this password will be automatically encrypted when Lisa runs).
- **ws.ssl.client.key.pass**: An optional password for the key entry if you are using the JKS keystore and the key has different password from the keystore. This password will be automatically encrypted using AES (Advanced Encryption Standard) when LISA runs.

> ⚠ This is currently not an available option to be set in the WS Test step and if required must be set in the **local.properties** file.

> ⚠ If you have duplicate values in **local.properties** and in the general tab, the values in the general tab will be used.

UDDI Tab

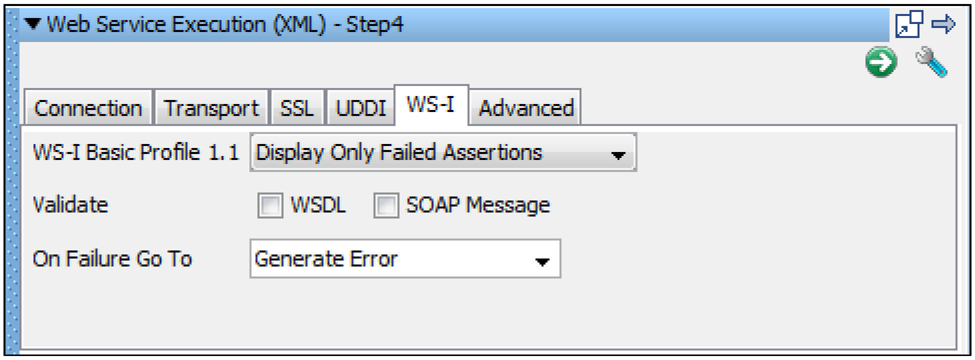

Web Service Execution (XML) - Step4
Connection
Transport
SSL
UDDI
WS-I
Advanced
UDDI
□ Perform Access Point (Web Service URL) Lookup
inquiry url not selected
binding template key not selected

- **Perform Access Point (Web Service URL) Lookup**

  Select the Inquiry URL and Binding Template: use the Search UDDI Server Find 🔍 button to navigate to the correct Binding Template to perform the lookup.

> When creating the step, if you used the UDDI Search function when specifying the Web Service WSDL URL, these values will be automatically filled in. If the Inquiry URL is specified but the Binding Template is not, you may have performed a Model search. To locate the Binding Template you must perform a search at a higher level of the hierarchy and drill down to the TModel through a particular Binding Template.

WS-I Tab



WS-I Basic Profile 1.1 Display Only Failed Assertions

Validate □ WSDL □ SOAP Message

On Failure Go To Generate Error

- **WS-I Basic Profile 1.1**:

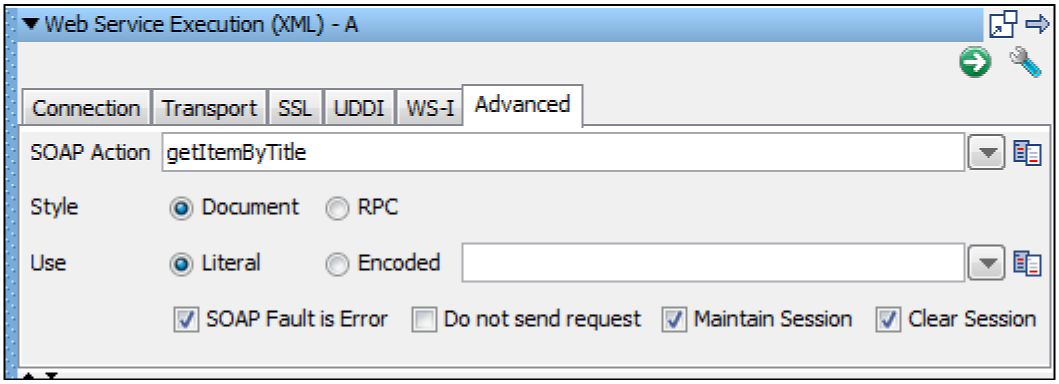
You can choose four different validation levels in the pull-down menu.

- Display All Assertions
- Display All But Info Assertions
- Display Only Failed Assertions
- Display Only Not Passed Assertions

- **Validate**: Check to validate the WSDL and/or the SOAP message.
- **On Failure Go To**: Select the step to redirect to on error.

> Validation failures are common, but usually should not affect the outcome of the test. It is good practice to set the next step to continue so that you can complete the test.

Advanced Tab



SOAP Action getItemByTitle

Style ● Document ○ RPC

Use ● Literal ○ Encoded

☑ SOAP Fault is Error □ Do not send request ☑ Maintain Session ☑ Clear Session

- **SOAP Action**: This field is auto-populated based on the WSDL operation definition. It is used as the value of the SOAPAction transport header for SOAP 1.1 request message. Change this field manually only in rare cases.

- **Style**: This field is auto-populated based on the WSDL operation definition. It is used to determine how a sample SOAP message is generated. Changing this field manually should only be done in rare cases.

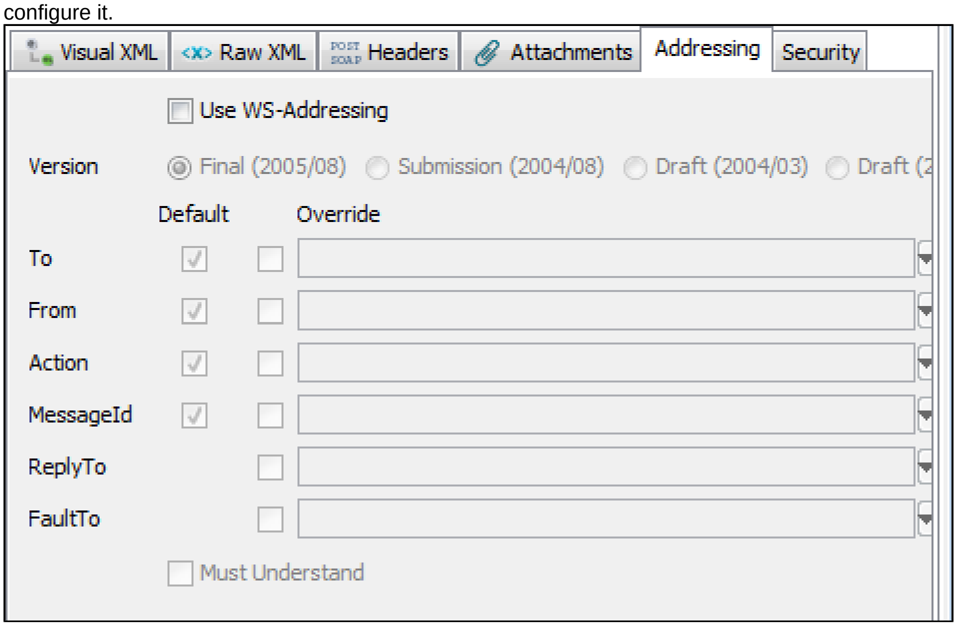
- **Use**: This field is auto-populated based on the WSDL operation definition. It is used to determine how a sample SOAP message is generated. If **Encoded** is selected, you can also edit the Encoded URI in the field next to the choice. Changing these fields manually should only be done in rare cases.

- **SOAP Fault is Error**: If a SOAP fault is returned, perform On Error handling.
- **Do Not Send Request**: When selected, the step execution will perform all the normal SOAP message processing but will not send the generated SOAP message. Instead it will set the response to be the request message that would have been sent.
- **Maintain Session**: Select to maintain cookies across invocations.
- **Clear Session**: Select to clear cookies across invocations. Clearing the session cookies will act like a new session was created. Any old session cookies will not be used and any new cookies in the response will not be set on the session, but the session will not be cleared so future steps can still use it.

Request Editor Tabs

Addressing Tab

You can send a WS-Addressing header with your request. The WSDL does not specify if WSAddressing information is required so you must configure it.



□ Use WS-Addressing

Version ● Final (2005/08) ○ Submission (2004/08) ○ Draft (2004/03) ○ Draft (2

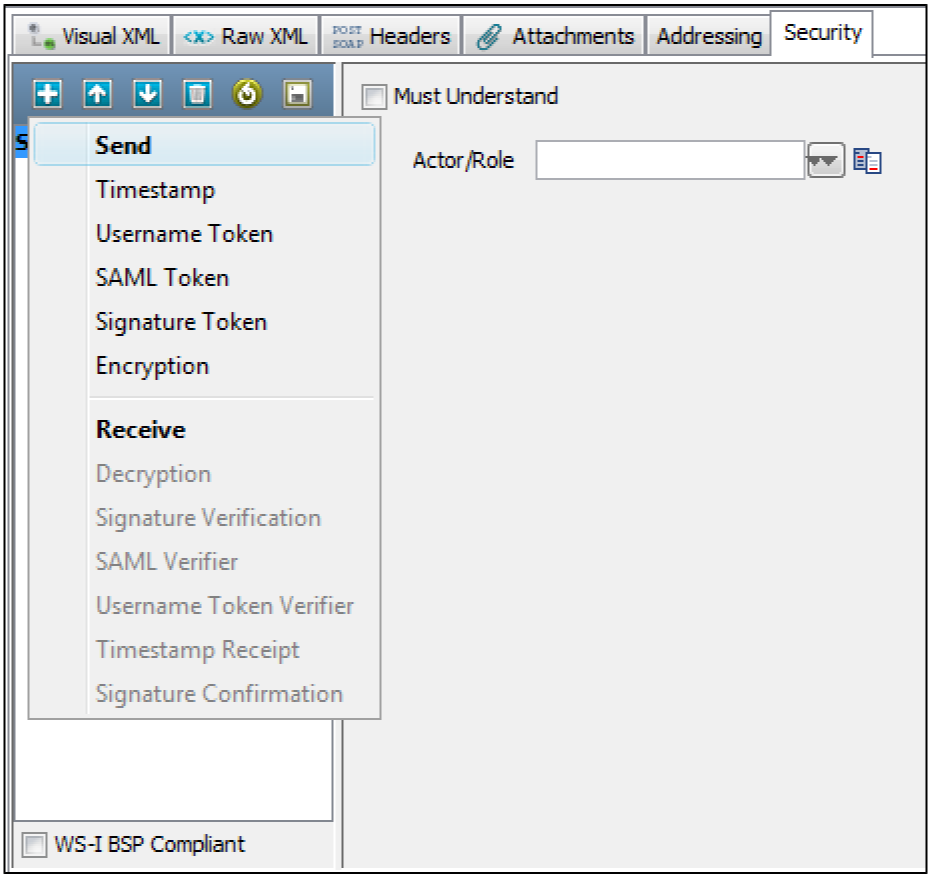
	Default	Override	
To	☑	□	
From	☑	□	
Action	☑	□	
MessageId	☑	□	
ReplyTo		□	
FaultTo		□	

□ Must Understand

Click the Addressing tab, and then specify:

- **Use WS-Addressing**: Click to use WS-Addressing.
- **Version**: Select appropriate version. Several versions of the WS-Addressing specification are listed as options because some web services platforms (for example, .NET) still use the older Draft specifications. You will need to determine which your web service platform is using.
- LISA will populate as many values as it can. Then you can choose to use the default value or override it. You can choose not to send some of the Default elements by clearing the Default check box for that element.
- Click the Must Understand check box if you want to assure that the web service can understand the WSAddressing header.

Security Tab



Click the Security tab. Click Send.

- **Must Understand**: To help ensure that the WS-Security header is processed by the server.
- **Actor/Role**: Enter name if needed: most web services do not use multiple Actors/Roles.

Click the Add icon and select the security action type to add. You will be presented with the configuration panel for that security action type.

Adding security verification to the Response is very similar:

Click the Security tab. Click the Add, icon and select Receive.

- Enter the Actor/Role name (if needed)
- Click the Add icon and select the security action type. You will be presented with the configuration panel for that security action type.

> ⚠ You can add as many security types as are needed to execute your Web Service.

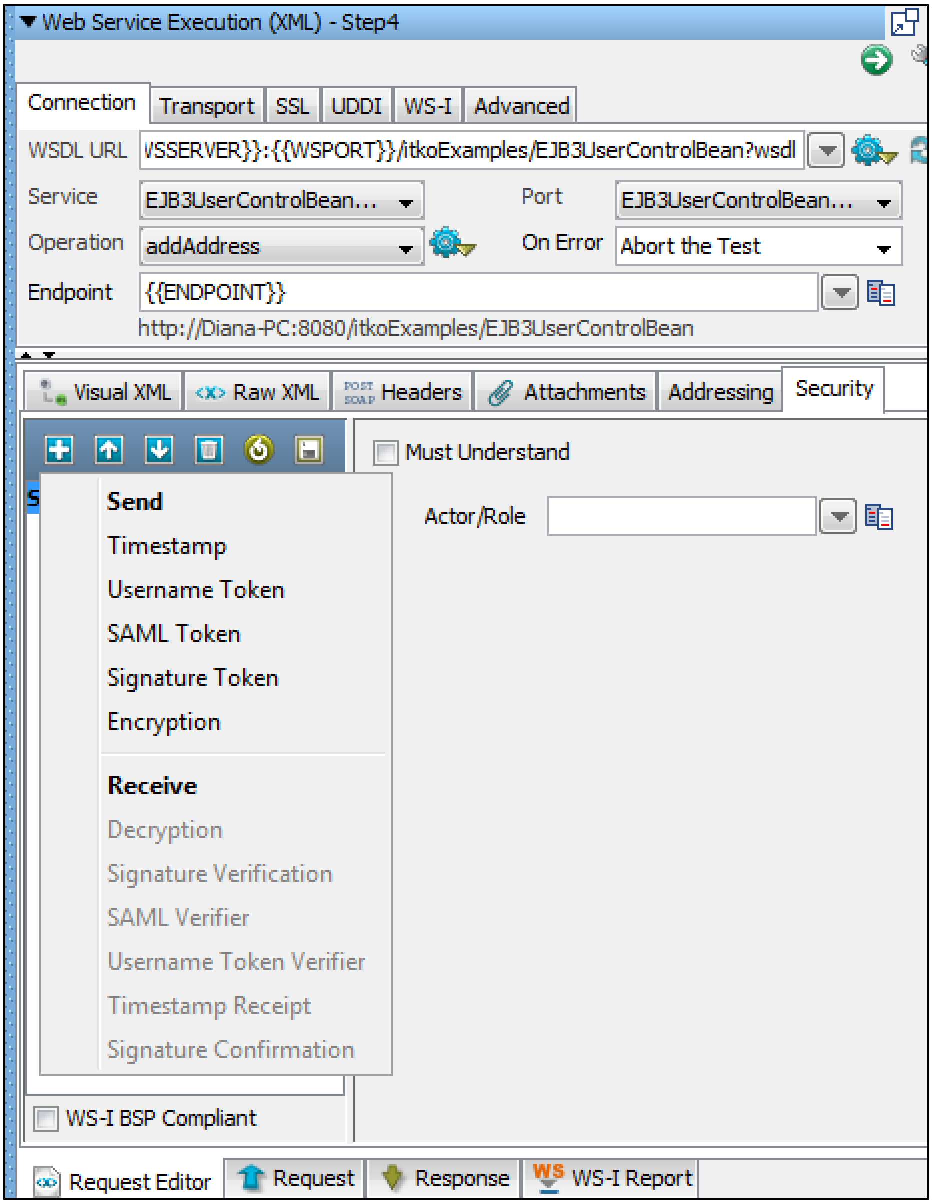
When a keystore is being used in a security action type configuration, you can verify your keystore settings to make sure you are using the correct format, password, alias and alias password. There is a Verify button on the editors for Signature, Encryption/Decryption and SAML Assertion Token. Clicking the Verify button will invoke the Keystore Verifier, which will produce a verification report.

If you do not know the expected alias name for a WS-Security setting, you can use the Keystore Verifier to list all of the aliases in the keystore. Leave the Keystore Alias and Alias Password boxes empty and click the Verify button. The Keystore Verifier is described at the end of this section on WS-Security.

> ⚠ New in LISA 5.0.25 and LISA 6.0, you can now load and save security configuration information from/to a .wss file, using the Load and Save icons. This allows for quick and easy creation of new steps connecting to the same service.

Security Example

This section describes the parameters needed to run the WS-security example.



Web Service Execution (XML) - Step4

WSDL URL VSSERVER}}:{{WSPORT}}/itkoExamples/EJB3UserControlBean?wsdl

Service EJB3UserControlBean...

Port EJB3UserControlBean...

Operation addAddress

On Error Abort the Test

Endpoint {{ENDPOINT}}

http://Diana-PC:8080/itkoExamples/EJB3UserControlBean

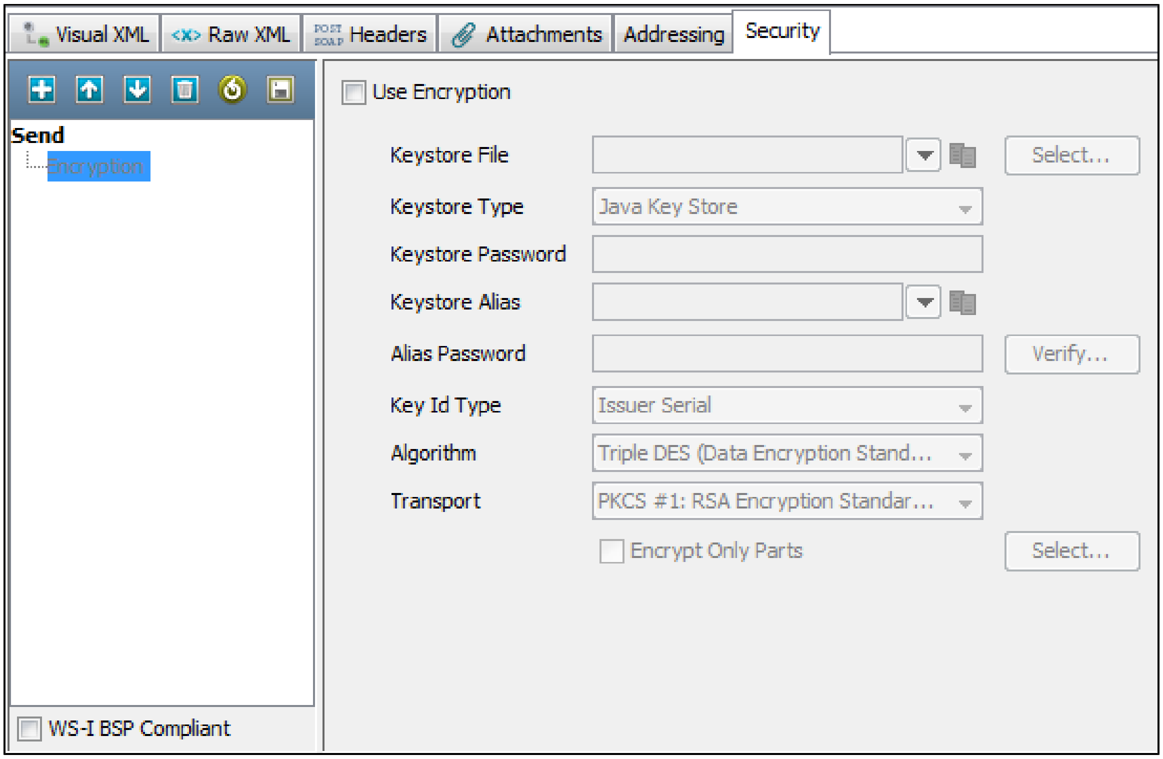
□ Must Understand

Actor/Role

□ WS-I BSP Compliant

XML Encryption/Decryption

Encryption



□ Use Encryption

Keystore File

Keystore Type: Java Key Store

Keystore Password

Keystore Alias

Alias Password

Key Id Type: Issuer Serial

Algorithm: Triple DES (Data Encryption Stand...

Transport: PKCS #1: RSA Encryption Standar...

□ Encrypt Only Parts

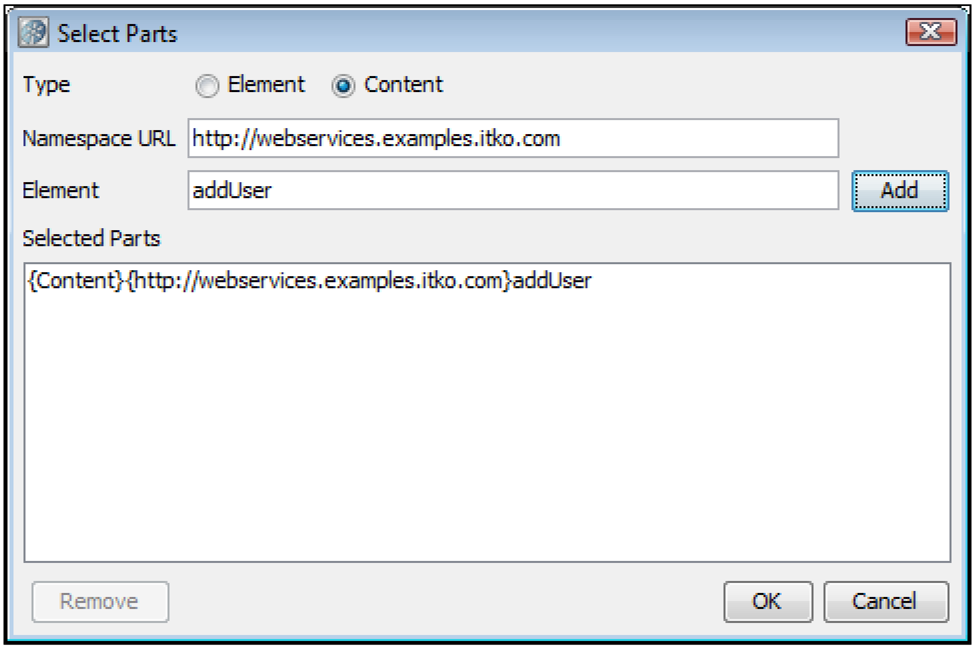
□ WS-I BSP Compliant

Click the Use Encryption check box.

- **Keystore File**: The location of the keystore file.
- **Keystore Type**: Select Java Key Store (jks) or Personal Information Exchange (PKCS #12).
- **Keystore Password**: Enter the password for the keystore.
- **Keystore Alias**: Enter an alias for a public key.
- **Alias Password**: Leave empty or make the same as Keystore Password for PKCS #12 files.
- **Key ID Type**: Select the appropriate key ID type from the pull-down menu.
- **Algorithm**: Select Triple DES, AES 128, AED 192, or AES 256.
- **Transport**: Select PKCS#1: RSA Encryption Standard v1.5 or Optimal Asymmetric Encryption Padding with RSA Encryption.

The default behavior is to encrypt only the SOAP Body contents.

- **Encrypt Only Parts**: If you want to specify different parts to encrypt. Click the Select button to identify the parts to be encrypted.



Type ○ Element ● Content

Namespace URL: http://webservices.examples.itko.com

Element: addUser

Selected Parts

{Content}{http://webservices.examples.itko.com}addUser

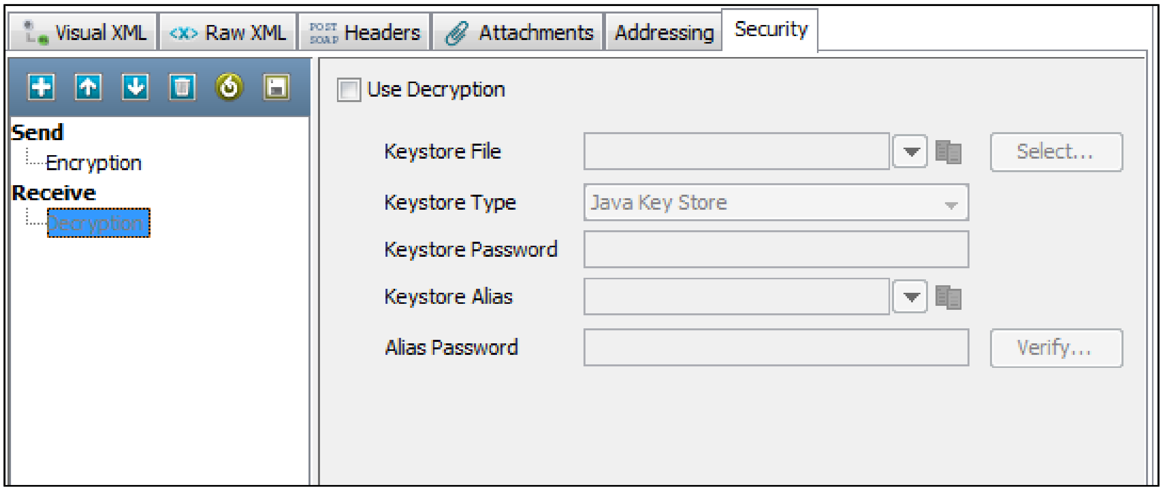
- **Type**: Select one of the following:

- **Element**: Select if you want to encrypt the element and the content.
- **Content:** Select if you want to encrypt just the content.
- **Namespace URL**: Enter the value for the element.
- **Element**: Enter the name of the element.

You can repeat this for as many elements as you want by clicking the Add button.

You will need to manually add the Body element if you want it to be included. If you want to include the Binary Security Token as a part, use the Element name "Token".

Decryption



□ Use Decryption

Keystore File

Keystore Type: Java Key Store

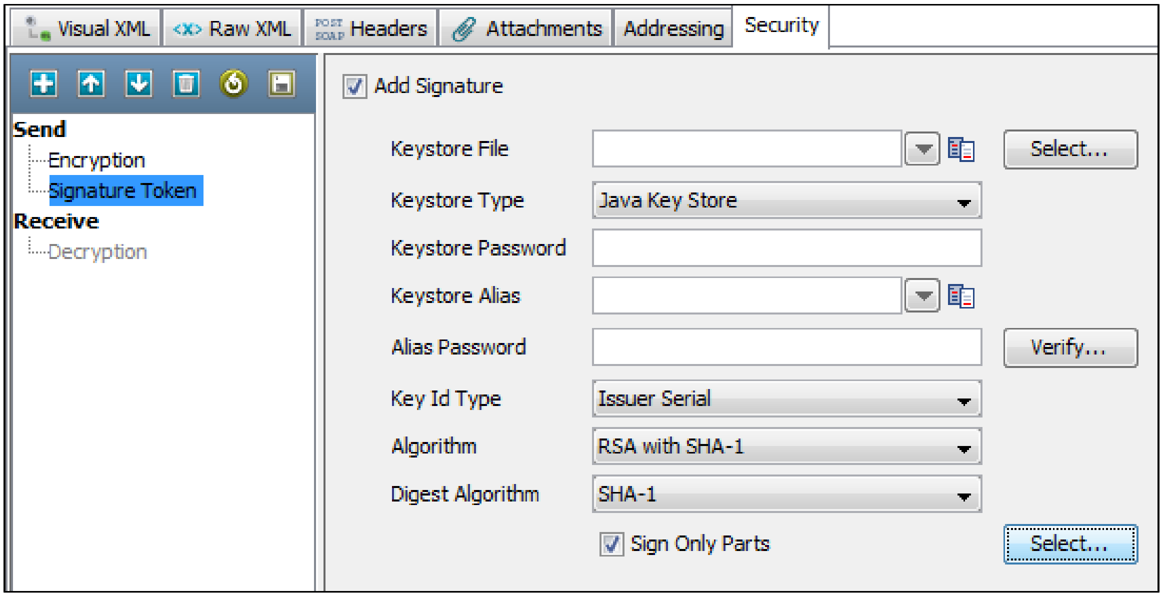
Keystore Password

Keystore Alias

Alias Password

XML Signature Token/Signature Verification

Signature Token



☑ Add Signature

Keystore File

Keystore Type: Java Key Store

Keystore Password

Keystore Alias

Alias Password

Key Id Type: Issuer Serial

Algorithm: RSA with SHA-1

Digest Algorithm: SHA-1

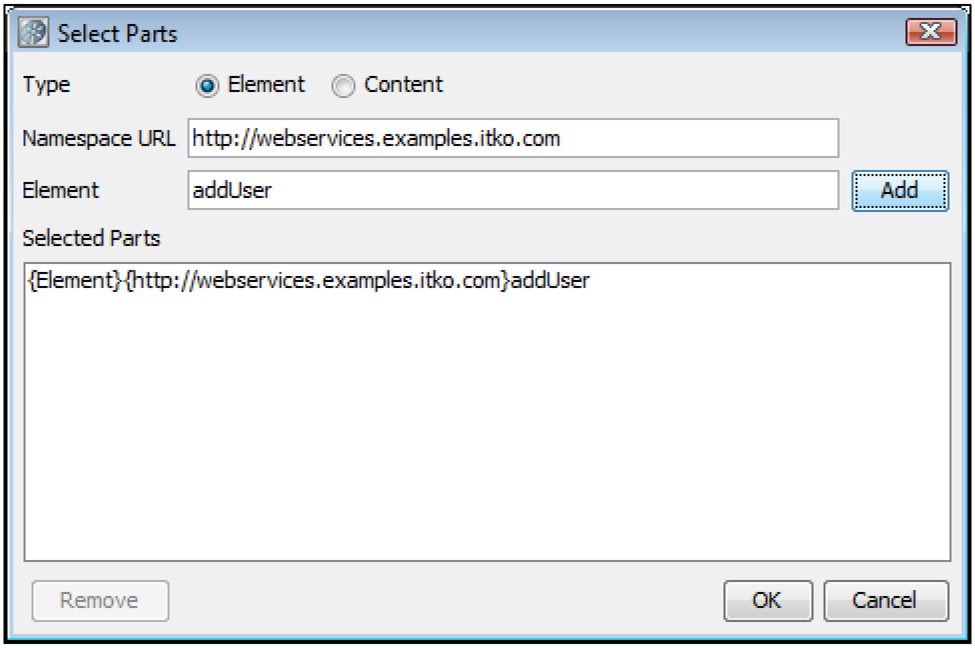
☑ Sign Only Parts

Click the Add Signature box.

- **Keystore File**: Enter the location of the keystore file.
- **Keystore Type**: Select Java Key Store (jks) or Personal Information Exchange (PKCS #12).
- **Keystore Password**: Enter the password for the keystore.
- **Keystore alias**: Enter an alias for a private key.
- **Alias Password**: Leave empty or enter the same as Keystore Password for PKCS #12 files.
- **Key ID type**: Select the appropriate key ID type from the pull-down menu.
- **Algorithm**: Select DSA with SHA-1.

The default behavior is to sign only the SOAP Body contents.

- **Sign Only Parts:** If you want to specify different parts to sign, click the Select button to identify the parts to be signed.



Type ● Element ○ Content

Namespace URL: http://webservices.examples.itko.com

Element: addUser

Selected Parts

{Element}{http://webservices.examples.itko.com}addUser

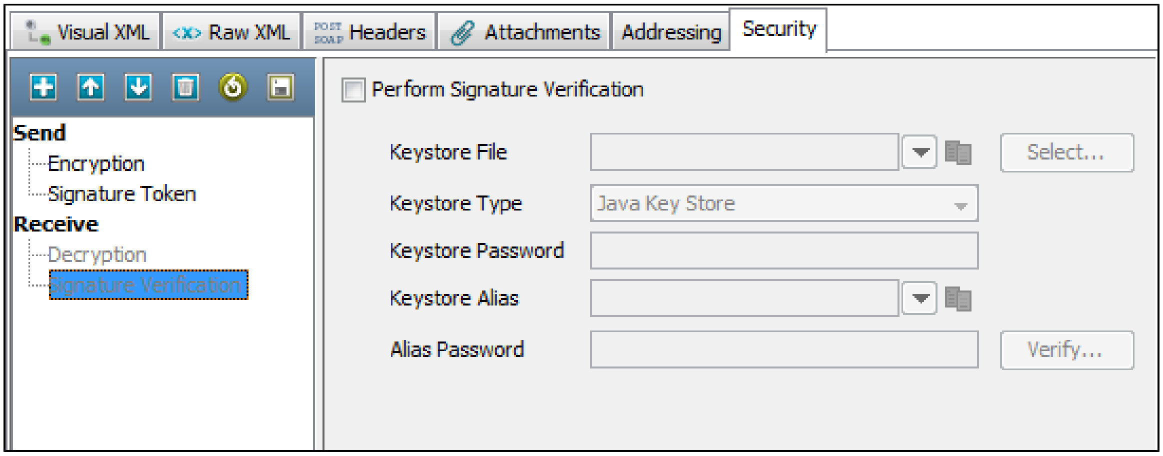
- **Type:**Select one of the following:
  - **Element**: Select if you want to sign the element and the content.
  - **Content**: Select if you want to sign just the content.
  - **Namespace URL**: Enter the value for the element.
  - **Element**: Enter the name of the element.

You can repeat this for as many elements as you want by clicking the Add button.

You will need to manually add the Body element if you want it to be included. If you want to include the Binary Security Token as a part, use the Element name "Token."

Signature Verification

The parameters required to configure signature verification are a subset of those required for signing.



□ Perform Signature Verification

Keystore File

Keystore Type: Java Key Store

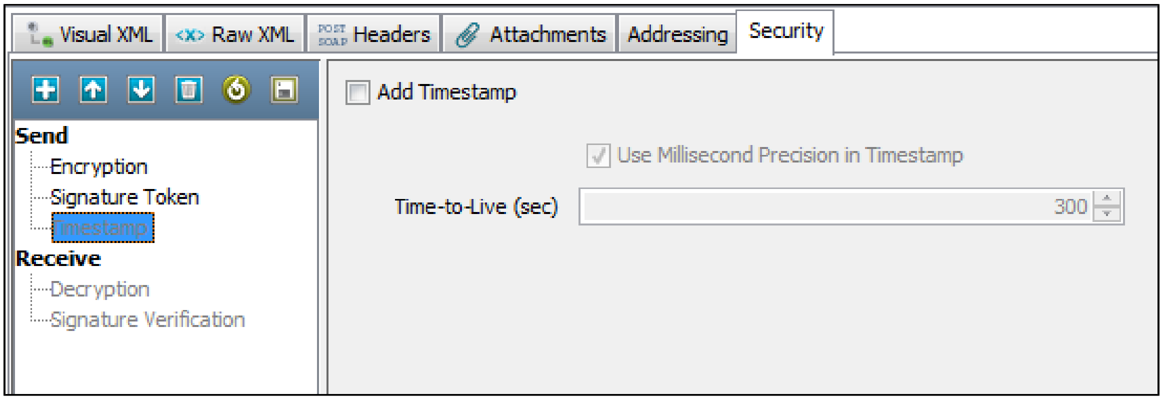
Keystore Password

Keystore Alias

Alias Password

Timestamp/Timestamp Receipt

Timestamp



□ Add Timestamp

☑ Use Millisecond Precision in Timestamp

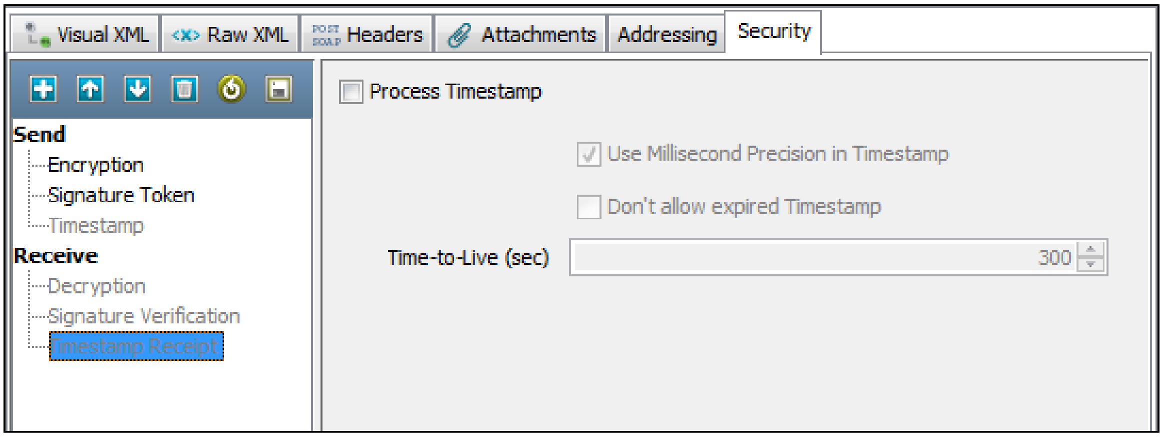
Time-to-Live (sec) 300

Click the Add Timestamp box.

- **Time-To-Live (sec)**: Enter the lifetime of the message in seconds. Enter 0 to not include an Expires element.

> ⚠ Some Web services, particularly .NET 1.x/2.0 with WSE 2.0, are not compliant with standard time stamp date formatting, and do not allow milliseconds. For these web services, clear the check box for Use Millisecond Precision in Timestamp.

Timestamp Receipt



□ Process Timestamp

☑ Use Millisecond Precision in Timestamp

□ Don't allow expired Timestamp

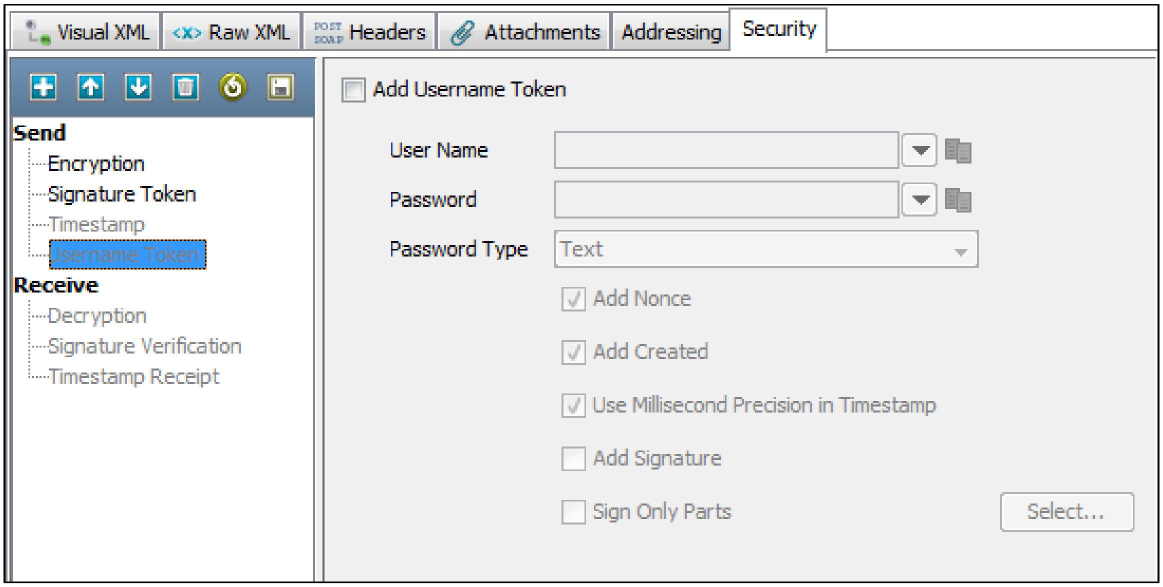
Time-to-Live (sec) 300

The parameters required for Timestamp Receipt are a super set of those required for Timestamp. The additional parameter is:

- **Don't allow expired**: Can be checked if you do not want to allow expired timestamps.

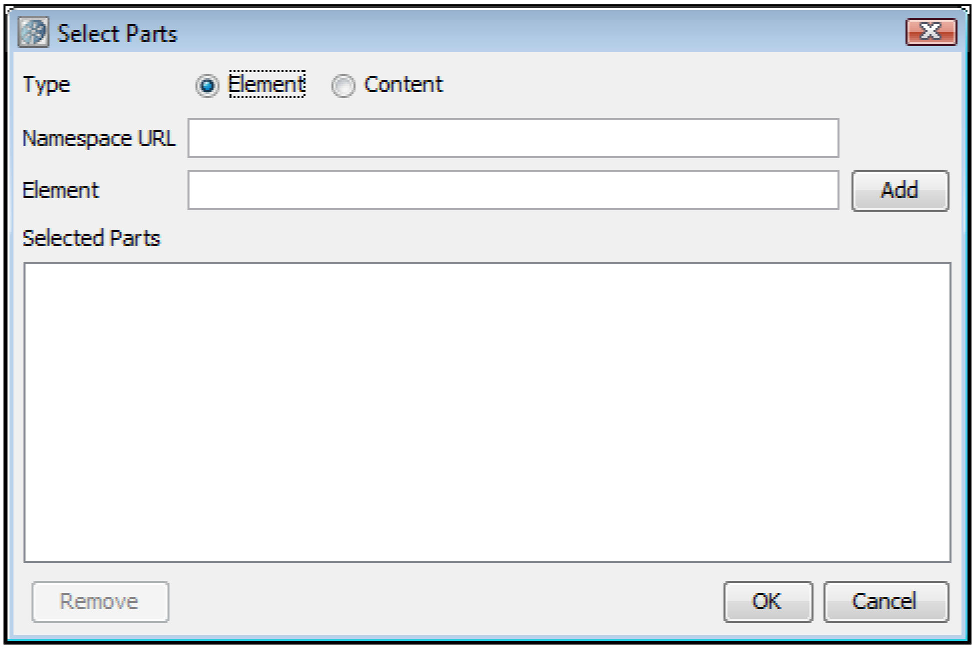
Username Token/Username Token Verifier

Username Token



Click the **Add Username Token** box.

- **User Name**: Enter the appropriate user name.
- **Password**: Enter the appropriate password.
- **Password Type**: Select the password type from the drop-down menu (Text, Digest, None). None is typically used with the Add Signature option.
- **Add Nonce**: Click if a Nonce is required – used to protect against replay attacks.
- **Add Created**: Click if a time stamp is required.
- **Use Millisecond Precision in Timestamp**: Select the check box to use millisecond precision. Some Web services, particularly .NET 1.x/2.0 with WSE 2.0, are not compliant with standard timestamp date formatting, and do not allow the use of milliseconds.
- **Add Signature**: Select to add a signature built using a combination of the username and password as the key.
- **Sign Only Parts**: Select if you want to specify different parts to sign and click the Select button to identify the parts to be signed.

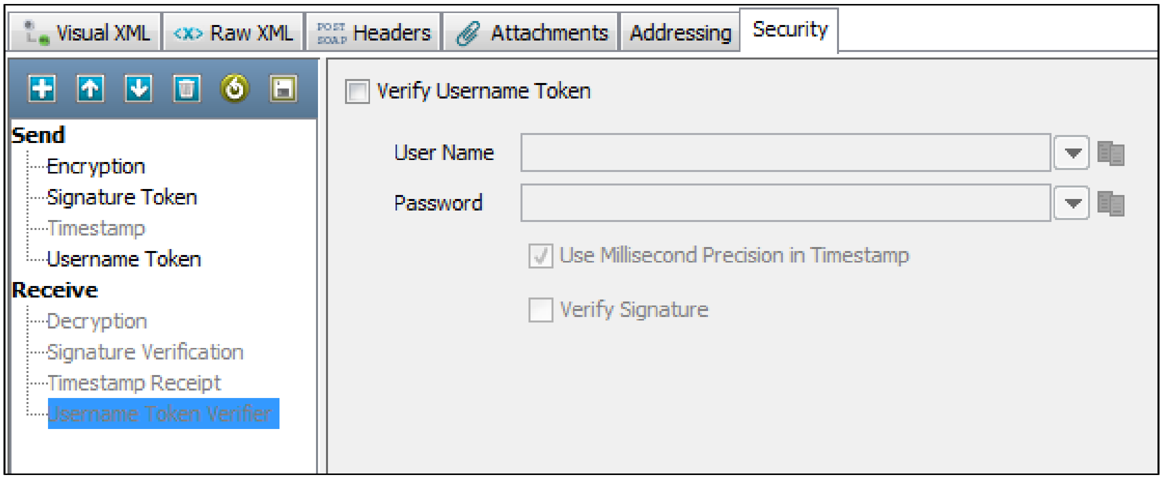


- **Type:**Select one of the following:
  - **Element**: Select if you want to encrypt the element and the content.
  - **Content**: Select if you want to encrypt just the content.
  - **Namespace URL**: Enter the value for the element.
  - **Element**: enter the name of the element.

You can repeat this for as many elements as you want by clicking the Add button.

You will need to manually add the Body element if you want it to be included. If you want to include the Binary Security Token as a part, use the Element name "Token."

UserName Token Verifier

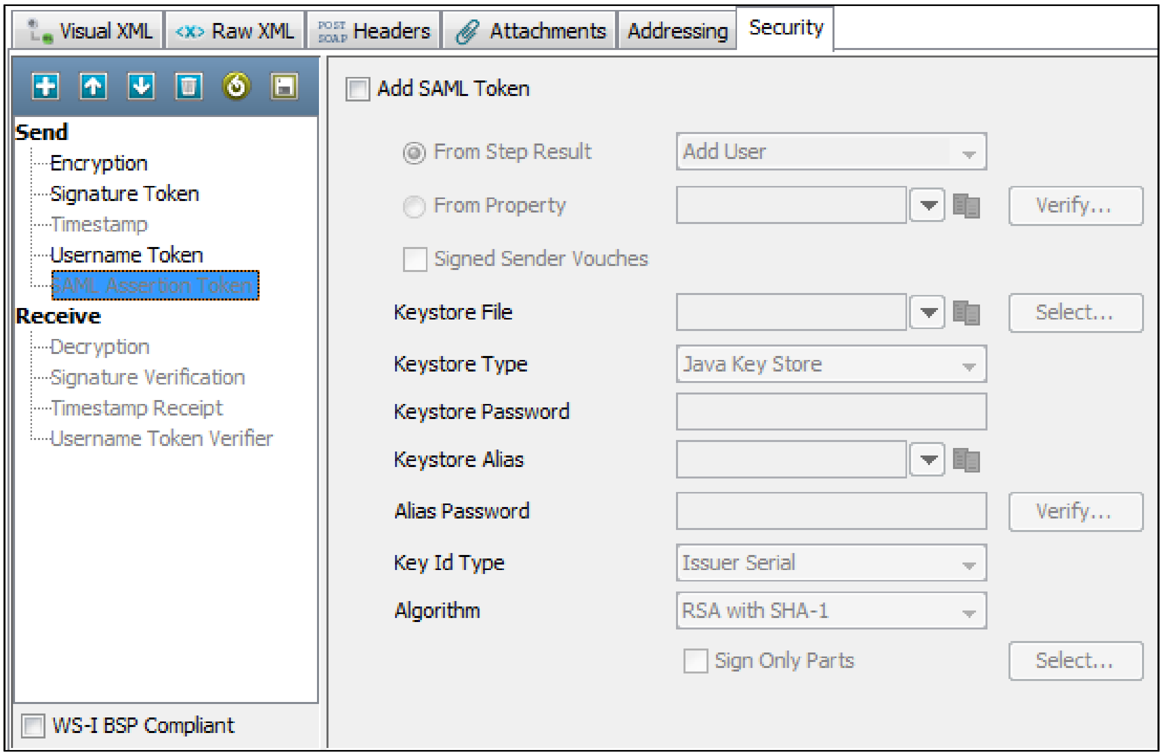


Click the Verify Username Token check box.

- **User Name**: Enter the verification user name.
- **Password**: Enter the verification password.
- **Use Millisecond Precision in Timestamp**: Select the check box to use millisecond precision. Some Web services, particularly .NET 1.x/2.0 with WSE 2.0, are not compliant with standard time stamp date formatting, and do not allow the use of milliseconds.
- **Verify Signature**: Click box if Signature verification is required.

SAML Assertion Token/SAML Assertion Receipt

SAML Assertion Token



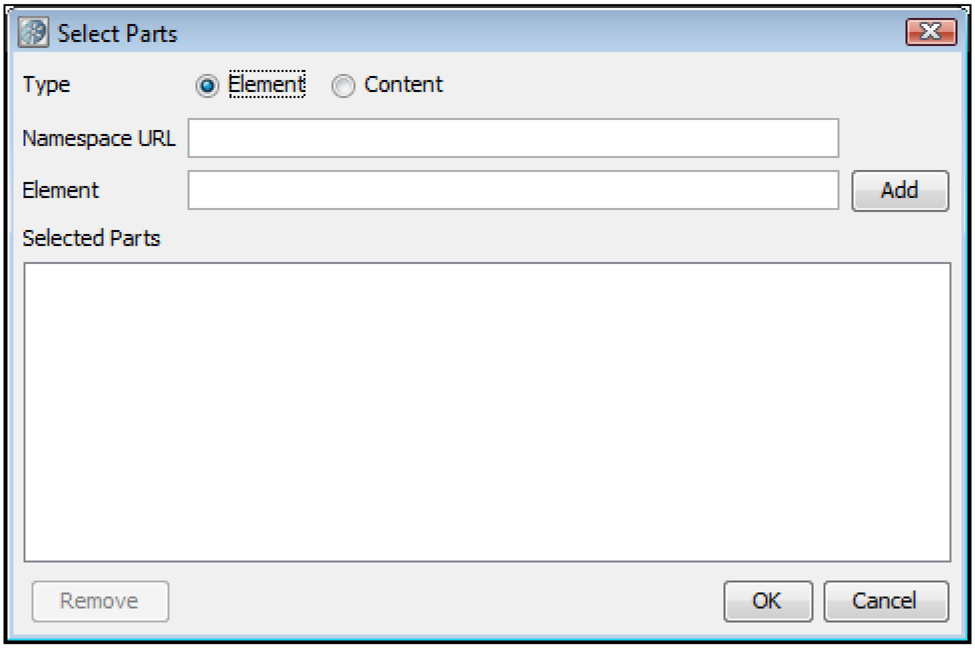
- Click the Add SAML Token box.
- Select the From Step Results option button. Select the step whose result is an XML SAML Assertion (like a SAML Query Step or a Parse Text Step with XML manually entered), or
- Select the From Property option button and enter the LISA property that contains the XML SAML Assertion.
- You can optionally select the **Verify** button to have LISA parse the SAML Assertion XML and build the SAML Assertion object as it would when sending the SOAP request. This is useful to confirm that the SAML Assertion that may have been created manually is a valid SAML Assertion. It will also attempt to verify any signatures associated with the assertion, but it is likely that LISA will not be able to verify the assertion without configuring a public certificate to verify with.
- **Signed Sender Vouches**: Select if the assertion needs to be signed by the sender (the sender vouches for its authenticity as opposed to

the bearer/creator of the SAML Assertion). When selected you will need to fill in the following information:

- **Keystore File**: Enter the location of the keystore file.
- **Keystore Type**: Select Java Key Store (jks) or Personal Information Exchange (PKCS #12).
- **Keystore Password**: Enter the password for the keystore.
- **Keystore alias**: Enter an alias for a private key.
- **Alias Password**: Leave empty or enter the same as Keystore Password for PKCS #12 files.
- **Key ID type**: Select the appropriate key ID type from the pull-down menu.
- **Algorithm**: Select DSA with SHA-1.

The default behavior is to sign only the SOAP Body contents.

- **Sign Only Parts**: If you want to specify different parts to sign click the Select button to identify the parts to be signed.


Select Parts

Type ● Element ○ Content

Namespace URL

Element

Selected Parts

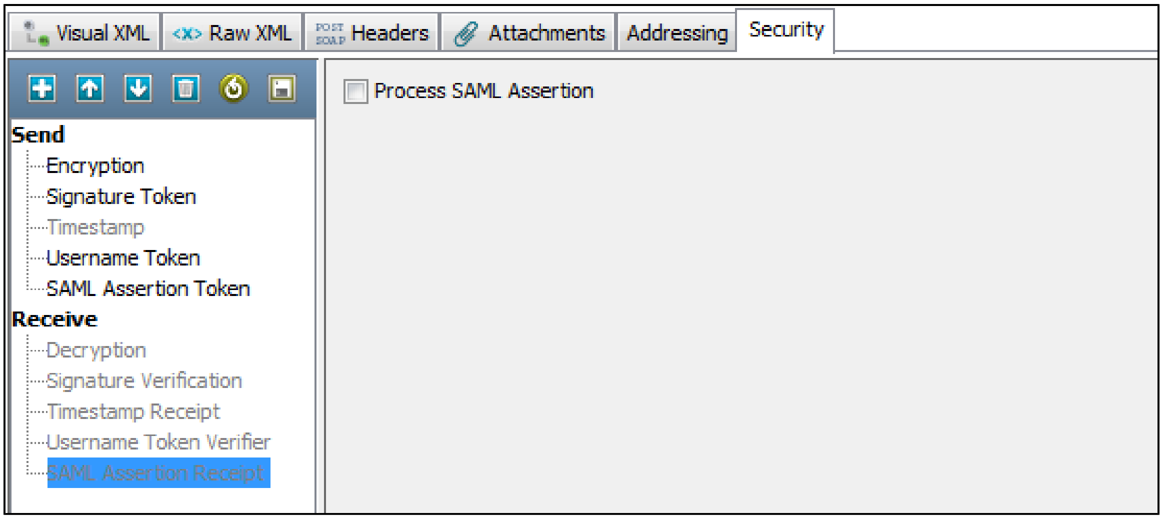
- **Type**: Select one of the following:
  - **Element**: Select if you want to sign the element and the content.
  - **Content**: Select if you want to sign just the content.
  - **Namespace URL**: Enter the value for the element.
  - **Element**: Enter the name of the element.

You can repeat this for as many elements as you want by clicking the Add button. You will need to manually add the Body element if you want it to be included.

To include the Binary Security Token as a part, use the Element name 'Token'.

SAML Assertion Receipt

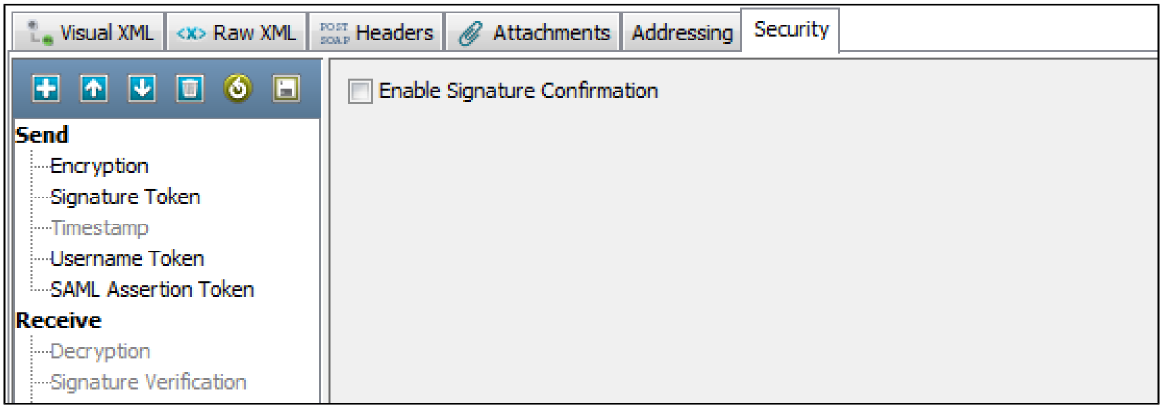
Click the Process SAML Assertion check box if you want to check for a SAML Assertion Receipt header in the response. If you select this option and there is no SAML Assertion Receipt header, an exception occurs.



□ Process SAML Assertion

Signature Confirmation

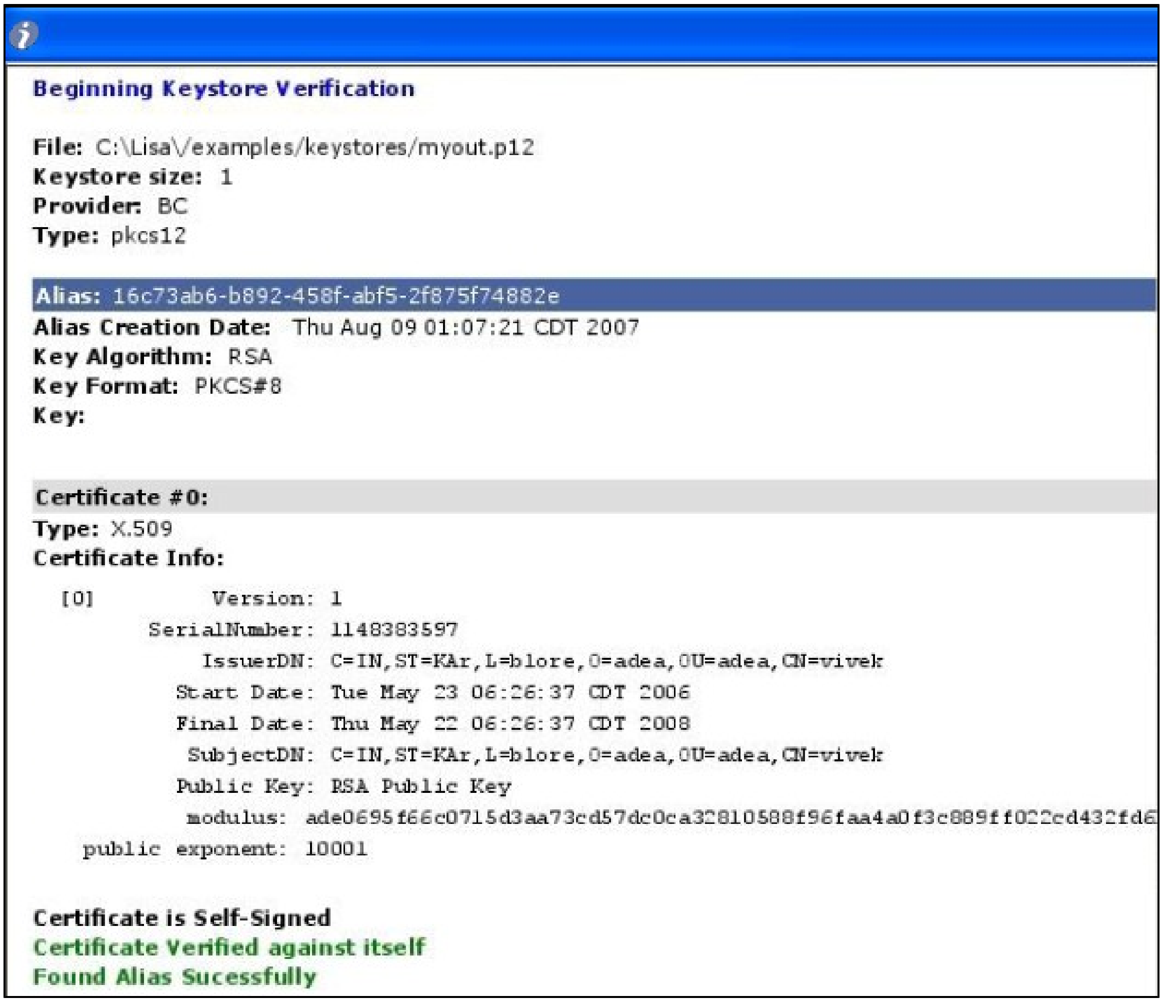
Click the Signature confirmation check box if you want to check for a signature confirmation header in the Response. If you select this option and there is no confirmation header, an exception occurs.



□ Enable Signature Confirmation

Using the Keystore Verifier

You can verify your keystore settings to make sure you are using the correct format, password, alias and alias password. There is a **Verify** button on the editors for SSL, Signature, Encryption/Decryption and SAML settings. Clicking Verify produces a verification report.



Beginning Keystore Verification

File: C:\Lisa\/examples/keystores/myout.p12
Keystore size: 1
Provider: BC
Type: pkcs12

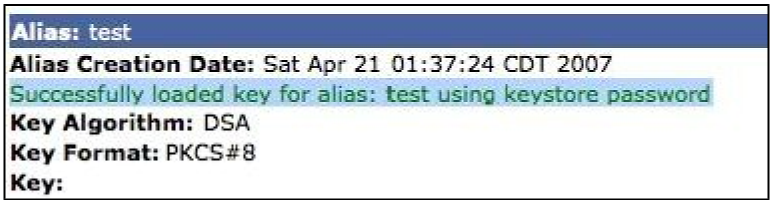
Alias: 16c73ab6-b892-458f-abf5-2f875f74882e
Alias Creation Date: Thu Aug 09 01:07:21 CDT 2007
Key Algorithm: RSA
Key Format: PKCS#8
Key:

Certificate #0:
Type: X.509
Certificate Info:

```
 [0]         Version: 1
        SerialNumber: 1148383597
            IssuerDN: C=IN,ST=KAr,L=blore,O=adea,OU=adea,CN=vivek
          Start Date: Tue May 23 06:26:37 CDT 2006
          Final Date: Thu May 22 06:26:37 CDT 2008
           SubjectDN: C=IN,ST=KAr,L=blore,O=adea,OU=adea,CN=vivek
          Public Key: RSA Public Key
           modulus: ade0695f66c0715d3aa73cd57dc0ca32810588f96faa4a0f3c889ff022cd432fd6
   public exponent: 10001
```

Certificate is Self-Signed
Certificate Verified against itself
Found Alias Sucessfully

SSL verification validates the Keystore password only and confirms that at least one of the keys in the keystore can be loaded using the keystore password.



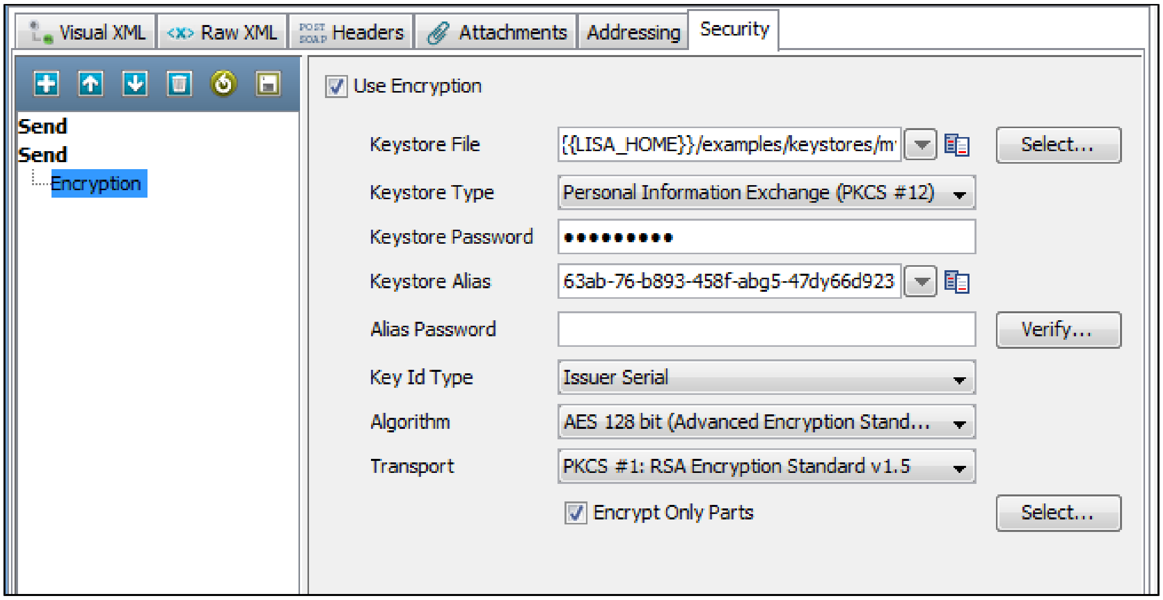
Alias: test
Alias Creation Date: Sat Apr 21 01:37:24 CDT 2007
Successfully loaded key for alias: test using keystore password
Key Algorithm: DSA
Key Format: PKCS#8
Key:

WS-Security verification validates the Keystore password, the alias and the alias password. Correct validation is indicated with a green entry. Any validation errors found will be shown in red. Warnings are shown in orange.

> ⚠ This verification only verifies the keystore parameters. There could still be issues with the web service, such as a mismatch in certificate sets or incorrect choice of algorithm. These issues will need to be validated independently.

Alias Search

If you do not know the expected alias name for a WS-Security setting, you can use the keystore verifier to list all of the aliases in the keystore. Leave the Keystore Alias and Alias Password boxes empty and click the Verify button.



☑ Use Encryption

Keystore File [{LISA_HOME}}/examples/keystores/m

Keystore Type Personal Information Exchange (PKCS #12)

Keystore Password ●●●●●●●●●

Keystore Alias 63ab-76-b893-458f-abg5-47dy66d923

Alias Password

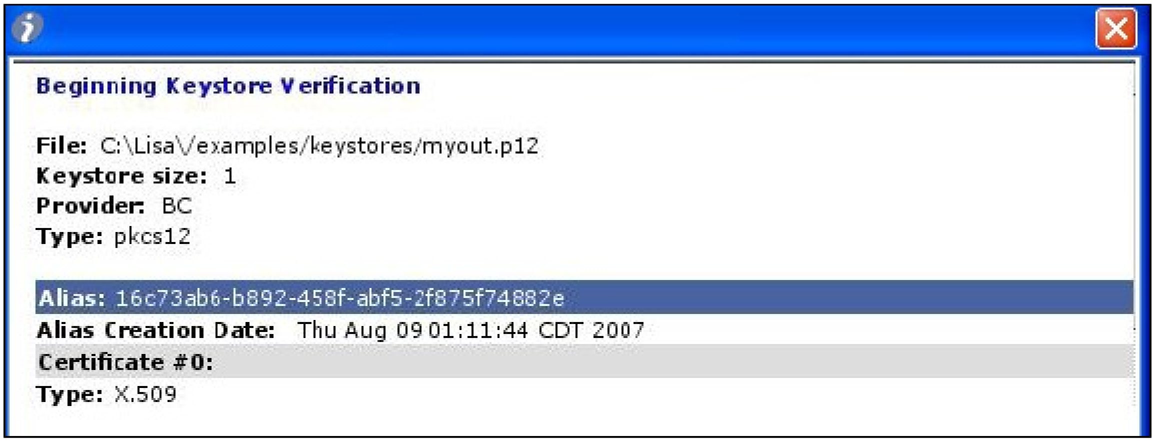
Key Id Type Issuer Serial

Algorithm AES 128 bit (Advanced Encryption Stand...

Transport PKCS #1: RSA Encryption Standard v1.5

☑ Encrypt Only Parts

Aliases are highlighted with a blue background.



Beginning Keystore Verification

File: C:\Lisa\/examples/keystores/myout.p12
Keystore size: 1
Provider: BC
Type: pkcs12

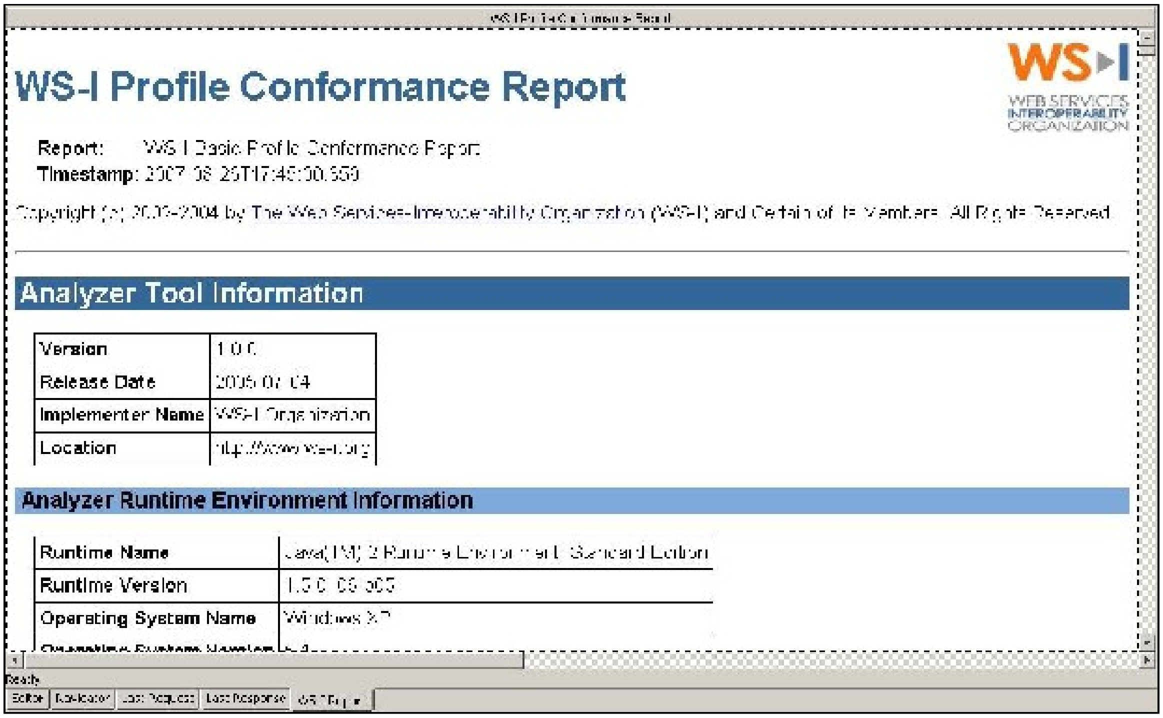
Alias: 16c73ab6-b892-458f-abf5-2f875f74882e
Alias Creation Date: Thu Aug 09 01:11:44 CDT 2007
Certificate #0:
Type: X.509

Verification will fail because the Keystore Alias and Alias Password boxes were left blank.

WS-I Report

While executing WS-I Basic profile Validation, when you click the Execute button on the Object Editor screen, the validation will be run and a report will be generated and saved (in the reports directory in the LISA install directory). You can view the report by pressing the WS-I Report tab at the bottom of the screen.

⚠ The format of this report is standard, and is dictated by the Web-Services-Interoperability Organization (WS-I).


WS-I Profile Conformance Report
WS-I WEB SERVICES INTEROPERABILITY ORGANIZATION
Ready
Editor
Navigator
Last Request
Last Response
WS-I Report

WS-I Profile Conformance Report

Report: WS-I Basic Profile Conformance Report
Timestamp: 2007-08-25T17:45:00.359

Copyright (c) 2002-2004 by The Web Services-Interoperability Organization (WS-I) and Certain of its Members. All Rights Reserved.

Analyzer Tool Information

Version	1.0.0
Release Date	2005-07-04
Implementer Name	WS-I Organization
Location	http://www.ws-i.org

Analyzer Runtime Environment Information

Runtime Name	Java(TM) 2 Runtime Environment, Standard Edition
Runtime Version	1.5.0_06-b05
Operating System Name	Windows XP

WSDL Validation

The WSDL Validation step lets you load a WSDL and add one or more assertions to validate the WSDL. This step is a little different in that it lets you load the static WSDL file and perform assertions on it. In most steps the assertions are being made on the response.

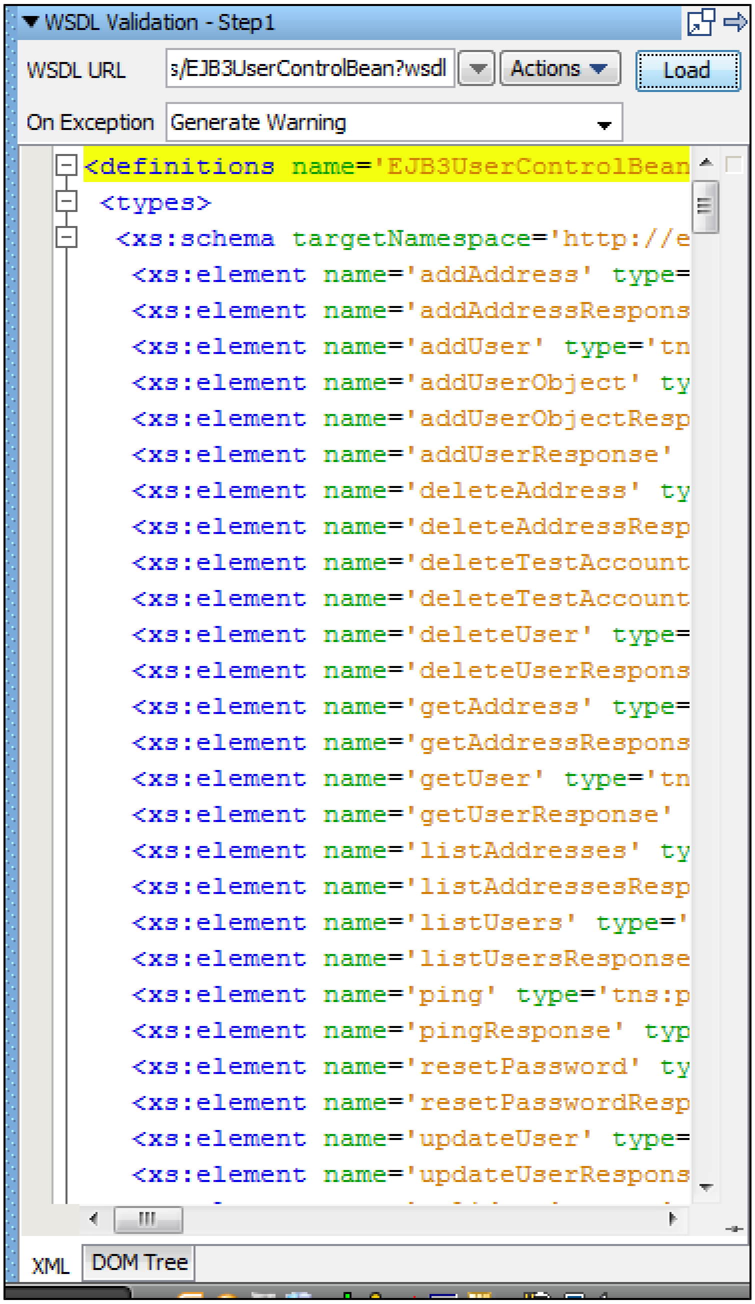
The most useful assertions are:

- **XML Diff Assertion**: Checks that the WSDL has not been changed by comparing it to a control copy of the original WSDL.
- **XML Validator**: Checks for valid XML using Schema or DTD.
- **WS-I Basic Profile 1.1 Assertion**: Checks for compliance with the WS-I Basic Profile.

These assertions are described in Types of Assertions.

Prerequisites: Familiarity with the three assertions named previously
Parameter Requirements: Location of the WSDL you want to validate

To configure the WSDL Validation step, enter a WSDL in the WSDL URL field and click Load.



WSDL URL: s/EJB3UserControlBean?wsdl

On Exception: Generate Warning

```
<definitions name='EJB3UserControlBean
 <types>
  <xs:schema targetNamespace='http://e
   <xs:element name='addAddress' type=
   <xs:element name='addAddressRespons
   <xs:element name='addUser' type='tn
   <xs:element name='addUserObject' ty
   <xs:element name='addUserObjectResp
   <xs:element name='addUserResponse'
   <xs:element name='deleteAddress' ty
   <xs:element name='deleteAddressResp
   <xs:element name='deleteTestAccount
   <xs:element name='deleteTestAccount
   <xs:element name='deleteUser' type=
   <xs:element name='deleteUserRespons
   <xs:element name='getAddress' type=
   <xs:element name='getAddressRespons
   <xs:element name='getUser' type='tn
   <xs:element name='getUserResponse'
   <xs:element name='listAddresses' ty
   <xs:element name='listAddressesResp
   <xs:element name='listUsers' type='
   <xs:element name='listUsersResponse
   <xs:element name='ping' type='tns:p
   <xs:element name='pingResponse' typ
   <xs:element name='resetPassword' ty
   <xs:element name='resetPasswordResp
   <xs:element name='updateUser' type=
   <xs:element name='updateUserRespons
```

The WSDL appears in the editor. You can view it in XML or DOM View.

You are now ready to add the assertions.

Web_ Raw SOAP Request

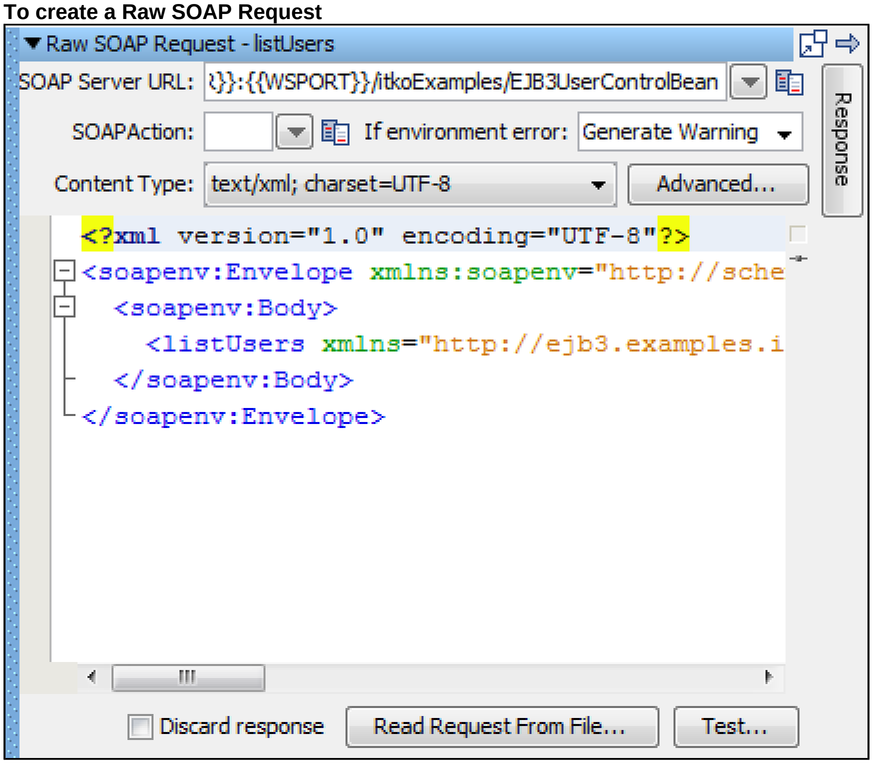
The Raw SOAP Request step lets you test a web service by sending a raw SOAP request (raw XML). It can be used to test legacy SOAP calls or web services that do not have a WSDL. It also lets you test a web service's reaction to data of an incorrect type (for example, sending a string when it expects a number), which is not allowed in the Web Service Execution step. Another use for the Raw SOAP request is to reduce overhead during intense load tests. The regular web service step has some additional overhead because it marshals an object into XML to make the request, and unmarshals the SOAP XML response back into an object. The Raw SOAP step avoids this overhead and only deals with the raw SOAP XML. It has less work to do, so it executes faster.

The SOAP request can be typed or pasted into the editor, or read from a file, and then parameterized using LISA properties.

There is no support for dynamic WS-Addressing or WS-Security headers. If you want to have these types of headers, they must be statically entered as part of the SOAP request entered in the input area. If your request does contain items like a WS-Security signature token, the signed elements cannot be parameterized or the signature will no longer be valid.

⚠ This step is not limited to SOAP calls. You can also do XML or text POSTs.

To create a Raw SOAP Request



SOAP Server URL: }}:{{WSPORT}}/itkoExamples/EJB3UserControlBean

SOAPAction:

If environment error: Generate Warning

Content Type: text/xml; charset=UTF-8

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://sche
  <soapenv:Body>
    <listUsers xmlns="http://ejb3.examples.i
  </soapenv:Body>
</soapenv:Envelope>
```

Enter the following parameters:

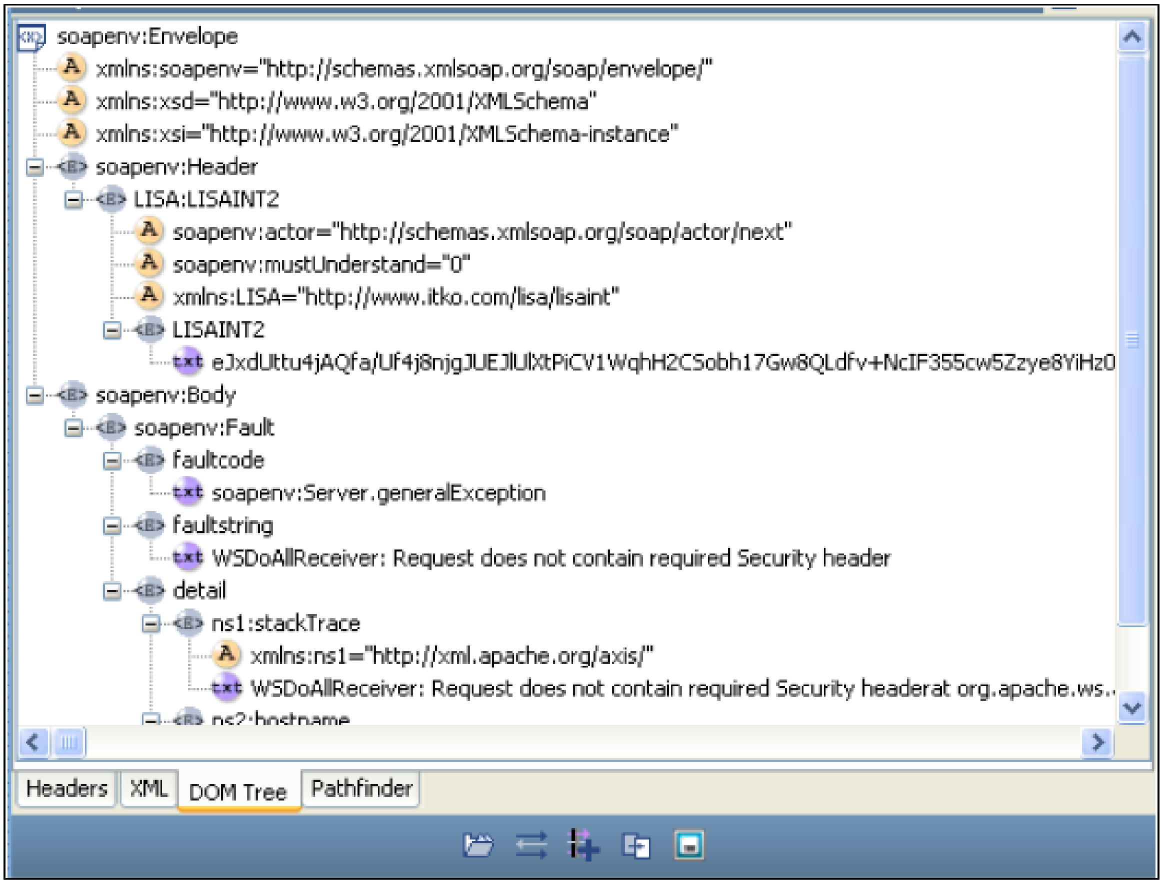
- **SOAP Server URL**: Enter the URL of the Web Service endpoint. The URL will be converted into a single property instead of just substituting the WSSERVER and PORT properties.
- **SOAP Action**: Fill in the SOAP action as indicated in <soap: operation> tag in the WSDL for the method being called. This is required for SOAP 1.1 and usually required to be left blank for SOAP 1.2.
- **Content Type**: Select the Content Type. Use text/HTML for SOAP 1.1, application/SOAP+XML for SOAP 1.2.
- **Advanced button**: Click to add any custom HTTP headers you want to send.
- **Discard response**: Check to discard the response, replacing it with a small valid but static SOAP text. This is meant to be used in load testing where processing a large response limits the scalability of the load generator computers.

Type or paste the SOAP Request into the editor, or click Read Request From File and browse to the file containing the SOAP Request.

Now you can parameterize the request, if you want, with properties.

Click the Test button to execute the call.

The response can be examined by clicking the Results tab:



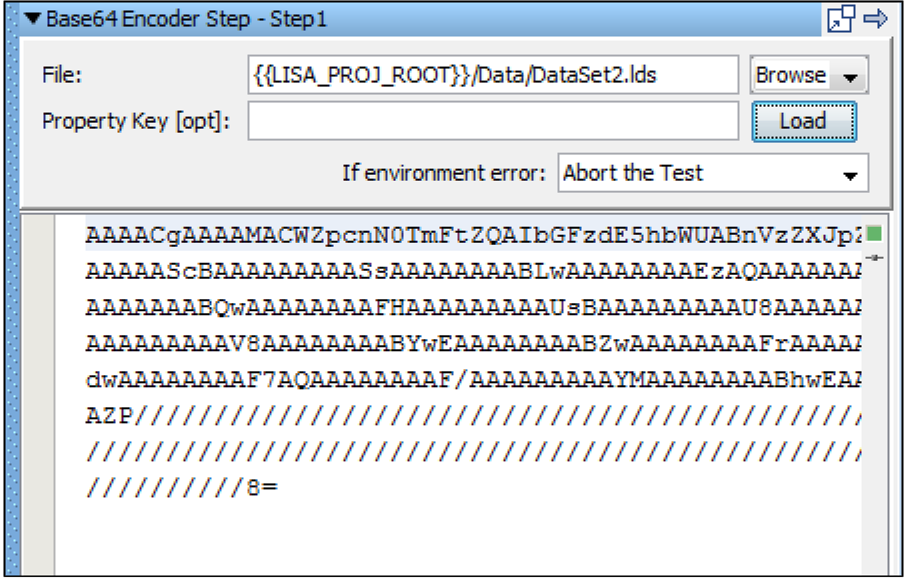
```
soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  soapenv:Header
    LISA:LISAINT2
      soapenv:actor="http://schemas.xmlsoap.org/soap/actor/next"
      soapenv:mustUnderstand="0"
      xmlns:LISA="http://www.itko.com/lisa/lisaint"
      LISAINT2
        eJxdUttu4jAQfa/Uf4j8njgJUEJlUlXtPiCV1WqhH2CSobh17Gw8QLdfv+NcIF355cw5Zzye8YiHz0
  soapenv:Body
    soapenv:Fault
      faultcode
        soapenv:Server.generalException
      faultstring
        WSDoAllReceiver: Request does not contain required Security header
      detail
        ns1:stackTrace
          xmlns:ns1="http://xml.apache.org/axis/"
          WSDoAllReceiver: Request does not contain required Security headerat org.apache.ws..
        ns2:hostname
```

You are now ready to add filters and assertions.

Base64 Encoder

The Base64 Encoder step is used to encode a file using the Base-64 encoding algorithm. The result can be stored into a LISA property for use elsewhere in the test case.

The Base64 Encoder step accepts a file as input and encodes the file using Base64 encoding. You can store the encoded file in a LISA property. Click the Load button to encode a file. The Base64 encoded text displayed in the editor is read-only.



Base64 Encoder Step - Step1

File: {{LISA_PROJ_ROOT}}/Data/DataSet2.lds

Property Key [opt]:

If environment error: Abort the Test

```
AAAACgAAAAMACWZpcnN0TmFtZQAIbGFzdE5hbWUABnVzZXJpZ
AAAAAScBAAAAAAAAAASsAAAAAAAABLwAAAAAAAAEzAQAAAAAAA
AAAAAAABQwAAAAAAAAFHAAAAAAAAAUsBAAAAAAAAAU8AAAAAA
AAAAAAAAAV8AAAAAAAAABYwEAAAAAAAABZwAAAAAAAAAFrAAAAA
dwAAAAAAAAF7AQAAAAAAAAF/AAAAAAAAAYMAAAAAAAAABhwEAA
AZP//////////////////////////////////////////////
/////////////////////////////////////////////////
//////////8=
```

Enter the following parameters:

- **File**: Enter the full path and path name, or browse to the file to be encoded.
- **Property Key: [opt]**: The name of the property in which to store the encoded file.

- **Load**: Click to load and test the encoding of the file. Optionally, store it in the specified property.
- **If environment error**: Action to take if an environment error occurs.

After the file is encoded, you can add filters and assertions. The valid options are:

- Random Selection Filter
- Parse Value Filter
- XML Xpath Filter
- Create HTML Table ResultSet Filter
- Make Assert on Selection

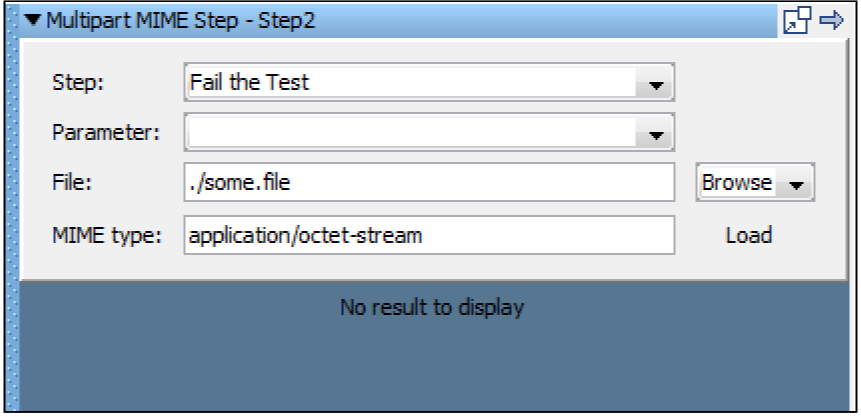
When you have completed adding filters and assertions, you can click Load to load the file or Save to save the contents of the editor in a new file.

Multipart MIME (Multipurpose Internet Mail Extensions) Step

The Multipart MIME step allows data to be loaded from a file, encoded, and stored in property to be used as a post parameter on an HTTP request. The encoded document will be stored in the LISA property that has been defined previously in an HTTP/HTML Request step.

When a multipart MIME form submit request is recorded, the contents of the file that was uploaded are recorded. Subsequent playback will result in the same content being submitted with "file upload" portion of the form again. The multipart MIME step can be used to change what file is uploaded when the test case is played back.

Prerequisite: The HTTP/HTML Request step containing the HTTP parameter must already exist, and it must be before the Multipart MIME step.



Multipart MIME Step - Step2

Step: Fail the Test

Parameter:

File: ./some.file

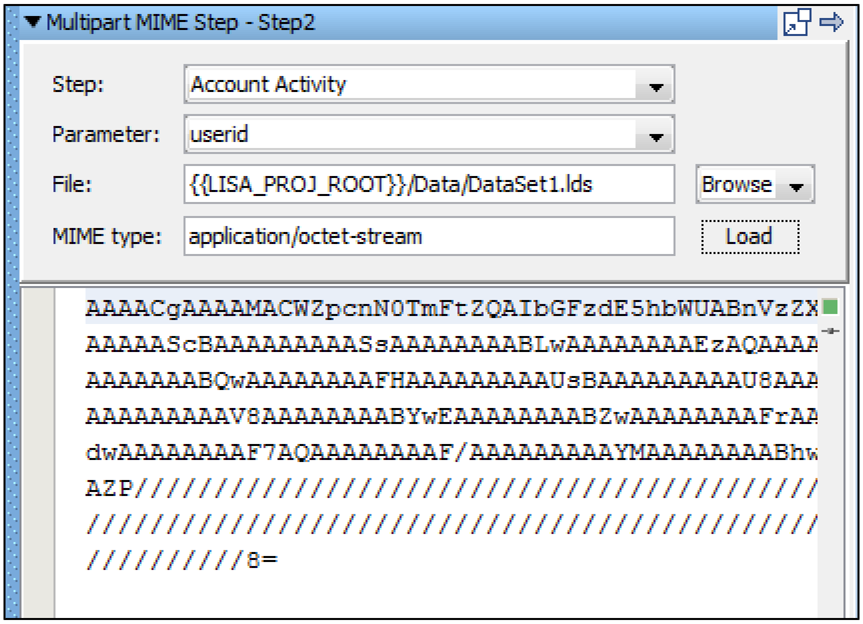
MIME type: application/octet-stream

No result to display

Enter the following parameters:

- **Step:** Select the name of the HTTP/HTML Request step, or select from the pull-down menu, the step that will receive the property containing the encoded document.
- **Parameter:** Select the name of the property, chosen from the pull-down menu, which is listed in the step named in the Step field.
- **File**: Enter the pathname or browse to the document to be encoded.
- **MIME Type**: Enter the MIME type expected by the server.

Click Load to encode the file.



Step: Account Activity

Parameter: userid

File: {{LISA_PROJ_ROOT}}/Data/DataSet1.lds

MIME type: application/octet-stream

```
AAAACgAAAAMACWZpcnN0TmFtZQAIbGFzdE5hbWUABnVzZX
AAAAAScBAAAAAAAAAASsAAAAAAAABLwAAAAAAAAAEzAQAAAA
AAAAAAABQwAAAAAAAAFHAAAAAAAAAUsBAAAAAAAAAAU8AAA
AAAAAAAAAV8AAAAAAAAABYwEAAAAAAAABZwAAAAAAAAFrAAA
dwAAAAAAAAAF7AQAAAAAAAAAF/AAAAAAAAAAYMAAAAAAAAAABhw
AZP///////////////////////////////////////////
//////////////////////////////////////////////
//////////8=
```

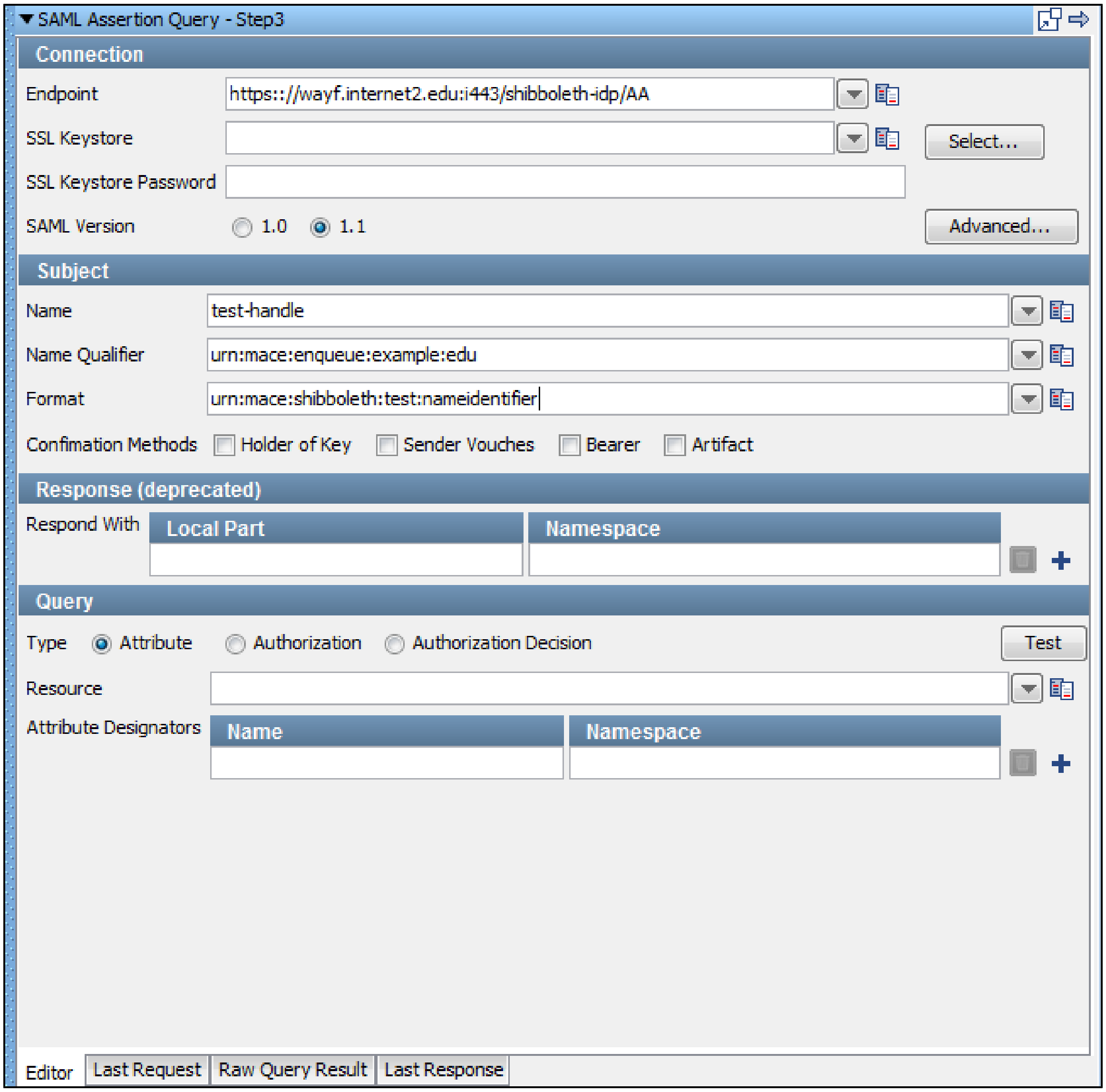
You can save the contents of the editor to a file using the Save icon.

In the example in the previous illustration, the Account Activity step has a post-parameter user id that contains the encoded version of the file Dataset1.lds.

SAML Assertion Query

The SAML Assertion Query step lets you obtain a SAML Assertion from an Identity Provider for later use in a Web Service Execution step that uses a WS-Security SAML 1.x Assertion Token.

Prerequisites: A cursory understanding of what type of SAML Assertion Query you need to perform. This information can be obtained from either the developer of the system that utilizes SAML Assertions as the form of identity security or from the Identity Provider administrator.
Parameter Requirements: At a minimum, you must know the URL to the Identity Providers SAML Query interface (Endpoint) and the Subject information (who/what you want to obtain a SAML Assertion for), what type of query you want to perform, and some extra information depending on which type of query.



SAML Assertion Query - Step3

Connection

Endpoint: https:://wayf.internet2.edu:i443/shibboleth-idp/AA
SSL Keystore:
SSL Keystore Password:
SAML Version: ○ 1.0 ● 1.1

Subject

Name: test-handle
Name Qualifier: urn:mace:enqueue:example:edu
Format: urn:mace:shibboleth:test:nameidentifier
Confimation Methods: □ Holder of Key □ Sender Vouches □ Bearer □ Artifact

Response (deprecated)

Respond With:

Local Part	Namespace

Query

Type: ● Attribute ○ Authorization ○ Authorization Decision
Resource:
Attribute Designators:

Name	Namespace

The SAML Assertion Query Editor has four tabs. The Editor tab lets you configure the query information. After doing so, you can test the query using the Test button in the Query section of the editor. After testing the query you can view the raw request that was sent in the Last Request tab. You can view the raw SOAP response in the Raw Query Result tab (for example, if it returned more than one assertion you can see that here). You can view the step response in the **Last Response** tab. This shows, for example, what will be used in the Web Service WS-Security token.

Connection

This information describes where the SAML Query API server lives and how you want to connect to it.

- **Endpoint**: The URL to the SAML Query API of the Identity Provider.
- **SSL Keystore**: If you must use client side identification certificates to connect to the Endpoint you can select the keystore file using the Select button, or you can select a previously-entered item from the pull-down list or enter one manually.
- **SSL Keystore Password**: The password for the SSL Keystore if used.
- **SAML Version**: The SAML version you want to use to query the Identity Provider.

Subject

This information describes who or what you want to request a SAML Assertion for. This could be a user or user group or other entity for which you want to provide an assertion about the subject's current authorization/privileges

- **Name**: The name of the entity (for example, username).
- **Name Qualifier**: A group or categorization used to qualify the Name (for example, domain).
- **Format**: This describes what format the name is being sent (for example, Full Name as opposed to Username).
- **Confirmation Methods**: Select which confirmation method types you want to include in the query. The query will only return assertions that contain at least one of the specified types. If you leave all types cleared it will return any assertion no matter what the confirmation method is.

Response (deprecated)

This information describes which assertion statements you want to be returned as part of the SAML Assertion. It has been deprecated as of SAML 1.1.

- **Local Part**: The element name (for example, AuthenticationStatement, AuthorizationDecisionStatement, and AttributeStatement).
- **Namespace**: The element namespace (for example, urn:oasis:names:tc:SAML:1.0:assertion).

Use the Add icon ➕ button to add more XML elements to the set to be returned. Use the Delete 🗑 button to remove any elements you have already added.

Query

A description of which type of query you want to perform. There are three different query types:

- Attribute
- Authorization
- Authorization Decision

Attribute

An attribute query responds with a set of Attribute Statements. For example, it may tell you which groups a subject is a member of.

- **Resource**: If you want to limit your query to a particular resource (for example, a particular web service, domain, file) you can specify the resource name.
- **Attribute Designators**: Each attribute is identified with a name and namespace (like XML elements). You can filter the set of attribute statements returned by specifying each attribute type you want to be returned. (for example, Name = urn:mace:dir:attribute-def:eduPersonScopedAffiliation, Namespace = urn:mace:shibboleth:1.0:attributeNamespace:uri).

Authorization

Used to request authentication statements related to a specific Subject SAML Assertions

- **Authorization Method**: If you want to limit your query to return Authorization Statements that are for a particular method of authorization (for example, urn:oasis:names:tc:SAML:1.0:am:X509-PKI, urn:oasis:names:tc:SAML:1.0:am:PGP, urn:oasis:names:tc:SAML:1.0:am:password). A set of predefined authorization methods are available from the pull-down list.

Authorization Decision

Used to request SAML Assertions for particular actions that a subject wants to perform given the evidence.

- **Resource**: If you want to limit your query to a particular resource (for example, a particular web service, domain, file) you can specify the resource name.
- **Actions**: You must specify at least one action for which you want to request authorization to perform (for example, login, view, edit) specified with a name (Data) and Namespace (like an XML element).
- **Evidence (Assertions):** Optionally specify one or more SAML Assertion to include with the Authorization Decision Query as advice to the Identity Provider. Specify the LISA property that holds the SAML Assertion XML. You can use the for l**isa.<stepname>.rsp** to use the response from a previous step (like another SAML Assertion Query or Parse Text step).
- **Evidence (Reference IDs):** Optionally specify assertion reference ids.

Web_Web Service Execution (Legacy)

The Web Service Execution step lets you construct a web service (SOAP protocol) test using the information in a Web Service Definition Language file (WSDL).

After you have identified the WSDL, LISA will construct a client with method calls for each operation specified in the WSDL. You use this web service client, and LISA's Object editor, to test the web service using your test data. Using additional information not included in the WSDL, you can test complex secure web services easily. The additional parameters are entered using several configuration screens available at the end of the web service execution wizard. These are discussed later in this section.

Prerequisites: Knowledge of LISA's Complex Object Editor is required to manipulate the Web Service Client built by LISA.

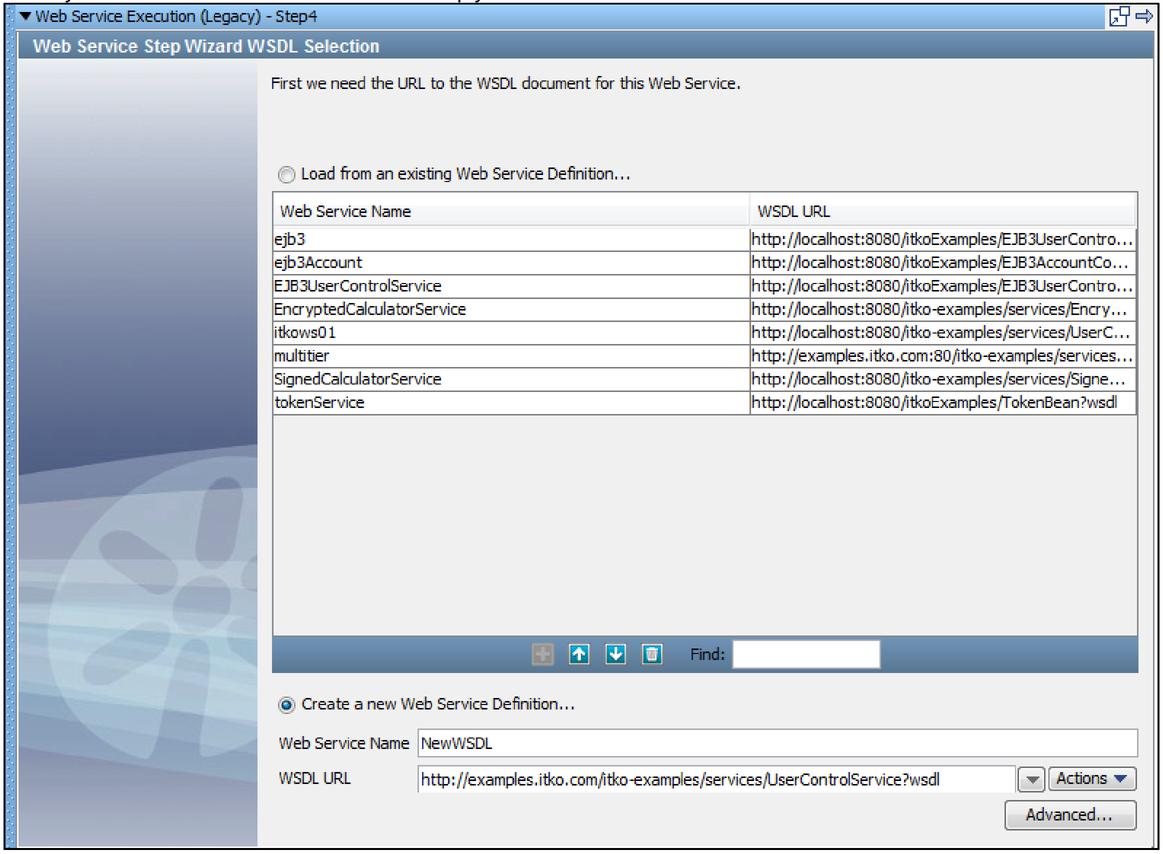
For more details, see the Complex Object Editor (COE).

Parameter Requirements: At a minimum, you must know the location of the WSDL (either from a local file or through the web). For complex web services, and secure web services there are several sets of parameters that you will need to gather before constructing your web service test step. These parameters are known to the web service developer, and are the same parameters required to use the web service in an application. These parameter sets will be specified when each advanced feature is discussed later in this section.

We will start by explaining how to test a simple web service, requiring only a WSDL. Then we will look at the advanced features that you may need to use. Finally we will look at the most complex of the advanced features, configuring WS-Security for a secure web service.

Testing a Simple Web Service

When you select the Web Service Execution step you will see the WSDL selection screen.


▼ Web Service Execution (Legacy) - Step4
Web Service Step Wizard WSDL Selection
Find:
Actions ▼
Advanced...

First we need the URL to the WSDL document for this Web Service.

○ Load from an existing Web Service Definition...

Web Service Name	WSDL URL
ejb3	http://localhost:8080/itkoExamples/EJB3UserContro...
ejb3Account	http://localhost:8080/itkoExamples/EJB3AccountCo...
EJB3UserControlService	http://localhost:8080/itkoExamples/EJB3UserContro...
EncryptedCalculatorService	http://localhost:8080/itko-examples/services/Encry...
itkows01	http://localhost:8080/itko-examples/services/UserC...
multitier	http://examples.itko.com:80/itko-examples/services...
SignedCalculatorService	http://localhost:8080/itko-examples/services/Signe...
tokenService	http://localhost:8080/itkoExamples/TokenBean?wsdl

● Create a new Web Service Definition...

Web Service Name: NewWSDL

WSDL URL: http://examples.itko.com/itko-examples/services/UserControlService?wsdl

There are two sections on this screen selectable by option button.

The upper section lets you select a WSDL that is available on the list, and has already constructed a web service client that is available to you.

To select one of these web services:

- Select Load from an existing Web Service Definition (optional: the option button is automatically selected if you select a web service name from the list).
- Click the web service name in the key column.

Click the Next button.

The lower section lets you enter a new WSDL.A client will be built corresponding to the contents of this WSDL.

- Select Create a new Web Service Definition (optional: the option button will automatically be selected if you begin entering information in the Web Service Name or WSDL URL fields)
- **Web Service Name**: Enter the name for the new service. This name must be unique. If a group of people plan on sharing test cases it is recommended to use a naming scheme that will help to ensure unique names for unique web services.
- **WSDL URL**: Enter the URL of the new WSDL, or select it from the pull-down list. You can also click the Actions list to browse the file system or search a UDDI server.
- **Advanced**: Opens a window where you can manage your namespaces.

Namespace mapping is described in the Advanced Features section.

We will use an example that is available on the Demo Server. The WSDL location is:
http://examples.itko.com/itko-examples/services/UserControlService?wsdl or
http://localhost:8080/itko-examples/services/UserControlService?wsdl,

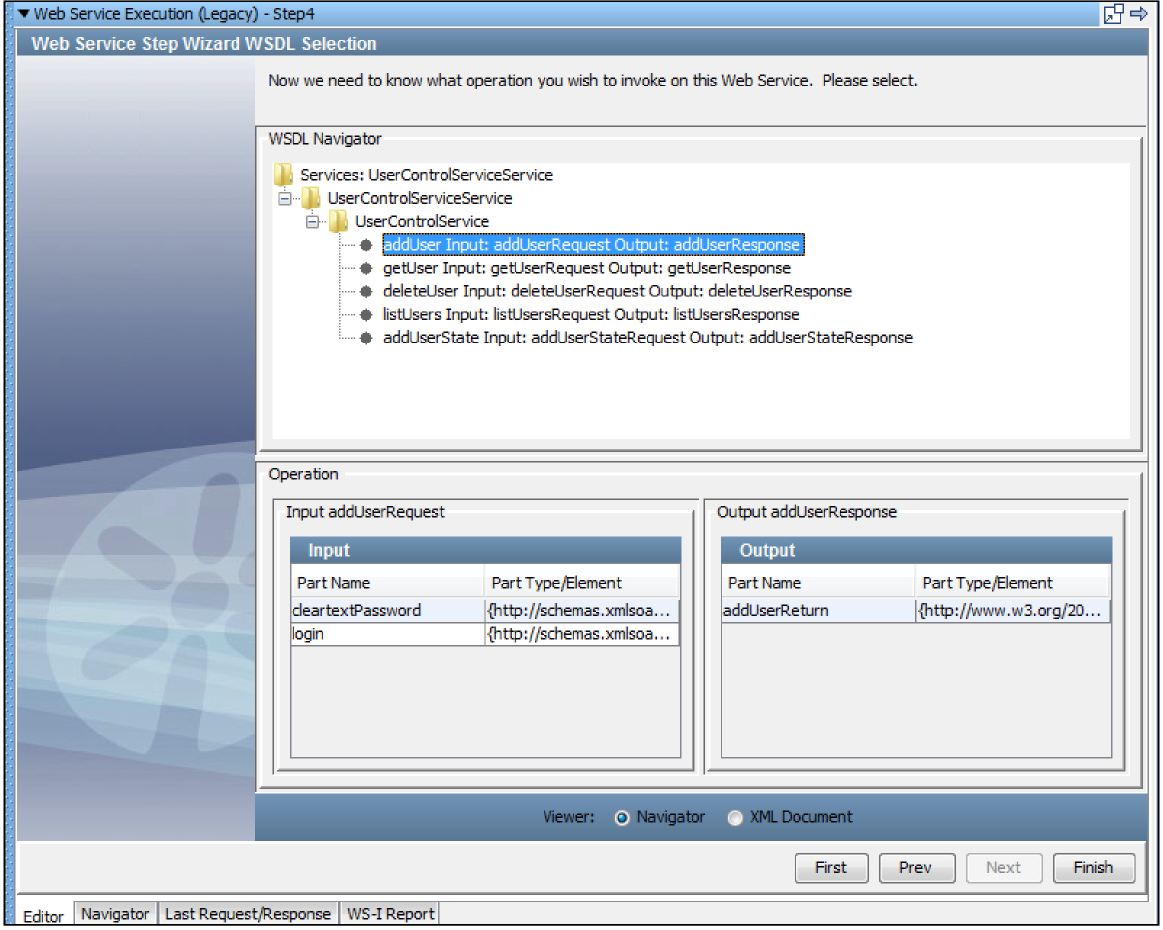
depending on the Demo Server you are using. We have chosen the first location in the previous illustration, and we have called the web service

NewWSDL.

Remove any spaces in the web service name.

Click the Next button.

You will now see the WSDL Navigator screen:



Web Service Execution (Legacy) - Step4

Web Service Step Wizard WSDL Selection

Now we need to know what operation you wish to invoke on this Web Service. Please select.

WSDL Navigator

- Services: UserControlServiceService
  - UserControlServiceService
    - UserControlService
      - addUser Input: addUserRequest Output: addUserResponse
      - getUser Input: getUserRequest Output: getUserResponse
      - deleteUser Input: deleteUserRequest Output: deleteUserResponse
      - listUsers Input: listUsersRequest Output: listUsersResponse
      - addUserState Input: addUserStateRequest Output: addUserStateResponse

Operation

Input addUserRequest

Part Name	Part Type/Element
cleartextPassword	{http://schemas.xmlsoa...
login	{http://schemas.xmlsoa...

Output addUserResponse

Part Name	Part Type/Element
addUserReturn	{http://www.w3.org/20...

Viewer: ● Navigator ○ XML Document

The upper section of the Navigator, titled **WSDL Navigator**, shows a hierarchy of folders, the last of which is the web service (User Control Service in our case), listing all the operations available.

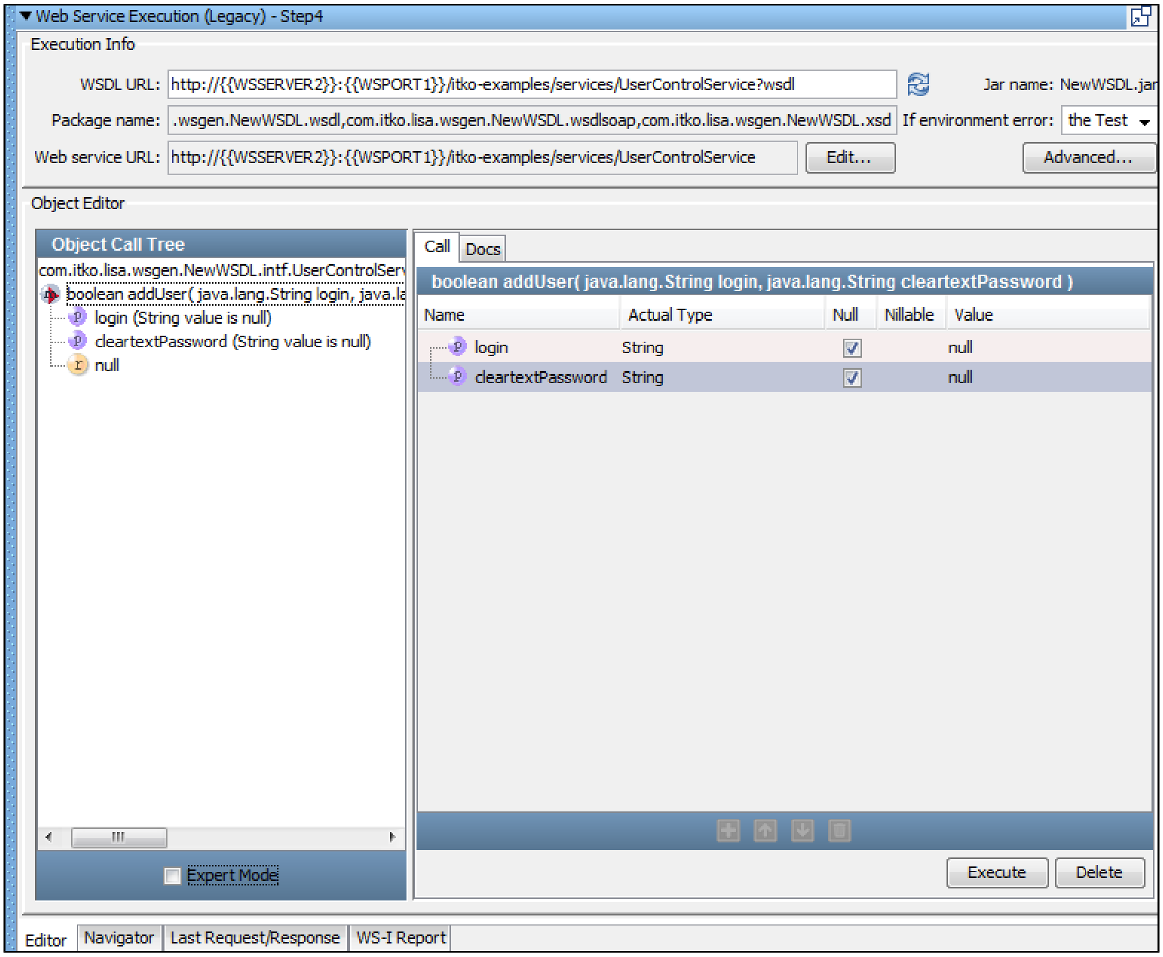
- Select an operation to invoke on the web service; we have chosen **addUser**.

The lower section now displays the input and output parameter information for the operation **addUser**.

> ⚠ Part Name and Part Types are standard web service notation for the input and output parameters. These are informational only; the only selection that needs to be made on this screen is the web service operation to execute in the WSDL Navigator.

Select the Viewer from Navigator or XML Document option. The Navigator option is the default.

Click Finish to bring up the Complex Object Editor with the **Execution Info** panel above it.



Web Service Execution (Legacy) - Step4

Execution Info

WSDL URL: http://{{WSSERVER2}}:{{WSPORT1}}/itko-examples/services/UserControlService?wsdl

Jar name: NewWSDL.jar

Package name: .wsgen.NewWSDL.wsdl,com.itko.lisa.wsgen.NewWSDL.wsdlsoap,com.itko.lisa.wsgen.NewWSDL.xsd

If environment error: the Test

Web service URL: http://{{WSSERVER2}}:{{WSPORT1}}/itko-examples/services/UserControlService

Object Editor

Object Call Tree

com.itko.lisa.wsgen.NewWSDL.intf.UserControlServ

boolean addUser(java.lang.String login, java.la

login (String value is null)

cleartextPassword (String value is null)

null

boolean addUser(java.lang.String login, java.lang.String cleartextPassword)

Name	Actual Type	Null	Nillable	Value
login	String	☑		null
cleartextPassword	String	☑		null

□ Expert Mode

The Execution Info panel comes populated with all the necessary fields like WSDL URL, Package name, Web Service URL and the JAR name. Other fields are:

- **If environment error**: Select the next step to execute, typically fail, if an environmental error is returned by the web service. Most SOAP faults will not trigger this workflow change.
- **Edit**: Click the Edit button if you wish to change the web service endpoint URL.

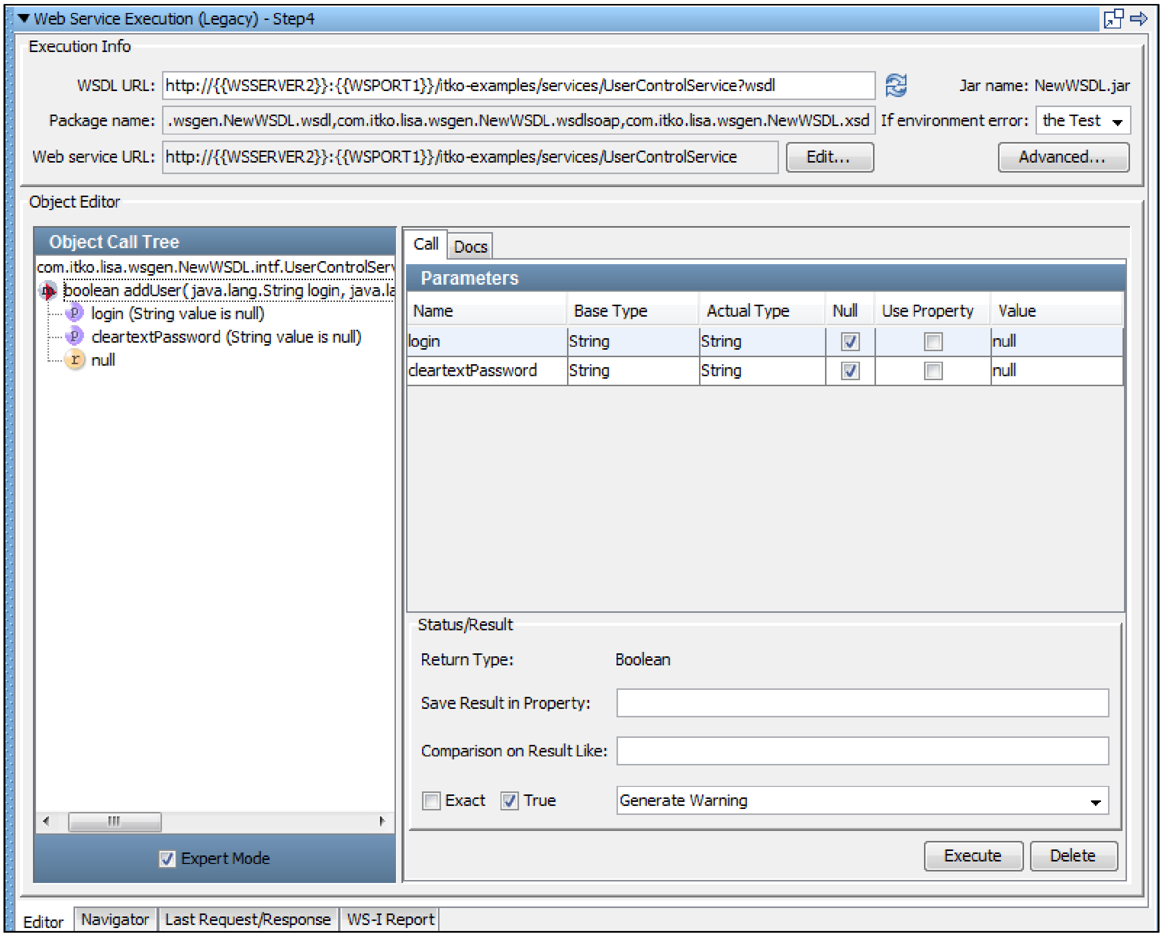
⚠ The host (WSSERVER) and port (WSPORT) for the WSDL and Web Service URL are automatically parameterized.

- **Advanced**: Opens a window where you can configure many advanced features associated with your web service.

⚠ The advanced features are described in the next section "Advanced Features". The WS-Security options are discussed in the section after that: "WS-Security"

The Object Editor panel uses the standard Object Editor to manipulate the web service client.

The previous illustration shows the object editor in the standard mode. Selecting the Expert Mode check box reveals the expert mode view:



WSDL URL: http://{{WSSERVER2}}:{{WSPORT1}}/itko-examples/services/UserControlService?wsdl

Jar name: NewWSDL.jar

Package name: .wsgen.NewWSDL.wsdl,com.itko.lisa.wsgen.NewWSDL.wsdlsoap,com.itko.lisa.wsgen.NewWSDL.xsd

If environment error: the Test

Web service URL: http://{{WSSERVER2}}:{{WSPORT1}}/itko-examples/services/UserControlService

com.itko.lisa.wsgen.NewWSDL.intf.UserControlServ
- boolean addUser(java.lang.String login, java.la
  - login (String value is null)
  - cleartextPassword (String value is null)
  - null

Name	Base Type	Actual Type	Null	Use Property	Value
login	String	String	☑	□	null
cleartextPassword	String	String	☑	□	null

Return Type: Boolean

Save Result in Property:

Comparison on Result Like:

□ Exact ☑ True Generate Warning

☑ Expert Mode

In Expert Mode you can configure inline filters and assertions. To use an inline filter, enter a property name in the Save Result in Property text box. To add an assertion, configure it using the 'Comparison on Result Like' text box, and the Exact and Tru' check boxes.

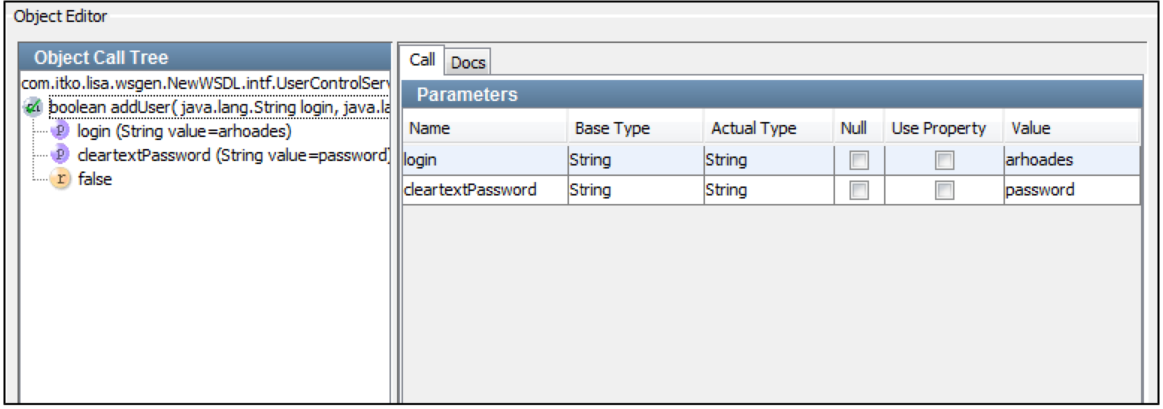
In either mode you can supply the values for each parameter listed. In our example, the **addUser** method takes two parameters, **login** and **cleartextPassword**. These parameter values can be static values, LISA properties, or **NULL**. Enter a static value by filling in the **Value** cell for the parameter. Enter a LISA property by selecting the Use Property check box and filling in the **Value** cell with the property name. Enter **NULL** by selecting the Null check box.

If the parameter type is a complex object, the **Value** cell shows **Ctrl-click to edit**. Press <Ctrl> and click the cell to open the Complex Object Editor can enter the fields of the complex object.

If the **Actual Type** cell displays the name of a Java interface instead of a Java class, you will have to change the Actual Type to be the name of a Java class. An error message is displayed if you try using a Java interface. The developers of the web service should be able to provide you with a class name that can be used in place of the interface name.
For detailed instructions on the use of the Object editor see "Complex Object Editor."

Click Execute.

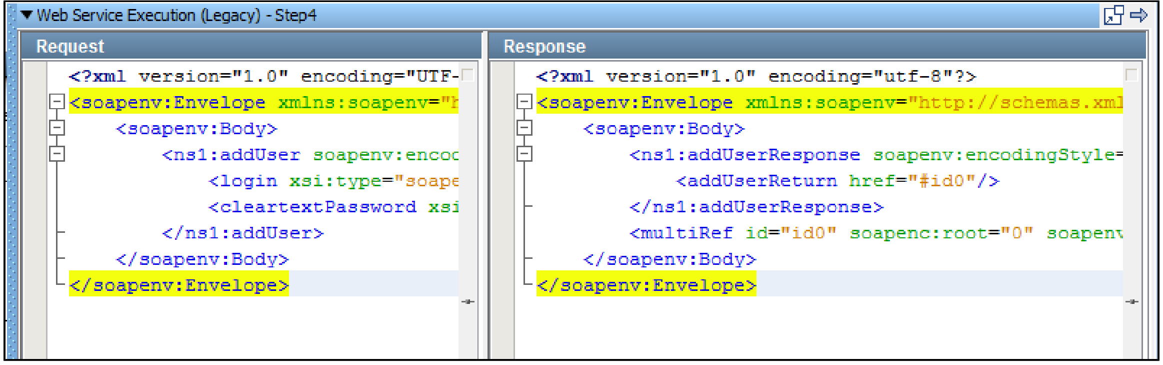
After execution, you can see the result in the **Object Call Tree** in the left panel:



com.itko.lisa.wsgen.NewWSDL.intf.UserControlServ
- boolean addUser(java.lang.String login, java.la
  - login (String value=arhoades)
  - cleartextPassword (String value=password)
  - false

Name	Base Type	Actual Type	Null	Use Property	Value
login	String	String	□	□	arhoades
cleartextPassword	String	String	□	□	password

To see the actual SOAP request/ SOAP response, click the **Last Request/Last Response** tab at the bottom of the panel:



Web Service Execution (Legacy) - Step4

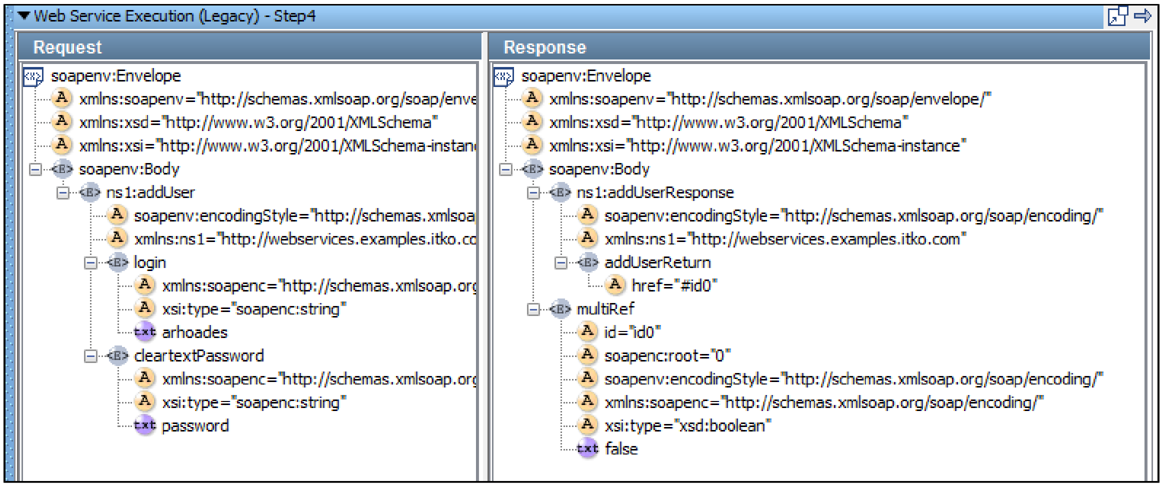
Request

```
<?xml version="1.0" encoding="UTF-
<soapenv:Envelope xmlns:soapenv="h
    <soapenv:Body>
        <ns1:addUser soapenv:encoc
            <login xsi:type="soape
            <cleartextPassword xsi
        </ns1:addUser>
    </soapenv:Body>
</soapenv:Envelope>
```

Response

```
<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xml
    <soapenv:Body>
        <ns1:addUserResponse soapenv:encodingStyle=
            <addUserReturn href="#id0"/>
        </ns1:addUserResponse>
        <multiRef id="id0" soapenc:root="0" soapenv
    </soapenv:Body>
</soapenv:Envelope>
```

You can view **Headers**, or examine the **XML** or the **DOM Tree**using the tabs at the bottom of the panel. The following example shows the DOM view.



Web Service Execution (Legacy) - Step4

Request

```
soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/enve
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instan
    soapenv:Body
        ns1:addUser
            soapenv:encodingStyle="http://schemas.xmlsoa
            xmlns:ns1="http://webservices.examples.itko.co
            login
                xmlns:soapenc="http://schemas.xmlsoap.org
                xsi:type="soapenc:string"
                arhoades
            cleartextPassword
                xmlns:soapenc="http://schemas.xmlsoap.org
                xsi:type="soapenc:string"
                password
```

Response

```
soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    soapenv:Body
        ns1:addUserResponse
            soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            xmlns:ns1="http://webservices.examples.itko.com"
            addUserReturn
                href="#id0"
        multiRef
            id="id0"
            soapenc:root="0"
            soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
            xsi:type="xsd:boolean"
            false
```

At this point, you are ready to add filters and assertions to the response.

You can add filters with the Add Filter icon, or by selecting the filter element for the current step.

You can add assertions with the Add Assertion icon, or by selecting the assertions element for the current step.

> Selecting the filters or assertions elements will give you greater choices.

This test step is now complete.

You can go back to the Object Editor, and select additional calls to other procedures in the web service, and execute them in this same step. This is useful if you need to add calls on request or response objects to "drill down" to values. The best practice for authoring test cases is to put each web service call in a separate step. It makes a test case easier to follow and easier to pinpoint which call is causing a problem if there are errors.

Web Services Advanced Features

This section describes the advanced features available in the Web Service Execution step. The following features are covered:

- Namespace mapping
- UDDI access point lookup
- Specific protocol requirement
- Maximum wait time
- Security in transport layer
- WSI basic profile 1.1 validation
- Advanced error handling
- Custom transport headers
- WS Addressing
- Customer SOAP Headers

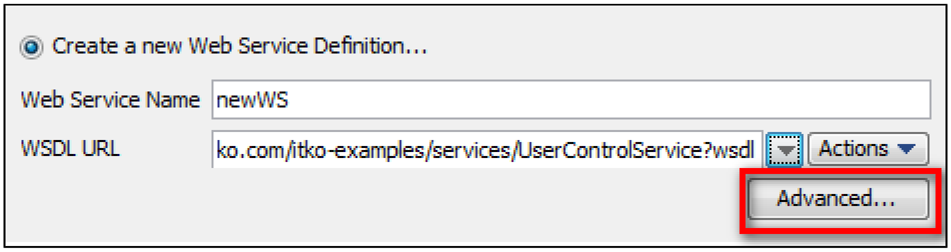
Namespace Mapping

For a nontrivial WSDL with multiple namespaces, or for reuse of client side stubs, you may want to customize the namespace mappings. This namespace mapping is used during the creation of the client classes that are used to execute the web service call. The default behavior is to separate out each type defined in the WSDL into a unique package based on its namespace. Use this dialog to change this default behavior.

The default namespace is com.itko.wsgen.<Web service name>

For each namespace in the WSDL, the namespace prefix (or a generated unique prefix if one is not available in the WSDL) is added to the default namespace to create a unique namespace: com.itko.wsgen.<web service name>**.**<namespace prefix>

To customize the namespace mappings, select the Advanced button on the WSDL Selection screen after you have entered your new Web Service Name and WSDL URL.

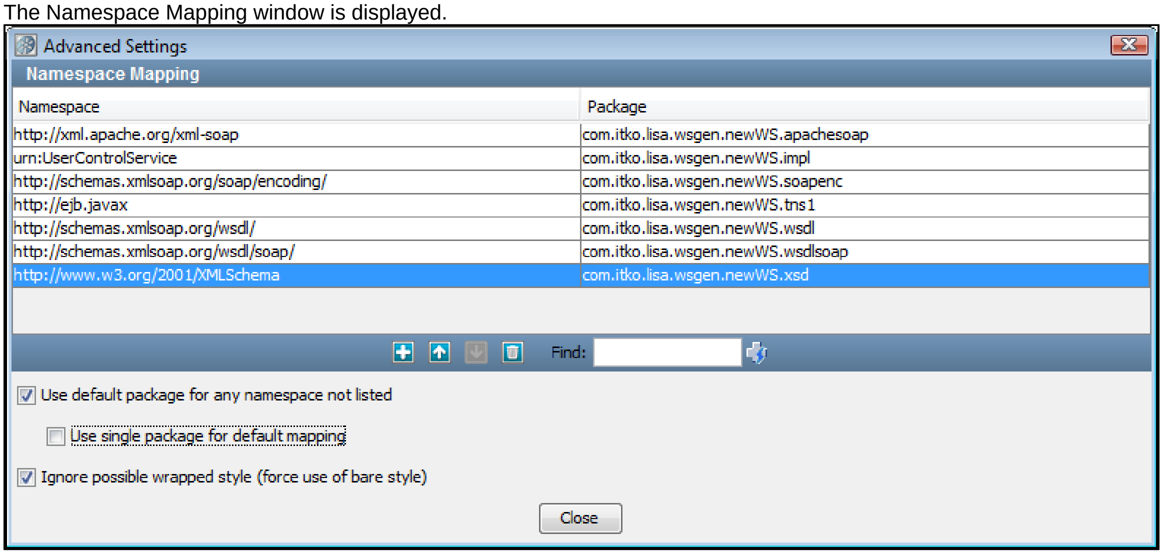


● Create a new Web Service Definition...

Web Service Name: newWS

WSDL URL: ko.com/itko-examples/services/UserControlService?wsdl

The Namespace Mapping window is displayed.



Namespace	Package
http://xml.apache.org/xml-soap	com.itko.lisa.wsgen.newWS.apachesoap
urn:UserControlService	com.itko.lisa.wsgen.newWS.impl
http://schemas.xmlsoap.org/soap/encoding/	com.itko.lisa.wsgen.newWS.soapenc
http://ejb.javax	com.itko.lisa.wsgen.newWS.tns1
http://schemas.xmlsoap.org/wsdl/	com.itko.lisa.wsgen.newWS.wsdl
http://schemas.xmlsoap.org/wsdl/soap/	com.itko.lisa.wsgen.newWS.wsdlsoap
http://www.w3.org/2001/XMLSchema	com.itko.lisa.wsgen.newWS.xsd

☑ Use default package for any namespace not listed

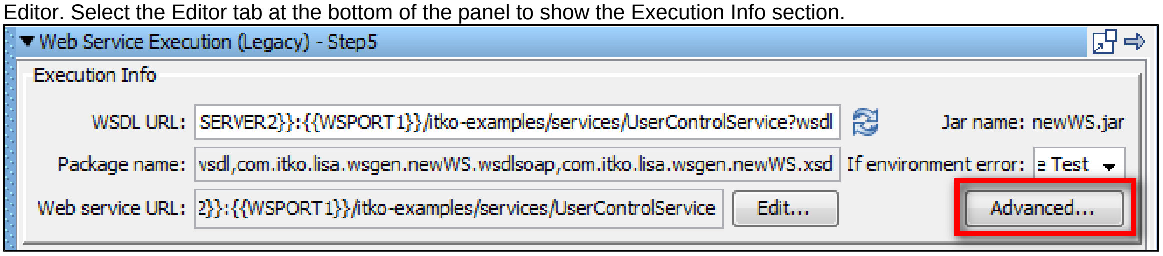
☐ Use single package for default mapping

☑ Ignore possible wrapped style (force use of bare style)

Click the Auto Populate icon to list all the WSDL namespaces and to see what the default LISA mappings will be. You can now add, remove, or edit the namespace mappings. At the bottom of the screen you can select the following options:

- Use default package for any namespace not listed.
- Use single package for default mapping.
- Ignore possible wrapped style (force use of bare style): For every operation in the WSDL, there is an attempt to auto-detect if the WSDL has been defined in the wrapped-style and if so, the set of parameters that are to be sent is unwrapped. For example, if you have an operation defined in the wrapped-style and it takes an OperationRequest type parameter. If you have this option cleared (default) it will expand the OperationRequest type and specify each of its sub-elements as parameters (for example, instead of passing an OperationRequest, it would take it individual elements that make up an OperationRequest like param1, param2). If you check this option it will not attempt to detect the wrapped-style and will generate the operation with the single OperationRequest parameter type. This can be useful if you plan on generating parameters using Excel DTOs.

For the rest of the advanced features in this section you will need to select the Advanced button on the Execution Info section of the Web Service Editor. Select the Editor tab at the bottom of the panel to show the Execution Info section.



WSDL URL: SERVER2}}:{{WSPORT1}}/itko-examples/services/UserControlService?wsdl

Jar name: newWS.jar

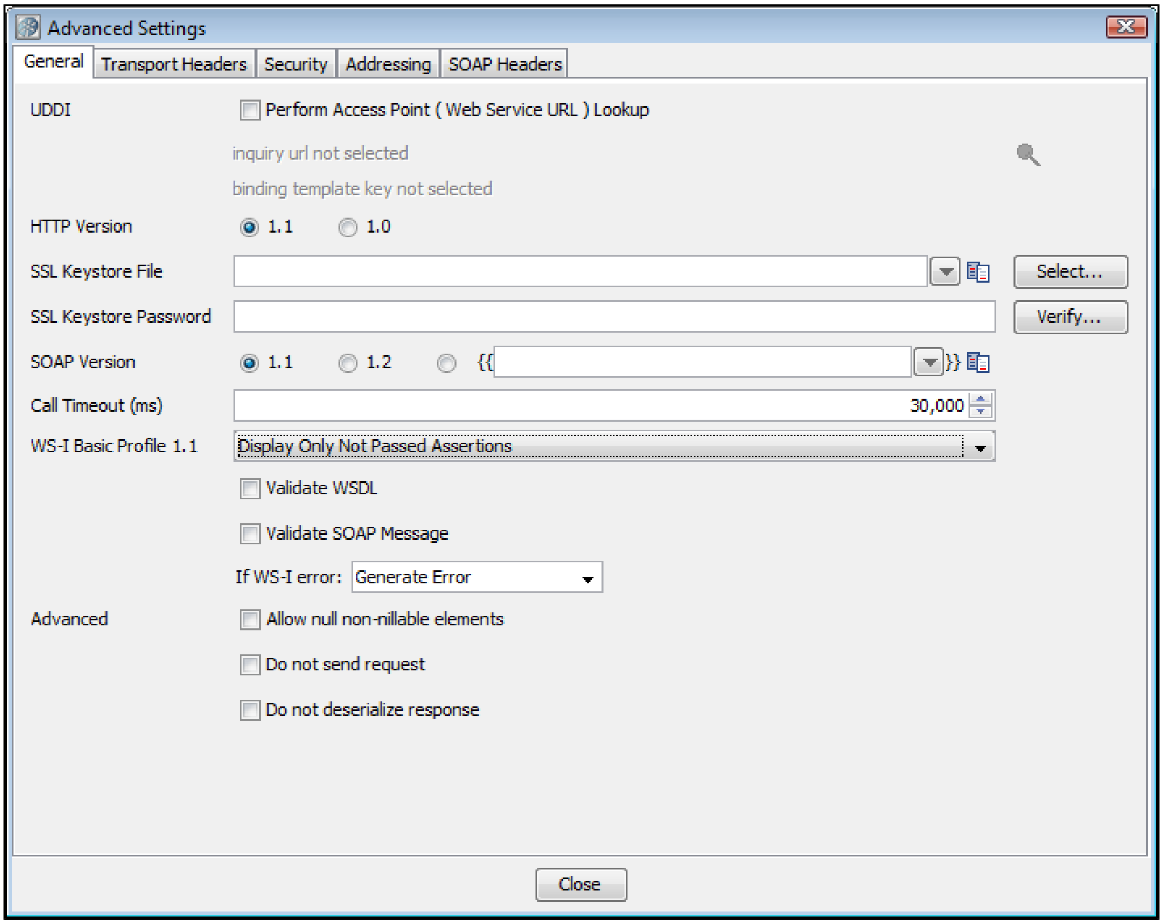
Package name: vsdl,com.itko.lisa.wsgen.newWS.wsdlsoap,com.itko.lisa.wsgen.newWS.xsd

If environment error: e Test

Web service URL: 2}}:{{WSPORT1}}/itko-examples/services/UserControlService

A window with five tabs is displayed. We will be referring to four of these tabs: General, Transport Headers, Addressing, and SOAP Headers in this section. The fifth, Security, is the topic of the next section.

General Tab



Advanced Settings

General | Transport Headers | Security | Addressing | SOAP Headers

UDDI □ Perform Access Point (Web Service URL) Lookup

inquiry url not selected

binding template key not selected

HTTP Version ● 1.1 ○ 1.0

SSL Keystore File

SSL Keystore Password

SOAP Version ● 1.1 ○ 1.2 ○ {{ }}

Call Timeout (ms) 30,000

WS-I Basic Profile 1.1 Display Only Not Passed Assertions

□ Validate WSDL

□ Validate SOAP Message

If WS-I error: Generate Error

Advanced □ Allow null non-nillable elements

□ Do not send request

□ Do not deserialize response

The following parameters are available on the General Tab:

- **UDDI Perform Access Point (Web Service URL) Lookup**

- **Inquiry URL and Binding Template**: Use the Search UDDI Server Find 🔍 button to navigate to the correct Binding Template to perform the lookup.

> ⚠ When creating the step if you used the UDDI Search function when specifying the Web Service WSDL URL, these values will be automatically filled in. If the Inquiry URL is specified but the Binding Template is not, you may have preformed a Model search. To locate the Binding Template you need to perform a search at a higher level of the hierarchy and drill down to the TModel through a particular Binding Template.

- **HTTP Version**: Select version 1.0 or 1.1.
- **SSL Keystore File**: Enter the name of the keystore file where the client identity certificate is stored. It can be in jks or pkcs format.
- **SSL Keystore Password**: Enter the password for the keystore.
- **SOAP Version**: Select version 1.1, 1.2, or a property to use for the version.
- **Call Timeout (ms)**: Enter the maximum wait time for a response from a web service call (milliseconds). This is especially useful when using SOAP over JMS.

You can specify global certificates properties for SSL in your local.properties file.

For global certificates (web server, raw SOAP, and web service steps):

- **ssl.client.cert.path**: A full path to the keystore.
- **ssl.client.cert.pass**: Password for the keystore. This password will be automatically encrypted using AES (Advanced Encryption Standard) when LISA runs.
- **ssl.client.key.pass:** An optional password for the key entry if you are using the JKS keystore and the key has different password from the keystore. This password will be automatically encrypted using AES (Advanced Encryption Standard) when LISA runs.

⚠ This is currently not an available option to be set in the WS Test step and if required must be set in the local.properties file.

For web service steps only certificates (not raw SOAP steps):

- **ws.ssl.client.cert.path**: A full path to the keystore.
- **ws.ssl.client.cert.pass**: Password for the keystore. This password will be automatically encrypted using AES (Advanced Encryption Standard) when LISA runs.
- **ws.ssl.client.key.pass**: An optional password for the key entry if you are using the JKS keystore and the key has a different password from the keystore. This password will be automatically encrypted using AES (Advanced Encryption Standard) when LISA runs.

⚠ This is currently not an available option to be set in the WS Test step and if required must be set in the local.properties file.

If you have values in both local.properties and in the General tab, the values in the General tab will be used.

⚠

- **WS-I Basic Profile 1.1**: You can choose four different validation levels in the pull-down menu:
  - Display All Assertions
  - Display All But Info Assertions
  - Display Only Failed Assertions
  - Display Only Not Passed Assertions
- **Validate WSDL**: Select to validate the WSDL.
- **Validate SOAP Message**: Select to validate the SOAP message.
- **If WS-I error**: Select the step to redirect to on error.

Validation failures are common, but usually should not affect the outcome of the test. It is good practice to set the next step to continue so that you can complete the test. When you click the **Execute** button on the Object Editor screen, the validation will run and a report will be generated and saved (in the reports directory in the LISA install directory). You can view the report by pressing the **WS-I Report** tab at the bottom of the screen. The format of this report is standard, and is dictated by the Web-Services-Interoperability Organization (WS-I).

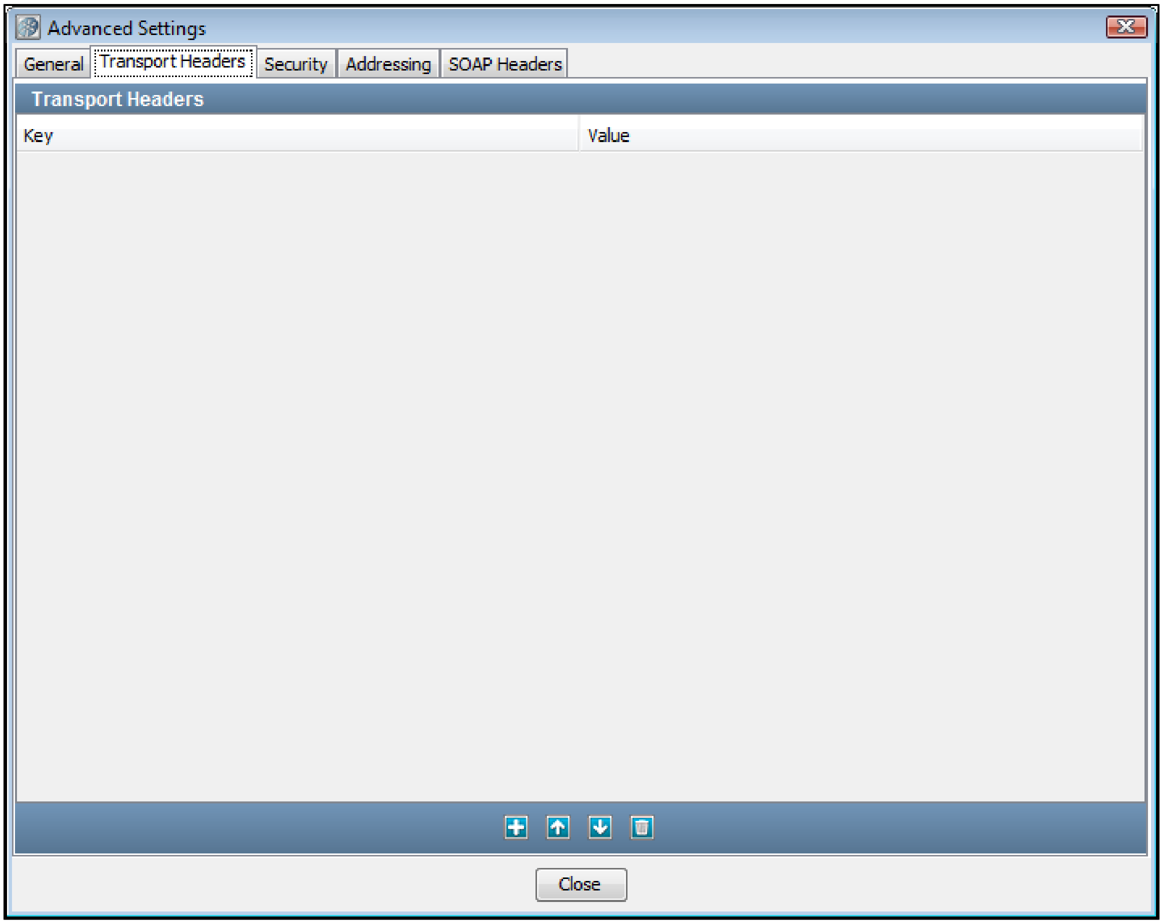
There are times when a valid XML SOAP response is received, but an exception is thrown when trying to process your results. The following options should let you process, by dealing with the XML response directly rather than processing the SOAP response.

- **Allow null non-nillable elements**: If there are non-nillable elements in the SOAP request, send the request even if the elements are null.

- **Do not send request**: Generate the SOAP request message, but do not send message to Web Service Endpoint. The step response will be the SOAP request message.

- **Do not deserialize response**: Send the SOAP request message and receive a SOAP response, but do not try to process the SOAP response message to generate a Java response object. The step response will not be affected, but the return value of the method invoked in the object editor will be null.

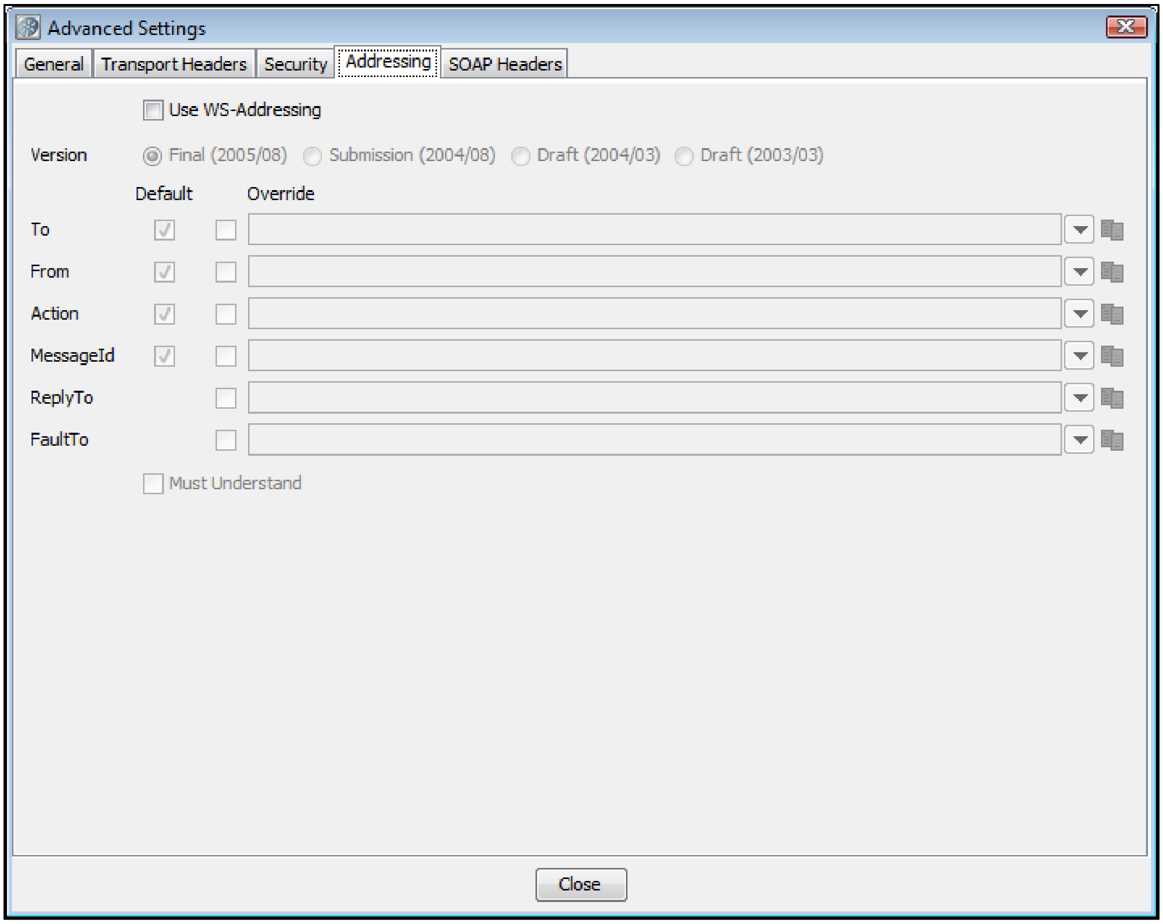
Transport Headers Tab



Click the Add ➕ icon and enter the header as a key/value pair. Repeat to add additional headers. These will be sent as HTTP Headers or JMS properties, based on which protocol is used to execute the web service operation.

Addressing Tab

You can send a WS-Addressing header with your request. The WSDL does not specify if WSAddressing information is required, so you must configure it.



☐ Use WS-Addressing

Version ● Final (2005/08) ○ Submission (2004/08) ○ Draft (2004/03) ○ Draft (2003/03)

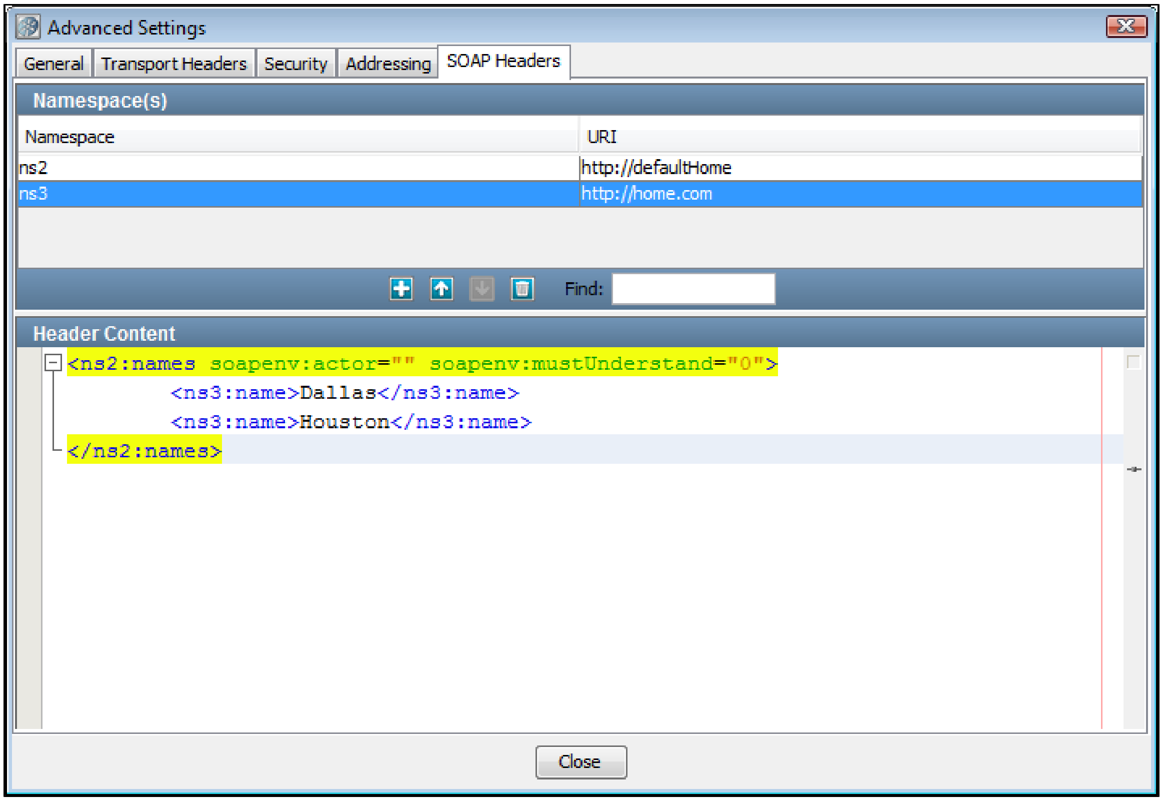
	Default	Override	
To	☑	☐	
From	☑	☐	
Action	☑	☐	
MessageId	☑	☐	
ReplyTo		☐	
FaultTo		☐	

☐ Must Understand

- **Use WS-Addressing**: Select to use WS-Addressing.
- **Version**: Select appropriate version. Several versions of the WS-Addressing specification are listed as options because some web services platforms (for example, .Net) still use the older draft specifications. You will need to determine which your web service platform is using. LISA will populate as many values as it can. Then you can choose to use the default value or override it. You can choose not to send some of the default elements by clearing the default check box for that element.
- **Must Understand** check box: Select if you want to assure that the web service can understand the WSAddressing header.

SOAP Headers Tab

Click the Add icon in the top panel and enter any namespace prefix(es) and URI(s) used in the custom header content that you want to be defined in the SOAP envelope element tag. Then enter your custom header as a valid XML fragment.



Namespace	URI
ns2	http://defaultHome
ns3	http://home.com

```
<ns2:names soapenv:actor="" soapenv:mustUnderstand="0">
        <ns3:name>Dallas</ns3:name>
        <ns3:name>Houston</ns3:name>
</ns2:names>
```

The header appears in the SOAP request:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns2=[http://defaulthome/]}[http://defaultHome] xmlns:ns3="http://home.com">
<soapenv:Header>
 <ns2:names soapenv:actor="" soapenv:mustUnderstand="0">
 <ns3:name>Dallas</ns3:name>
 <ns3:name>Houston</ns3:name>
 </ns2:names>
</soapenv:Header>
```

WS-Security

Prerequisites: Before configuring a secure web service for testing, familiarize yourself with the testing of simple web services as described in this section .

Parameter Requirements: When testing secured web services, there are several sets of parameters you will need to gather before constructing your web service test step. These parameters are known to the web service developer, and are the same parameters that would be required to call the web service from an application. The parameter sets needed must be made available to you before you can configure your web service for testing. You will need to know which security options are being used by the web service, the configuration parameter values for the options used, and the order in which they are expected in the SOAP headers for the request and the response.

This example uses the parameters needed for the security test case available on the Demo Server. This test case, named **ws_security**, (located in the example directory), uses a timestamp, encryption and a signature token. The WSDL location is:

{+}http://localhost:8080/itko-examples/services/EncryptedCalculatorService?wsdl+, or
{+}http://examples.itko.com:80/itko-examples/services/EncryptedCalculatorService?wsdl+ for encryption
{+}http://localhost:8080/itko-examples/services/SignedCalculatorService?wsdl+, or
{+}http://examples.itko.com:80/itko-examples/services/SignedCalculatorService?wsdl+ for the signature token, depending on the Demo Server you are using.

The web service names are EncryptedCalculatorService and SignedCalculatorService.

LISA has extensive support for WS-Security including:

- **Send (request)**

  Encryption

Signature Token

Timestamp

Username Token

SAML Assertion Token

- **Receive (response)**

Decryption

Signature Verification

Timestamp Receipt

Username Token Verifier

SAML Assertion Receipt

Signature Confirmation

To add security headers to the web service request:

Select the Security tab. Click Send.

- Select the Must Understand check box (if needed or if you want to verify that the WS-Security header is processed by the server).
- Enter the Actor/Role name (if needed: most web services do not use multiple Actors/Roles).
- Click the Add ⊞ icon and select the security action type to add. You will be presented with the configuration panel for that security action type. These are discussed in the following section.

Adding security verification to the response is very similar.

Click the Security tab. Click the Add ⊞ icon and select Receive.

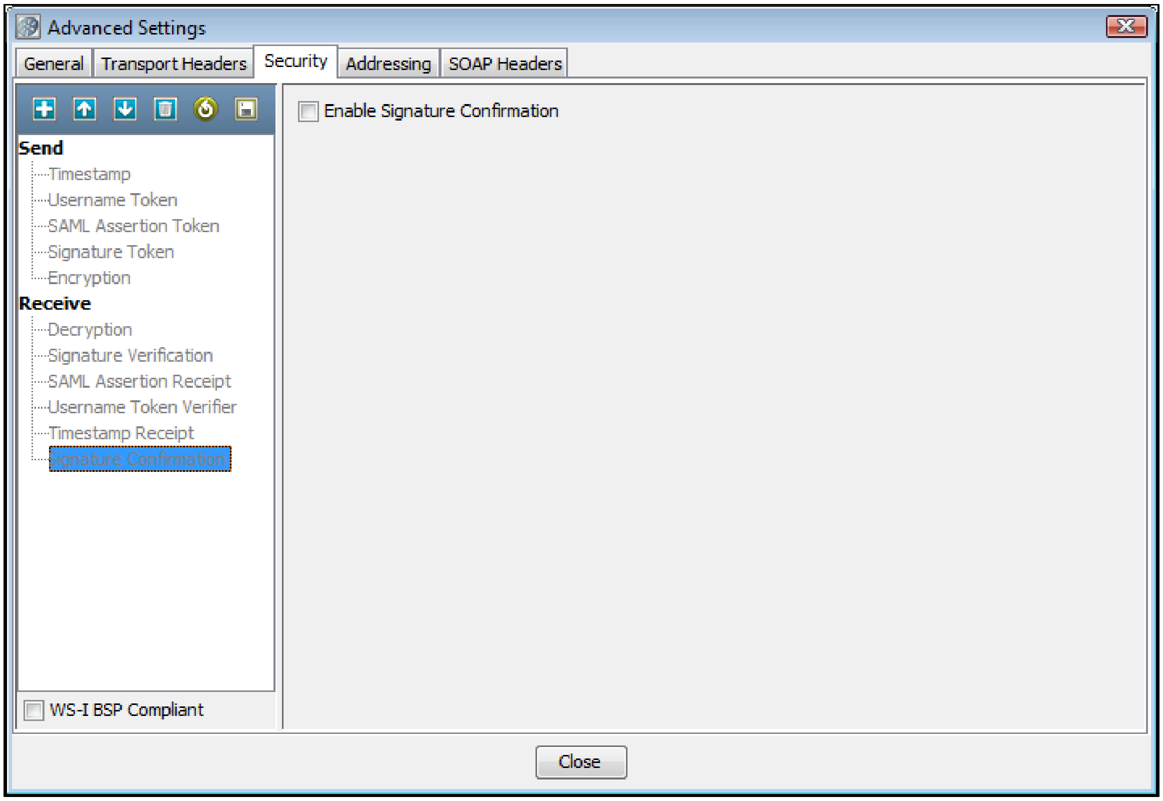
- Enter the Actor/Role name (if needed).
- Click the Add ⊞ icon and select the security action type. You will be presented with the configuration panel for that security action type. These are discussed in the following section.

> ⚠ You can add as many security types as are needed to execute your Web Service.

When a keystore is being used in a security action type configuration (described in the following section for each type), you can verify your keystore settings to make sure you are using the correct format, password, alias and alias password. There is a Verify button on the editors for Signature, Encryption/Decryption and SAML Assertion Token. Clicking the Verify button will invoke the Keystore Verifier, which will produce a verification report.

If you do not know the expected alias name for a WS-Security setting, you can use the Keystore Verifier to list all the aliases in the keystore. Leave the Keystore Alias and Alias Password boxes empty and click the Verify button. The Keystore Verifier is described at the end of this section on WS-Security.

An example screen with several security types chosen is shown in the following screenshot (disabled actions are dimmed).



☐ Enable Signature Confirmation

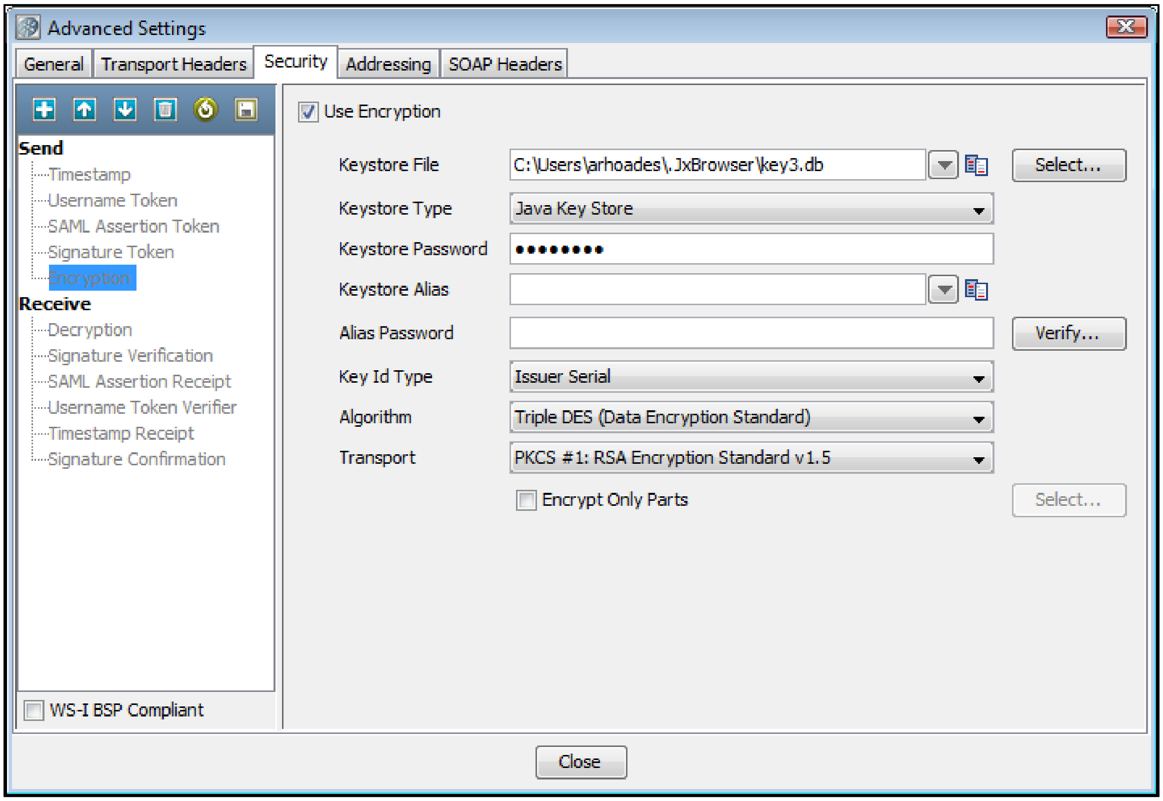
☐ WS-I BSP Compliant

In the following sections we describe the parameters needed to run the ws-security example.

XML Encryption/Decryption

The parameters required for configuring XML Encryption and Decryption are shown here.

Encryption



☑ Use Encryption

Keystore File C:\Users\arhoades\.JxBrowser\key3.db

Keystore Type Java Key Store

Keystore Password ●●●●●●●●

Keystore Alias

Alias Password

Key Id Type Issuer Serial

Algorithm Triple DES (Data Encryption Standard)

Transport PKCS #1: RSA Encryption Standard v1.5

☐ Encrypt Only Parts

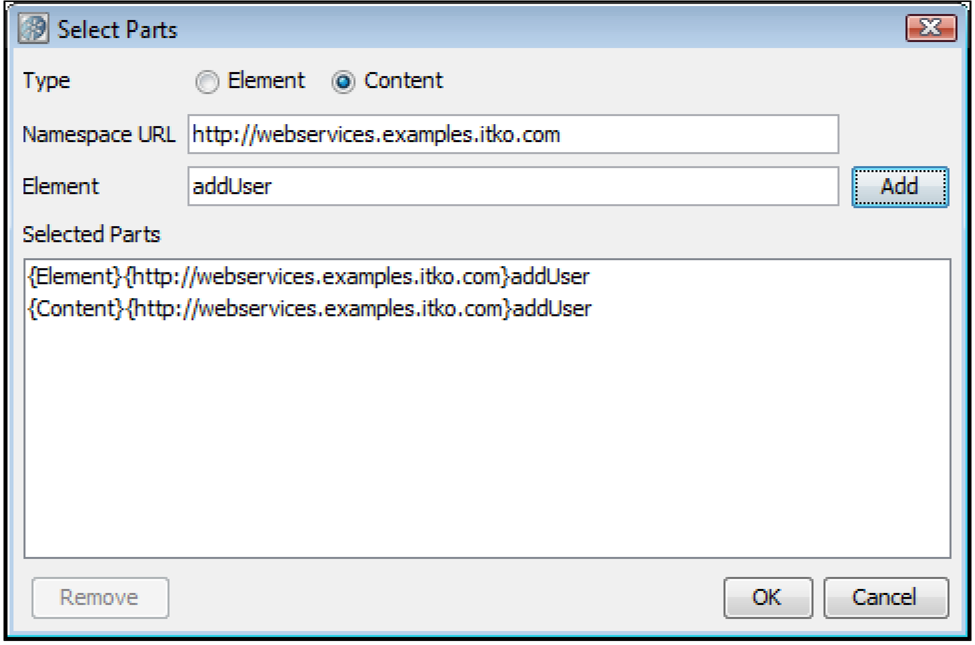
☐ WS-I BSP Compliant

Select the Use Encryption check box.

- **Keystore File**: Enter the location of the keystore file.
- **Keystore Type**: Select **Java Key Store (jks)** or **Personal Information Exchange (PKCS #12)**.
- **Keystore Password**: Enter the password for the keystore.
- **Keystore Alias**: Enter an alias for a public key.
- **Alias Password**: Leave blank or enter the same as Keystore Password for PKCS #12 files.
- **Key ID Type**: Select the appropriate key ID type from the pull-down menu.
- **Algorithm**: Enter **Triple DES**, **AES 128**, **AED 192**, or **AES 236**.
- **Transport**: Enter **PKCS#1**: RSA Encryption Standard v1.5 is the only option.

The default behavior is to encrypt only the SOAP body contents.

- **Encrypt Only Parts**: If you want to specify different parts to encrypt click the Select button to identify the parts to be encrypted.



Select Parts

Type ○ Element ● Content

Namespace URL http://webservices.examples.itko.com

Element addUser

Selected Parts

{Element}{http://webservices.examples.itko.com}addUser
{Content}{http://webservices.examples.itko.com}addUser

Type: Select one of the following:

- **Element**: Select if you want to encrypt the element and the content.
- **Content**: Select if you want to encrypt just the content.
- **Namespace URL**: Enter the value for the element.
- **Element**: Enter the name of the element.

You can repeat this for as many elements as you wish by clicking the Add button.

You will need to manually add the Body element if you wish it to be included.

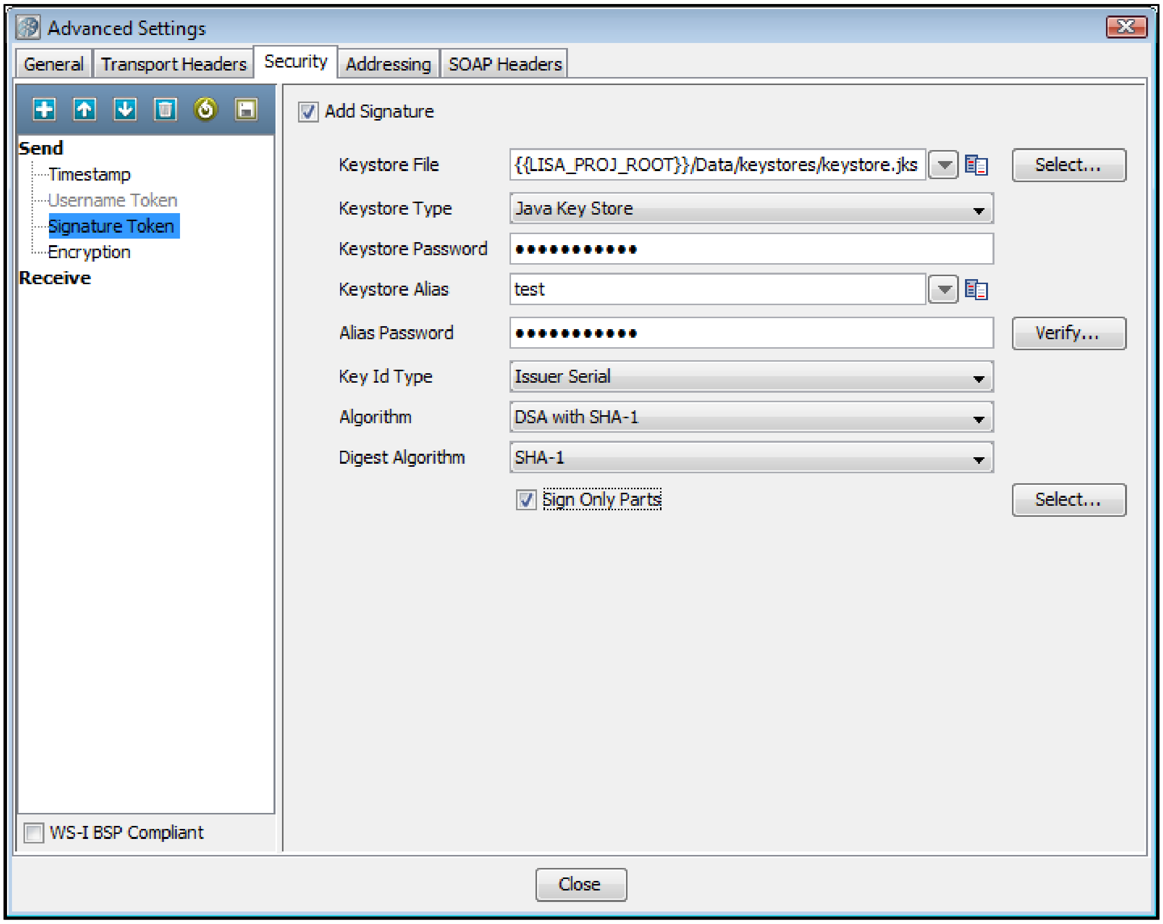
If you want to include the Binary Security Token as a part, use the Element name "Token."

Decryption

The parameters required to configure decryption are a subset of those required for encryption.

XML Signature Token/Signature Verification

Signature token



☑ Add Signature

Keystore File: {{LISA_PROJ_ROOT}}/Data/keystores/keystore.jks

Keystore Type: Java Key Store

Keystore Password: ●●●●●●●●●●●

Keystore Alias: test

Alias Password: ●●●●●●●●●●●

Key Id Type: Issuer Serial

Algorithm: DSA with SHA-1

Digest Algorithm: SHA-1

☑ Sign Only Parts

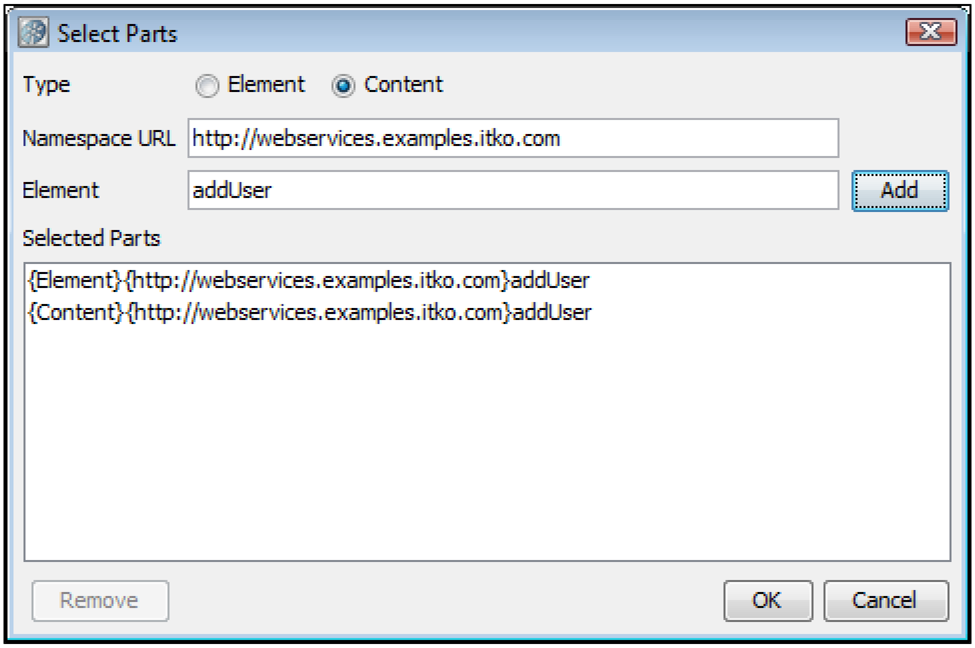
□ WS-I BSP Compliant

Select the Add Signature check box.

- **Keystore File**: Enter the location of the keystore file.
- **Keystore Type**: Select **Java Key Store (jks)** or **Personal Information Exchange (PKCS #12)**.
- **Keystore Password**: Enter the password for the keystore.
- **Keystore alias**: Enter an alias for a private key.
- **Alias Password**: Leave blank, or same as Keystore Password for PKCS #12 files.
- **Key ID type**: Select the appropriate key ID type from the pull-down menu.
- **Algorithm**: Select **DSA with SHA-1**.
- **Digest Algorithm**: Defaults to **SHA-1**.

The default behavior is to sign only the SOAP body contents.

Sign Only Parts: If you want to specify different parts to sign click the Select button to identify the parts to be signed.



Type ○ Element ● Content

Namespace URL http://webservices.examples.itko.com

Element addUser

Selected Parts

{Element}{http://webservices.examples.itko.com}addUser
{Content}{http://webservices.examples.itko.com}addUser

- **Type:** Select one of the following:
- **Element**: Select if you want to sign the element and the content.
- **Content**: Select if you want to sign just the content.
- **Namespace URL**: Enter the value for the element.
- **Element**: Enter the name of the element.

You can repeat this for as many elements as you want by clicking the Add button.

You will need to manually add the Body element if you want it to be included.

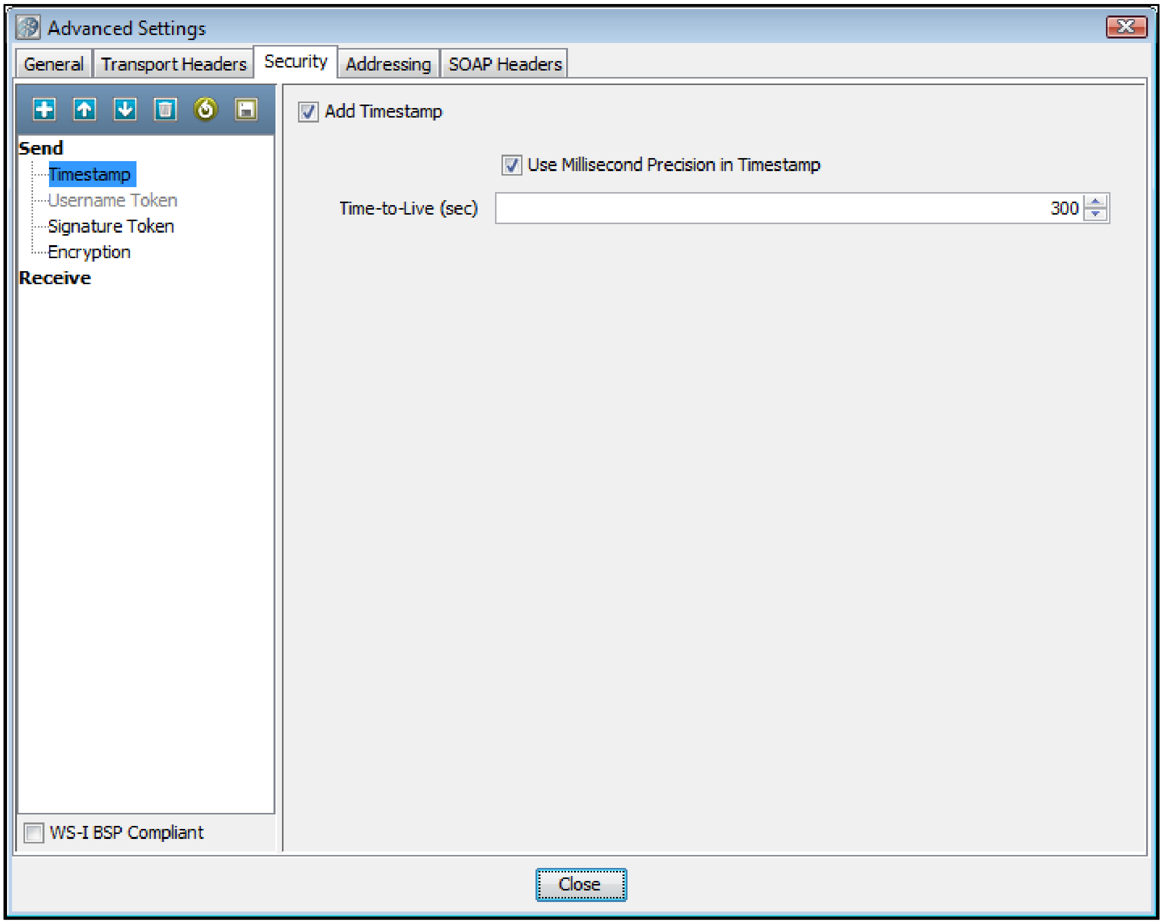
If you want to include the Binary Security Token as a part, use the Element name "Token."

Signature Verification

The parameters required to configure signature verification are a subset of those required for signing.

Timestamp/Timestamp Receipt

Timestamp



☑ Add Timestamp

☑ Use Millisecond Precision in Timestamp

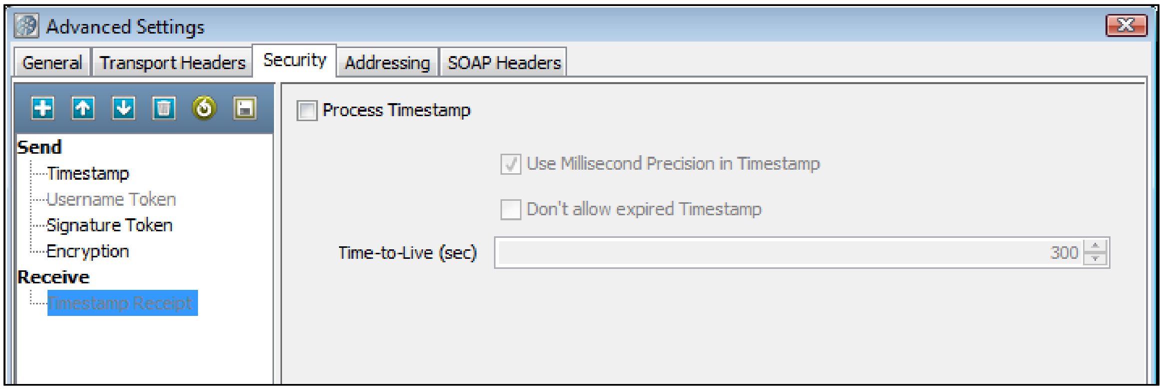
Time-to-Live (sec) 300

□ WS-I BSP Compliant

Select the Add Timestamp check box.

- **Time-To-Live (sec)**: Enter the lifetime of the message in seconds. Enter 0 to not include an Expires element.
  Some Web services, particularly .NET 1.x/2.0 with WSE 2.0, are not compliant with standard timestamp date formatting, and do not allow milliseconds. For these web services, clear the Use Millisecond Precision in Timestamp check box.

Timestamp Receipt



□ Process Timestamp

☑ Use Millisecond Precision in Timestamp

□ Don't allow expired Timestamp

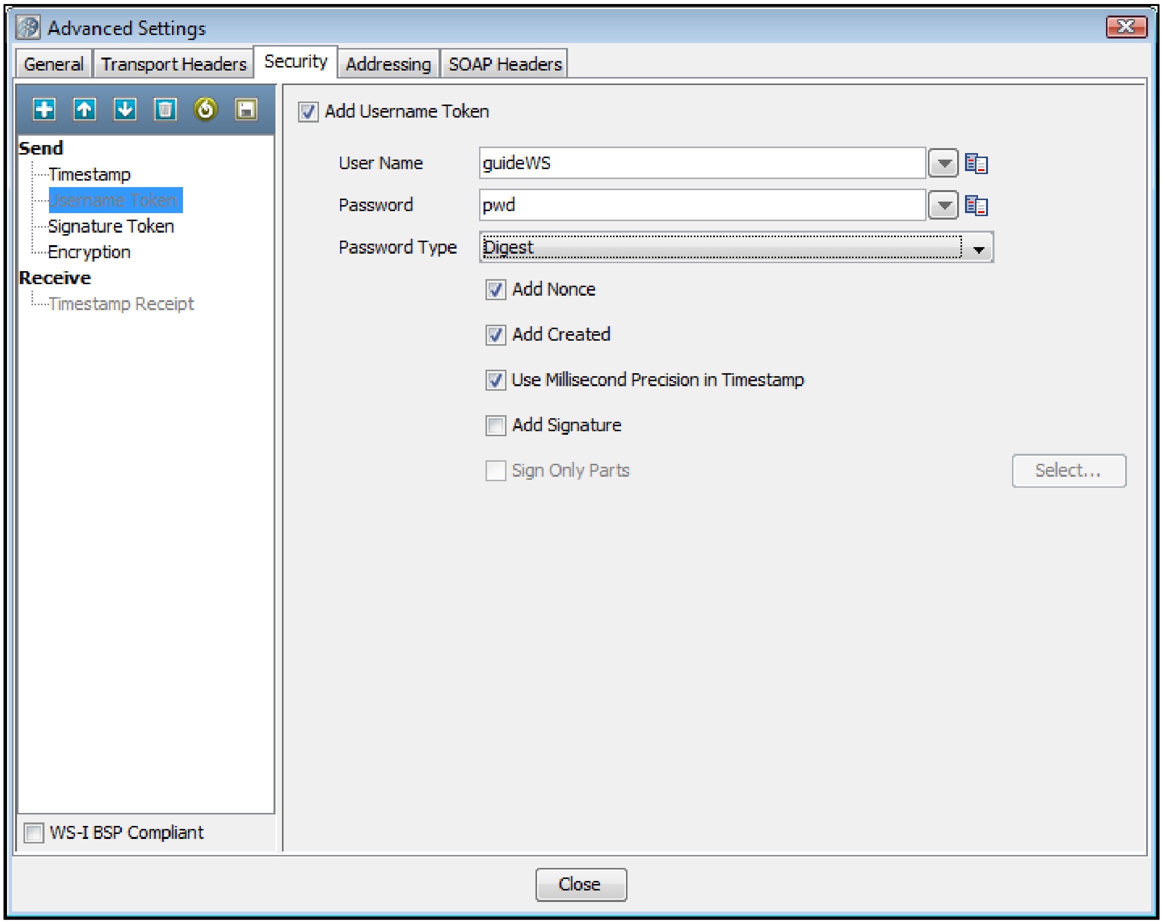
Time-to-Live (sec) 300

The parameters required for Timestamp Receipt are a superset of those required for Timestamp. The additional parameter:

- **Don't allow expired**: Timestamp can be selected if you do not want to allow expired timestamps.

Username Token/Username Token Verifier

Username Token



☑ Add Username Token

User Name: guideWS

Password: pwd

Password Type: Digest

☑ Add Nonce

☑ Add Created

☑ Use Millisecond Precision in Timestamp

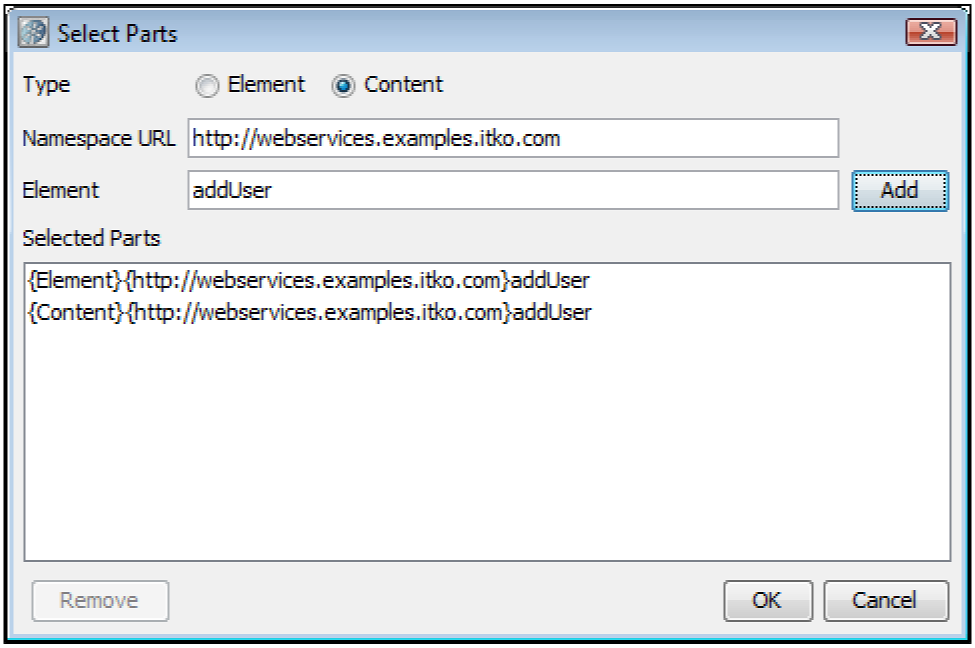
☐ Add Signature

☐ Sign Only Parts

☐ WS-I BSP Compliant

Select the Add Username Token check box.

- **User Name**: Enter the appropriate user name.
- **Password**: Enter the appropriate password.
- **Password Type**: Select the password type from the drop-down menu (Text, Digest, None). None is typically used with the Add Signature option.
- **Add Nonce**: Click if a Nonce is required: used to protect against replay attacks.
- **Add Created**: Click if a timestamp is required.
- **Use Millisecond Precision in Timestamp**: Select the check box to use millisecond precision. Some Web services, particularly .NET 1.x/2.0 with WSE 2.0, are not compliant with standard timestamp date formatting, and do not allow the use of milliseconds.
- **Add Signature**: Select to add a signature built using a combination of the user name and password as the key.
- **Sign Only Parts**: Select if you want to specify different parts to sign and click the Select button to identify the parts to be signed.



Type ○ Element ● Content

Namespace URL: http://webservices.examples.itko.com

Element: addUser

Selected Parts

{Element}{http://webservices.examples.itko.com}addUser
{Content}{http://webservices.examples.itko.com}addUser

Type: select one of the following:

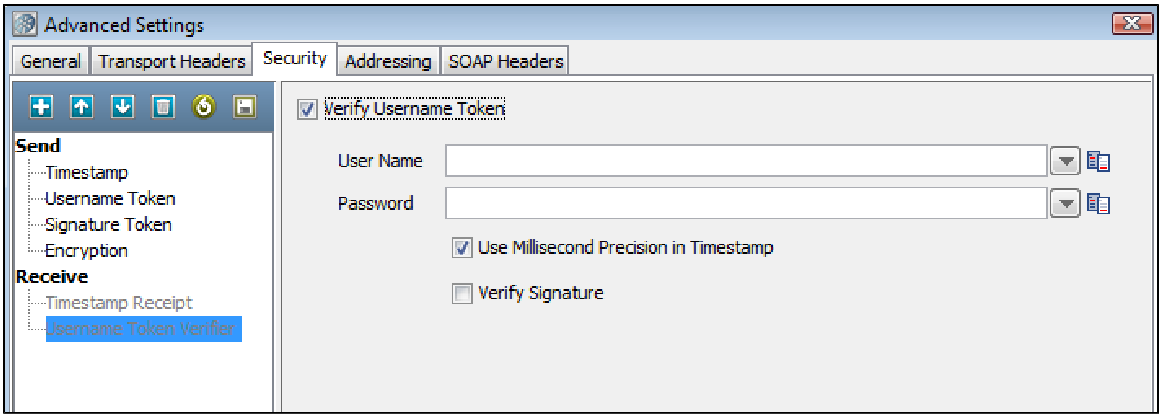
- **Element**: Select if you want to encrypt the element and the content.
- **Content**: Select if you want to encrypt just the content.
- **Namespace URL**: Enter the value for the element.
- **Element**: Enter the name of the element.

You can repeat this for as many elements as you wish by clicking the Add button.

You will need to manually add the Body element if you wish it to be included.

If you want to include the Binary Security Token as a part, use the Element name "Token."

UserName Token Verifier



☑ Verify Username Token

User Name:

Password:

☑ Use Millisecond Precision in Timestamp

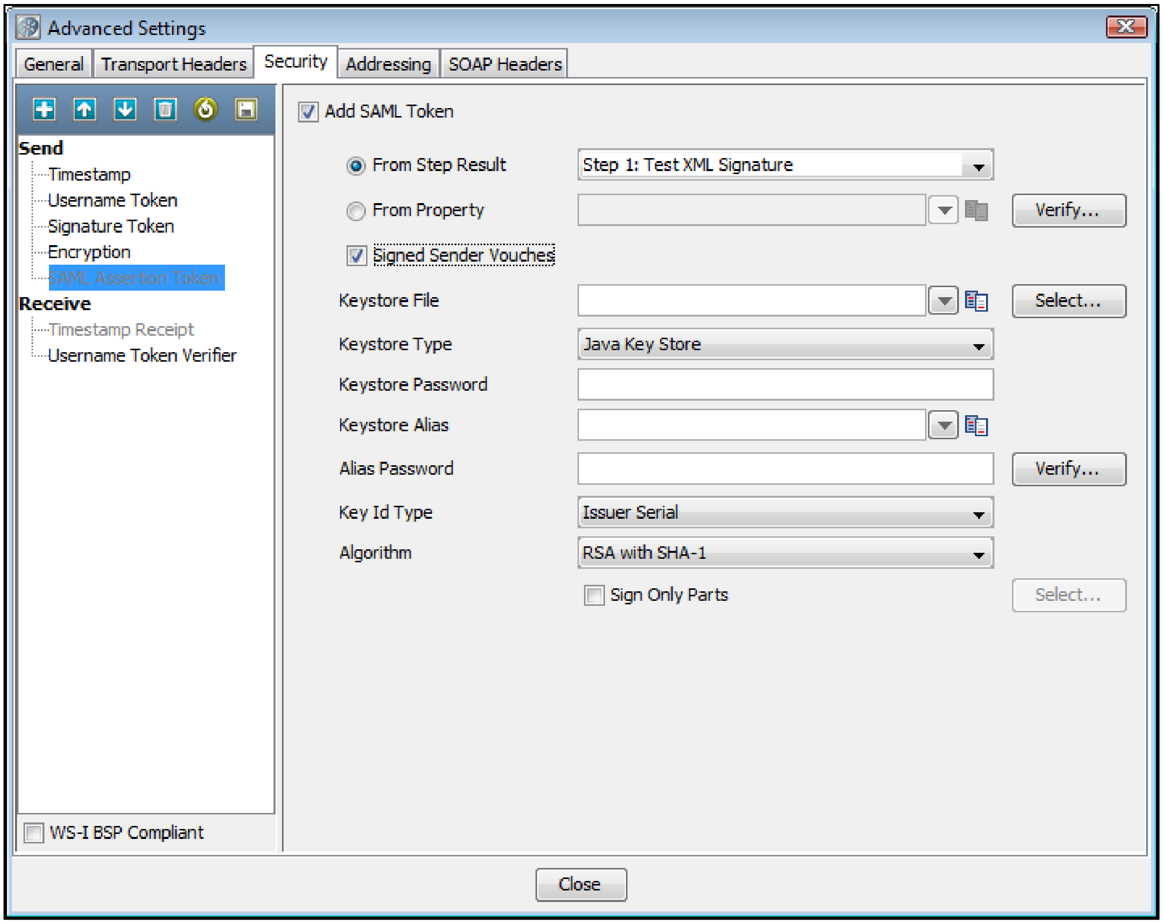
□ Verify Signature

Click the **Verify Username Token** box.

- **User Name**: Enter the verification user name.
- **Password**: Enter the verification password.
- **Use Millisecond Precision in Timestamp**: Select the check box to use millisecond precision. Some Web services, particularly .NET 1.x/2.0 with WSE 2.0, are not compliant with standard timestamp date formatting, and do not allow the use of milliseconds.
- **Verify Signature**: Select if Signature verification is required.

SAML Token/SAML Verifier

SAML Token



☑ Add SAML Token

● From Step Result: Step 1: Test XML Signature

○ From Property

☑ Signed Sender Vouches

Keystore File:

Keystore Type: Java Key Store

Keystore Password:

Keystore Alias:

Alias Password:

Key Id Type: Issuer Serial

Algorithm: RSA with SHA-1

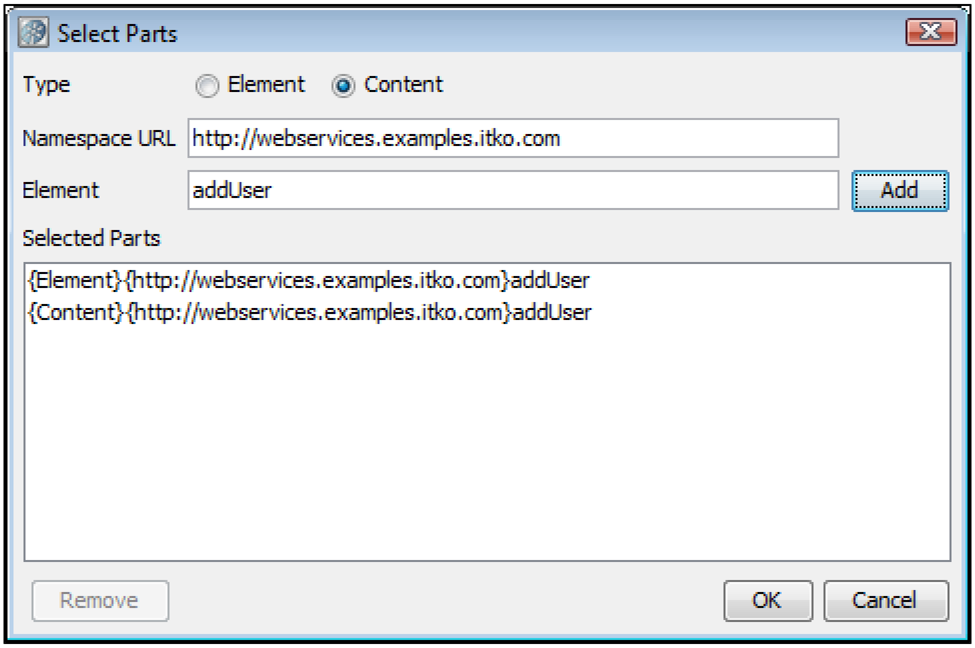
□ Sign Only Parts

□ WS-I BSP Compliant

- Select the Add SAML Token check box.
- Select the From Step Results drop-down: Select the step whose result is an XML SAML Assertion (like a SAML Query Step or a Parse Text Step with XML manually entered), or
- Select the From Property drop-down: Enter the LISA property that contains the XML SAML Assertion.
- You can optionally select the Verify button to have LISA parse the SAML Assertion XML and build the SAML Assertion object as it would when sending the SOAP request. This is useful to confirm that the SAML Assertion that may have been created manually is a valid SAML Assertion. It will also attempt to verify any signatures associated with the assertion, but it is likely that LISA will not be able to verify the assertion without configuring a public certificate to verify with.
- **Signed Sender Vouches**: If the assertion needs to be signed by the sender (the sender vouches for its authenticity as opposed to the bearer/creator of the SAML Assertion). When selected you will need to fill in the following information:
- **Keystore File**: Enter the location of the keystore file.
- **Keystore Type**: Select Java Key Store (jks) or Personal Information Exchange (PKCS #12).
- **Keystore Password**: Enter the password for the keystore.
- **Keystore alias**: Enter an alias for a private key.
- **Alias Password**: Leave blank or the same as Keystore Password for PKCS #12 files.
- **Key ID type**: Select the appropriate key ID type from the pull-down menu.
- **Algorithm**: Select **DSA with SHA-1**.

The default behavior is to sign only the SOAP Body contents.

- **Sign Only Parts**: If you want to specify different parts to sign, click the Select button to identify the parts to be signed.



Select Parts

Type ○ Element ● Content

Namespace URL: http://webservices.examples.itko.com

Element: addUser

Selected Parts

{Element}{http://webservices.examples.itko.com}addUser
{Content}{http://webservices.examples.itko.com}addUser

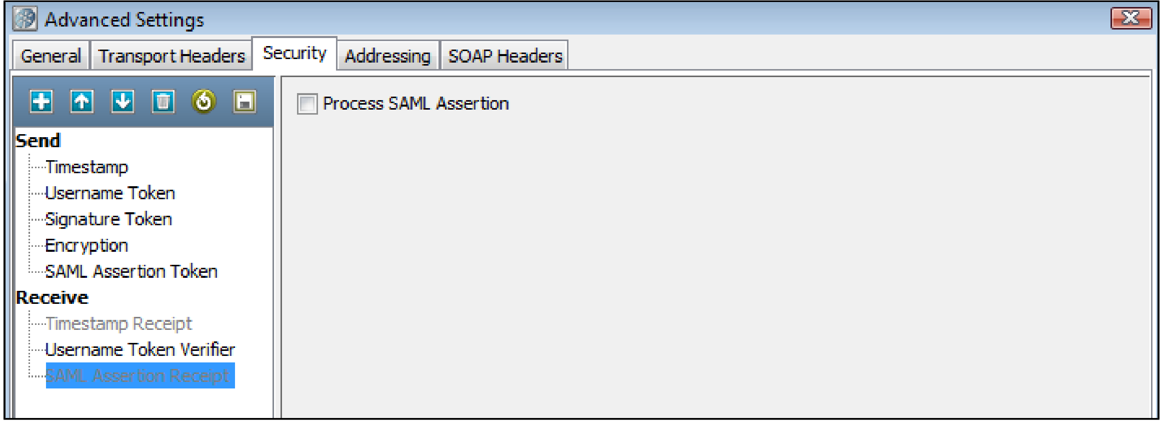
- **Type**: Select one of the following:
  - **Element**: Select if you want to sign the element and the content.
  - **Content**: Select if you want to sign just the content.
  - **Namespace URL**: Enter the value for the element.
  - **Element**: Enter the name of the element.

You can repeat this for as many elements as you wish by clicking the Add button. You will need to manually add the Body element if you wish it to be included.

If you want to include the Binary Security Token as a part, use the Element name "Token."

SAML Verifier

Select the Process SAML Assertion check box if you want LISA to check for a SAML Assertion Receipt header in the response. If you select this option and there is no SAML Assertion Receipt header then an exception occurs.



Advanced Settings

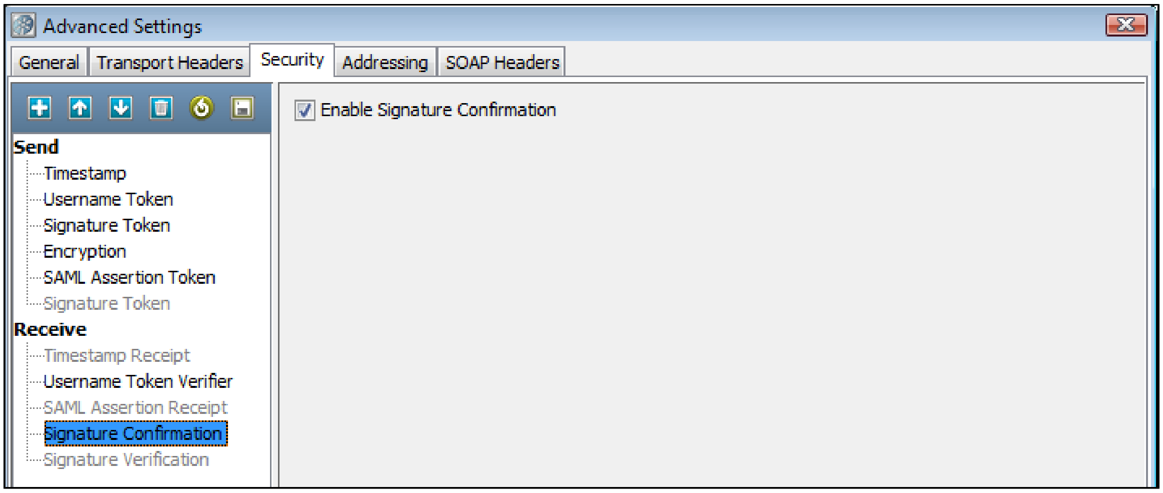
Send
- Timestamp
- Username Token
- Signature Token
- Encryption
- SAML Assertion Token

Receive
- Timestamp Receipt
- Username Token Verifier
- SAML Assertion Receipt

□ Process SAML Assertion

Signature Confirmation

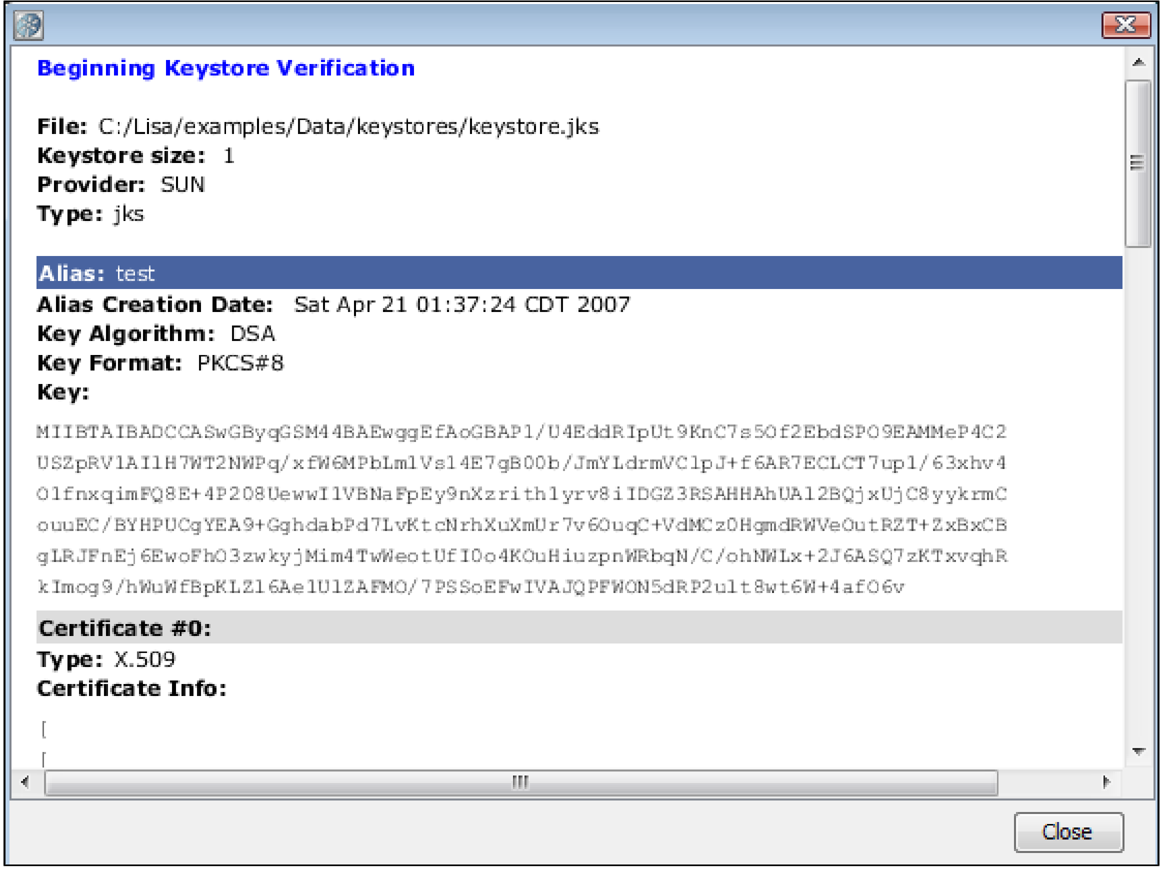
Select the Signature confirmation check box if you want LISA to check for a signature confirmation header in the response. If you select this option and there is no confirmation header then an exception occurs.



☑ Enable Signature Confirmation

Using the Keystore Verifier

You can verify your keystore settings to make sure you are using the correct format, password, alias and alias password. There is a Verify button on the editors for SSL, Signature, Encryption/Decryption and SAML settings. Clicking Verify produces a verification report.



Beginning Keystore Verification

File: C:/Lisa/examples/Data/keystores/keystore.jks
Keystore size: 1
Provider: SUN
Type: jks

Alias: test
Alias Creation Date: Sat Apr 21 01:37:24 CDT 2007
Key Algorithm: DSA
Key Format: PKCS#8
Key:

```
MIIBTAIBADCCASwGByqGSM44BAEwggEfAoGBAP1/U4EddRIpUt9KnC7s5Of2EbdSPO9EAMMeP4C2
USZpRV1AIlH7WT2NWPq/xfW6MPbLm1Vsl4E7gB00b/JmYLdrmVClpJ+f6AR7ECLCT7up1/63xhv4
O1fnxqimFQ8E+4P208UewwI1VBNaFpEy9nXzrith1yrv8iIDGZ3RSAHHAhUAl2BQjxUjC8yykrmC
ouuEC/BYHPUCgYEA9+GghdabPd7LvKtcNrhXuXmUr7v6OuqC+VdMCz0HgmdRWVeOutRZT+ZxBxCB
gLRJFnEj6EwoFhO3zwkyjMim4TwWeotUfI0o4KOuHiuzpnWRbqN/C/ohNWLx+2J6ASQ7zKTxvqhR
kImog9/hWuWfBpKLZl6Ae1UlZAFMO/7PSSoEFwIVAJQPFWON5dRP2ult8wt6W+4afO6v
```

Certificate #0:
Type: X.509
Certificate Info:

[
[

SSL verification validates the Keystore password only and confirms that at least one of the keys in the keystore can be loaded using the keystore password.

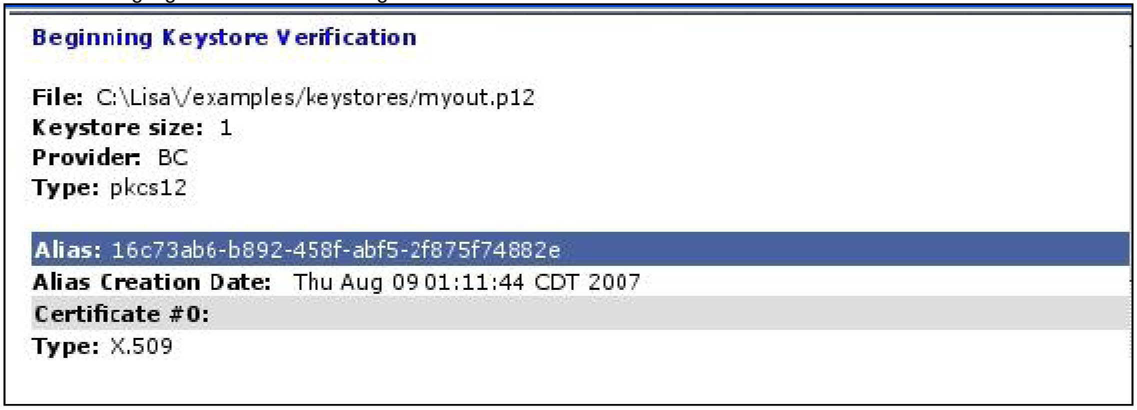
WS-Security verification validates the Keystore password, the alias and the alias password. Correct validation is indicated with a green entry. Any validation errors found will be shown in red. Warnings are shown in orange.

> ⚠ This verification only verifies the keystore parameters. There could still be issues with the web service, such as a mismatch in certificate sets or incorrect choice of algorithm. These issues will need to be validated independently.

Alias Search

If you do not know the expected alias name for a WS-Security setting, you can use the keystore verifier to list all the aliases in the keystore. Leave the **Keystore Alias** and **Alias Password** boxes empty and click the Verify button:

Aliases are highlighted with a blue background.



Beginning Keystore Verification

File: C:\Lisa\/examples/keystores/myout.p12
Keystore size: 1
Provider: BC
Type: pkcs12

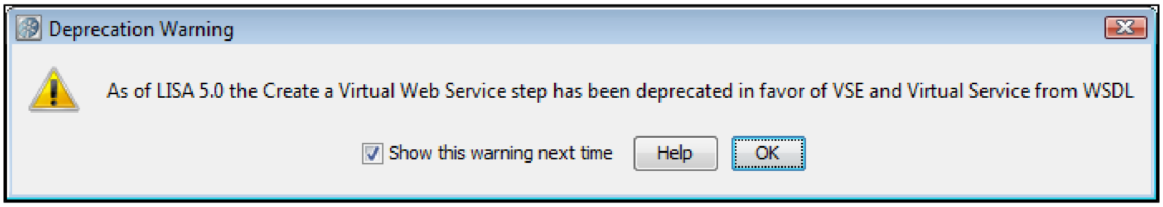
Alias: 16c73ab6-b892-458f-abf5-2f875f74882e
Alias Creation Date: Thu Aug 09 01:11:44 CDT 2007
Certificate #0:
Type: X.509

Verification will fail because the Keystore Alias and Alias Password boxes were left blank.

Start or Stop Web Server (Legacy)

In LISA 5.0 the Virtual Web Service support was deprecated in favor of using a WSDL to generate a true Virtual Service for LISA Virtual Service Environment.

This warning appears when you open the Virtual Web Service step or the Start Web Server step. Its purpose is to make you aware of the new VSE functionality in LISA that replaces these older virtualization steps. In the future these steps will be removed completely as virtualization will only be possible through LISA VSE.



Deprecation Warning

As of LISA 5.0 the Create a Virtual Web Service step has been deprecated in favor of VSE and Virtual Service from WSDL

☑ Show this warning next time

Clearing the check box will prevent the warning from showing up in the future for that step type. However, both the Start Web Server and Virtualize Web Service steps will continue to log a warning to the LISA log each time they are run.

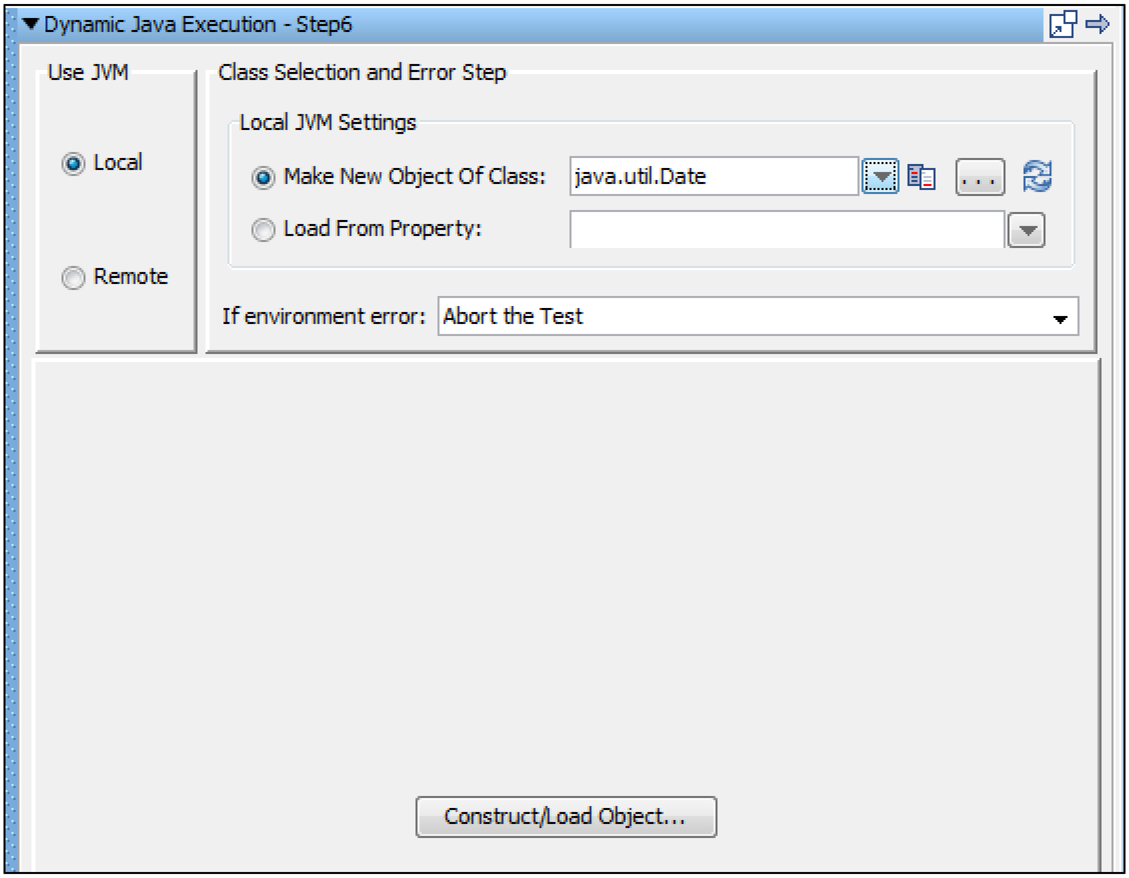
Java_J2EE Steps

The following steps are available in this chapter.

- Dynamic Java Execution
- RMI Server Execution
- Enterprise JavaBean Execution

Dynamic Java Execution

The Dynamic Java Execution step lets you instantiate and manipulate a Java object. All Java classes on the LISA classpath are available, including the classes in the JRE's classpath. Any user classes can be placed on the classpath by copying them into the hotdeploy directory. The class under test is loaded into the LISA Complex Object Editor where it can be manipulated without having to write any Java code.



Dynamic Java Execution - Step6

Use JVM

● Local

○ Remote

Class Selection and Error Step

Local JVM Settings

● Make New Object Of Class: java.util.Date

○ Load From Property:

If environment error: Abort the Test

In our example we are using a Java date instance of class **java.util.Date**.

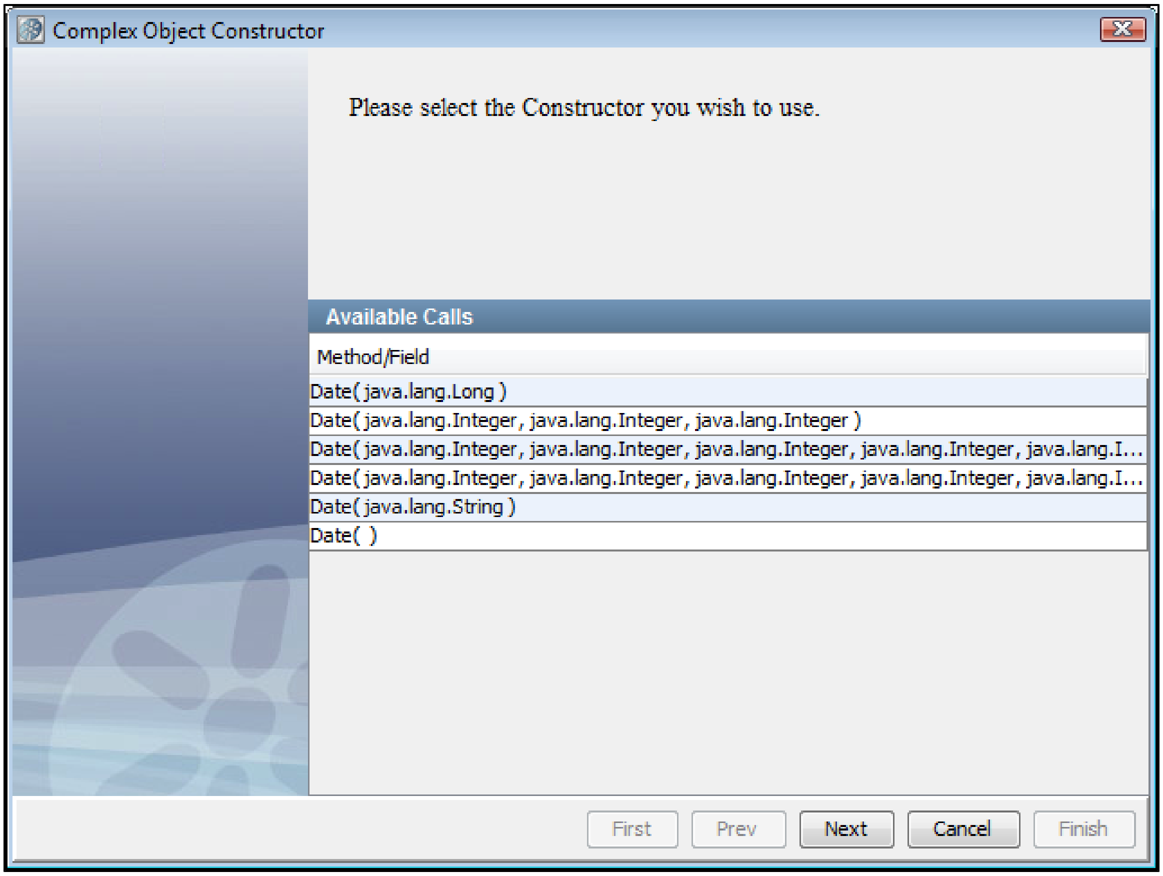
1. Enter the following parameters in the Dynamic Java Execution editor:

- **Use JVM**: Select the Local button. It is possible to execute a Java object remotely, using In-Container Testing (ICT), by clicking the Remote option button, but this mode requires some extra setup before it can be used. This is discussed in the *Developer's Guide (SDK)*.
- **Local JVM Settings**: Select one of the following option buttons:
  - **Make New Object of Class**: Click the option button and enter, select or browse to the Java class you want to instantiate. This must be the fully qualified class name including the package of the Java class; for example, **com.example.MyClass**.
  - **Load from Property**: Click the option button and enter the name of the property that has the serialized object as its value.
- **If environment error**: Select the step to redirect to if an environment error occurs while trying to create an object.

> ⚠ If you require that the Java object is loaded by its own classloader, you must add the companion **Class Loader Sandbox Companion**.

2. Click the Construct/Load Object button.

3. The Complex Object Constructor window is now displayed listing the available constructors for your object. Select a constructor and click the Next button.

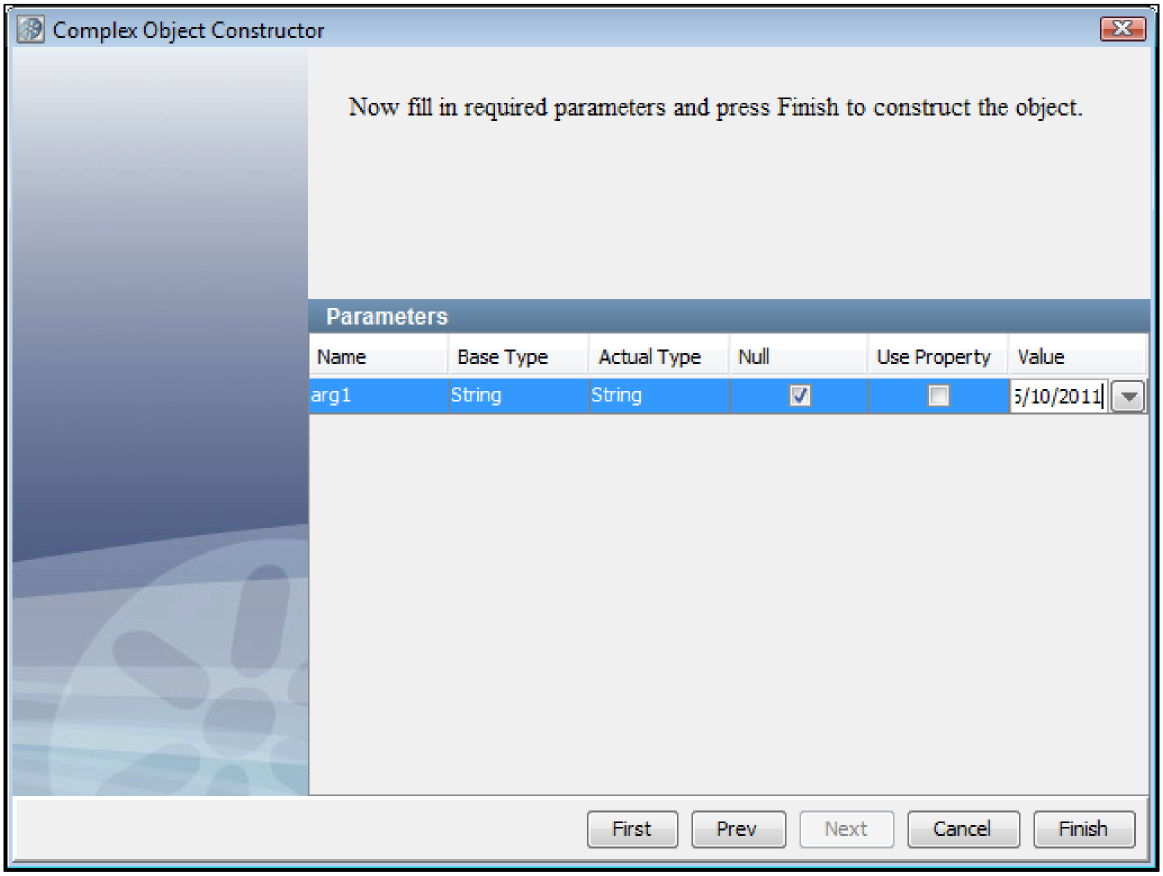


Please select the Constructor you wish to use.

Available Calls

Method/Field
Date(java.lang.Long)
Date(java.lang.Integer, java.lang.Integer, java.lang.Integer)
Date(java.lang.Integer, java.lang.Integer, java.lang.Integer, java.lang.Integer, java.lang.I...
Date(java.lang.Integer, java.lang.Integer, java.lang.Integer, java.lang.Integer, java.lang.I...
Date(java.lang.String)
Date( )

4. Enter any input parameters needed by the constructor.

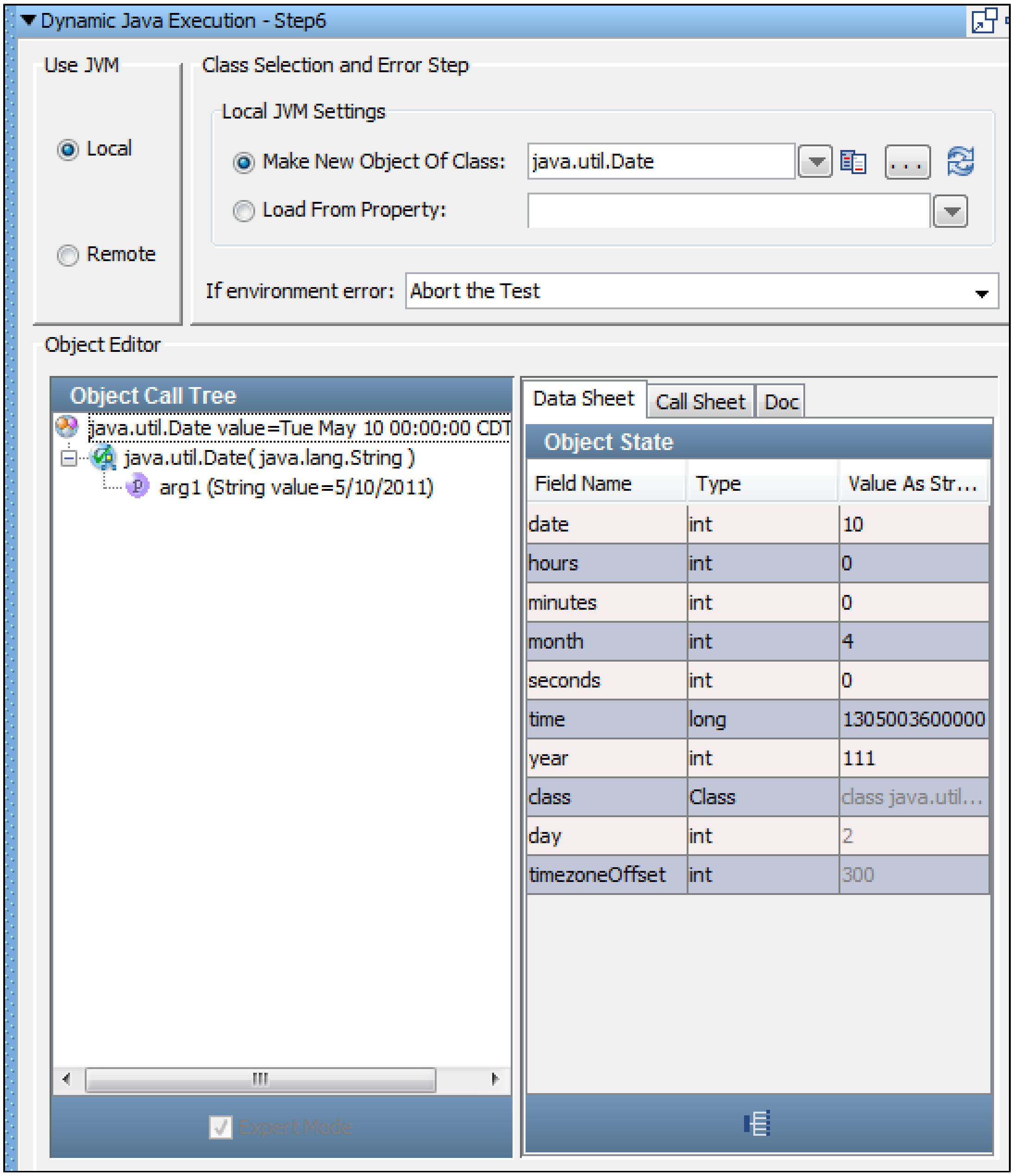


Now fill in required parameters and press Finish to construct the object.

Parameters

Name	Base Type	Actual Type	Null	Use Property	Value
arg1	String	String	☑	□	5/10/2011

5. In this example, enter a string representation of a day (5/10/2011). You can enter a value, a property or null. LISA constructs the object and loads it into the LISA Complex Object Editor.



Dynamic Java Execution - Step6

Use JVM

● Local

○ Remote

Class Selection and Error Step

Local JVM Settings

● Make New Object Of Class: java.util.Date

○ Load From Property:

If environment error: Abort the Test

Object Editor

Object Call Tree

java.util.Date value=Tue May 10 00:00:00 CDT

java.util.Date(java.lang.String)

arg1 (String value=5/10/2011)

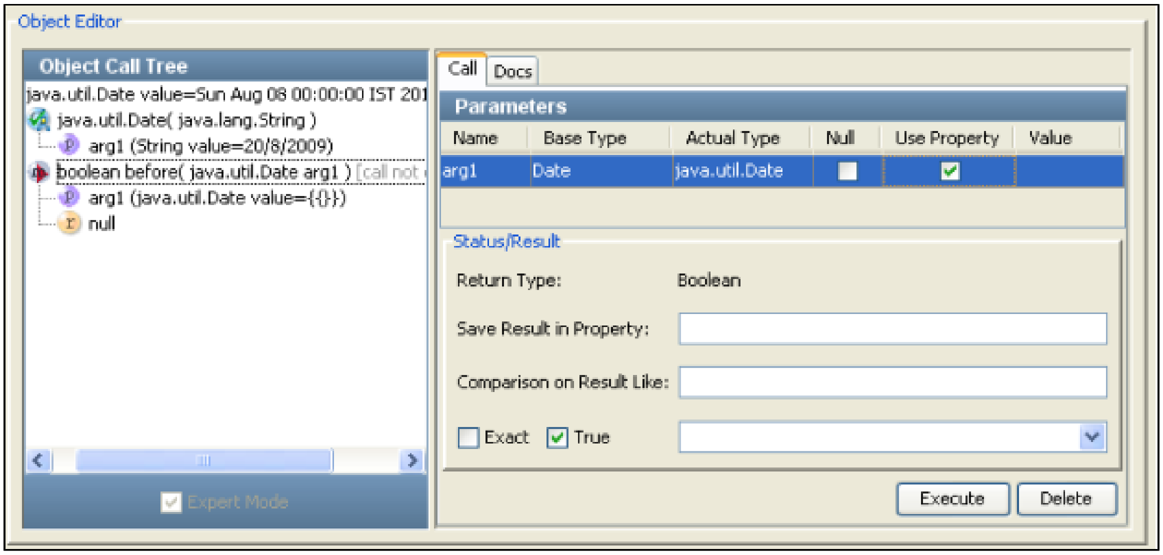
☑ Expert Mode

Object State

Field Name	Type	Value As Str...
date	int	10
hours	int	0
minutes	int	0
month	int	4
seconds	int	0
time	long	1305003600000
year	int	111
class	Class	class java.util...
day	int	2
timezoneOffset	int	300

6. You can now manipulate the object, and execute methods, using the LISA Complex Object Editor. For more information about how to use the Complex Object Editor, see Complex Object Editor (COE).

7. You can also add filters and add assertions, either by using the inline filter/assertion form (which is part of the Complex Object Editor) or manually by selecting **filter** under your test step in the test case tree. For example, here is a screen just before we execute the before method on the Date class.



java.util.Date value=Sun Aug 08 00:00:00 IST 201
java.util.Date(java.lang.String)
arg1 (String value=20/8/2009)
boolean before(java.util.Date arg1) [call not
arg1 (java.util.Date value={{}})
null

Name	Base Type	Actual Type	Null	Use Property	Value
arg1	Date	java.util.Date	□	☑	

Return Type: Boolean

Save Result in Property:

Comparison on Result Like:

□ Exact ☑ True

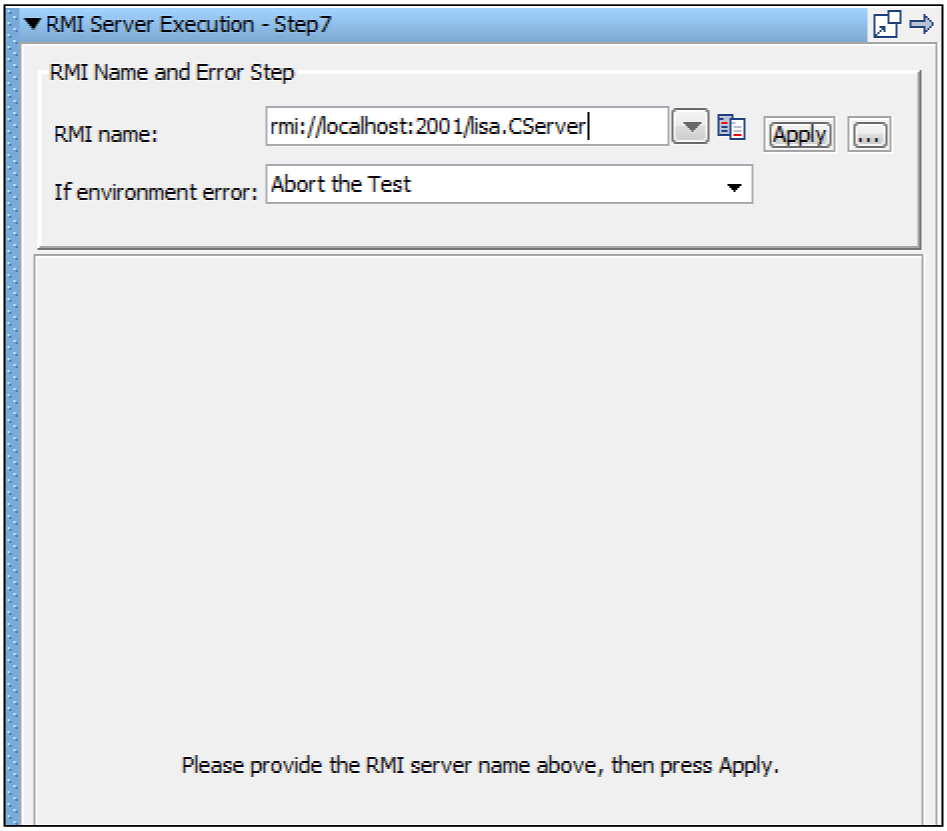
You can see the Status/Result section where you can add an inline filter, in the Save Results in Property text box, and an inline assertion, in the Comparison on Result Like text box.

RMI Server Execution

The RMI Server Execution step lets you acquire a reference to a remote Java object through RMI (Remote method Invocation), and make calls on the Java object.

Prerequisites: Knowledge of the LISA Complex Object Editor is assumed. You also need to copy the interface and stub classes for the remote object into the hot deploy directory. LISA requires these to contact and interact with the remote object. Get these classes from the remote object developer.

Parameter Requirement: You will need to know how to connect to the RMI Server, usually a host name and port, and you will need to know the RMI name of the object you want to invoke.



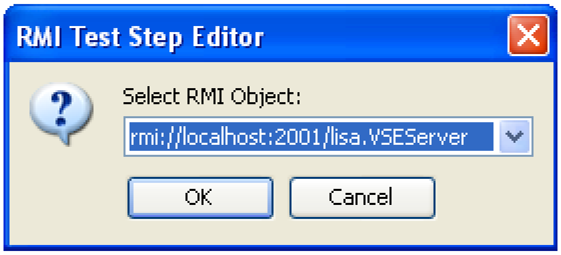
RMI name: rmi://localhost:2001/lisa.CServer

If environment error: Abort the Test

Please provide the RMI server name above, then press Apply.

The RMI Server step editor lets you enter the following parameters:

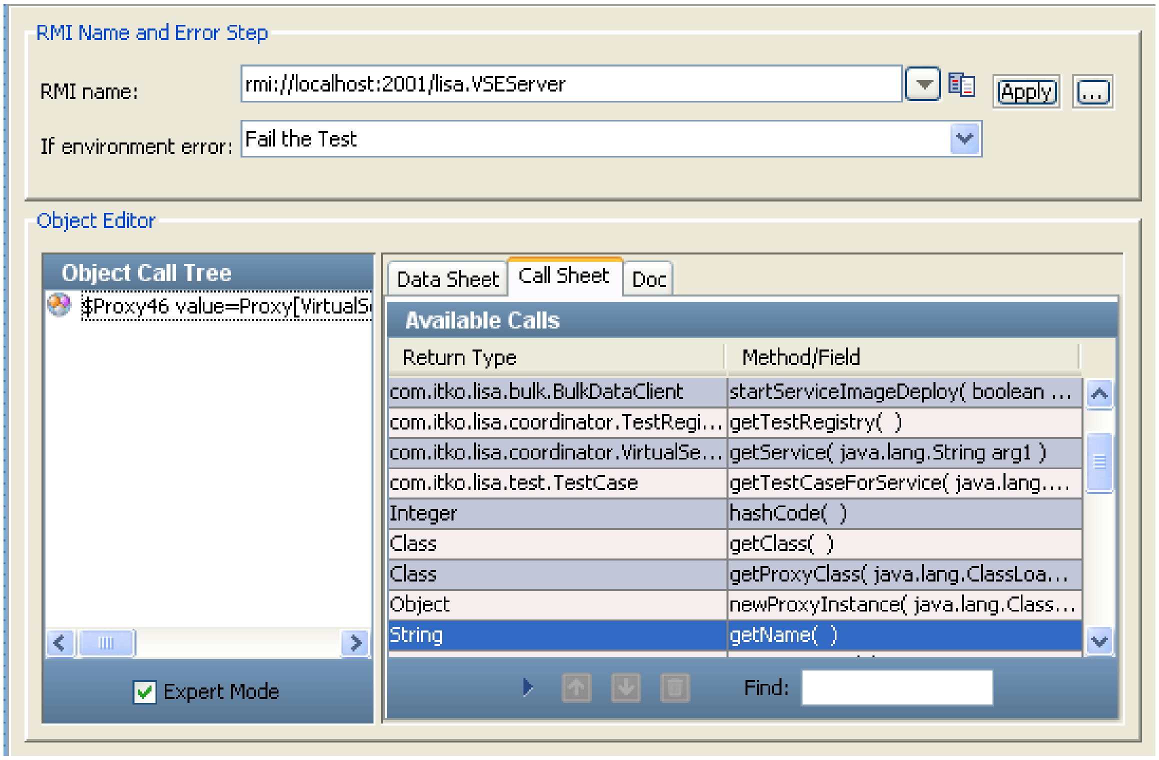
- **RMI Name**: Enter or select the full RMI name of the object (as shown previously), or enter the RMI Server name and click the Browse button to get a list of available objects.



Select RMI Object:

rmi://localhost:2001/lisa.VSEServer

LISA constructs the object and loads it into the LISA complex object editor.



RMI name: rmi://localhost:2001/lisa.VSEServer

If environment error: Fail the Test

$Proxy46 value=Proxy[VirtualS

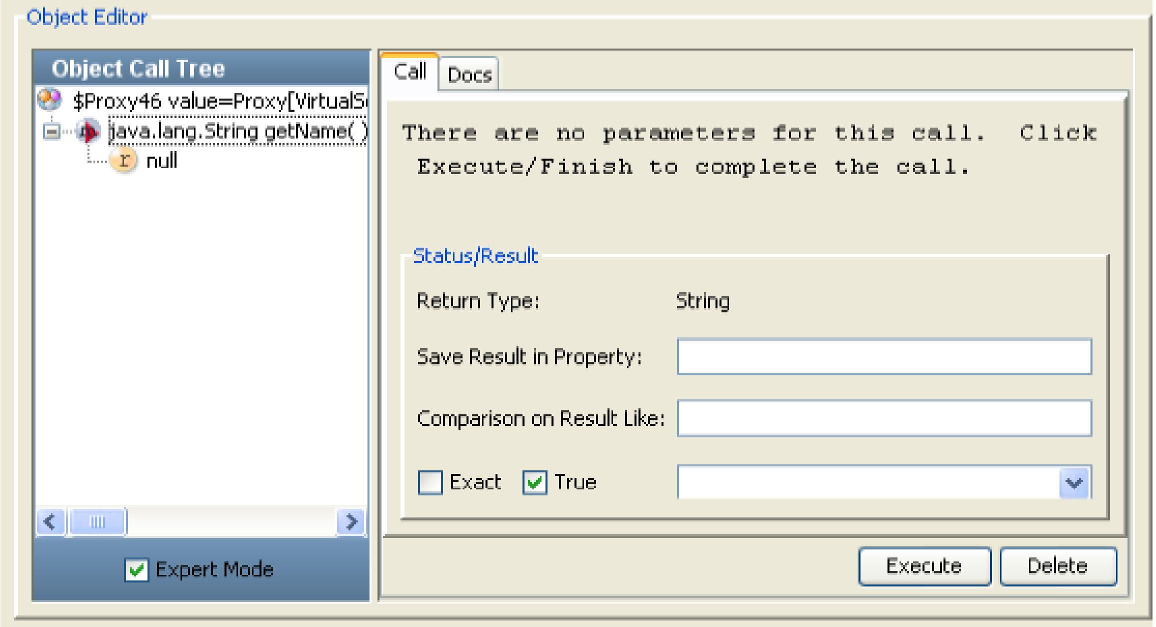
Return Type	Method/Field
com.itko.lisa.bulk.BulkDataClient	startServiceImageDeploy(boolean ...
com.itko.lisa.coordinator.TestRegi...	getTestRegistry( )
com.itko.lisa.coordinator.VirtualSe...	getService(java.lang.String arg1)
com.itko.lisa.test.TestCase	getTestCaseForService(java.lang....
Integer	hashCode( )
Class	getClass( )
Class	getProxyClass(java.lang.ClassLoa...
Object	newProxyInstance(java.lang.Class...
String	getName( )

☑ Expert Mode

You can now manipulate the object, and execute methods, using the LISA Complex Object Editor.

In the previous illustration we have selected the getName method. If we double-click it, it gets added in the Object Call Tree, and then it can be executed using the Execute button that appears in the dialog, with any method arguments that need to be passed to the method, and information

about how to process the result.
NEED UPDATED SCREEN CAPTURE



$Proxy46 value=Proxy[VirtualS
java.lang.String getName( )
r null

There are no parameters for this call. Click Execute/Finish to complete the call.

Status/Result

Return Type: String

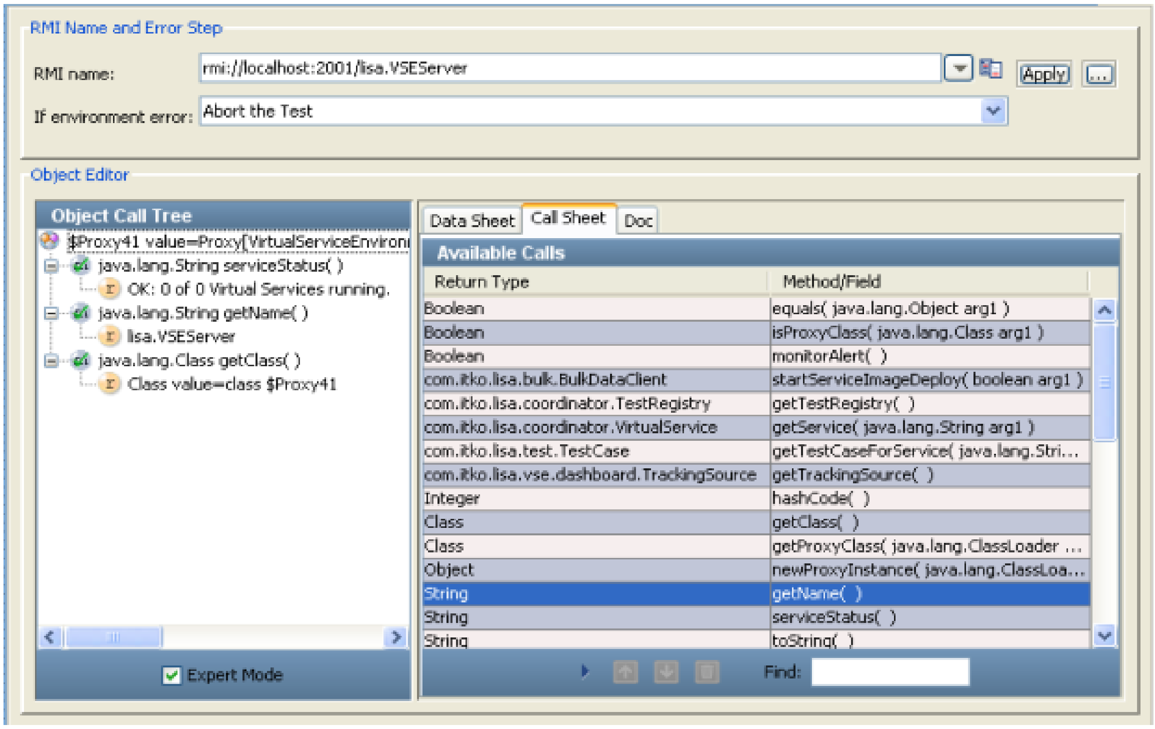
Save Result in Property:

Comparison on Result Like:

□ Exact ☑ True

☑ Expert Mode

The previous illustration shows null as the return value because the method is not yet executed. After the **Execute** button is clicked, it gets executed and the correct return type is shown. The following illustration shows how the Object Call Tree tracks the results of execution of several methods.



RMI name: rmi://localhost:2001/lisa.VSEServer

If environment error: Abort the Test

$Proxy41 value=Proxy[VirtualServiceEnviron
java.lang.String serviceStatus( )
r OK: 0 of 0 Virtual Services running.
java.lang.String getName( )
r lisa.VSEServer
java.lang.Class getClass( )
r Class value=class $Proxy41

Return Type	Method/Field
Boolean	equals(java.lang.Object arg1)
Boolean	isProxyClass(java.lang.Class arg1)
Boolean	monitorAlert( )
com.itko.lisa.bulk.BulkDataClient	startServiceImageDeploy(boolean arg1)
com.itko.lisa.coordinator.TestRegistry	getTestRegistry( )
com.itko.lisa.coordinator.VirtualService	getService(java.lang.String arg1)
com.itko.lisa.test.TestCase	getTestCaseForService(java.lang.Stri...
com.itko.lisa.vse.dashboard.TrackingSource	getTrackingSource( )
Integer	hashCode( )
Class	getClass( )
Class	getProxyClass(java.lang.ClassLoader ...
Object	newProxyInstance(java.lang.ClassLoa...
String	getName( )
String	serviceStatus( )
String	toString( )

☑ Expert Mode

Find:

You can also add filters and add assertions, either by using the inline filter/assertion form or manually by selecting filter under your test step in the test case tree. You can see the **Status/Result** section where you can add an inline filter, in the Save Results in Property textbox, and an inline assertion, in the Comparison on Result Like text box.

⚠ If you have multiple network cards, using localhost in the RMI name can cause errors. You may need to use the IP address, or the host name that corresponds to the particular IP address.

Enterprise JavaBean Execution

The Enterprise JavaBean Execution step lets you acquire a reference to and make calls on an Enterprise JavaBean (EJB) running in a J2EE application server.

Testing an EJB is similar to testing a Java object. LISA will dynamically connect to the EJB using the Home EJB interface, and then from it create an instance of an EJB object. This process is a little different for EJBs, as they do not require a home interface. The EJB under test is loaded into the LISA Complex Object Editor where it can be manipulated without having to write any Java code.

Prerequisites: Knowledge of the LISA Complex Object Editor is assumed. The application client JAR and client EJB JAR must be in the LISA classpath. Both of these JAR files are usually copied into the hotDeploy directory. The LISA hotDeploy directory contains the jboss-all-client.jar file for the JBoss application server and the LISA examples JAR file so that you can run the EJB examples immediately.

Parameter Requirements:

- Server connection information (JNDI connection) and user ID and password (if required).
- The global JNDI lookup name of your EJB home interface.

This information should be provided by the EJB deployer.

Connecting to WebSphere with LISA using SIBC

IBM has an EJB and JMS client that you can download and use with the Sun JVM. It is available here.

Instructions:

1. Download this file; then run their installer.
2. Issue the command

```
java -jar sibc_install-o0902.06.jar jms_jndi_sun <output_directory>
```

3. Get the following files from your <output_directory>:

- lib\sibc.jms.jar
- lib\sibc.jndi.jar
- lib\sibc.orb.jar

4. Create a LISA_PRE_CLASSPATH environment variable, to reference the previous three JAR files; for example, LISA_PRE_CLASSPATH=C:\sibc.jms.jar;C:\sibc.jndi.jar;C:\sibc.orb.jar;

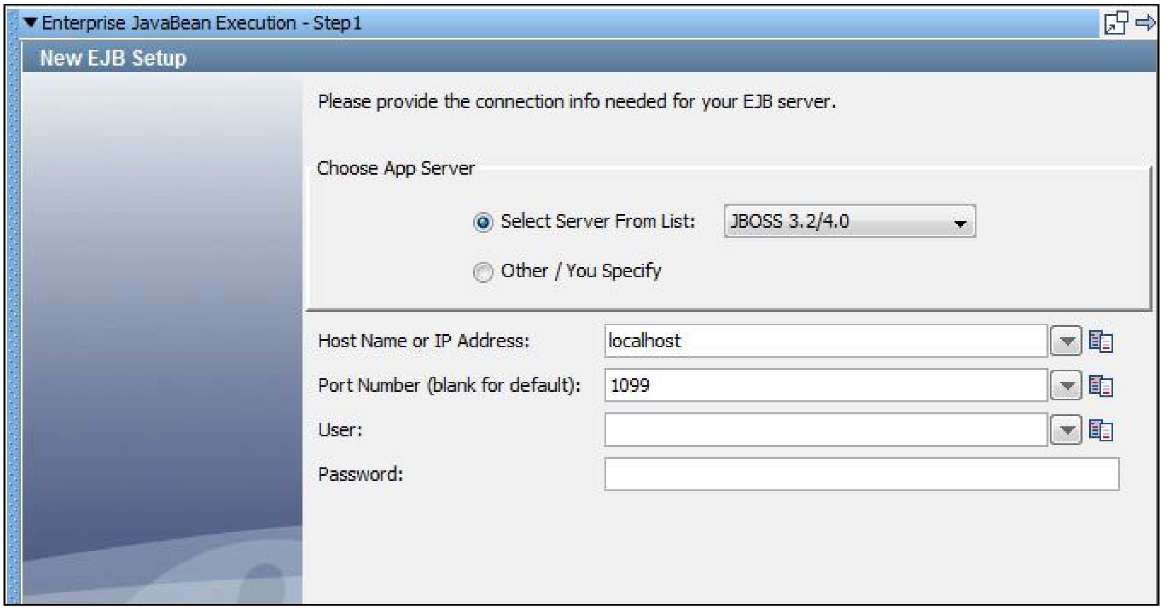
5. Edit local.properties and add the following line:

```
com.ibm.CORBA.ORBInit=com.ibm.ws.sib.client.ORB
```

6. On the JMS step, use the following settings:

- JNDI factory class: com.ibm.websphere.naming.WsnInitialContextFactory
- JNDU URL: iiop://SERVER:PORT

Example



Please provide the connection info needed for your EJB server.

Choose App Server

● Select Server From List: JBOSS 3.2/4.0

○ Other / You Specify

Host Name or IP Address: localhost

Port Number (blank for default): 1099

User:

Password:

This example uses the ITKO example server, a JBoss server. To use your local Demo Server, use localhost as the host name.

1. Enter the following parameters:

- **Choose App Server**: Select your application server from the list. If your application server is not on the list, click the Other/You Specify option button.

2. The lower section of the editor changes, depending on your selection. The previous illustration shows the configuration panel for JBoss.
3. For the JBoss panel, enter the following parameters:

- **Host Name or IP Address**: Enter hostname or IP address of your application server
- **Port Number**: Enter port number
- **User**: Enter if a User ID is required for the application server
- **Password**: Enter if a password is required for the application server

4. Click Next.

For the Other/You Specify window:

1. Enter the following parameters:

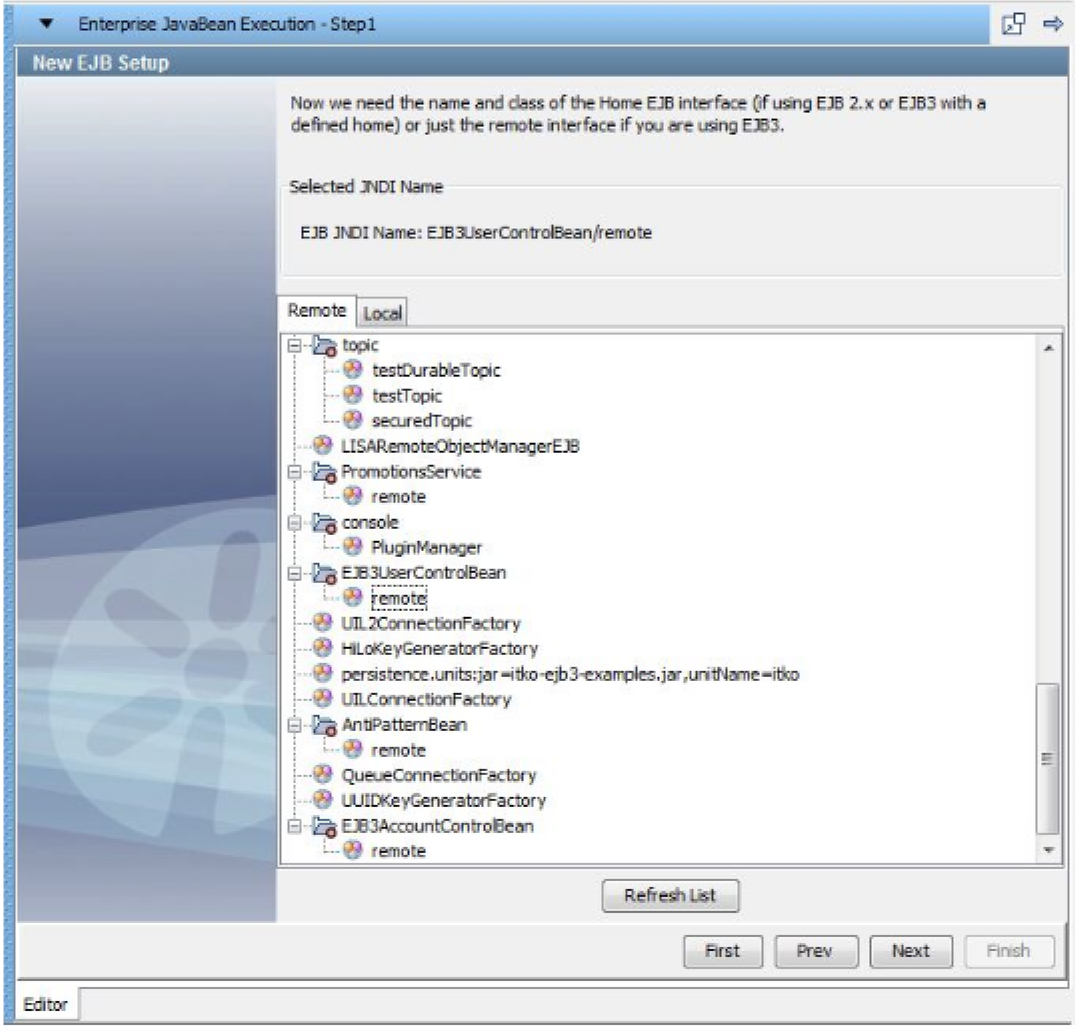
- **JNDI factory**: Enter or select the fully qualified JNDI factory class name for your application server.

- **URL**: Enter or select the JNDI server name.

- **User**: Enter if a user ID is required for the application server.

- **Password**: Enter if a password is required for the application server.

2. Click Next.
3. The New EJB Setup window displays a list of all the JNDI names registered with the application server.



Now we need the name and class of the Home EJB interface (if using EJB 2.x or EJB3 with a defined home) or just the remote interface if you are using EJB3.

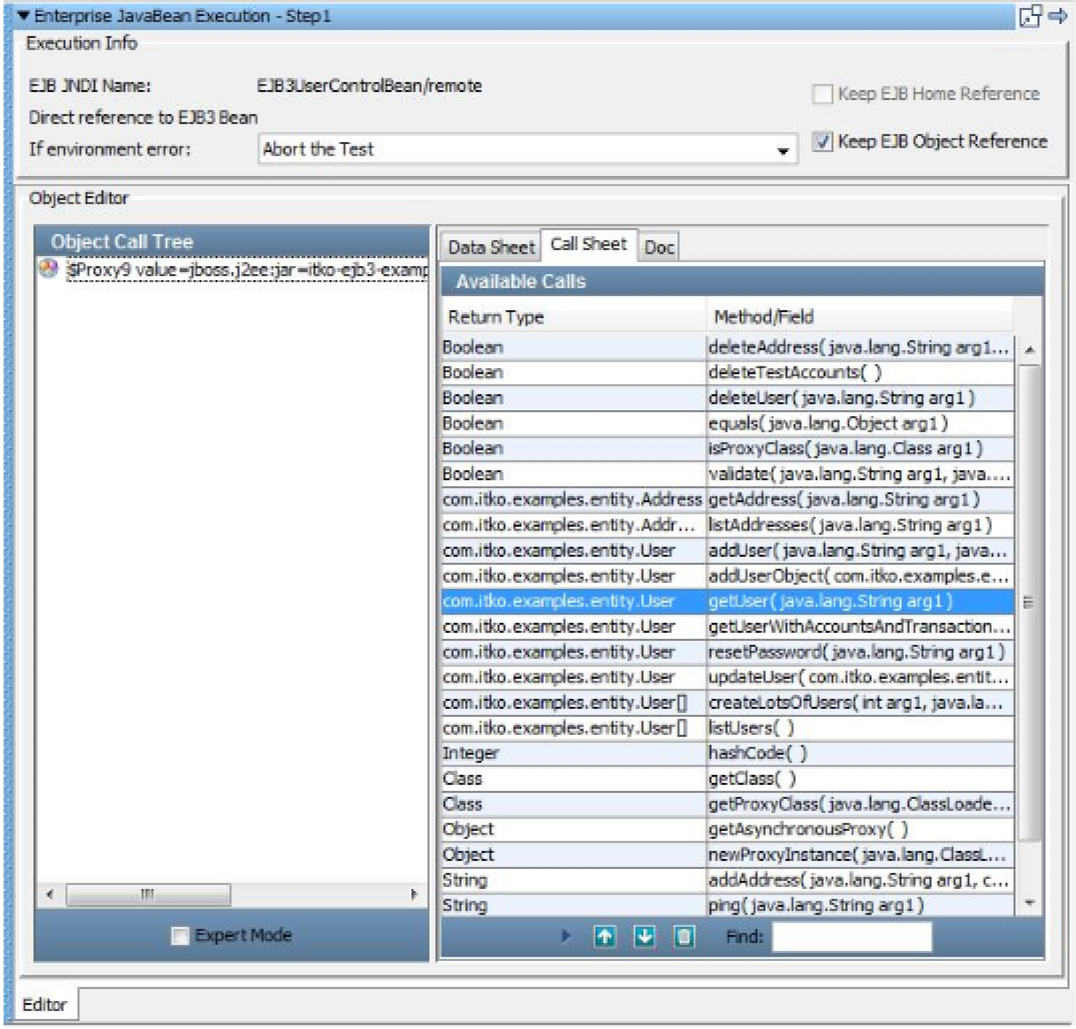
Selected JNDI Name

EJB JNDI Name: EJB3UserControlBean/remote

```
topic
    testDurableTopic
    testTopic
    securedTopic
LISARemoteObjectManagerEJB
PromotionsService
    remote
console
    PluginManager
EJB3UserControlBean
    remote
UIL2ConnectionFactory
HiLoKeyGeneratorFactory
persistence.units:jar=itko-ejb3-examples.jar,unitName=itko
UILConnectionFactory
AntiPatternBean
    remote
QueueConnectionFactory
UUIDKeyGeneratorFactory
EJB3AccountControlBean
    remote
```

4. Select the name of the appropriate EJB home interface. In this example, the JNDI name is: com.itko.examples.ejb.UserControlBean. The EJB3 specification enables stateful and stateless beans to bind to the JNDI tree directly and not require a home interface. If this is the case, the bean can be selected directly and LISA will not need to create a new instance.

5. Click Next.
6. The object is constructed and loaded into the LISA Complex Object Editor.



EJB JNDI Name: EJB3UserControlBean/remote

Direct reference to EJB3 Bean

If environment error: Abort the Test

☐ Keep EJB Home Reference

☑ Keep EJB Object Reference

$Proxy9 value=jboss.j2ee:jar=itko-ejb3-examp

Return Type	Method/Field
Boolean	deleteAddress(java.lang.String arg1...
Boolean	deleteTestAccounts( )
Boolean	deleteUser(java.lang.String arg1)
Boolean	equals(java.lang.Object arg1)
Boolean	isProxyClass(java.lang.Class arg1)
Boolean	validate(java.lang.String arg1, java....
com.itko.examples.entity.Address	getAddress(java.lang.String arg1)
com.itko.examples.entity.Addr...	listAddresses(java.lang.String arg1)
com.itko.examples.entity.User	addUser(java.lang.String arg1, java...
com.itko.examples.entity.User	addUserObject(com.itko.examples.e...
com.itko.examples.entity.User	getUser(java.lang.String arg1)
com.itko.examples.entity.User	getUserWithAccountsAndTransaction...
com.itko.examples.entity.User	resetPassword(java.lang.String arg1)
com.itko.examples.entity.User	updateUser(com.itko.examples.entit...
com.itko.examples.entity.User[]	createLotsOfUsers(int arg1, java.la...
com.itko.examples.entity.User[]	listUsers( )
Integer	hashCode( )
Class	getClass( )
Class	getProxyClass(java.lang.ClassLoade...
Object	getAsynchronousProxy( )
Object	newProxyInstance(java.lang.ClassL...
String	addAddress(java.lang.String arg1, c...
String	ping(java.lang.String arg1)

7. In the Execution Info area, the current EJB information is displayed. If you plan to reuse this EJB you can keep the references to the EJB object and the EJB Home by clicking the Keep EJB Home Reference, and Keep EJB Object Reference check boxes on the right. If the bean is an EJB3 bean without a Home interface, the Keep EJB Home Reference check box will be disabled. Set the If Exception to the step to redirect to if an exception occurs.

8. You can now manipulate the object, and execute methods, using the LISA Complex Object Editor. The usage is the same as in the RMI Server Execution step.

Other Transaction Steps

The following steps are available in this chapter.

- SQL Database Execution (JDBC)
- CORBA Execution

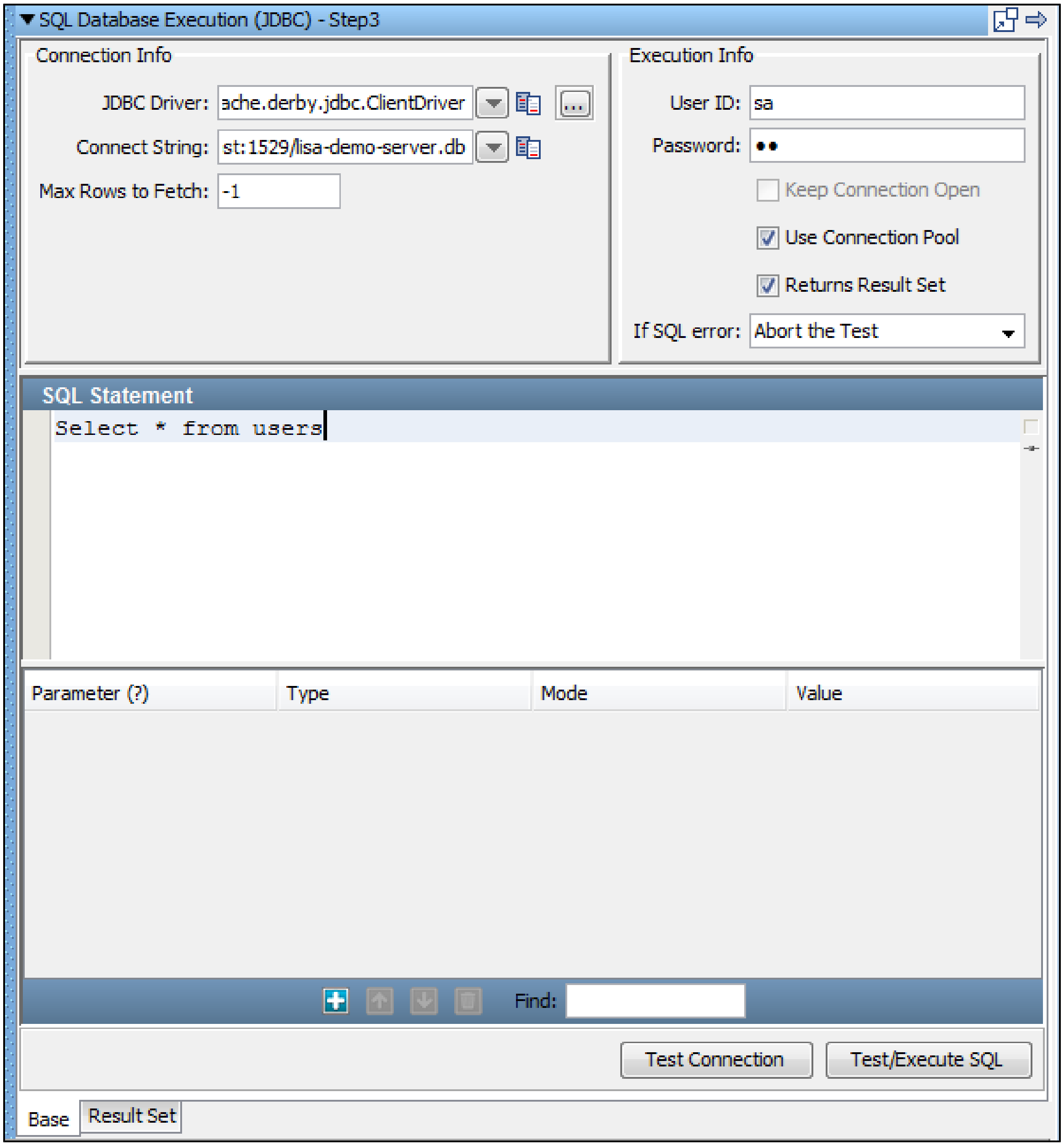
SQL Database Execution (JDBC)

The SQL Database Execution step lets you connect to a database using JDBC (Java Database Connectivity) and make SQL queries on the database.

Full SQL syntax is supported, but your SQL is not validated. It is passed through to the database where it will be validated. If you get an SQL error, it will be captured in the response so that you can assert on it. Make sure that the SQL is valid for the database manager you are using.

Prerequisites: The JDBC driver appropriate for your database must be on the LISA classpath. You can place the driver JAR file in the hot deploy directory. The Derby client driver is included in the LISA classpath, so you do not need to add it again.

Parameter Requirements: You will need to have the name of the JDBC driver class, the JDBC URL for your database, and if required, a user ID and password for the database. You will also need to know the schemas for the tables in the database to construct your SQL queries.



SQL Database Execution (JDBC) - Step3

Connection Info

JDBC Driver: ache.derby.jdbc.ClientDriver

Connect String: st:1529/lisa-demo-server.db

Max Rows to Fetch: -1

Execution Info

User ID: sa

Password: ••

□ Keep Connection Open

☑ Use Connection Pool

☑ Returns Result Set

If SQL error: Abort the Test

SQL Statement

```
Select * from users
```

Parameter (?)	Type	Mode	Value

1. Enter the following parameters in the SQL Database step editor:

Connection Info

- **JDBC Driver**: Enter or select the full package name of the appropriate driver class. Standard driver classes are available in the drop-down menu. You can also use the Browse button to browse the LISA class path for the driver class.

- **Connect String**: This is the standard JDBC URL for your database. Enter or select the URL. JDBC URL templates for common database managers are available in the drop-down menu.

- **Max Rows to Fetch**: Enter the maximum number of rows you want returned in the result set. This is a required field. Enter **-1** for unlimited rows.

Execution Info

- **User ID**: Enter a user ID (if it is required by the database).

- **Password**: Enter a password (if it is required by the database).

- **Keep Connection Open**: If selected, the database connection opened the first time the step executes is cached, and is closed when the step is garbage collected. If Keep Connection Open is not selected, the connection is closed each time the step executes.

- **Returns Result Set**: Select this check box if your query will result in a Result Set being returned; that is, a SELECT type query. Leave cleared for an UPDATE, INSERT, or DELETE. Your query will cause an error if this check box is set incorrectly.

- **If SQL error**: Select the step to redirect to if an error occurs.

To communicate with the local Demo Server, use:

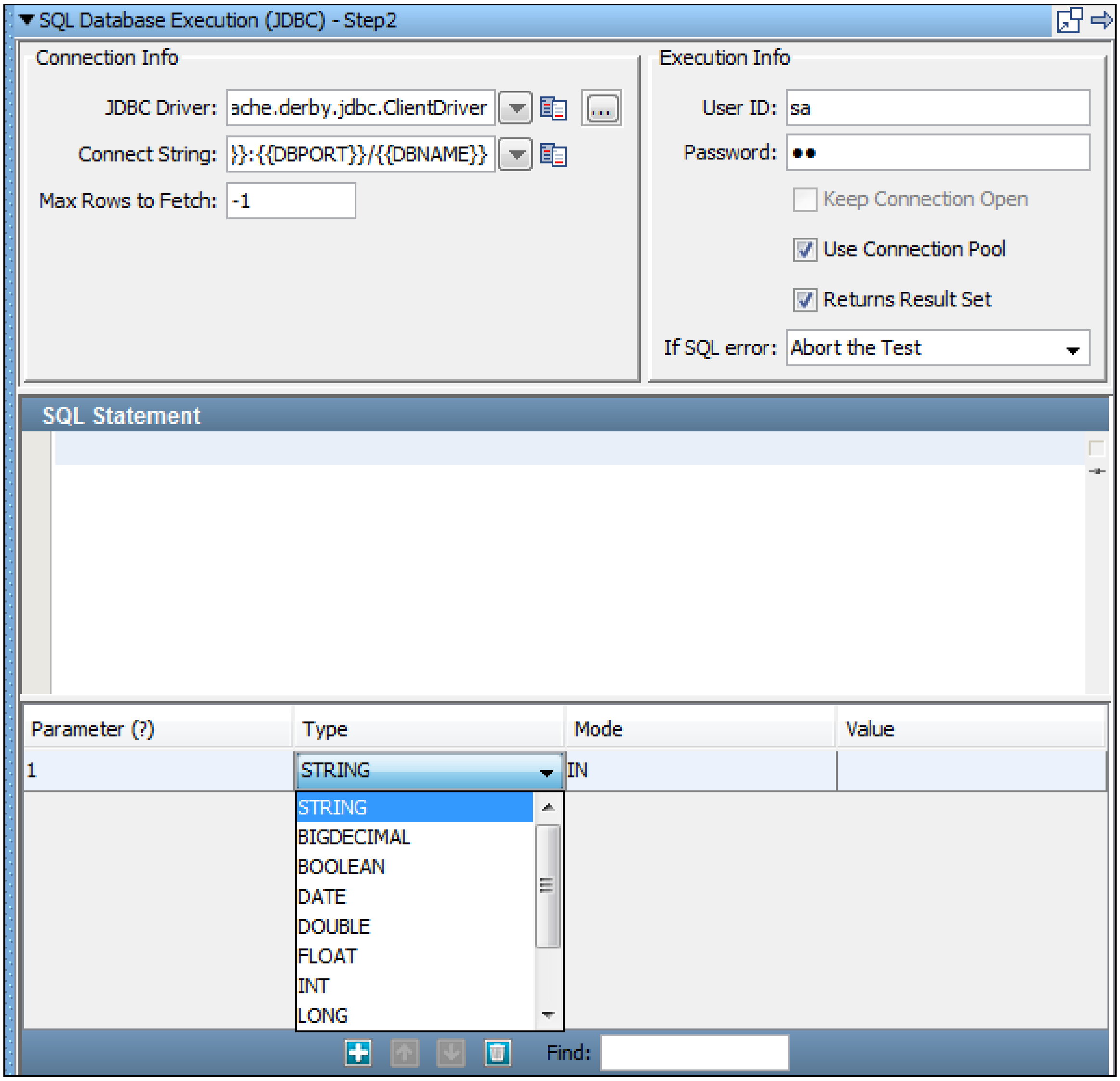
- **JDBC Driver**: org.apache.derby.jdbc.ClientDriver
- **Connect String**: jdbc:derby://localhost:1527/reports/lisa-reports.db
- **User ID**: sa
- **Password**: sa

2. After you have entered the database connection information, including the user ID and password (if required), you can use the Test Connection button to test your connection. If the information is correct, you will get a success message in a window. Otherwise you will get an error message indicating what the problem might be.

3. You are now ready to enter your SQL statement in the lower window. Properties can be used in your SQL. LISA will make the parameter substitution before passing the SQL string to the database.

The JDBC step supports stored procedure calls. Basic data types (strings, numbers, dates, Boolean) are supported as arguments into and returned by a stored procedure. Click the Add icon to add a parameter. The numbers in the Parameter column are not editable. As you add, delete, and move rows, the numbers in the Parameter column are automatically renumbered.

The JDBC step also can use JDBC prepared statements. You can use ? markers in a SQL statement and add named {{properties}} without being concerned about the type of the argument or escaping single quotes in parameter values. A statement "insert into MYTABLE(COL1,COL2) values (?, ?)" with a reference to {{col1}} and {{col2}} is easier to understand. The type and escape characters are automatically converted.



SQL Database Execution (JDBC) - Step2

Connection Info

JDBC Driver: ache.derby.jdbc.ClientDriver

Connect String: }}:{{DBPORT}}/{{DBNAME}}

Max Rows to Fetch: -1

Execution Info

User ID: sa

Password: ●●

□ Keep Connection Open

☑ Use Connection Pool

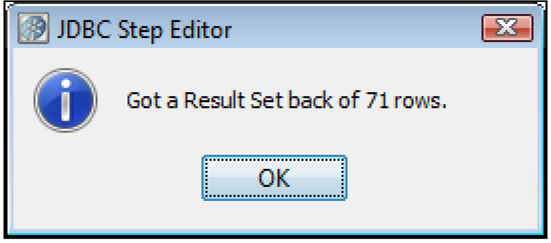
☑ Returns Result Set

If SQL error: Abort the Test

SQL Statement

Parameter (?)	Type	Mode	Value
1	STRING	IN	

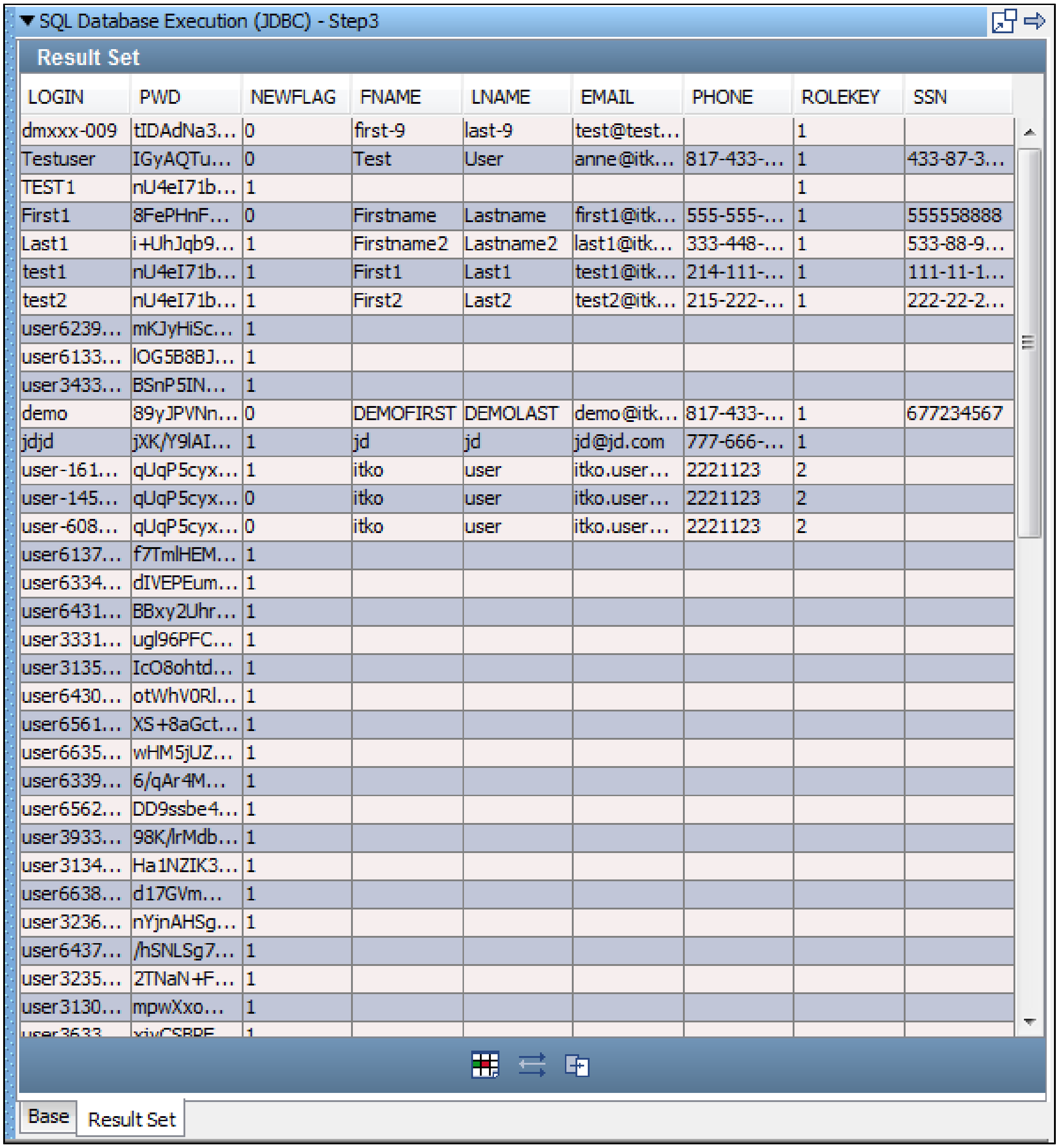
4. After you have created the SQL query, use the Test/Execute SQL button to execute the query. A message indicates the result status.



JDBC Step Editor

Got a Result Set back of 71 rows.

5. Click OK. Your results are displayed in the Result Set tab.



LOGIN	PWD	NEWFLAG	FNAME	LNAME	EMAIL	PHONE	ROLEKEY	SSN
dmxxx-009	tIDAdNa3...	0	first-9	last-9	test@test...		1	
Testuser	IGyAQTu...	0	Test	User	anne@itk...	817-433-...	1	433-87-3...
TEST1	nU4eI71b...	1					1	
First1	8FePHnF...	0	Firstname	Lastname	first1@itk...	555-555-...	1	555558888
Last1	i+UhJqb9...	1	Firstname2	Lastname2	last1@itk...	333-448-...	1	533-88-9...
test1	nU4eI71b...	1	First1	Last1	test1@itk...	214-111-...	1	111-11-1...
test2	nU4eI71b...	1	First2	Last2	test2@itk...	215-222-...	1	222-22-2...
user6239...	mKJyHiSc...	1						
user6133...	lOG5B8BJ...	1						
user3433...	BSnP5IN...	1						
demo	89yJPVNn...	0	DEMOFIRST	DEMOLAST	demo@itk...	817-433-...	1	677234567
jdjd	jXK/Y9lAI...	1	jd	jd	jd@jd.com	777-666-...	1	
user-161...	qUqP5cyx...	1	itko	user	itko.user...	2221123	2	
user-145...	qUqP5cyx...	0	itko	user	itko.user...	2221123	2	
user-608...	qUqP5cyx...	0	itko	user	itko.user...	2221123	2	
user6137...	f7TmlHEM...	1						
user6334...	dIVEPEum...	1						
user6431...	BBxy2Uhr...	1						
user3331...	ugl96PFC...	1						
user3135...	IcO8ohtd...	1						
user6430...	otWhV0Rl...	1						
user6561...	XS+8aGct...	1						
user6635...	wHM5jUZ...	1						
user6339...	6/qAr4M...	1						
user6562...	DD9ssbe4...	1						
user3933...	98K/lrMdb...	1						
user3134...	Ha1NZIK3...	1						
user6638...	d17GVm...	1						
user3236...	nYjnAHSg...	1						
user6437...	/hSNLSg7...	1						
user3235...	2TNaN+F...	1						
user3130...	mpwXxo...	1						
user3633	xjvCSBPE	1						

6. You are now ready to create filters and assertions on the result set.

The three icons on the bottom of the Result Set tab give you easy access to the following filters and assertions:

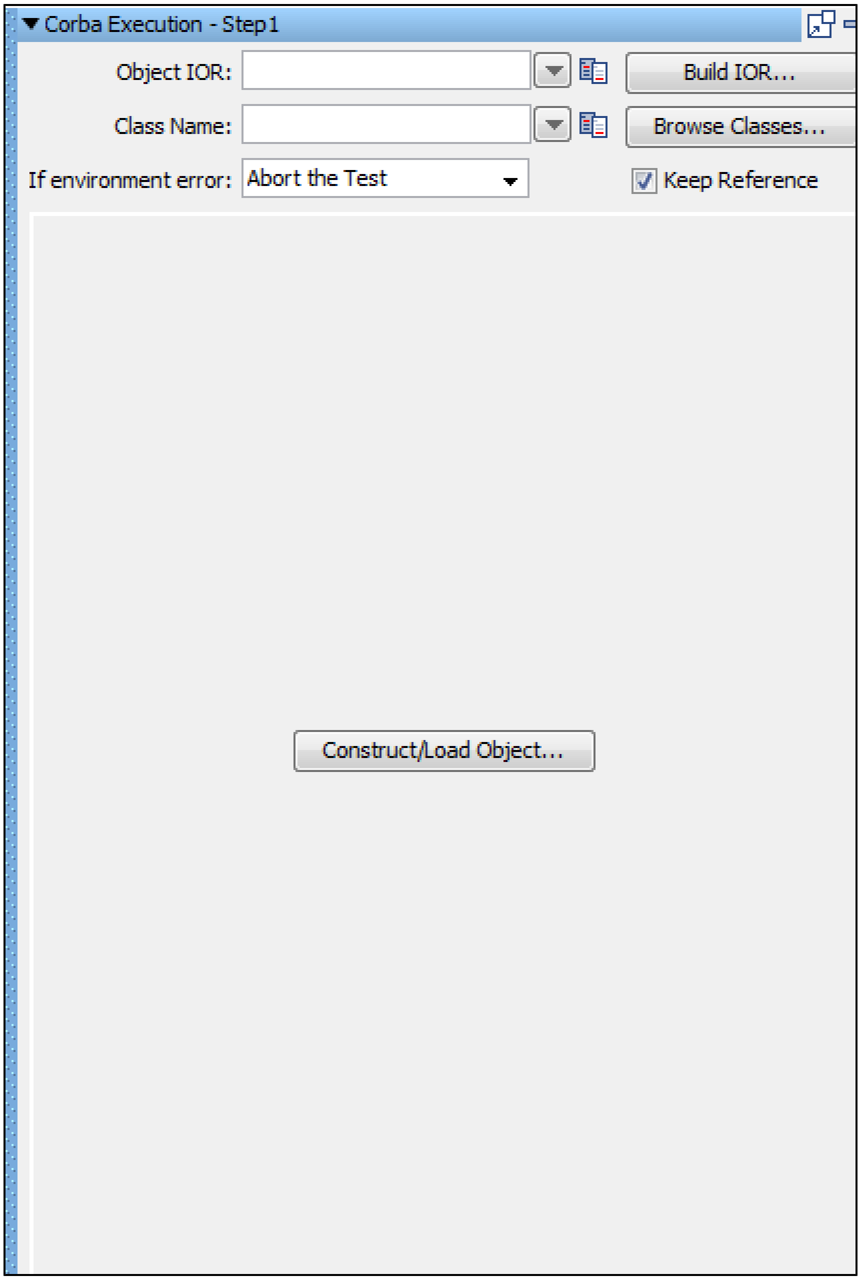
- **Get value for another value in a Result Set Row**: You select a search field cell, a value field filter and enter a property name. If the cell value in the search field is found, the value in the value field in that row will be set as the value of the property that is entered.
- **Parse Result for Value filter**: The value in the selected cell will be set as the value of the property that is entered.
- **Result Set Contents Assertion**: The values in the chosen field (column) will be compared to the regular expression that is entered.

For more information about these and other filters and assertions appropriate for result sets, see Types of Filters and Types of Assertions.

CORBA Execution

The CORBA Execution step is used to make CORBA calls using Java RMI-IIOP library. You are expected to provide appropriate skeleton classes.

1. Before starting execution, copy the corbaserver.jar file (this JAR is available in CORBA server lib directory) to the LISA **lib** directory.
2. Select the CORBA step to open its editor.



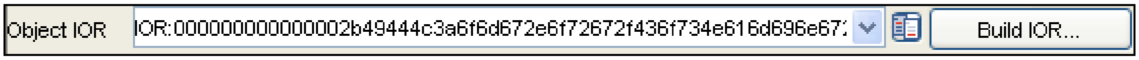
Object IOR:

Class Name:

If environment error: Abort the Test

☑ Keep Reference

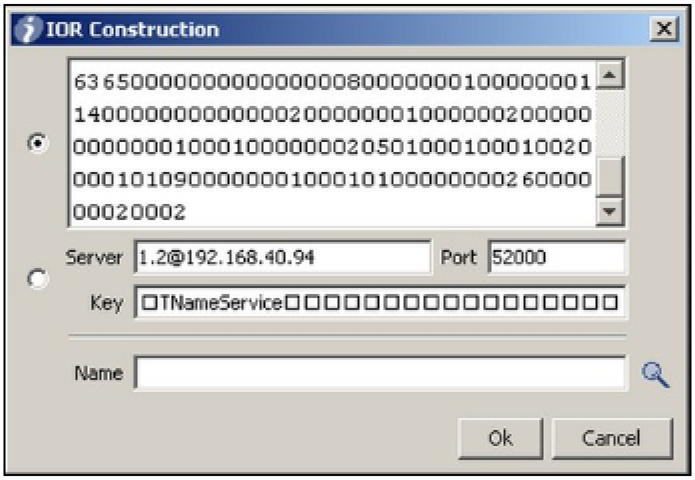
3. Complete the fields.
   Object IOR: This field contains the raw IOR string for the object or the name service. This string can be taken from the output (generated IOR) of running nameserver.sh batch file.

   Object IOR IOR:000000000000002b49444c3a6f6d672e6f72672f436f734e616d696e67

   The entire string should not contain any spaces, which may happen if it is directly copied and pasted from nameserver printed output to

the Object IOR field.
Class Name: This is the object class that IOR references.



IOR Construction

●
```
6365000000000000000080000000100000001
14000000000000002000000010000000200000
0000000100010000000205010001000100020
000101090000000100010100000000260000
00020002
```

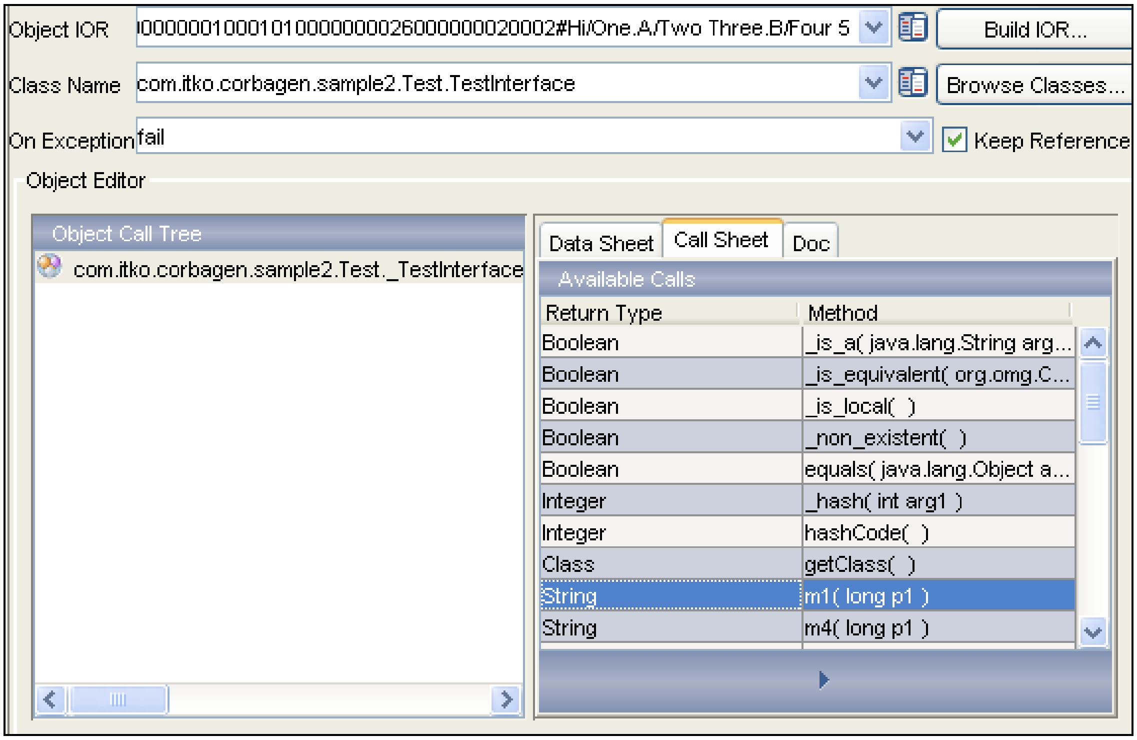
○ Server 1.2@192.168.40.94 Port 52000

Key □TNameService□□□□□□□□□□□□□□□□□

Name

You can also use the IOR construction dialog. When open, it will parse any IOR string entered and fill in the individual parts. Don't worry about the strange looking key. IORs store the key in a byte format so not every byte can be displayed properly. Just don't edit the field if you want to use the parsed version of a raw IOR.

4. Click Construct/Load Object. The dynamic object editor will show the object's call sheet.



Object IOR 000000100010100000000260000000020002#Hi/One.A/Two Three.B/Four 5

Class Name com.itko.corbagen.sample2.Test.TestInterface

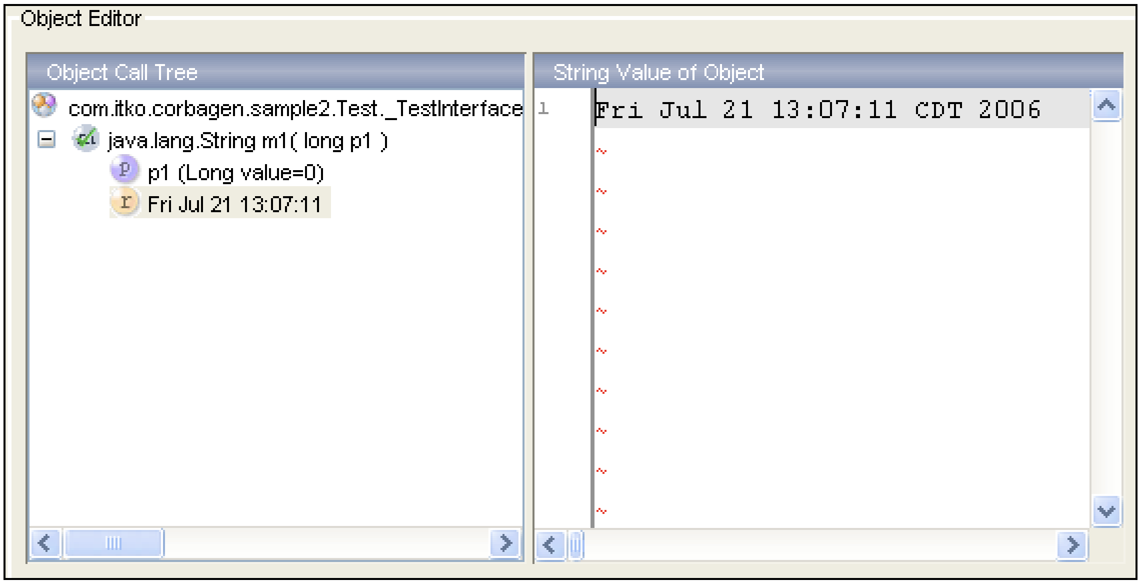
On Exception fail

☑ Keep Reference

com.itko.corbagen.sample2.Test._TestInterface

Return Type	Method
Boolean	_is_a(java.lang.String arg...
Boolean	_is_equivalent(org.omg.C...
Boolean	_is_local( )
Boolean	_non_existent( )
Boolean	equals(java.lang.Object a...
Integer	_hash(int arg1)
Integer	hashCode( )
Class	getClass( )
String	m1(long p1)
String	m4(long p1)

5. Select the method you want to call and execute it.



```
Fri Jul 21 13:07:11 CDT 2006
```

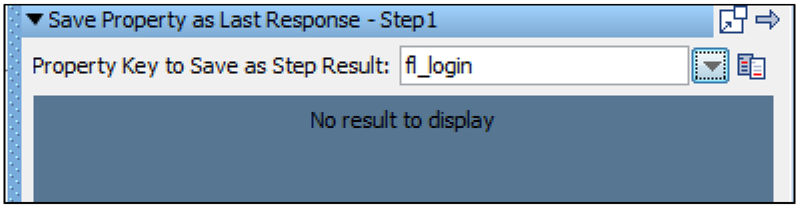
Utilities Steps

The following steps are available in this chapter.

- Save Property as Last Response
- Output Log Message
- Write Properties to File
- Read Properties from a File
- Do-Nothing Step
- Parse Text as Response
- Audit Step
- Base64 Encoder Step
- Checksum Step
- Convert XML to Element Object
- Utilities_Compare Strings for Response Lookup Step
- Utilities Compare Strings for Next Step Lookup Step

Save Property as Last Response

The Save Property as Last Response step lets you save the value of a LISA property as the last response.



Property Key to Save as Step Result: fl_login

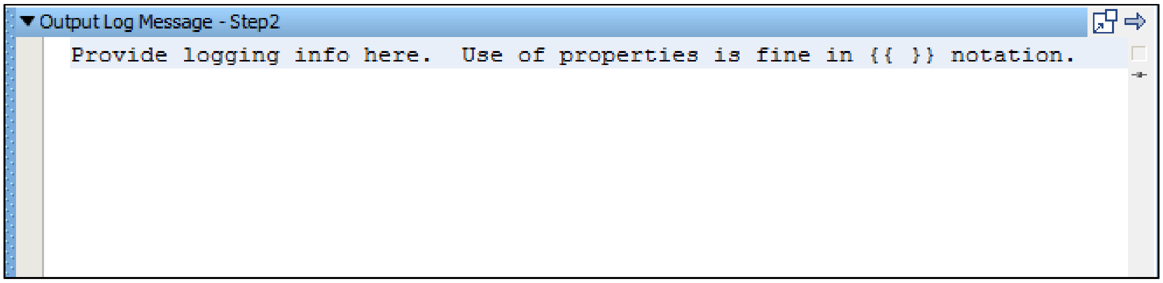
No result to display

Enter the property name of an existing LISA property or select it from the pull-down menu. The value of the property will be loaded as the last response (and the step response). It can then be accessed immediately as the last response, or later in the test case using the property lisa.thisStepname.rsp, where thisStepName is the name of the current step.

Each step in a LISA test case has a response associated with it, and when that step is executed, its response is automatically saved in two LISA properties: **LASTRESPONSE** and **lisa.thisStepName.rsp**. You use this step often so that you can have filters and assertions apply to a property value rather than the real response of the step.

Output Log Message

The Output Log Message step lets you output a text message to the log. This step is useful for tracking and logging test cases as they execute. Your log message will usually be a combination of text and LISA properties.

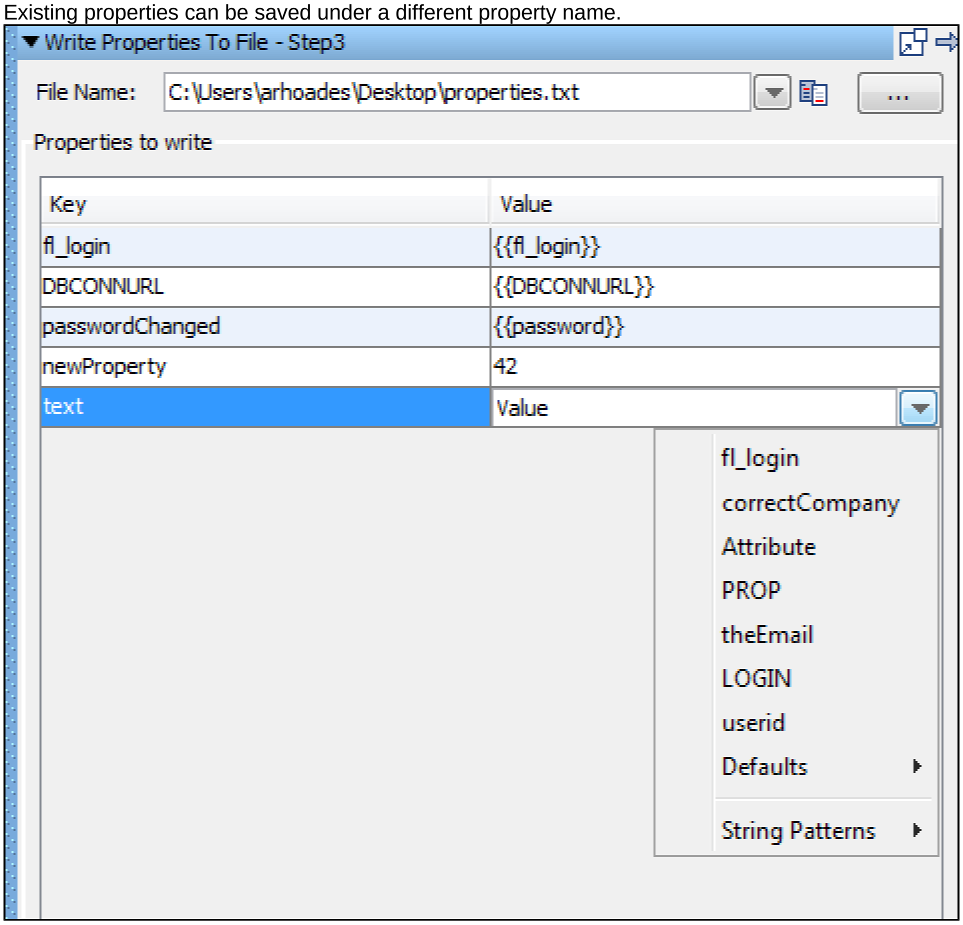


```
Provide logging info here.  Use of properties is fine in {{ }} notation.
```

Enter your log message into the editor. This log message appears when this step executes in the Interactive Test Run (ITR), and will be logged when during a test run.

Write Properties to File

The Write Properties to File step lets you save a list of LISA properties in a text file. The properties can be existing properties or new properties. Existing properties can be saved under a different property name.



File Name: C:\Users\arhoades\Desktop\properties.txt

Properties to write

Key	Value
fl_login	{{fl_login}}
DBCONNURL	{{DBCONNURL}}
passwordChanged	{{password}}
newProperty	42
text	Value

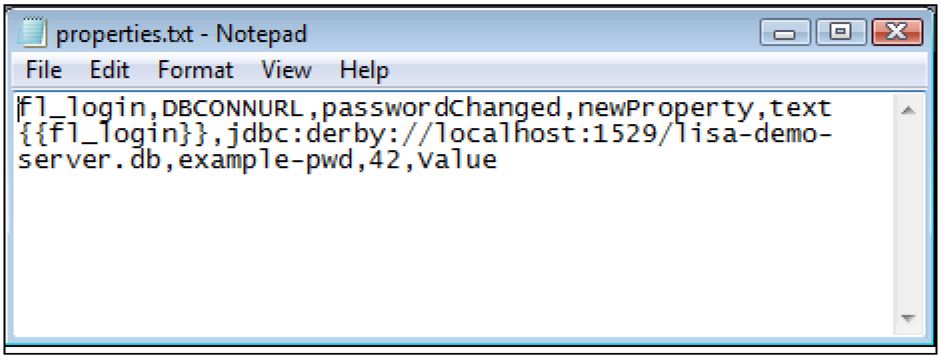
Enter the following:

- **File Name**: Enter path name of file, or browse to file using the Browse button.
- **Properties to write**: Use the Add button to add a row. Then enter the properties (Key) and corresponding values (Value) you want to save. Your list of properties can contain existing or new properties. When specifying existing properties (Key), you can override their current value by specifying a new value, or you can use their existing value by selecting it from the drop-down list.

In the previous example:

- The first two properties were stored without change.
- The third property was stored under a new name (key).
- The fourth property is new.
- The fifth entry is in progress showing the pull-down menu in the Value column.

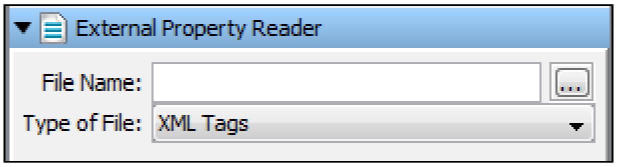
The properties are stored in a comma delimited text file where the first line in the file is the set of property names being saved, and the second line contains the values corresponding to each of the properties. This example shows the file produced using the four properties previously described.



```
fl_login,DBCONNURL,passwordChanged,newProperty,text
{{fl_login}},jdbc:derby://localhost:1529/lisa-demo-
server.db,example-pwd,42,value
```

Read Properties from a File

The Read Properties from a File step is used to read the properties from an external file. You can read the properties in two ways: name-value-pairs or in XML tags.

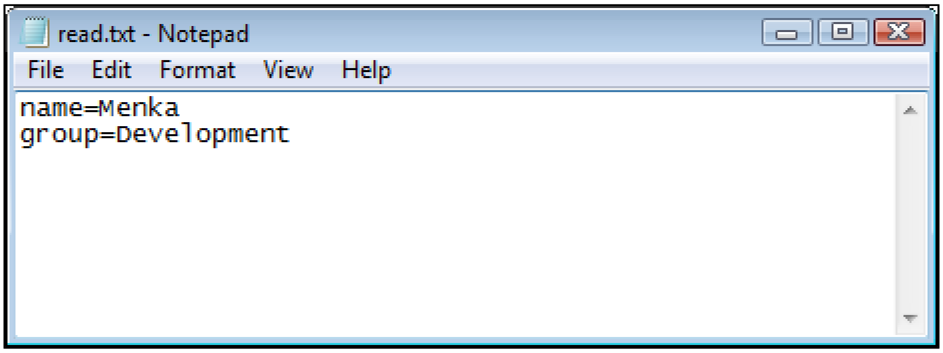


File Name:

Type of File: XML Tags

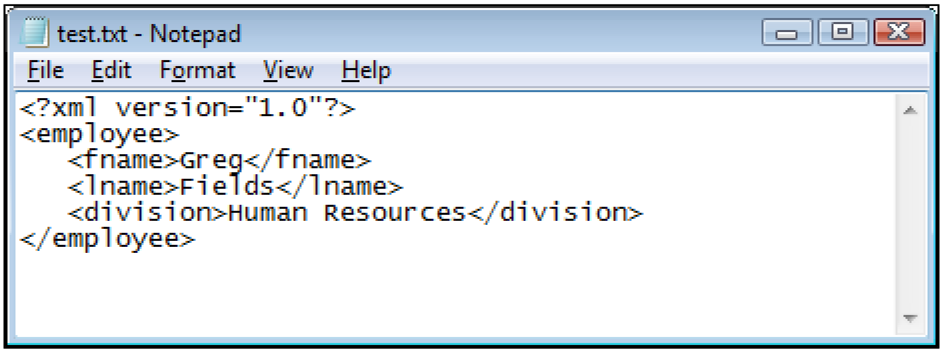
Enter the following parameters:

- **File Name**: Enter the file name or browse to the file containing the properties.
- **Type of File**: Select the type of a file, depending upon the way it has stored the properties.



```
name=Menka
group=Development
```

In the previous illustration, you can see the Name-Value-Pair type of a properties file, where *name* is a property and *Menka* is a value of the name property.



```
<?xml version="1.0"?>
<employee>
   <fname>Greg</fname>
   <lname>Fields</lname>
   <division>Human Resources</division>
</employee>
```

In the previous illustration you can see the XML Tags type of properties file, where *fname* is a property and *Greg* is a value of the fname property.

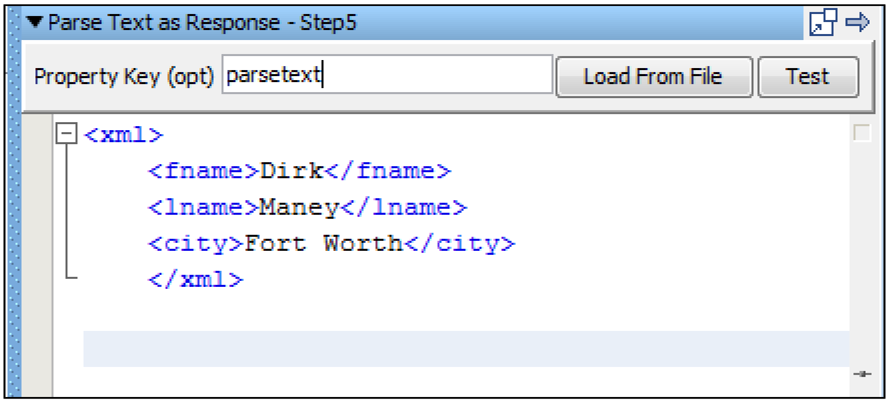
Do-Nothing Step

The Do-Nothing step does not take any parameters, nor does it have any functionality by itself. However, this step is useful in certain situations, as you can add assertions to the step.

For example, you can use the scripted assertion to add a quick custom assertion, to compare numbers or dates, or some numerical comparison test (greater than or less than).

Parse Text as Response

The Parse Text as Response step lets you enter textual content from a file that can be saved as the last response. The content can be stored in a LISA property (optional). You can type or paste the text into the editor, or load it from a file.



Property Key (opt) parsetext

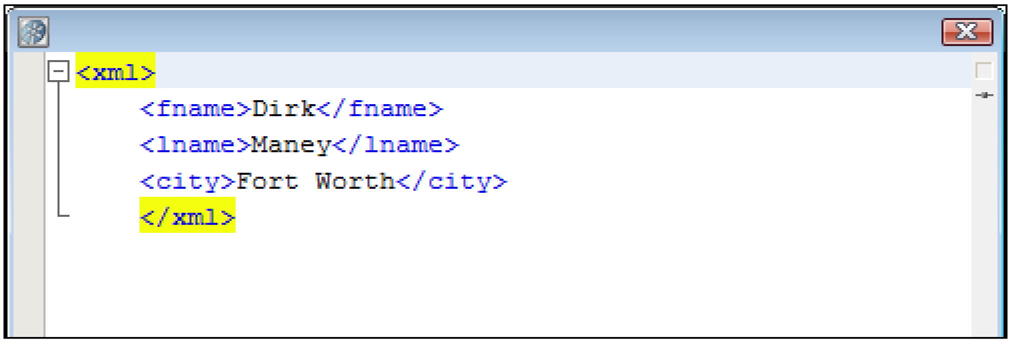
```
<xml>
    <fname>Dirk</fname>
    <lname>Maney</lname>
    <city>Fort Worth</city>
    </xml>
```

Enter the following parameters:

- **Property Key**: The name of the property to store the content (optional).
- **Load from File**: Click to browse to the file. Otherwise type or paste the text into the editor.

The content is now available for you to parameterize, filter and add assertions.

Click the Test button to show the resulting text.



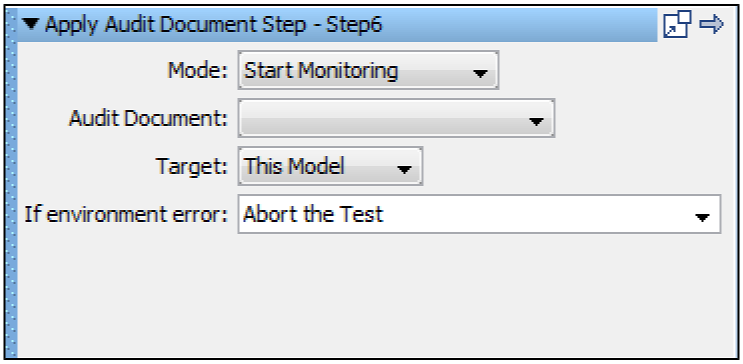
```
<xml>
    <fname>Dirk</fname>
    <lname>Maney</lname>
    <city>Fort Worth</city>
    </xml>
```

Audit Step

The Audit Step lets you apply an audit document against a current test step, remote test, or virtual service.

This allows a LISA model to be verified in terms of the events it produces during execution.

Click the step to open its editor:



Apply Audit Document Step - Step6

Mode: Start Monitoring

Audit Document:

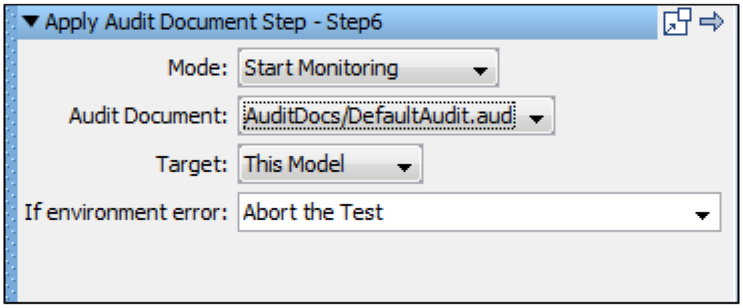
Target: This Model

If environment error: Abort the Test

Mode: This step has two modes: Start Monitoring and Apply Audit Document.

Start Monitoring Mode

If you select the Start Monitoring Mode:



Apply Audit Document Step - Step6

Mode: Start Monitoring

Audit Document: AuditDocs/DefaultAudit.aud

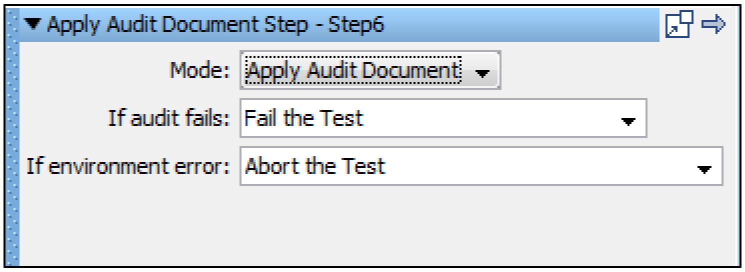
Target: This Model

If environment error: Abort the Test

- **Audit Document**: Enter or browse for the audit document.
- **Target**: Select the target for the audit document:
  - **This Model**: Will apply to the current model.
  - **Test Run**: Will apply to the test run.
  - **Virtual Model**: Will apply to the virtual model.
- **If Environment Error**: Select the step to be executed if there is an environment error.

Apply Audit Document Mode

If you select the Apply Audit document mode:



Apply Audit Document Step - Step6

Mode: Apply Audit Document

If audit fails: Fail the Test

If environment error: Abort the Test

- **If Audit Fails**: Select the step to be executed if the audit fails.
- **If Environment Error**: Select the step to be executed if there is an environment error.

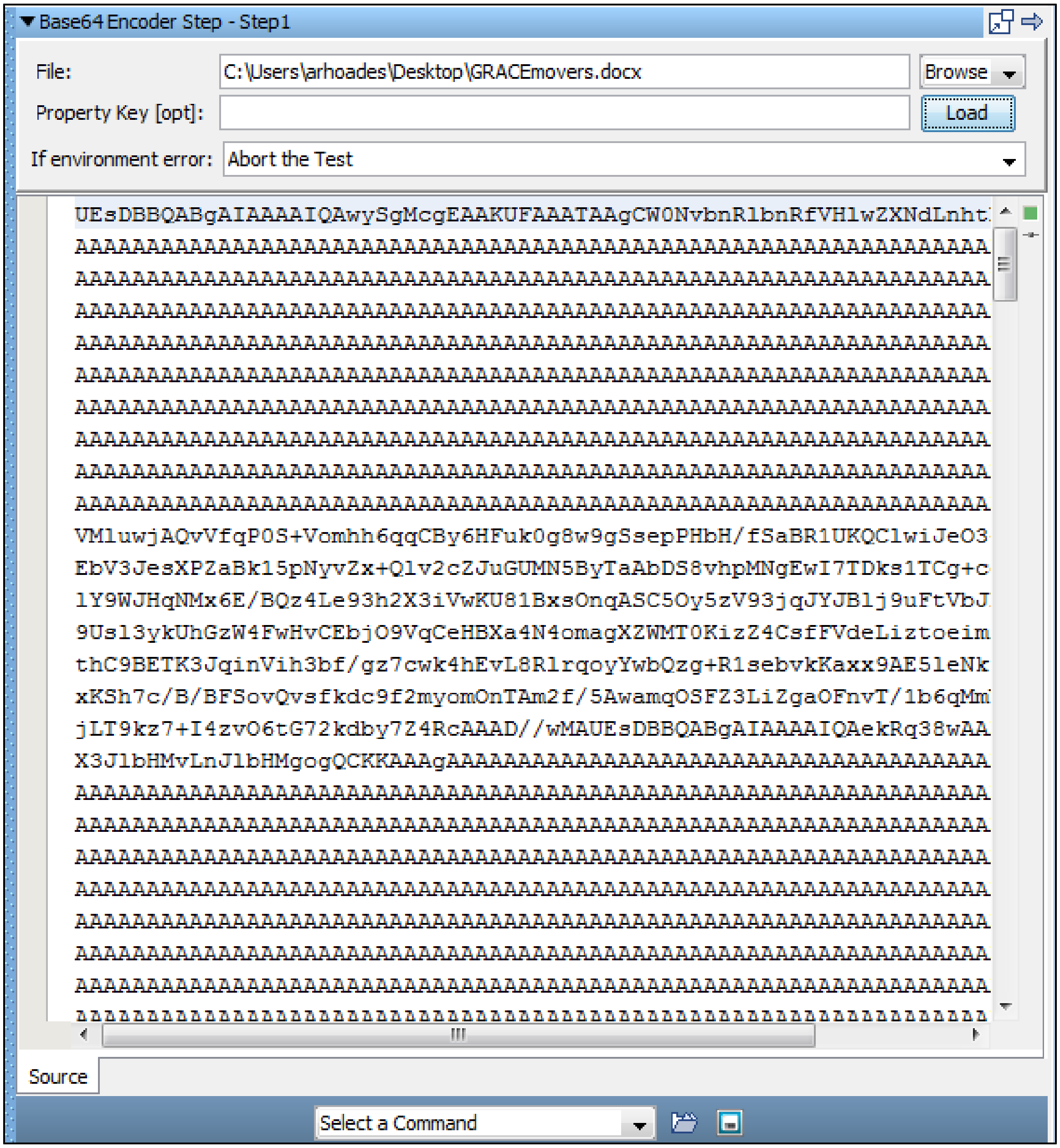
Base64 Encoder Step

Base64 Encoder Step

This step is used to encode a file using Base 64 encoding algorithm.

The result can be stored in a LISA property file to be used anywhere within the test run.

Click the step to open its editor:



File: C:\Users\arhoades\Desktop\GRACEmovers.docx

Property Key [opt]:

If environment error: Abort the Test

```
UEsDBBQABgAIAAAAIQAwySgMcgEAAKUFAAATAAgCW0NvbnRlbnRfVHlwZXNdLnht
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
VMluwjAQvVfqP0S+Vomhh6qqCBy6HFuk0g8w9gSsepPHbH/fSaBR1UKQClwiJeO3
EbV3JesXPZaBk15pNyvZx+Qlv2cZJuGUMN5ByTaAbDS8vhpMNgEwI7TDks1TCg+c
lY9WJHqNMx6E/BQz4Le93h2X3iVwKU81BxsOnqASC5Oy5zV93jqJYJBlj9uFtVbJ
9Usl3ykUhGzW4FwHvCEbjO9VqCeHBXa4N4omagXZWMT0KizZ4CsfFVdeLiztoeim
thC9BETK3JqinVih3bf/gz7cwk4hEvL8RlrqoyYwbQzg+R1sebvkKaxx9AE5leNk
xKSh7c/B/BFSovQvsfkdc9f2myomOnTAm2f/5AwamqOSFZ3LiZgaOFnvT/1b6qMm
jLT9kz7+I4zvO6tG72kdby7Z4RcAAAD//wMAUEsDBBQABgAIAAAAIQAekRq38wAA
X3JlbHMvLnJlbHMgogQCKKAAAgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

Enter the following parameters:

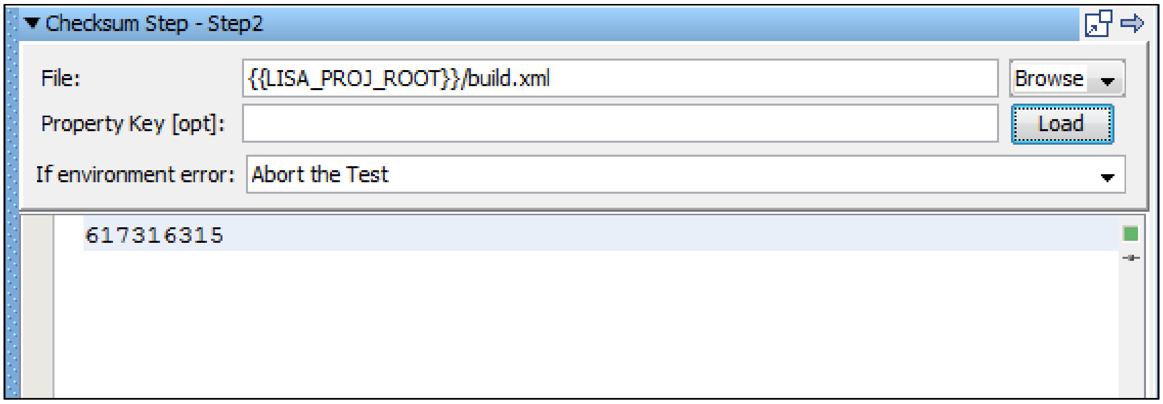
- **File**: Enter the name of file to be encoded or browse to the target location.
- **Property Key**: The name of the property to store the encoded file. This is optional.
- **If Environment Error**: Select the step to be executed if there is an environment error.

Click the Load button to load the file in the editor.

Here you can apply filters and/or assertions if required from the Command menu at the bottom.

Checksum Step

The Checksum step calculates the checksum of a file and can save that value in a LISA property.



Checksum Step - Step2

File: {{LISA_PROJ_ROOT}}/build.xml

Property Key [opt]:

If environment error: Abort the Test

```
617316315
```

Enter the following parameters:

- **File**: Enter the pathname or browse to the file on which you want the checksum calculated.
- **Property key**: Enter the name of the property that will store the checksum value.

Click the Load button.

The checksum will be displayed as the response, and if a property name was entered the value will be stored in that property.

The checksum value is now available for you to filter and add assertions.

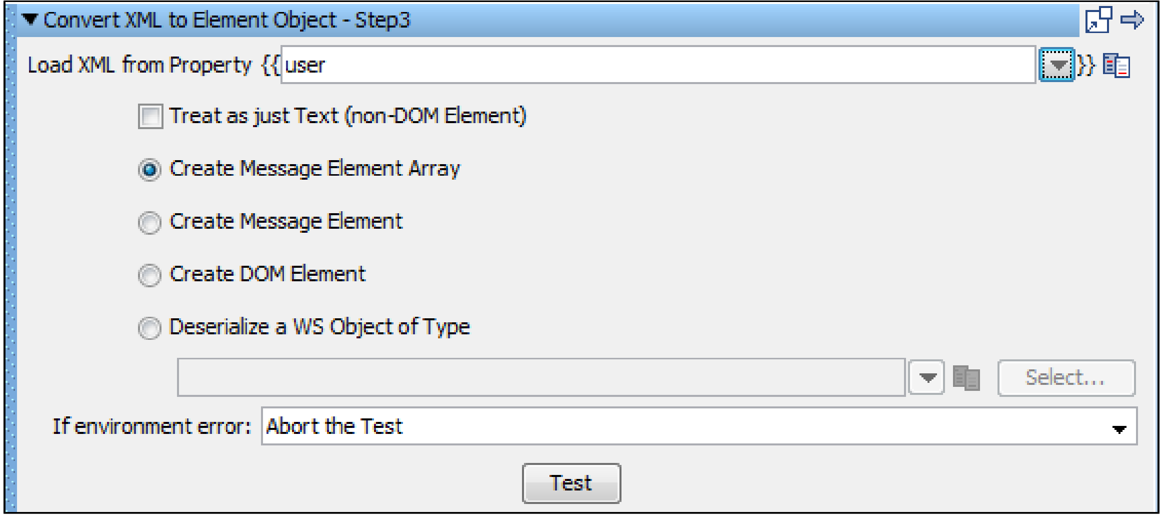
Convert XML to Element Object

The Convert XML to Element Object step converts a raw XML into an object of one of the following types:

- Message Element Array
- Message Element
- DOM Element

This is useful when you have a web service API that takes any type using strict processing. This type of WSDL element will require a Message Element Array as an input parameter. You can capture the raw XML from a previous step (such as Read from File or Parse Text as Response) and store it in a LISA property. That property becomes an input parameter for this step.

Prerequisites: The XML must be already stored in a LISA property.



Convert XML to Element Object - Step3

Load XML from Property {{user}}

□ Treat as just Text (non-DOM Element)

● Create Message Element Array

○ Create Message Element

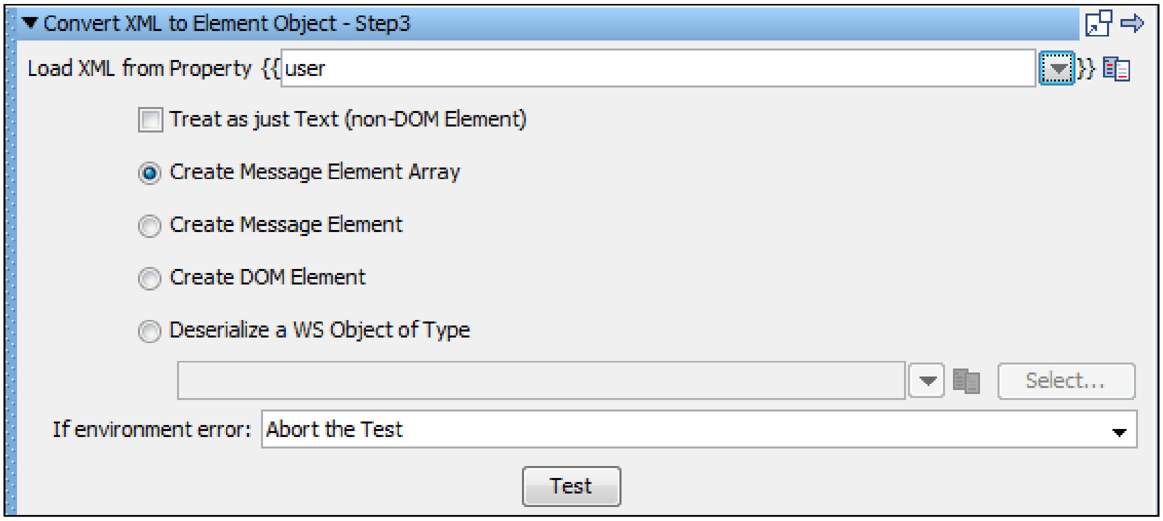
○ Create DOM Element

○ Deserialize a WS Object of Type

If environment error: Abort the Test

Enter the following parameters:

- **Load XML from Property**: Enter the property that contains the XML. This can be a user-defined property or a built-in LISA property.
- Check the Treat as just Text check box if you need to use plain text as your input rather than XML. This will result in a message element that contains plain text.
- Select the type of object you want from the available types by clicking the respective option button.
- Click the Test button to do the conversion.



Convert XML to Element Object - Step3

Load XML from Property {{user}}

☐ Treat as just Text (non-DOM Element)

● Create Message Element Array

○ Create Message Element

○ Create DOM Element

○ Deserialize a WS Object of Type

If environment error: Abort the Test

Use the response from this step when this object is required as a parameter in another step.

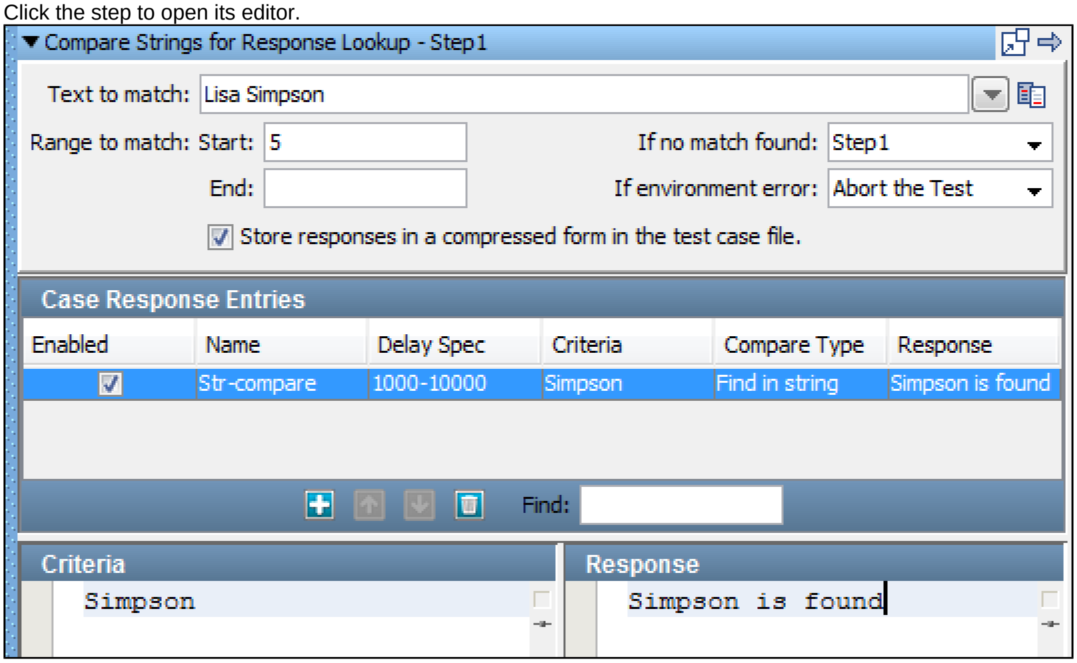
You can use the Save Step Response as a Property filter to save the response in a filter, or you can refer to it as lisa.<stepname>.rsp.

Utilities_Compare Strings for Response Lookup Step

The Compare Strings for Response Lookup step is used to look at an incoming request to a virtual service and determine the appropriate response. You can match the incoming requests using partial text match, regular expression, and others.

This step is automatically filled out and added to a virtual service when using the Virtual Web Service HTTP Recorder.

Click the step to open its editor.



Compare Strings for Response Lookup - Step1

Text to match: Lisa Simpson

Range to match: Start: 5

End:

If no match found: Step1

If environment error: Abort the Test

☑ Store responses in a compressed form in the test case file.

Case Response Entries

Enabled	Name	Delay Spec	Criteria	Compare Type	Response
☑	Str-compare	1000-10000	Simpson	Find in string	Simpson is found

Criteria

```
Simpson
```

Response

```
Simpson is found
```

Enter the following parameters:

- **Text to match**: Enter the text against which criteria should be matched. This is typically a property reference, such as LASTRESPONSE.
- **Range to match**: Enter the Start and End of the range.
- **If no match found**: Select the step to be executed, if no match is found.
- **If environment error**: Select the step to be executed, if the test fails.
- **Store responses in a compressed form...**: Selected by default. This option compresses the responses in the test case file.
- **Case Response Entries**: In this table you can add, move, and delete entries by clicking the Add, Move and Delete icons. The columns in the table are:
  - **Enabled Column**: Selected by default when you add an entry. Clear to ignore an entry.
  - **Name Column**: Enter a unique name for the case response entry.
  - **Delay Spec Column**: Enter the delay specification range. The default is 1000-10000, which indicates to use a randomly selected delay time between 1000 and 10000 milliseconds. (The syntax is the same format as Think Time specifications.)
  - **Criteria Column**: This area provides the response of this step if the entry matches the Text to match field. To edit the criteria, in

    the Criteria Column area select the appropriate row, and enter a different setting in the criteria list. Click Enter to get it updated in the row below.
    - **Compare Type Column**: Select an option for the Compare type from the list:
        - **Find in string (default)**
        - **Regular expression**
        - **Starts with**
        - **Ends with**
        - **Exactly equals**
    - **Response Column**: Allows for the update of the step response for an entry.
- **Response**: This area provides the response of this step if the entry matches the Text to match field. To edit the response, in the Case Response Entries, area select the appropriate row, and enter a different setting in the Response list. Click Enter to get it updated in the row above.
- **Criteria**: Allows for the update of the criteria string for an entry.
- **Response**: Allows for the update of the step response for an entry.

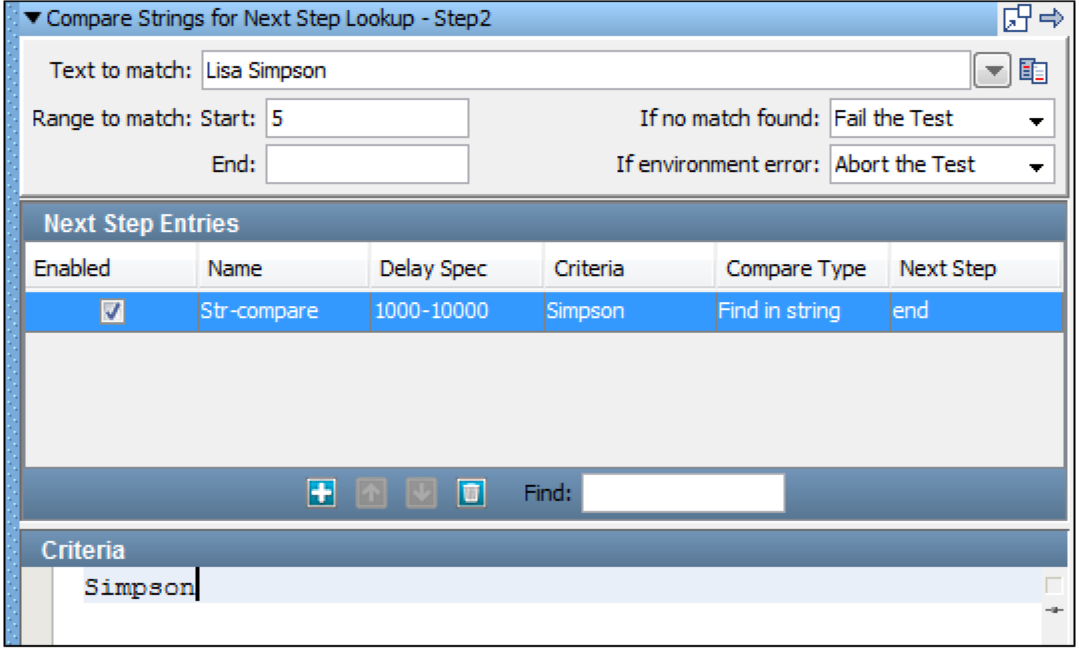
Utilities Compare Strings for Next Step Lookup Step

This step is used to look at an incoming request and determine the appropriate next step.

You can match incoming requests using partial text match, regular expression, among others. Each matching criterion specifies the name of the step to which to transfer if the match succeeds.

This step is automatically filled out and matches to a Virtual Service when using the JDBC Database Traffic Recorder.

Click the step to open its editor.



Compare Strings for Next Step Lookup - Step2

Text to match: Lisa Simpson

Range to match: Start: 5

End:

If no match found: Fail the Test

If environment error: Abort the Test

Next Step Entries

Enabled	Name	Delay Spec	Criteria	Compare Type	Next Step
☑	Str-compare	1000-10000	Simpson	Find in string	end

Find:

Criteria

Simpson

Enter the following parameters:

- **Text to match**: Enter the text against which criteria should be matched. This is typically a property reference, such as **`LASTRESPONSE`**.
- **Range to match**: Enter the Start and End of the range.
- **If no match found**: Select the step to execute if no match is found.
- **If environment error**: Select the step to execute if there is an environment error.
- **Next Step Entries**: Click the Add icon to add an entry. Use the Move and Delete icons to move or delete an entry. Enter text in the **Find** field to find a entry.
  The columns in the Next Step Entry are:
- **Enabled Column**: Selected by default when you add an entry. Clear to ignore an entry.
- **Name Column**: Enter a unique name for the next step entry.
- **Delay Spec Column**: Enter the delay specification range. The default is 1000-10000, which indicates to use a randomly selected delay time between 1000 and 10000 milliseconds. The syntax is the same format as Think Time specifications.
- **Criteria Column**: This area defines the criteria to compare against the Text to match field.
- **Compare Type Column**: This has five options to select from:
    - Find in string (default)
    - Regular expression
    - Starts with
    - Ends with
    - Exactly equals
- **Criteria**: Allows for the update of the criteria string for an entry.

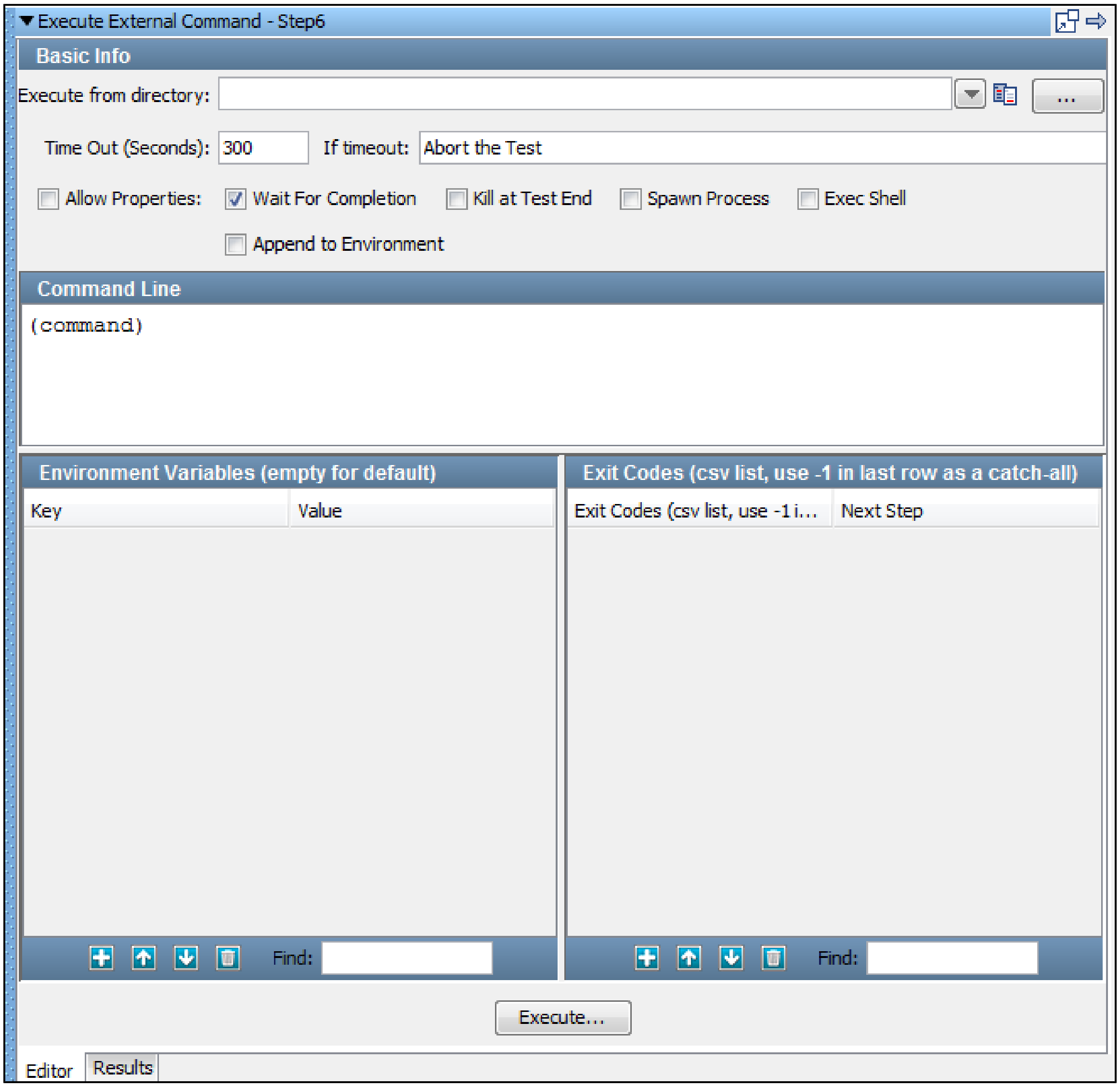
External_Subprocess Steps

The following steps are available in this chapter.

Execute External Command

The Execute External Command step lets you execute an external program, such as an operating system script, an operating system command, or an executable, and capture its contents for filtering or making assertions.

The external program syntax will depend on your operating system.



Execute External Command - Step6

Basic Info

Execute from directory:

Time Out (Seconds): 300 If timeout: Abort the Test

□ Allow Properties: ☑ Wait For Completion □ Kill at Test End □ Spawn Process □ Exec Shell

□ Append to Environment

Command Line

```
(command)
```

Environment Variables (empty for default)

Key	Value

Exit Codes (csv list, use -1 in last row as a catch-all)

Exit Codes (csv list, use -1 i...	Next Step

Enter the following parameters in the Execute External Command editor:

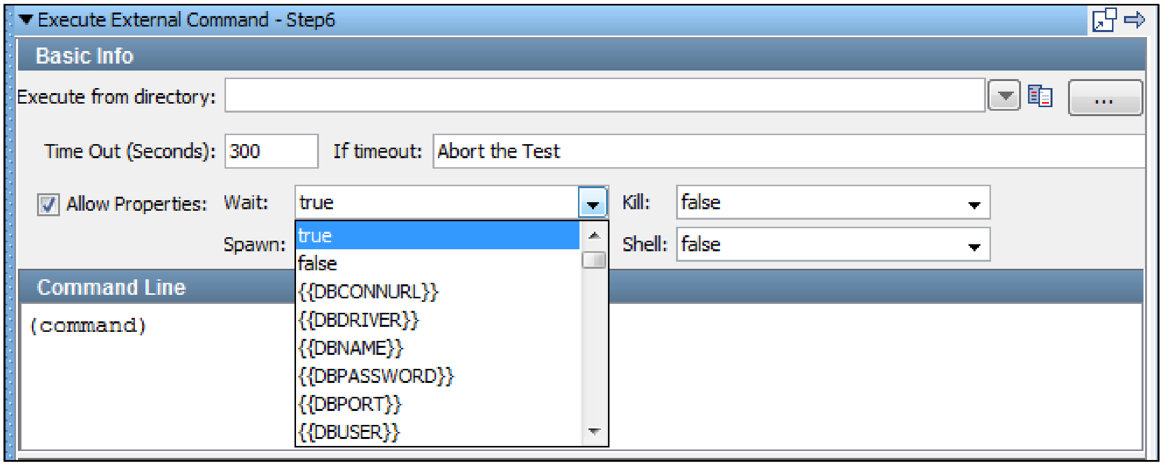
- **Execute from directory**: The directory that will be considered current when the external command is executed. If the directory does not exist on the system that is running the test, the directory will be created, subject to file system permissions. If the directory does not exist and cannot be created the step will fail.
- **Time Out (Seconds)**: How long to wait before transferring to the step defined by On Time Out Execute.
- **If timeout**: The step to execute if the external command does not complete execution before the given timeout value.

- **If environment error**: The step to execute when an environment error occurs.
- **Allow Properties**: This check box determines whether properties are allowed for the next four parameters. It changes the look of the command editor interface.

With the Allow Properties check box cleared you will see five check boxes as shown in the previous illustration. Here your only choice is to select the parameter or not.

- **Wait for Completion**: If this check box is selected, the step waits until the execution has completed, and then the results can be filtered or asserted upon. If this check box is not selected, filters and assertions will execute; however execution will not wait for the result of the command being executed.
- **Kill at Test End**: Can be used to kill the process after the test case has completed, if the **Wait for Completion** box is not selected. This lets a process run while a test case executes and then be shut down. A LISA property will contain the process ID of the started command.
- **Spawn Process**: Creates a new process in the operating system to run the command in. This is helpful if you want to have a long-running background process or need to make sure that a new set of environment variables is set and nothing set by LISA is in your environment.
- **Exec Shell**: Lets you run the contents of the Command Line within a system shell. This is required if you need to use the features of a shell process such as redirecting (pipe) output streams to files or other commands. Depending upon your system this option may be required for you to run system commands like **dir** or **ls**. This must be checked for a Windows operating system.
- **Append to Environment**: When environment variables are defined (in the step) they will be appended to the existing environment, as opposed to creating a new empty environment where only these variables are defined.

With the **Allow properties** check box selected, you will see pull-down menus that have the same functionality as shown previously, but each parameter can now be a property.



Execute External Command - Step6

Basic Info

Execute from directory:

Time Out (Seconds): 300 If timeout: Abort the Test

☑ Allow Properties: Wait: true Kill: false

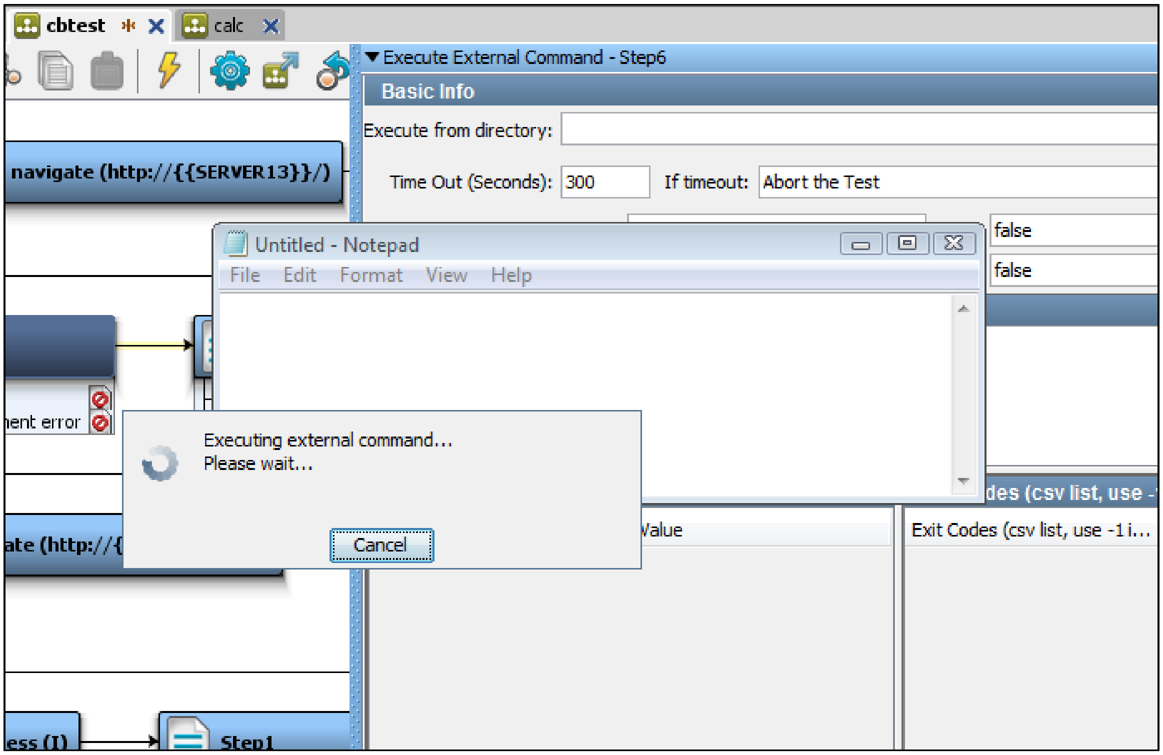
Spawn: Shell: false

Command Line

(command)

- **Command Line**: The external command is generally just one command written as a shell script or batch file. It is possible to execute multiple commands when the **Exec Shell** option is also selected. The command string must be valid for the operating system that you are running LISA on.
- **Environment Variables**: Allows existing environment variables to be overridden with new environment variables. If nothing is entered the existing environment variables are used for the command. If an environment variable is specified, the new variables sets are used; the environment variables that were used to start LISA are not used.
- **Exit Codes**: Lets you change the outcome of the test based upon the exit code of the executed process. You can enter a comma-delimited string of exit codes with a corresponding step to execute when the process exits with this code.

To test your command, press the **Execute** button. Here is a sample of its output, when the command entered was "notepad."



Execute External Command - Step6

Basic Info

Execute from directory:

Time Out (Seconds): 300 If timeout: Abort the Test

false

false

Exit Codes (csv list, use -1 i...

Executing external command...
Please wait...

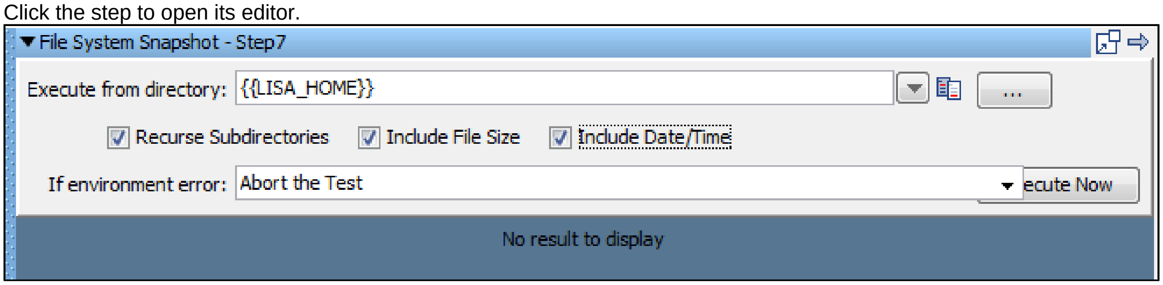
The content is now available for you to filter and add assertions.

File System Snapshot

The File System Snapshot step lets you list the files in a directory in a format that is operating system independent.

You can list a single file, all the files in a directory, or all the files in a directory tree.

1. Click the step to open its editor.



File System Snapshot - Step7

Execute from directory: {{LISA_HOME}}

☑ Recurse Subdirectories ☑ Include File Size ☑ Include Date/Time

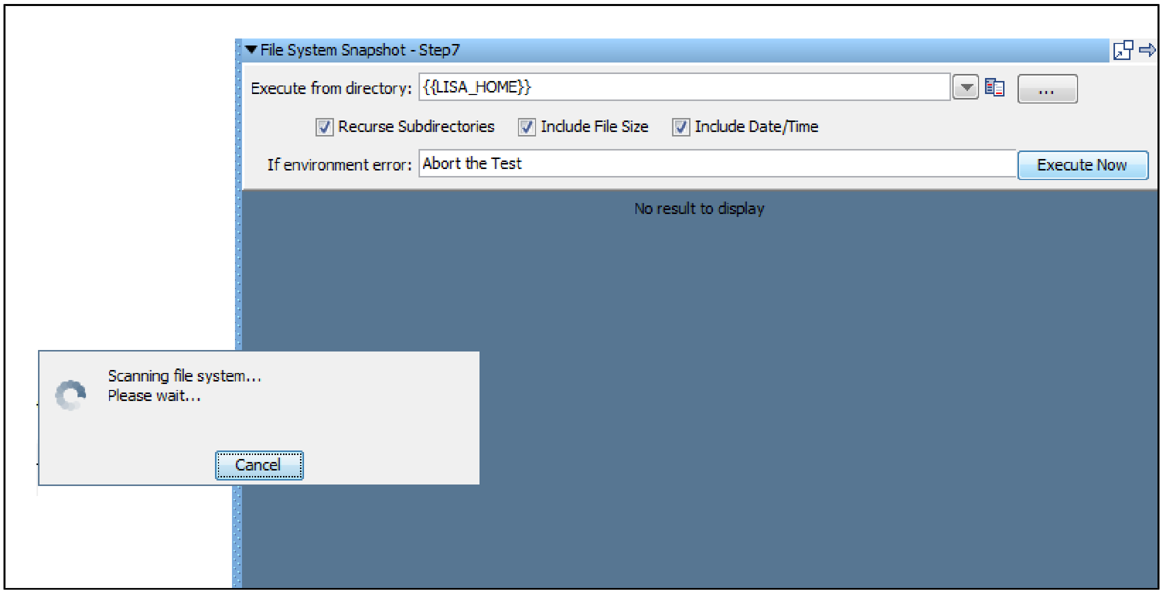
If environment error: Abort the Test

No result to display

2. Enter the following parameters:

- **Execute from directory**: Enter the path name or browse to the file or directory.
- **Recurse Subdirectories**: Check if you want the complete directory tree including sub directories.
- **Include File Size**: Check if you want the file sizes to be listed.
- **Include Date/Time**: Check if you want the last modified date to be listed.
- **If environment error**: Select a step to be executed if there is an environment error.

3. Click Execute Now to execute. This will start the scanning of the file system.

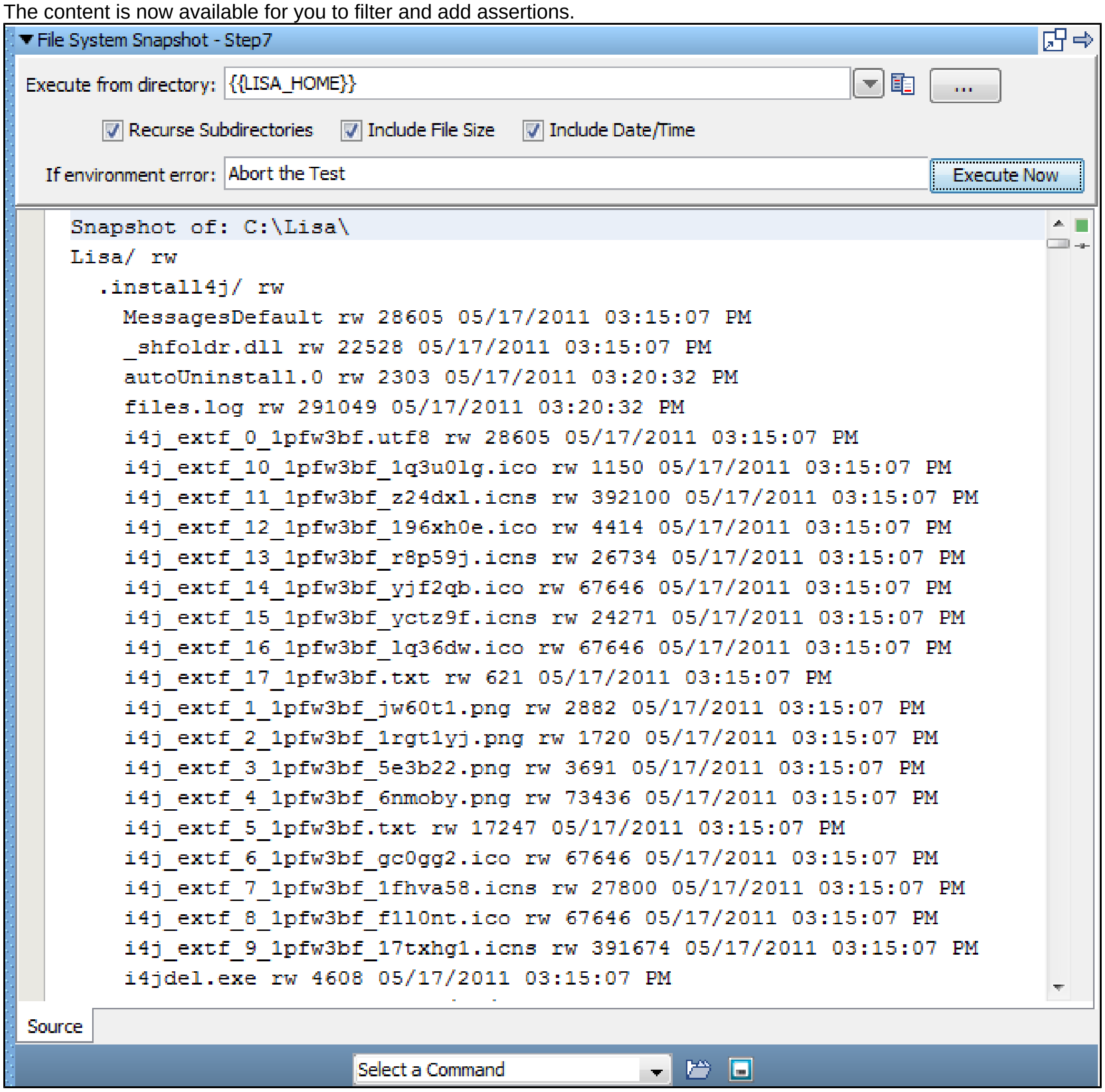


Execute from directory: {{LISA_HOME}}

☑ Recurse Subdirectories ☑ Include File Size ☑ Include Date/Time

If environment error: Abort the Test

The content is now available for you to filter and add assertions.



Execute from directory: {{LISA_HOME}}

☑ Recurse Subdirectories ☑ Include File Size ☑ Include Date/Time

If environment error: Abort the Test

```
Snapshot of: C:\Lisa\
Lisa/ rw
  .install4j/ rw
    MessagesDefault rw 28605 05/17/2011 03:15:07 PM
    _shfoldr.dll rw 22528 05/17/2011 03:15:07 PM
    autoUninstall.0 rw 2303 05/17/2011 03:20:32 PM
    files.log rw 291049 05/17/2011 03:20:32 PM
    i4j_extf_0_1pfw3bf.utf8 rw 28605 05/17/2011 03:15:07 PM
    i4j_extf_10_1pfw3bf_1q3u0lg.ico rw 1150 05/17/2011 03:15:07 PM
    i4j_extf_11_1pfw3bf_z24dxl.icns rw 392100 05/17/2011 03:15:07 PM
    i4j_extf_12_1pfw3bf_196xh0e.ico rw 4414 05/17/2011 03:15:07 PM
    i4j_extf_13_1pfw3bf_r8p59j.icns rw 26734 05/17/2011 03:15:07 PM
    i4j_extf_14_1pfw3bf_yjf2qb.ico rw 67646 05/17/2011 03:15:07 PM
    i4j_extf_15_1pfw3bf_yctz9f.icns rw 24271 05/17/2011 03:15:07 PM
    i4j_extf_16_1pfw3bf_1q36dw.ico rw 67646 05/17/2011 03:15:07 PM
    i4j_extf_17_1pfw3bf.txt rw 621 05/17/2011 03:15:07 PM
    i4j_extf_1_1pfw3bf_jw60t1.png rw 2882 05/17/2011 03:15:07 PM
    i4j_extf_2_1pfw3bf_1rgt1yj.png rw 1720 05/17/2011 03:15:07 PM
    i4j_extf_3_1pfw3bf_5e3b22.png rw 3691 05/17/2011 03:15:07 PM
    i4j_extf_4_1pfw3bf_6nmoby.png rw 73436 05/17/2011 03:15:07 PM
    i4j_extf_5_1pfw3bf.txt rw 17247 05/17/2011 03:15:07 PM
    i4j_extf_6_1pfw3bf_gc0gg2.ico rw 67646 05/17/2011 03:15:07 PM
    i4j_extf_7_1pfw3bf_1fhva58.icns rw 27800 05/17/2011 03:15:07 PM
    i4j_extf_8_1pfw3bf_f110nt.ico rw 67646 05/17/2011 03:15:07 PM
    i4j_extf_9_1pfw3bf_17txhg1.icns rw 391674 05/17/2011 03:15:07 PM
    i4jdel.exe rw 4608 05/17/2011 03:15:07 PM
```

Execute Subprocess

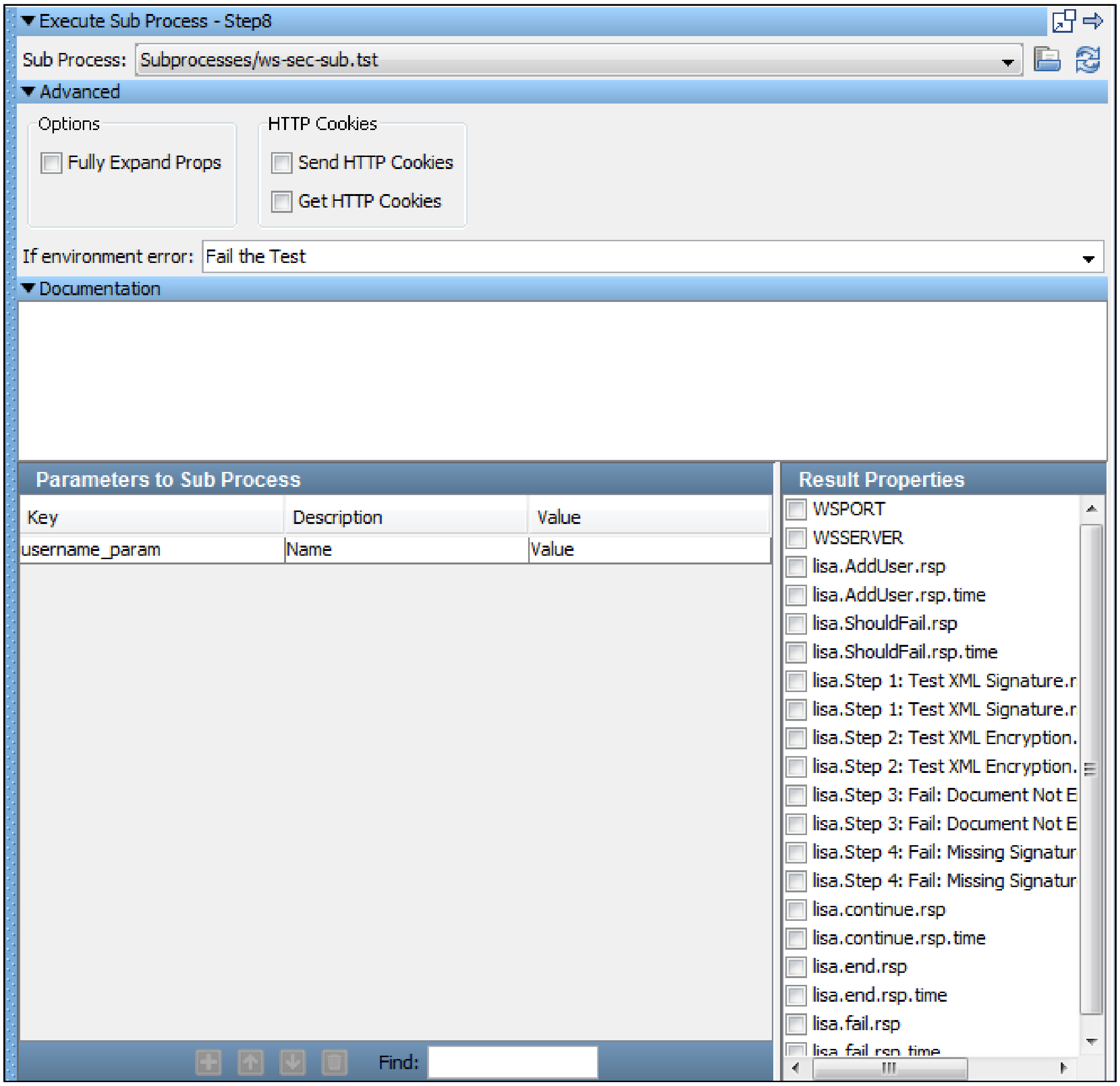
The Execute Subprocess step lets you execute a subprocess as a single step.

This step is used to execute a LISA subprocess and receive the outputs. This is commonly used when a certain function is performed in numerous test cases.

For example, a particular validation may always work the same way, so a subprocess is created to perform the validation and is added to different test cases.

For more information about subprocesses, and how to create them, see Building Subprocesses.

Parameter Requirements: Knowledge of the input and output requirements needed to execute the subprocess.



Execute Sub Process - Step8

Sub Process: Subprocesses/ws-sec-sub.tst

Advanced

Options

□ Fully Expand Props

HTTP Cookies

□ Send HTTP Cookies

□ Get HTTP Cookies

If environment error: Fail the Test

Documentation

Parameters to Sub Process

Key	Description	Value
username_param	Name	Value

Find:

Result Properties

□ WSPORT
□ WSSERVER
□ lisa.AddUser.rsp
□ lisa.AddUser.rsp.time
□ lisa.ShouldFail.rsp
□ lisa.ShouldFail.rsp.time
□ lisa.Step 1: Test XML Signature.r
□ lisa.Step 1: Test XML Signature.r
□ lisa.Step 2: Test XML Encryption.
□ lisa.Step 2: Test XML Encryption.
□ lisa.Step 3: Fail: Document Not E
□ lisa.Step 3: Fail: Document Not E
□ lisa.Step 4: Fail: Missing Signatur
□ lisa.Step 4: Fail: Missing Signatur
□ lisa.continue.rsp
□ lisa.continue.rsp.time
□ lisa.end.rsp
□ lisa.end.rsp.time
□ lisa.fail.rsp
□ lisa.fail.rsp.time

Enter the following parameters:

- **Subprocess**: Enter the name, select from the pull-down menu, or browse to the LISA test case that is the subprocess.
- **Open**: Use the Open button to open the subprocess test case. The subprocess test case is opened under a new project.

Options

- **Fully Expand Props**: When parameters have nested properties, fully expand the properties before sending to the subprocess.

HTTP Cookies

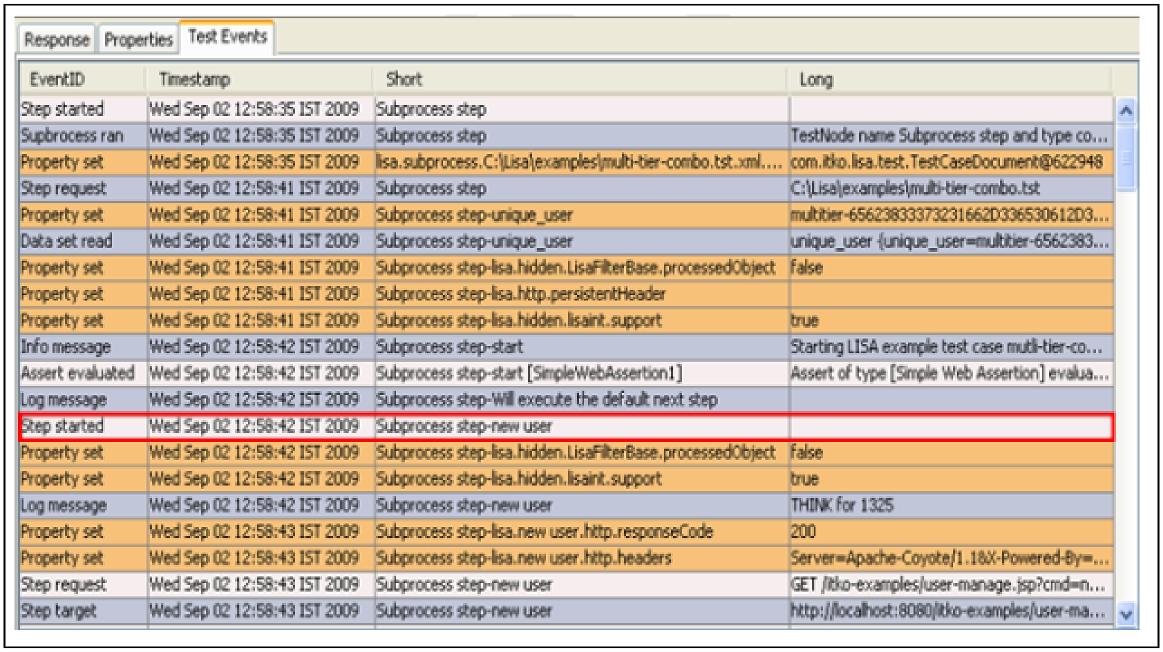
- **Send HTTP Cookies**: Select if you want to forward cookies to the subprocess.
- **Get HTTP Cookies**: Get HTTP cookies from the subprocess.
- **If environment error**: Select the step to redirect to if the subprocess fails.

In the Parameters to Subprocess panel there is a list of the parameters required by the subprocess. These keys and values must be present in your current test case. Edit the values column as needed to supply the correct values.

In the middle of the **Subprocess** information screen, LISA displays the documentation that was entered into the Documentation of the subprocess test case.

LISA also lists all the properties produced in the subprocess in Result Properties. Select the properties you want returned from the subprocess. These will be used in your test case. You are not limited to a single return value.

When this step is executed, it will appear to run as a single step. In the Interactive Test Run (ITR) you will be able to see the events fired during the execution of the step. The short name of the events will be a combination of the name of the current step and the name of the subprocess step where the event was fired.



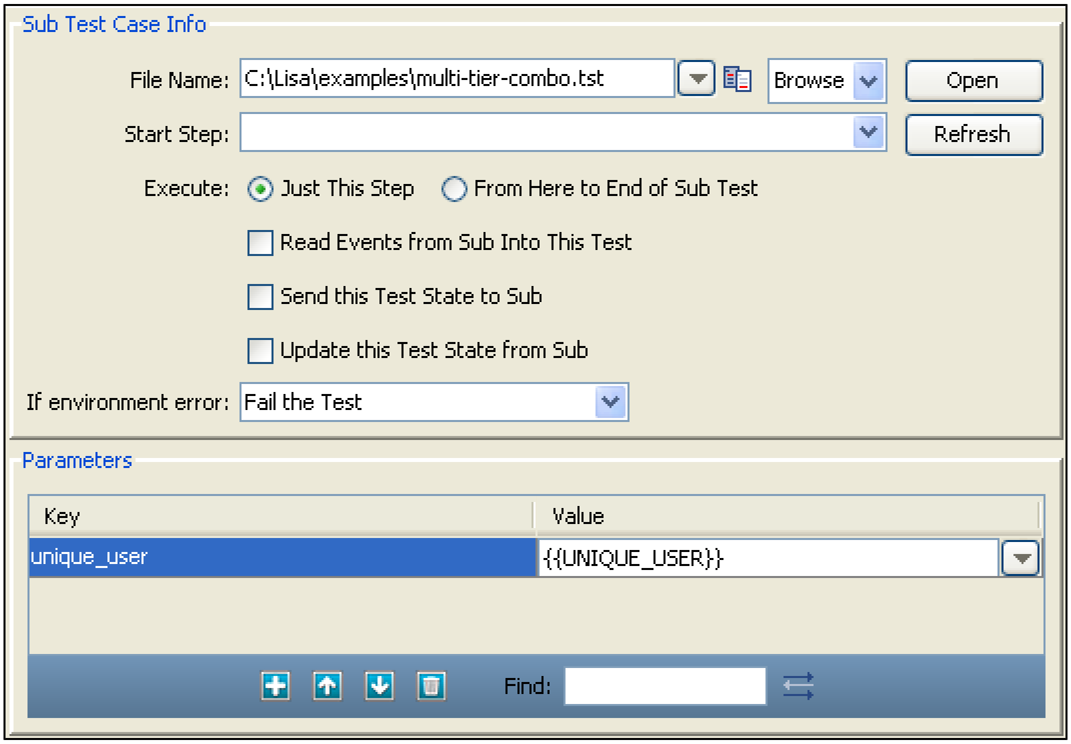
EventID	Timestamp	Short	Long
Step started	Wed Sep 02 12:58:35 IST 2009	Subprocess step	
Supbrocess ran	Wed Sep 02 12:58:35 IST 2009	Subprocess step	TestNode name Subprocess step and type co...
Property set	Wed Sep 02 12:58:35 IST 2009	lisa.subprocess.C:\Lisa\examples\multi-tier-combo.tst.xml....	com.itko.lisa.test.TestCaseDocument@622948
Step request	Wed Sep 02 12:58:41 IST 2009	Subprocess step	C:\Lisa\examples\multi-tier-combo.tst
Property set	Wed Sep 02 12:58:41 IST 2009	Subprocess step-unique_user	multitier-65623833373231662D336530612D3...
Data set read	Wed Sep 02 12:58:41 IST 2009	Subprocess step-unique_user	unique_user {unique_user=multitier-6562383...
Property set	Wed Sep 02 12:58:41 IST 2009	Subprocess step-lisa.hidden.LisaFilterBase.processedObject	false
Property set	Wed Sep 02 12:58:41 IST 2009	Subprocess step-lisa.http.persistentHeader	
Property set	Wed Sep 02 12:58:41 IST 2009	Subprocess step-lisa.hidden.lisaint.support	true
Info message	Wed Sep 02 12:58:42 IST 2009	Subprocess step-start	Starting LISA example test case mutli-tier-co...
Assert evaluated	Wed Sep 02 12:58:42 IST 2009	Subprocess step-start [SimpleWebAssertion1]	Assert of type [Simple Web Assertion] evalua...
Log message	Wed Sep 02 12:58:42 IST 2009	Subprocess step-Will execute the default next step	
Step started	Wed Sep 02 12:58:42 IST 2009	Subprocess step-new user	
Property set	Wed Sep 02 12:58:42 IST 2009	Subprocess step-lisa.hidden.LisaFilterBase.processedObject	false
Property set	Wed Sep 02 12:58:42 IST 2009	Subprocess step-lisa.hidden.lisaint.support	true
Log message	Wed Sep 02 12:58:42 IST 2009	Subprocess step-new user	THINK for 1325
Property set	Wed Sep 02 12:58:43 IST 2009	Subprocess step-lisa.new user.http.responseCode	200
Property set	Wed Sep 02 12:58:43 IST 2009	Subprocess step-lisa.new user.http.headers	Server=Apache-Coyote/1.1&X-Powered-By=...
Step request	Wed Sep 02 12:58:43 IST 2009	Subprocess step-new user	GET /itko-examples/user-manage.jsp?cmd=n...
Step target	Wed Sep 02 12:58:43 IST 2009	Subprocess step-new user	http://localhost:8080/itko-examples/user-ma...

Notice the new user notation in the **Short** column.

JUnit Test Case_Suite

The Execute JUnit Test Case/Suite step lets you run a JUnit test case or a JUnit test suite in a LISA step. If the JUnit test passes, then so does the LISA step. On failure you can redirect to another LISA step.

Prerequisite: Your JUnit test must be on the classpath. Drop it into the hot deploy directory.



Sub Test Case Info

File Name: C:\Lisa\examples\multi-tier-combo.tst

Start Step:

Execute: ● Just This Step ○ From Here to End of Sub Test

□ Read Events from Sub Into This Test

□ Send this Test State to Sub

□ Update this Test State from Sub

If environment error: Fail the Test

Parameters

Key	Value
unique_user	{{UNIQUE_USER}}

Enter the following parameters:

- **Test Class**: Enter the package name of the JUnit test class or test suite class. You can browse the classpath using the browse button. This will also confirm that your class is on your classpath.
- **If environment error**: Select the step to redirect to if the JUnit test fails.

Click the Load button to load the class files. The class tree is displayed in the left panel.

Click the Execute button to execute the JUnit test. The standard JUnit results are displayed in the right panel.

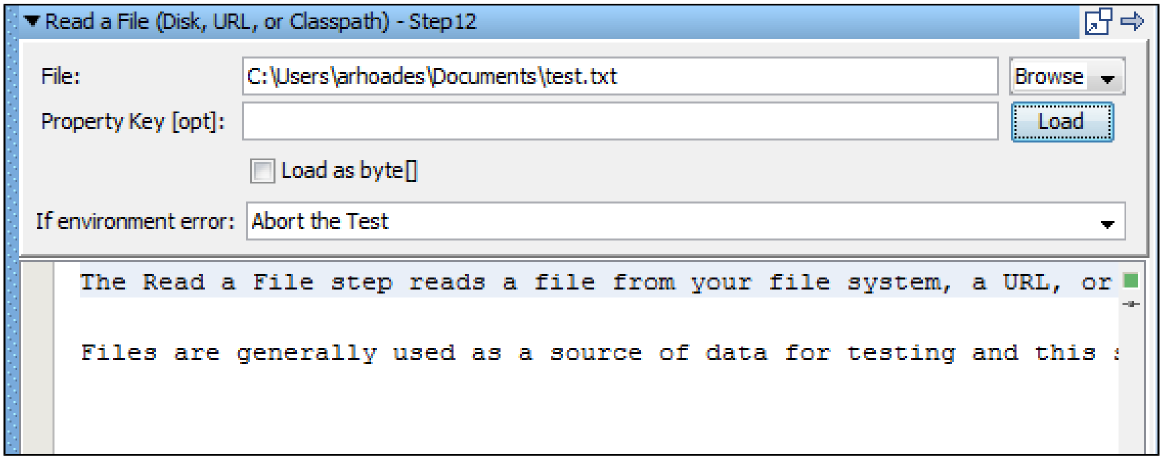
The previous illustration shows the editor after the test case has been executed. The text in the right panel is set as the response for this step and is available for you to add filters and assertions.

Read a File (Disk, URL or Classpath)

The Read a File step reads a file from your file system, a URL, or the classpath.

Files are generally used as a source of data for testing and this step can be paired with the Load a set of File Names data set to provide source data for testing.

You can read a text file or a binary file. The contents of the file can optionally be stored in a LISA property. In previous versions of LISA, this step was known as "Read Result from Stream".



Read a File (Disk, URL, or Classpath) - Step12

File: C:\Users\arhoades\Documents\test.txt

Property Key [opt]:

□ Load as byte[]

If environment error: Abort the Test

```
The Read a File step reads a file from your file system, a URL, or

Files are generally used as a source of data for testing and this s
```

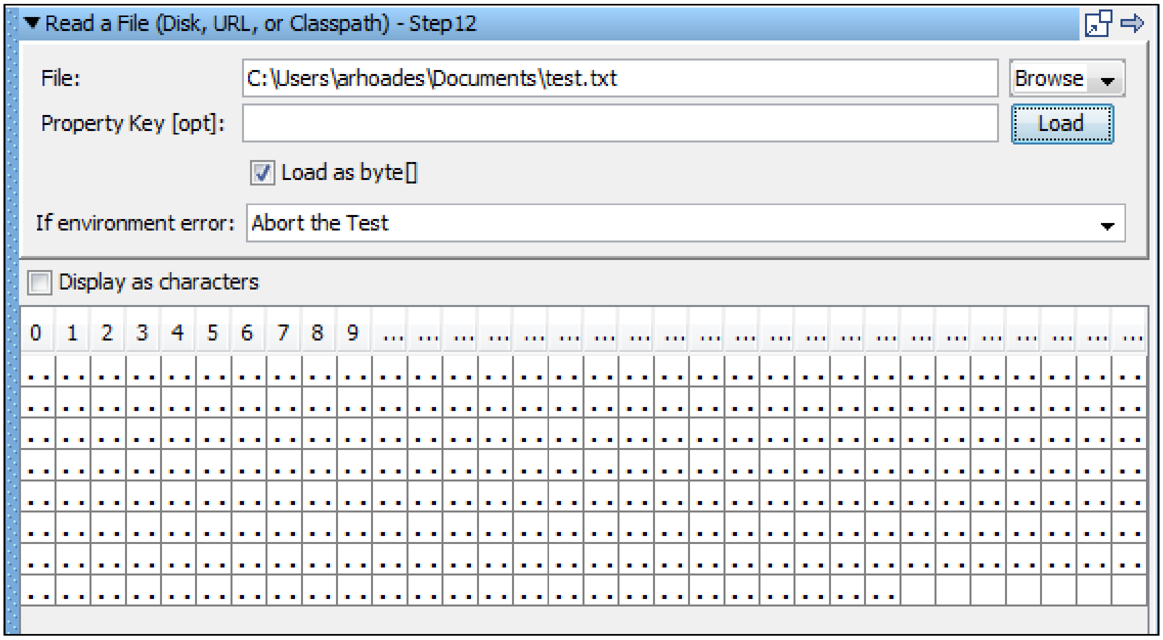
Enter the following parameters:

- **File**: Enter the path name, a URL, or a class path, or, browse to the file using the Browse button.
- **Property Key**: Enter the name of the property to store the file contents (optional).
- **Load as Byte []**: Select this check box to load contents as a byte array. (This is useful when loading a file to be used as a binaryData type in a web service execution parameter).
- **If environment error**: Select the step to redirect to if the Read a File test fails.
- **Display as characters**: The contents will be displayed as hexadecimal encoded bytes unless you select this check box. This check box is only visible if the Load as Byte [] box is selected.

Click the Load button to load and display the file. The content is now ready for you to filter and add assertions.

The following illustrations show a small binary file displayed as bytes and characters respectively.

Display as Bytes:



▼ Read a File (Disk, URL, or Classpath) - Step12

File: C:\Users\arhoades\Documents\test.txt

Property Key [opt]:

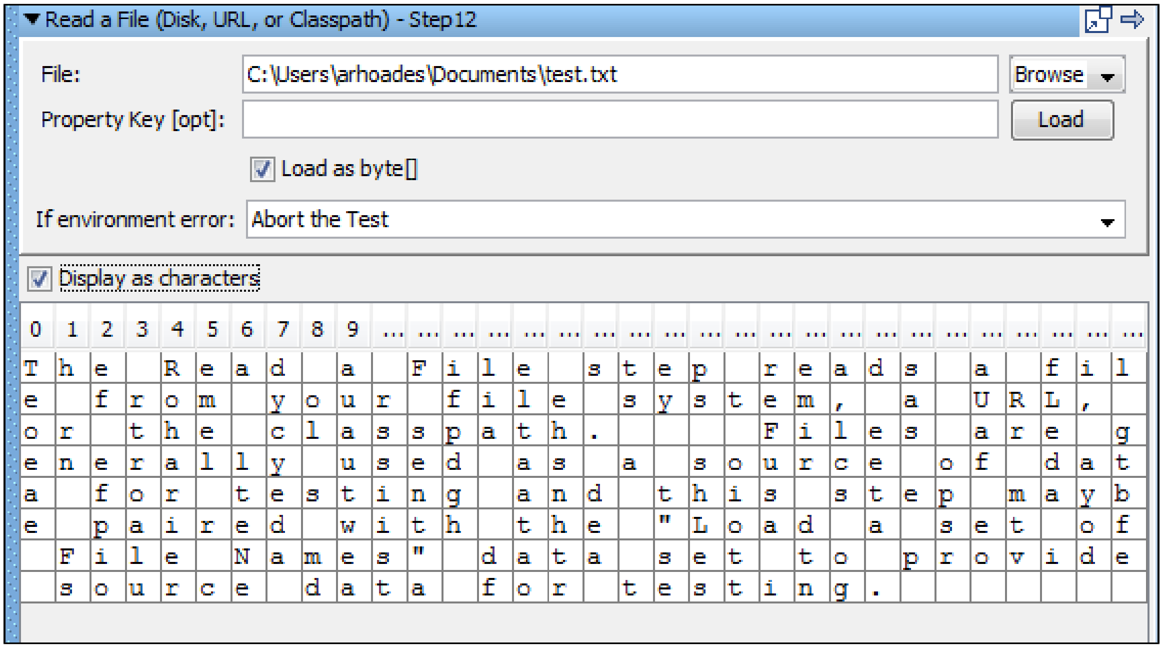
☑ Load as byte[]

If environment error: Abort the Test

☐ Display as characters

0	1	2	3	4	5	6	7	8	9	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..
..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..
..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..
..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..
..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..
..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..
..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..
..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..							

Display as Characters:



▼ Read a File (Disk, URL, or Classpath) - Step12

File: C:\Users\arhoades\Documents\test.txt

Property Key [opt]:

☑ Load as byte[]

If environment error: Abort the Test

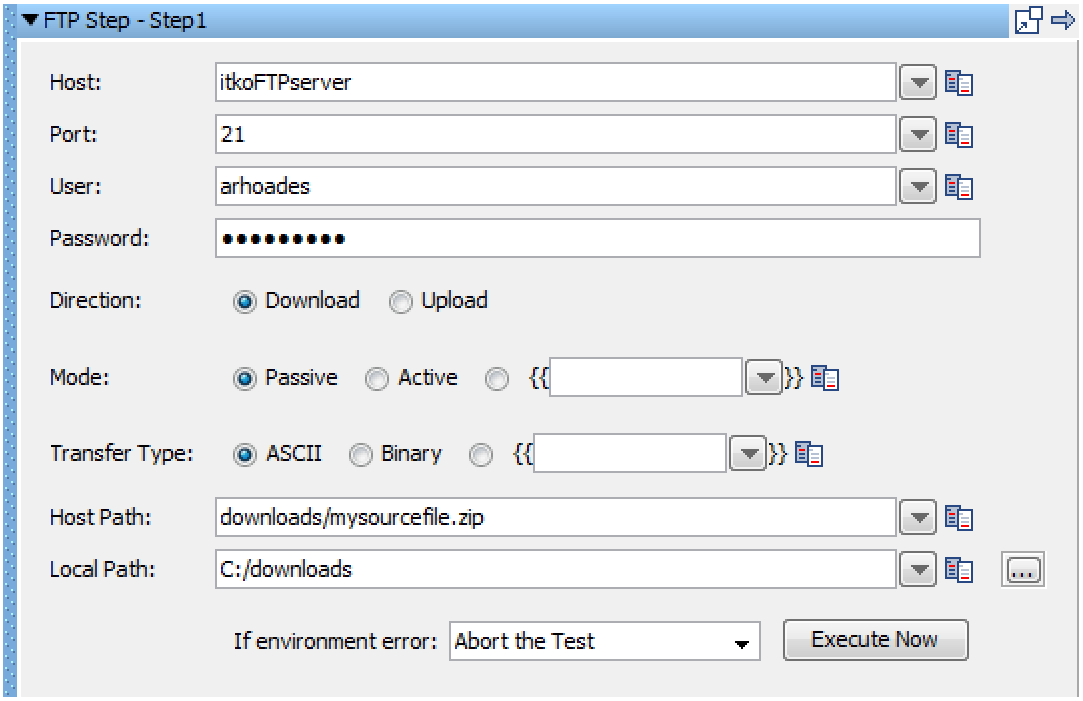
☑ Display as characters

0	1	2	3	4	5	6	7	8	9	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
T	h	e		R	e	a	d		a		F	i	l	e		s	t	e	p		r	e	a	d	s		a		f	i	l
e		f	r	o	m		y	o	u	r		f	i	l	e		s	y	s	t	e	m	,		a		U	R	L	,	
o	r		t	h	e		c	l	a	s	s	p	a	t	h	.					F	i	l	e	s		a	r	e		g
e	n	e	r	a	l	l	y		u	s	e	d		a	s		a		s	o	u	r	c	e		o	f		d	a	t
a		f	o	r		t	e	s	t	i	n	g		a	n	d		t	h	i	s		s	t	e	p		m	a	y	b
e		p	a	i	r	e	d		w	i	t	h		t	h	e		"	L	o	a	d		a		s	e	t		o	f
	F	i	l	e		N	a	m	e	s	"		d	a	t	a		s	e	t		t	o		p	r	o	v	i	d	e
	s	o	u	r	c	e		d	a	t	a		f	o	r		t	e	s	t	i	n	g	.							

If a binary file is loaded but you do not choose to load as byte, LISA will convert the binary data into characters, many of which will be unreadable, and the step response will be a string.

External - FTP Step

The FTP step lets you send or receive a file using FTP protocol. After entering the FTP information, user name, and user password you can either upload a file or download a file.



FTP Step - Step1

Host: itkoFTPserver
Port: 21
User: arhoades
Password: •••••••••
Direction: ● Download ○ Upload
Mode: ● Passive ○ Active ○ {{ }}
Transfer Type: ● ASCII ○ Binary ○ {{ }}
Host Path: downloads/mysourcefile.zip
Local Path: C:/downloads
If environment error: Abort the Test

On the FTP step editor screen, you may need to drag the window to the left to display the Execute Now button. Enter the following parameters:

- **Host**: Enter the host name of the FTP server (without the protocol).
- **Port**: Enter the port for FTP server access. A port is optional; the default port is 21.
- **User**: Enter the user id for FTP server access.
- **Password**: Enter the password for FTP server access.
- **Direction:** Indicate if the data flow will be an upload or a download.
- **Mode**: Specify passive or active FTP, or enter a LISA property that indicates passive or active.
- **Transfer Type**: Select the file transfer type; Binary or ASCII.
- **Host Path**: Enter the path to the source file (either on the FTP server or local computer).
- **Local Path**: Enter the path for the destination file (either on the FTP server or local computer).
- **If environment error**: Select the step to redirect to in the advent of an error.

Click the Execute Now button to initiate the send/receive action.

The response from a download is the file itself. The response from a successful upload is the string *success.*

If a file by the same name already exists, it will be overwritten without a warning message.

JMS Messaging Steps

The following steps are available in this chapter.

- JMS Messaging (JNDI)
- JMS Messaging - Message Consumer

JMS Messaging (JNDI)

The JMS Messaging (JNDI) step lets you send messages to, and receive messages from, topics and queues. You can also receive, modify, and forward an existing message. The list of possible queues and topics can be browsed using JNDI. You must provide client libraries where they can be read by LISA.

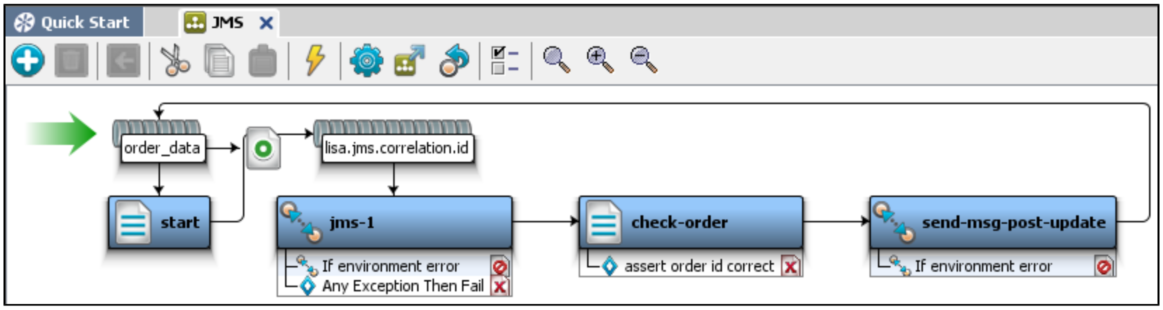
All the common message types including Empty, Text, Object, Bytes, Message, and Mapped (Extended) are supported.

The JMS Messaging (JNDI) step is configured regardless of the messaging requirements. Input options will vary on the messaging requirements. The editor will only allow valid configurations, so when you enable certain features, others may become inactive.

Prerequisites: You must supply the necessary JAR files for your chosen configuration and connection. They will typically need to be provided in the **LISA_HOME/hotDeploy** directory.

Parameter Requirements: You need the connection parameters and the queue or topic names used in the application under test. Other parameters may be required, depending on your environment. Get these from the application developers. In most cases, you can browse for server resources to get some of these needed parameters.

Review the **jms.tst** test case in the examples project to illustrate what is being described in this section.

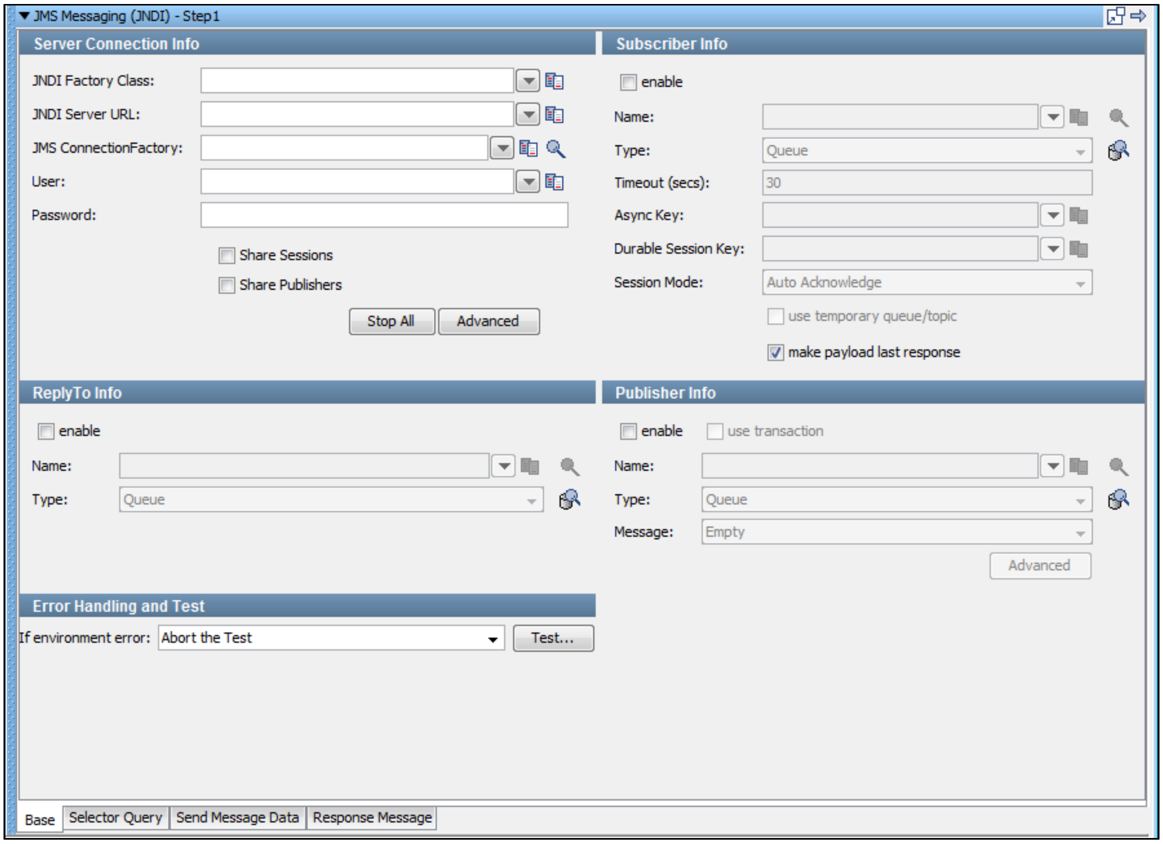


The **jms.tst** test case uses a publish/subscribe JMS step to send a message and listen on a temporary queue. The message is handled by a Message Driven Bean (MDB) on the server, which drops the message onto the temporary queue. The message type is text. It is an XML payload that is created by inserting LISA properties dynamically into the XML elements. The properties are read from the **order_data** data set. After the response message is received, the XML from the JMS message is put into a LISA property. The next step does an assertion validating the order ID. After this check asserts true, the existing message object is modified, and the message is sent to another JMS destination.

The **jms.tst** test case shows how LISA can listen in and intercept messages as they move through a multi-point messaging service backbone. You can run this test case against the JBoss Demo Server on your computer. The application backend is available there.

The editor for the JMS Messaging (JNDI) step contains the following tabs:

- The **Base** tab is where you define the connection and messaging parameters.
- The **Selector Query** tab lets you specify a selector query to be run when listening for a message on a queue.
- The **Send Message Data** tab is where you create the message content.
- The **Response Message** tab is where the response messages are posted.



Server Connection Info:
□ Share Sessions
□ Share Publishers

Subscriber Info:
□ enable
Type: Queue
Timeout (secs): 30
Session Mode: Auto Acknowledge
□ use temporary queue/topic
☑ make payload last response

ReplyTo Info:
□ enable
Type: Queue

Publisher Info:
□ enable
□ use transaction
Type: Queue
Message: Empty

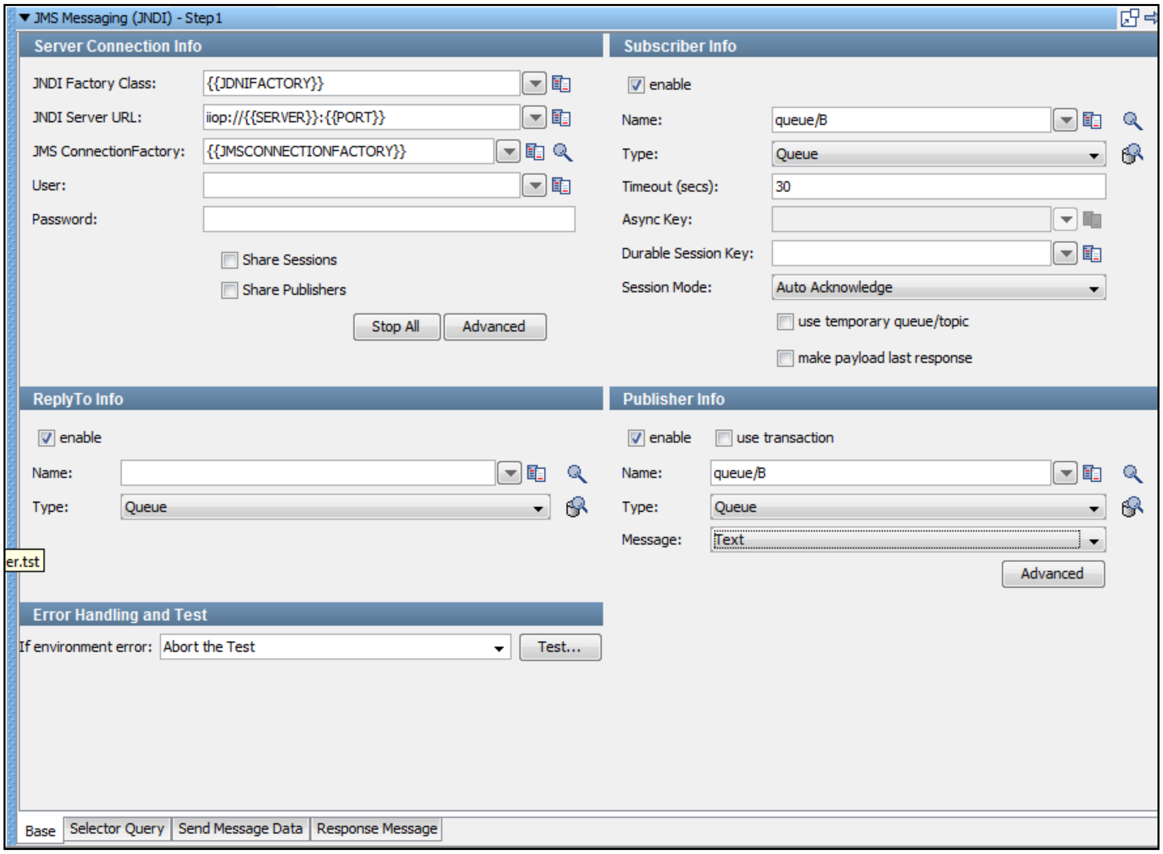
Error Handling and Test:
If environment error: Abort the Test

JMS Messaging: Base Tab

The Base tab is where you define the connection and messaging parameters.

The following image shows the Base tab. The tab is divided into the following sections:

- Server Connection Info
- Subscriber Info
- ReplyTo Info
- Publisher Info
- Error Handling and Test



▼ JMS Messaging (JNDI) - Step1

Server Connection Info

JNDI Factory Class: {{JDNIFACTORY}}

JNDI Server URL: iiop://{{SERVER}}:{{PORT}}

JMS ConnectionFactory: {{JMSCONNECTIONFACTORY}}

User:

Password:

□ Share Sessions

□ Share Publishers

Subscriber Info

☑ enable

Name: queue/B

Type: Queue

Timeout (secs): 30

Async Key:

Durable Session Key:

Session Mode: Auto Acknowledge

□ use temporary queue/topic

□ make payload last response

ReplyTo Info

☑ enable

Name:

Type: Queue

Publisher Info

☑ enable □ use transaction

Name: queue/B

Type: Queue

Message: Text

Error Handling and Test

If environment error: Abort the Test

You can enable and disable the Subscriber Info, Publisher Info, and ReplyTo Info sections by using the enable check box in the top left corner of each section. Thus, you can configure the step to be a publish step, a subscribe step, or both. You can also choose to include a JMS reply to component in the step.

When you finish configuring the test step, use the Test button in the Error Handling and Test section to test the configuration settings.

Server Connection Info

Enter the JNDI information in the Server Connection Info section of the Base tab.

These values should be parameterized with properties that are in your configuration, making it easy to change the application under test. The preceding image shows an example of this approach.

The following parameters should be available to you for the system under test:

- **JNDI Factory Class**
- **JNDI Server URL**
- **JMS Connection Factory**: Use the Search icon 🔍 to browse available resources on the server. Select or enter a connection factory per the JMS specification to use for this step execution.

The pull-down menus contain common examples or templates for these values.

The user and password may or may not be needed.

- **User**: The user name for connecting to the JNDI provider and getting a handle to the connection factory.
- **Password**: The password for connecting to the JNDI provider and getting a handle to the connection factory.
- **Share Sessions** and **Share Publishers**: These check boxes are used to share JMS sessions and publishers throughout the test case. This approach can lower overhead, but does not always provide a realistic simulation because typically JMS clients want to release resources. If you select the Share Publishers check box, the Share Sessions check box is also selected because you cannot share publishers without sharing sessions.
- **Stop All**: Lets you stop any listeners at design time now. This is to resolve issues during test case creation where some listeners can get orphaned, but will still consume messages making it difficult or impossible to finish test case creation.
- **Advanced**: Displays a panel where you can add custom properties that will be sent with the connection information, and configure second-level authentication.

> ⚠ The user and password fields in the main Server Connection Info section are for connecting to the JNDI provider and getting a handle to the connection factory. The user and password fields in the Second Level Authentication tab are for getting a handle to the actual JMS connection.

Publisher Info

Select the **enable** check box to set up the ability to send messages.

Select the **use transaction** check box to execute a commit when the message is sent.

Enter the following parameters:

- **Name**: The name of the topic or queue. Use the Search icon to browse the JNDI server for the topic or queue name.
- **Type**: Select whether you are using a topic or queue. Use the Browse icon to the right of this field to see what messages are waiting to be consumed from a queue (only).
- **Message**: Select the type of message you are sending. The supported types are Empty, Text, Object, Bytes, Message, and Mapped (Extended).
- **Advanced**: Displays a panel where you can edit the message headers and add message properties.

Subscriber Info

Select the **enable** check box to set up the ability to receive messages.

Enter the following parameters:

- **Name**: The name of the topic or queue. Use the Search icon to browse the JNDI server for the topic or queue name.
- **Type**: Select whether you are using a topic or queue, and whether you want to listen in synchronous or asynchronous mode. For asynchronous mode, you also must have an entry in the **Async Key** field. Use the Browse icon to the right of this field to see what messages are waiting to be consumed from a queue.
- **Timeout (secs**): The period to wait before there is an interrupt waiting for a message. Use 0 for no timeout.
- **Async Key**: The value needed to identify asynchronous messages. This field is needed only in asynchronous mode. It will be used in a subsequent Message Consumer step to retrieve asynchronous messages.
- **Durable Session Key**: By entering a name here, you are requesting a durable session. You are also providing a key for that session. A durable session lets you receive all of your messages from a topic even if you log off and then log on again.
- **Session Mode**: The options available are:
  - **Auto Acknowledge**: JMS Messages are automatically acknowledged by the JMS client libraries as soon as they are received.
  - **Client Acknowledge**: JMS Messages must be explicitly acknowledged by the JMS client.
  - **Use Transaction**: The JMS Session operates under a transaction. The acknowledge mode is ignored.
  - **Auto (Duplicates Okay)**: The JMS client library will automatically acknowledge at unknown intervals. As a result, duplicate messages may be received if the automatic acknowledgment does not arrive before the JMS Provider retries the delivery.

There is no practical difference between **Auto Acknowledge** and **Client Acknowledge** and **Auto (Duplicates Only)**. With **Client Acknowledge**, each received message is acknowledged immediately upon receipt. The only difference is that the acknowledge call is made explicitly instead of letting the JMS client library do it. With **Auto (Duplicates Only)**, the behavior is exactly the same as Auto Acknowledge except under very high loads.

The **Use Transaction** option is not strictly an Acknowledgment mode setting. It is included in the list for two reasons:

1. If the JMS Session is operating under a transaction, then the acknowledgment modes are ignored. Messages are essentially acknowledged by committing the session transaction.
2. The **Use Transaction** mode is still a way of controlling JMS's guaranteed delivery of messages. If a message is received and the session transaction is not committed, then the message will be resent, just like if it was not acknowledged.

- **Use temporary queue/topic**: Select this check box if you want the JMS provider to set up a temporary queue/topic on your behalf. When a temporary queue/topic is used, LISA automatically sets the **JMS ReplyTo** parameter of the message you send to the temporary queue/topic. The temporary queue/topic feature must always be used with a publisher so that a reply can be sent. If you use a temporary queue/topic, the **ReplyTo** section is disabled.
- **Make payload last response**: Select this check box if you want to make the payload response as a last response.

ReplyTo Info

Select the **enable** check box to set up a destination queue/topic.

If your application needs a destination, it is set up in this section.

Enter the following parameters:

- **Name**: The name of the topic or queue. Use the Search icon to browse the JNDI server for the topic or queue name.
- **Type**: Select whether you are using a topic or queue. Use the Browse icon to the right of this field to see what messages are waiting to be consumed from a queue (only).

Error Handling and Test

The Error Handling and Test section lets you redirect to a step if an error occurs.

- **If environment error**: Select the step to redirect to if an environment error occurs.

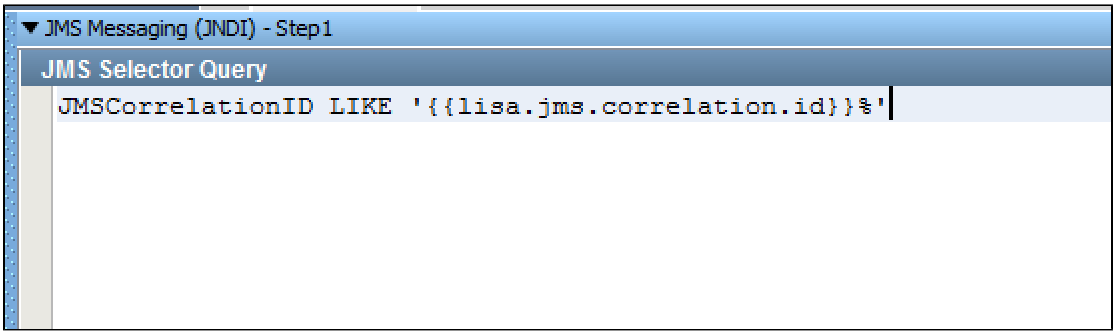
Click the Test button to test your step configuration settings.

JMS Messaging: Selector Query Tab

You can enter a JMS selector query in the Selector Query tab. The syntax closely follows SQL. It is a subset of SQL92.

A JMS selector query can be specified when listening for a message on a queue that is a response to a published message.

The following image shows a query looking for a **JMSCorrelationID** that matches the **lisa.jms.correlation.id** property as sent with the original message.



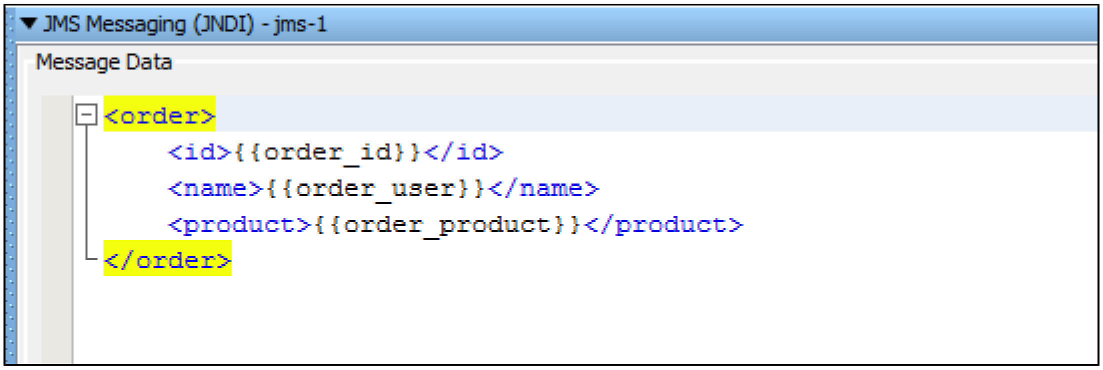
```
JMSCorrelationID LIKE '{{lisa.jms.correlation.id}}%'
```

JMS Messaging: Send Message Data Tab

If your step is configured to publish, the Send Message Data tab is where you compose the message.

The text can be typed in, or it can be read from a file using the Read Message From File button in the bottom right corner of the tab. You can also store text in a LISA property, in which case you would place the property in the editor, for example, **{{property_name}}**.

The following image shows an XML fragment with LISA properties. Using properties allows the message to be created dynamically during the test run.



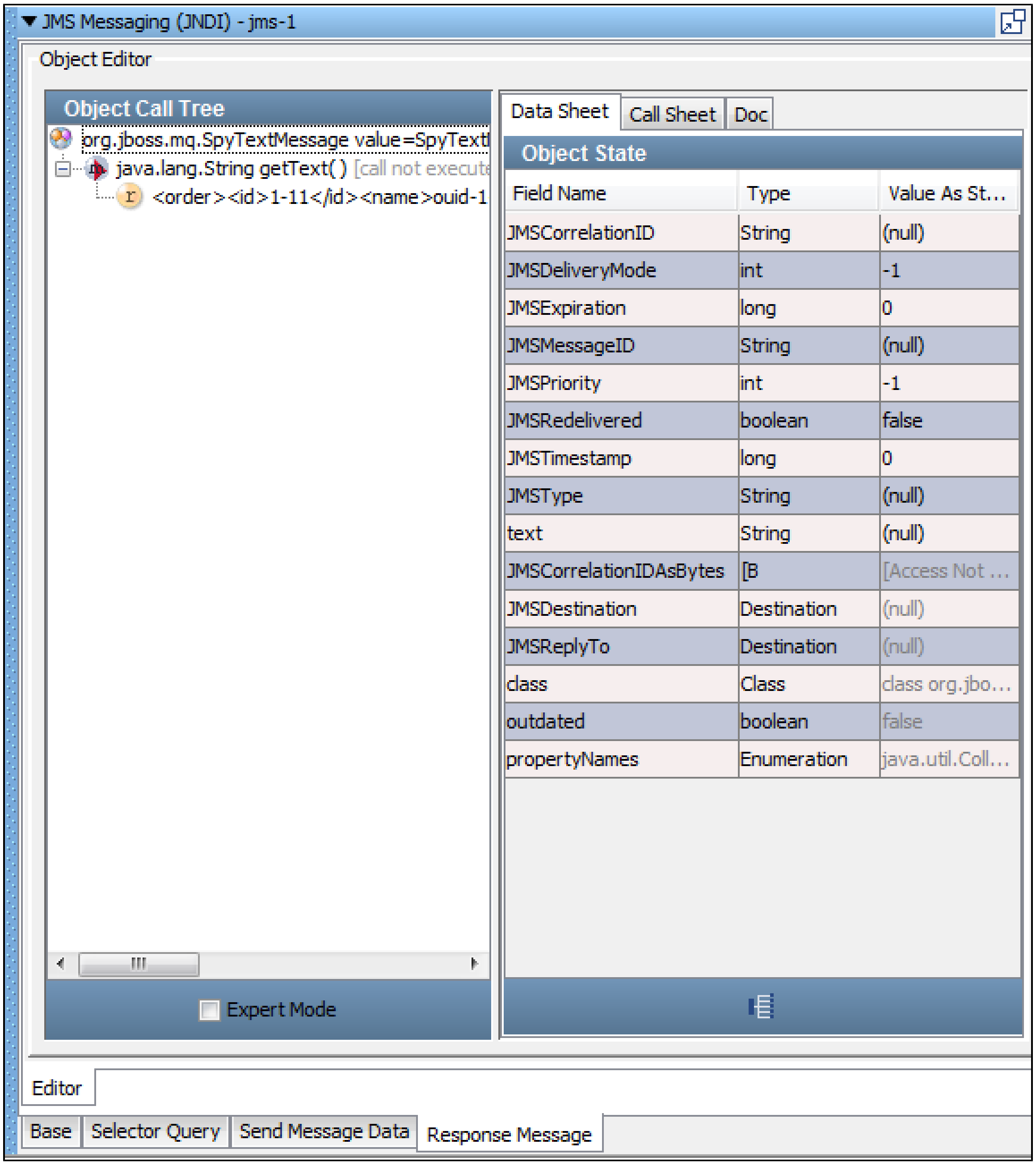
```
<order>
    <id>{{order_id}}</id>
    <name>{{order_user}}</name>
    <product>{{order_product}}</product>
</order>
```

JMS Messaging: Response Message Tab

If your step is configured to subscribe, the response appears in the Response Message tab after clicking the Test button in the Base tab.

The tab shows the Complex Object Editor for the returned object. The returned object varies with the type of application server. You have access to all the JMS parameters returned in addition to the message itself. The object is loaded into the Complex Object Editor, where it can be manipulated like any other Java object.

The following image shows a text response from a JBoss object.



org.jboss.mq.SpyTextMessage value=SpyTextl

java.lang.String getText( ) [call not execute

<order><id>1-11</id><name>ouid-1

Field Name	Type	Value As St...
JMSCorrelationID	String	(null)
JMSDeliveryMode	int	-1
JMSExpiration	long	0
JMSMessageID	String	(null)
JMSPriority	int	-1
JMSRedelivered	boolean	false
JMSTimestamp	long	0
JMSType	String	(null)
text	String	(null)
JMSCorrelationIDAsBytes	[B	[Access Not ...
JMSDestination	Destination	(null)
JMSReplyTo	Destination	(null)
class	Class	class org.jbo...
outdated	boolean	false
propertyNames	Enumeration	java.util.Coll...

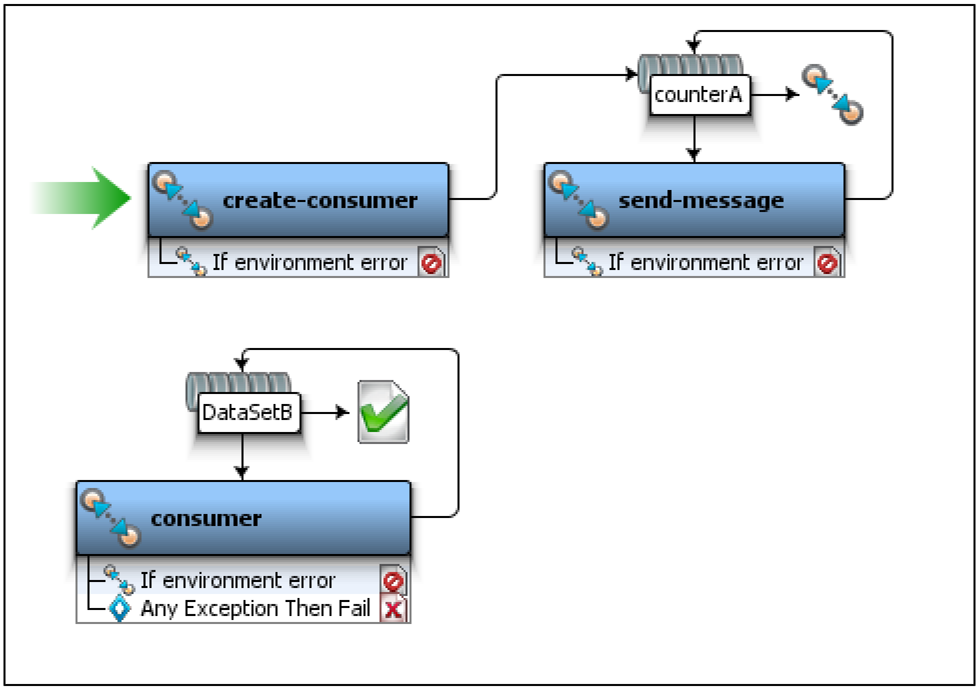
JMS Messaging - Message Consumer

The Message Consumer step lets you consume asynchronous messages in a test case. This step can attach to a known queue/topic and get messages posted for this subscriber. You identify yourself with a unique key. You must have already subscribed to the queue/topic and the messages have been pushed to that destination.

Prerequisite: Before executing the example test case you must have the demo server running.

Parameter Requirements: Knowledge of the queue/topic being used in the application under test.

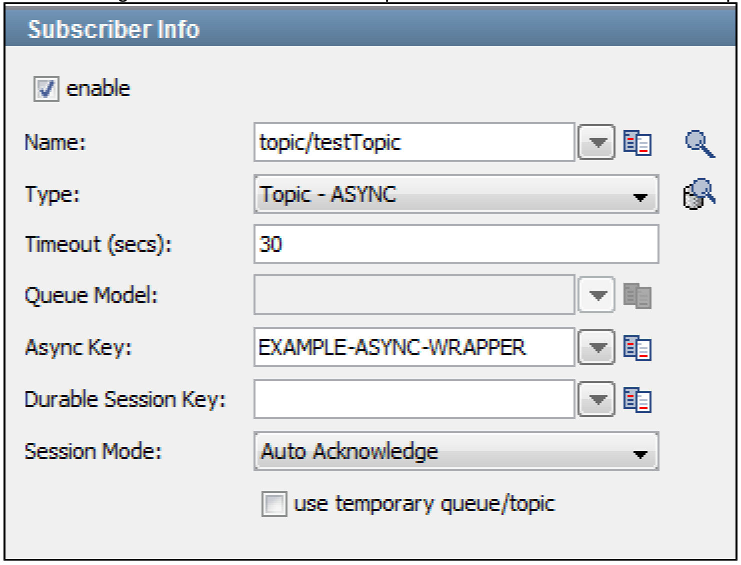
An example test case, **async-consumer-jms.tst**, is included in the LISA examples directory. Look at that test case to illustrate what is being described in this section.



The create-consumer step subscribes to asynchronous message (topic/testTopic) using the Async Key (EXAMPLE-ASYNC-WRAPPER). The send-message step publishes message to a queue (queue/C). The number of messages to be published is controlled using a data set (counterA). The message consumer step has an Async queue (EXAMPLE-ASYNC-WRAPPER) using which message subscribed by the create-consumer step is consumed. The number of messages to be consumed is controlled using the data set (DataSetB).

⚠ The Async Key specified in create-consumer and consumer steps must match.

The following illustration shows an example of the subscriber section of a step.



Subscriber Info

☑ enable

Name: topic/testTopic

Type: Topic - ASYNC

Timeout (secs): 30

Queue Model:

Async Key: EXAMPLE-ASYNC-WRAPPER

Durable Session Key:

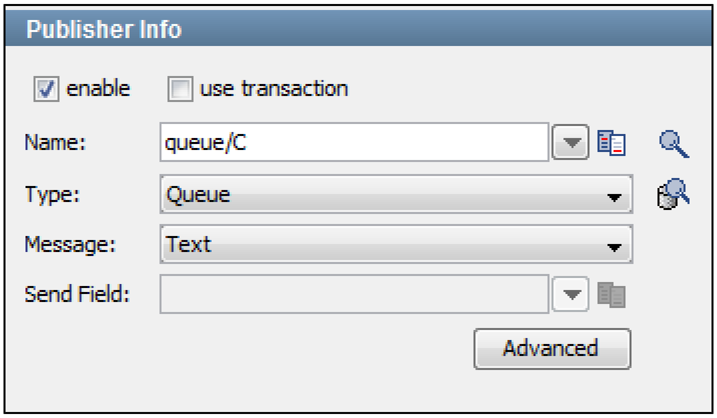
Session Mode: Auto Acknowledge

☐ use temporary queue/topic

Select the enable check box to set up and enable the ability to listen to (subscribe to) messages.

Notice that there is an asynchronous topic specified in the Type field, and an Async Key parameter defined. This key will be needed as an input in the current step.

This illustration shows an example of the publisher section of a step.



Publisher Info

☑ enable ☐ use transaction

Name: queue/C

Type: Queue

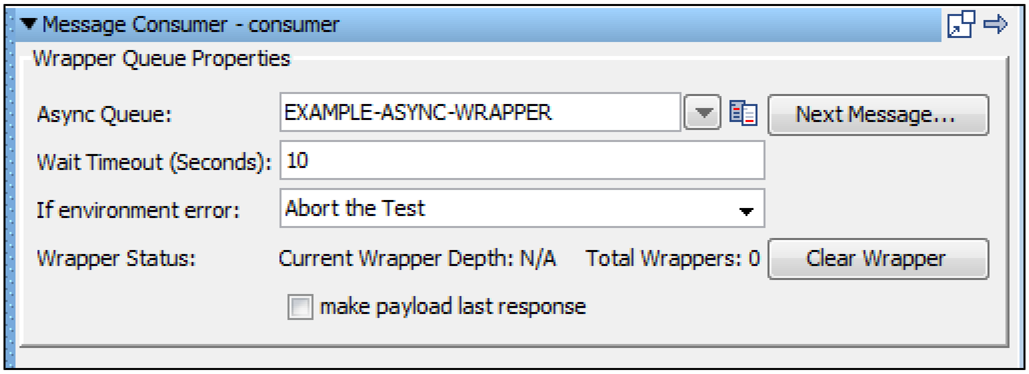
Message: Text

Send Field:

Select the enable check box to set up the ability to send (publish) messages. Select the use transaction check box to execute a commit when the message is sent.

Enter the following parameters:

- **Name**: Enter the name of the topic or queue to use. Use the Search icon 🔍 to browse the JNDI server for the topic or queue name.
- **Type**: Select whether you are using a topic or queue. Use the Browse icon to the right of this field to see what messages are waiting to be consumed from a queue (only).
- **Message**: Select the type of message you are sending from the pull-down menu. The supported messages are: Empty, Text, Object, Bytes, Message, and Mapped (Extended).
- **Advanced** button: Select to display a panel where you can add custom message properties to be sent with the message.



▼ Message Consumer - consumer

Wrapper Queue Properties

Async Queue: EXAMPLE-ASYNC-WRAPPER

Wait Timeout (Seconds): 10

If environment error: Abort the Test

Wrapper Status: Current Wrapper Depth: N/A Total Wrappers: 0

☐ make payload last response

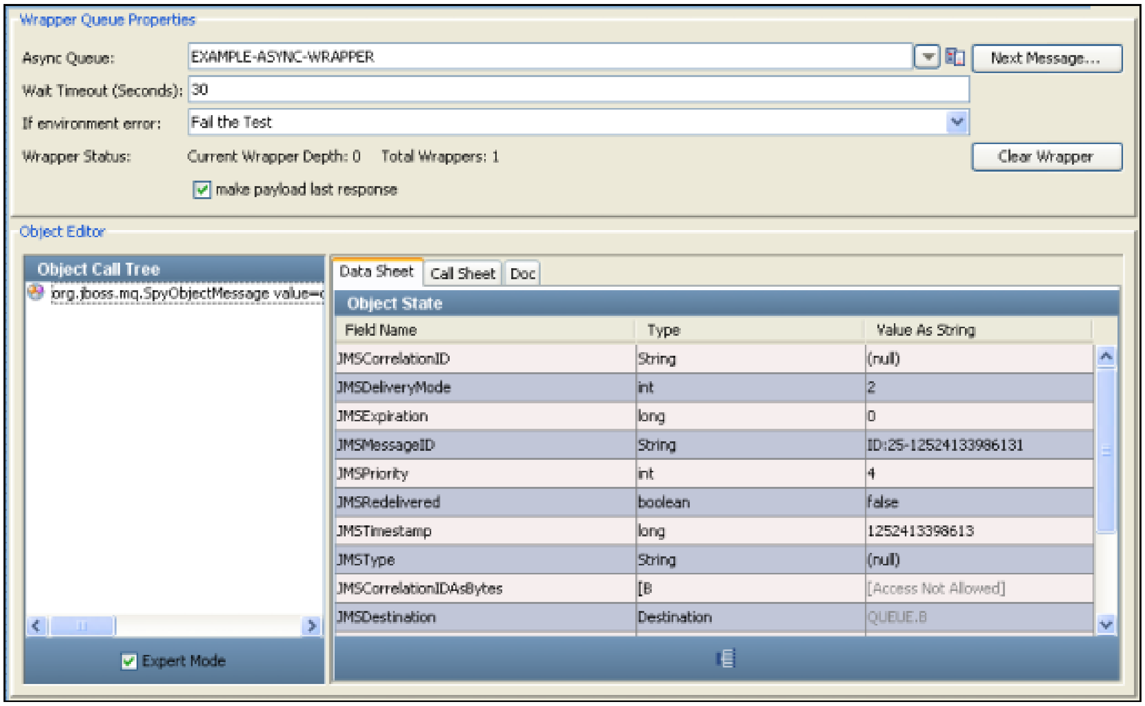
Enter the following parameters:

- **Async Queue**: Enter or select the Async Key parameter that was named in a preceding subscriber step (EXAMPLE-ASYNC-WRAPPER). These names must match.
- **Wait Timeout (Seconds)**: Enter the time, in seconds, to wait for the next message.
- **If environment error**: Select the step to redirect to if an environment error occurs.

Wrapper Status contains two output status values:

- **Current Wrapper Depth**: Number of messages left to be read in current wrapper
- **Total Wrappers**: Number of wrappers (destinations).
- **Make payload last response**: Select this option if you want to make the payload as the last response in the step.

If there are messages waiting, they can be read by clicking the Next Message button. This will typically show the message in the LISA Complex Object Editor.



Async Queue: EXAMPLE-ASYNC-WRAPPER

Wait Timeout (Seconds): 30

If environment error: Fail the Test

Wrapper Status: Current Wrapper Depth: 0 Total Wrappers: 1

☑ make payload last response

org.jboss.mq.SpyObjectMessage value=c

☑ Expert Mode

Field Name	Type	Value As String
JMSCorrelationID	String	(null)
JMSDeliveryMode	int	2
JMSExpiration	long	0
JMSMessageID	String	ID:25-12524133986131
JMSPriority	int	4
JMSRedelivered	boolean	false
JMSTimestamp	long	1252413398613
JMSType	String	(null)
JMSCorrelationIDAsBytes	[B	[Access Not Allowed]
JMSDestination	Destination	QUEUE.B

You are now ready to manipulate this object.

A wrapper is a FIFO list for holding responses from asynchronous topics and queues. It provides a place for the application to put responses for later consumption. Messages wait in this list for subsequent LISA processing (in this Message Consumer step).

BEA Steps

The following steps are included in this chapter.

WebLogic JMS (JNDI)

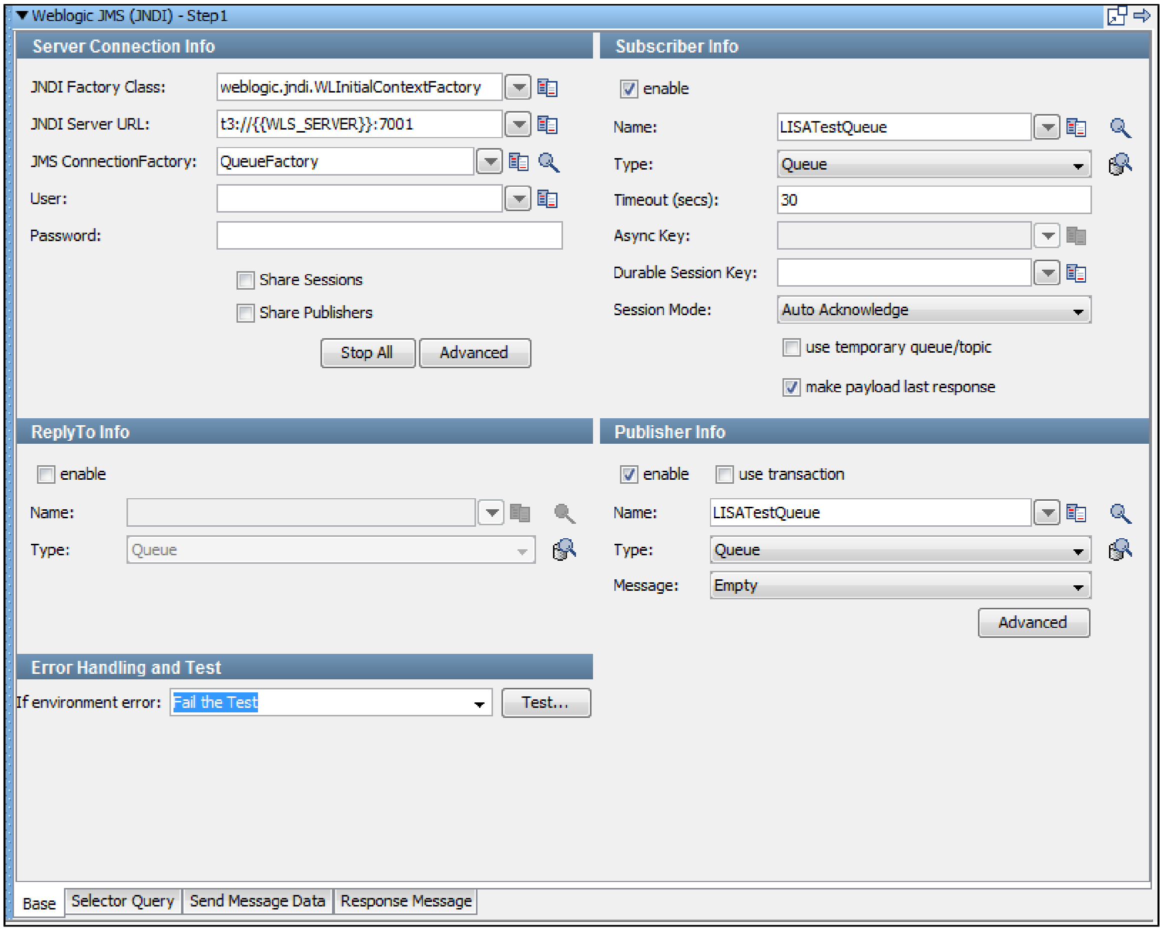
The WebLogic JMS (JNDI) step lets you send messages to, and receive messages from, topics and queues. You can also receive, modify and forward an existing message. WebLogic JMS (JNDI) supports all the common message types including Empty, Text, Object, Bytes, Message, and Mapped (Extended).

The WebLogic JMS (JNDI) step is configured using a single editor regardless of the messaging requirements. Input options will vary on the messaging requirements. The editor will only allow valid configurations, so when you enable certain features, others may become inactive.

Prerequisites: You are required to supply the necessary JAR files for your chosen configuration and connection.

WebLogic requires that the WebLogic JAR file (**weblogic.jar**) be added to your classpath. You can put it in the hot deploy directory. There may be other JAR files required if you are using security or JMX. See the WebLogic documentation for more information about the JAR files you might need. The JAR files can be found in the **lib** directory of the WebLogic installation.

Parameter Requirements: You need the connection parameters and the subject names used in the application under test. The following sections describe the parameters that you will need. There may be other parameters required, depending on your environment. Get these from the developers of the application.



Weblogic JMS (JNDI) - Step1

Server Connection Info

JNDI Factory Class: weblogic.jndi.WLInitialContextFactory

JNDI Server URL: t3://{{WLS_SERVER}}:7001

JMS ConnectionFactory: QueueFactory

User:

Password:

☐ Share Sessions

☐ Share Publishers

Subscriber Info

☑ enable

Name: LISATestQueue

Type: Queue

Timeout (secs): 30

Async Key:

Durable Session Key:

Session Mode: Auto Acknowledge

☐ use temporary queue/topic

☑ make payload last response

ReplyTo Info

☐ enable

Name:

Type: Queue

Publisher Info

☑ enable ☐ use transaction

Name: LISATestQueue

Type: Queue

Message: Empty

Error Handling and Test

If environment error: Fail the Test

There are four tabs available at the bottom of the editor.

- The Base tab is where you define your connection and messaging parameters.
- The Selector Query tab lets you specify a selector query to be run when listening for a message on a queue.
- The Send Message Data tab is where you create your message content.
- The Response Message tab is where your response messages will be posted.

Base Tab

The Base tab view is divided into five major sections:

- Server Connection Info
- Subscriber Info
- Publisher Info
- ReplyTo Info
- Error Handling and Test

The Server Connection Info, Error Handling and Test, Publisher Info are always active. The Subscriber Info and, ReplyTo Info sections can be enabled or disabled using the enable check box in the top left corner of each section.

Using these check boxes you can configure the step to be a publish step, a subscribe step, or both. You can also choose to include a replyto component in the step. When you have completely configured your test step, use the Test button in the Error Handling and Test section to test your configuration settings.

Server Connection Info

Enter the JNDI information.

These values should be parameterized with properties that are in your configuration, making it easy to change the application under test. By default the `WLS_SERVER` property in the JNDI Server URL is used. This property must be added to your configuration if you plan to use it.

The five parameters should be available to you for the system under test. The pull-down menus contain common examples or templates for these values.

- **JNDI Factory Class**: LISA pre-populates this field with the default values.

- **JNDI Server URL**: LISA pre-populates this field with the default values.
- **JMS Connection Factory**: Use the Search icon to browse available resources on the server. Select or type in a Connection Factory per the JMS specification to use for this step execution.
- **User**
- **Password**
- **Share Sessions** and **Share Publishers**: Use these check boxes to share JMS Sessions and Publishers throughout the test case. This can lower overhead, but does not always provide a realistic simulation because typically JMS clients want to release resources. If you check Share Publishers, the Share Sessions check box is also checked for you as you cannot share publishers without sharing sessions.
- The **Stop All** button lets you stop any listeners at design time now. This is to resolve issues during test case creation where some listeners can get orphaned, but will still consume messages making it difficult or sometimes impossible to finish test case creation.
- The **Advanced** button displays a panel where you can add custom properties that will be sent with the connection information.

Publisher Info

Select the **enable** check box to set up the ability to send (publish) messages. Click the **use transaction** check box to execute a commit when the message is sent.

Enter the following parameters:

- **Name**: The name of the topic or queue to use. Use the Search icon to browse the JNDI server for the topic or queue name.
- **Type**: Select whether you are using a topic or queue. Use the Browse icon to the right of this field to see what messages are waiting to be consumed from a queue (only).
- **Message**: Select the type of message you are sending. The supported types are Empty, Text, Object, Bytes, Message, and Mapped (Extended).
- **Advanced**: Displays a panel where you can edit the message headers and add message properties.

Subscriber Info

Select the enable check box to set up to enable the ability to receive (subscribe to) messages.

Enter the following parameters:

- **Name**: Enter the name of the topic or queue to use. Use the Search icon to browse the JNDI server for the topic or queue name.
- **Type**: Select whether you are using a topic or queue, and whether you want to listen in synchronous or asynchronous mode. For asynchronous mode you will also have to have an entry in the Async key field. Use the Browse icon to the right of this field to see what messages are waiting to be consumed from a queue (only).
- **Timeout (secs)**: Enter the period to wait before the application interrupts waiting for a message (this field can be left blank for no timeout).
- **Async Key**: Enter the value needed to identify asynchronous messages. This is only needed in asynchronous mode. It will be used in a subsequent Message Consumer step to retrieve asynchronous messages.
- **Durable Session** key: By entering a name here you are requesting a durable session. You are also providing a key for that session. A durable session lets you receive all of your messages from a topic even if you log off, and then log on again.
- **use transaction** check box: Select the **use transaction** check box to execute a Commit when a message is received.
- **use temporary queue/topic** check box: Select the **use temporary queue/topic** check box if you want the JMS.provider to set up a temporary queue/topic on your behalf. When a temporary queue/topic is used, the JMS **ReplyTo** parameter of the message you send to the temporary queue/topic is automatically set. The temporary queue/topic feature must always be used with a publisher so that a reply can be sent. If you use a temporary queue/topic the **ReplyTo** section is disabled.
- **make payload last response**: Select this option if you want to make payload as the response of this step.

ReplyTo Info

Select the **enable** check box to set up a destination queue/topic.

If your application needs a destination, it is set up in this section.

Enter the following parameters:

- **Name**: Enter the name of the topic or queue to use. Use the Search icon to browse the JNDI server for the topic or queue name.
- **Type**: Select whether you are using a topic or queue. Use the Browse icon to the right of this field to see what messages are waiting to be consumed from a queue (only).

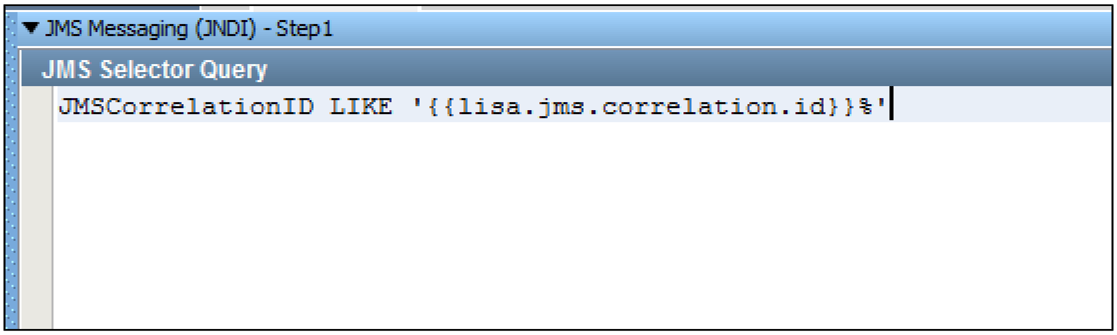
Error Handling and Test

Error Handling and Test lets you redirect to a step if an exception occurs.

- **If environment error**: Select the step to redirect to if an error occurs.

Click the Test button to test your step configuration settings.

Selector Query Tab



```
JMSCorrelationID LIKE '{{lisa.jms.correlation.id}}%'
```

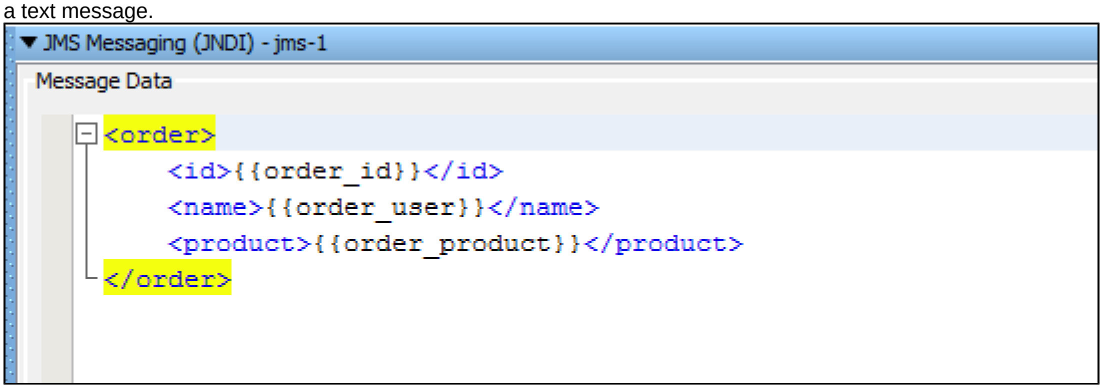
You can enter a JMS selector query in this editor. The syntax closely follows SQL. It is a subset of SQL92. A JMS selector query can be specified when listening for a message on a queue that is a response to a published message. The previous illustration shows a specific query looking for a **JMSCorrelationID** that matches one set in a LISA property as sent with the original message.

There is a built-in mechanism for allowing a test creator to set the **JMSCorrelationID** for a message before sending it. Any time before the message is sent you can set the correlation ID by setting the LISA property **lisa.jms.correlation.id**.

A non-zero value will be detected, and the message **JMSCorrelationID** property will be set before the message is sent.

Send Message Data Tab

If your step is configured to publish, this is where you compose your message. The Send Message Data tab view in the following example shows a text message.



```
<order>
    <id>{{order_id}}</id>
    <name>{{order_user}}</name>
    <product>{{order_product}}</product>
</order>
```

This particular example shows an XML fragment with LISA properties being used. The text can be typed in or it can be read from a file using the Read Message from the File button in the bottom-right corner, or it can be stored in a LISA property, in which case you would just place the LISA property in the editor: for example, **`LISA_PROP`**.

Notice that LISA properties are used in the message XML allowing the message to be created dynamically during the test run.

Response Message Tab

If your step is configured to subscribe, your response will be shown here. For more information, see JMS Messaging (JNDI).

Message Consumer

For detailed information about this step, see JMS Messaging - Message Consumer.

Read a File (Disk, URL, or Classpath)

For detailed information about this step, see Read a File (Disk, URL or Class Path).

Web Service Execution (XML)

For detailed information about this step, see Web Service Execution (XML) Step.

Raw Soap Request

For detailed information about this step, see Web_ Raw SOAP Request.

FTP Step

For detailed information about this step, see External - FTP Step.

Web Service Execution (Legacy)

For detailed information about this step, see Web_Web Service Execution (Legacy).

Sun JCAPS Steps

The following steps are included in this chapter.

JCAPS Messaging (Native)

The JCAPS Messaging (Native) step lets you send messages to, and receive messages, from topics and queues. You can also receive, modify, and forward an existing message.

JCAPS Messaging (Native) supports all the common message types including Empty, Text, Object, Bytes, Message, and Mapped (Extended).

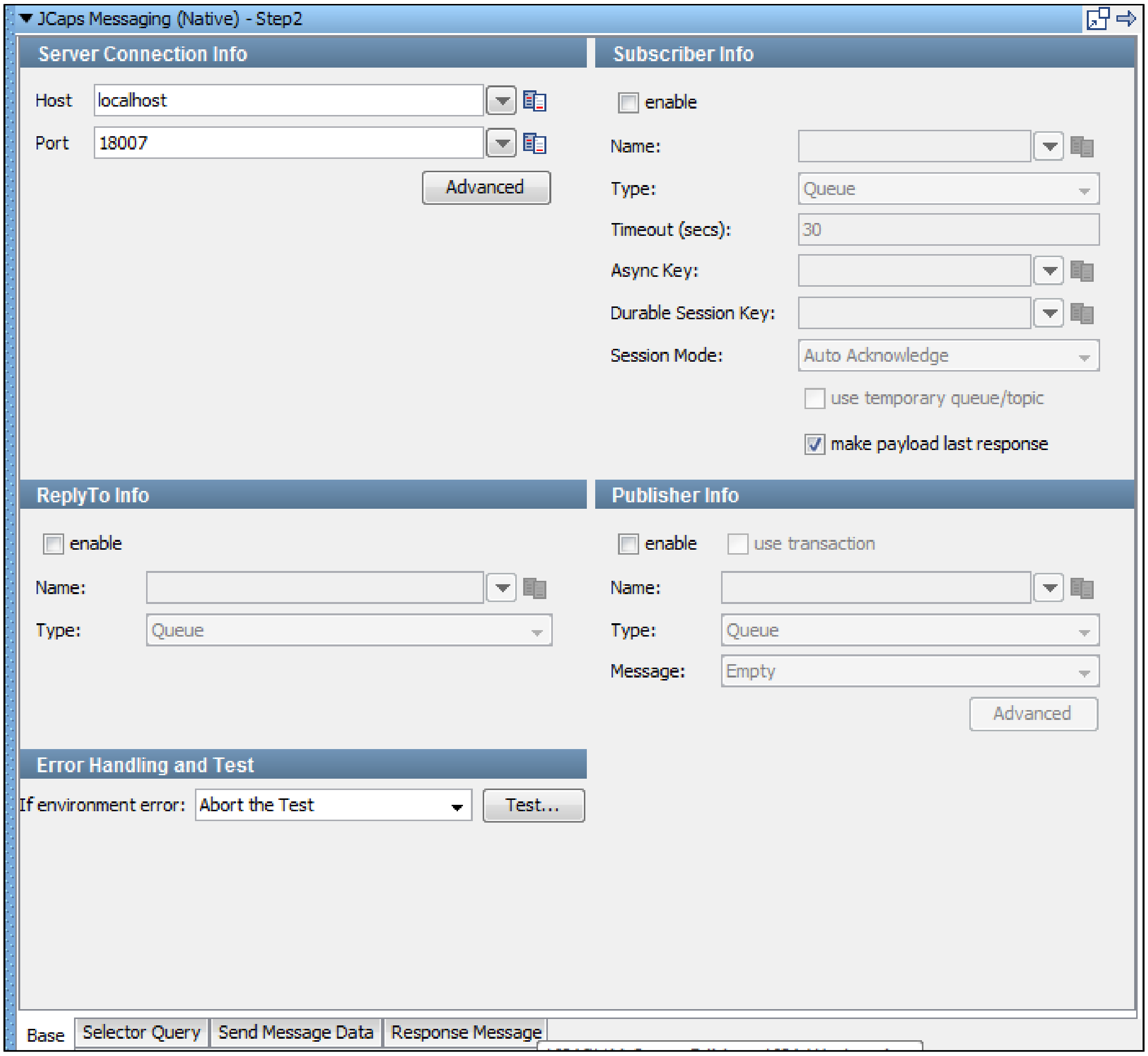
The JCAPS Messaging (Native) step is configured using a single editor regardless of the messaging requirements. Input options will vary on the messaging requirements. The editor will only allow valid configurations, so when you enable certain features others may become inactive.

Prerequisites: You are required to supply the necessary JAR files for your chosen configuration and connection.

JCAPS messaging requires that several JAR files be added to your CLASSPATH. You can put them in the hot deploy directory. Consult the JCAPS documentation for more information about the JAR files you might need. The JAR files can be found in the lib directory of the JCAPS installation.

Parameter Requirements: You need the connection parameters and the subject names used in the application under test. The following sections describe the parameters that you will need. There may be other parameters required, depending on your environment. Get these from the developers of the application.

The messaging step editor for JCAPS Messaging (Native) is used to configure this step.



JCaps Messaging (Native) - Step2

Server Connection Info

Host: localhost

Port: 18007

Subscriber Info

□ enable

Name:

Type: Queue

Timeout (secs): 30

Async Key:

Durable Session Key:

Session Mode: Auto Acknowledge

□ use temporary queue/topic

☑ make payload last response

ReplyTo Info

□ enable

Name:

Type: Queue

Publisher Info

□ enable □ use transaction

Name:

Type: Queue

Message: Empty

Error Handling and Test

If environment error: Abort the Test

There are four tabs available at the bottom of the editor.

- The Base tab is where you define your connection and messaging parameters.
- The Selector Query tab lets you specify a selector query to be run when listening for a message on a queue.
- The Send Message Data tab is where you create your message content.
- The Response Message tab is where your response messages will be posted.

Base Information

The Base tab is divided into five major sections:

- Server Connection Info
- Subscriber Info
- Publisher Info
- ReplyTo Info
- Error Handling and Test

The Server Connection Info and Error Handling and Test sections are always active. The Subscriber Info, Publisher Info, and, ReplyTo Info sections can be enabled or disabled using the enable check box in the top left corner of each section. Using these check boxes you can configure the step to be a publish step, a subscribe step, or both. You can also choose to include a replyto component in the step. When you have completely configured your test step, use the Test button in the Error Handling and Test section to test your configuration settings.

Server Connection Info

The Server Connection Info section displays two parameters available to you for the system under test.

- **Host**: The name of the JMS server.
- **Port**: The port number the JMS server is running on.

The Advanced button displays a panel where you can add custom properties that will be sent with the connection information.

All other tabs are defined in detail in JMS Messaging (JNDI).

JCAPS Messaging (JNDI)

The JCAPS Messaging (JNDI) step lets you send messages to, and receive messages from topics and queues. You can also receive, modify and forward an existing message.

JCAPS Messaging (JNDI) supports all the common message types including Empty, Text, Object, Bytes, Message, and Mapped (Extended).

The JCAPS Messaging (JNDI) step is configured using a single LISA editor regardless the messaging requirements. Input options will vary on the messaging requirements. The editor will only allow valid configurations, so when you enable certain features others may become inactive.

Prerequisites: You will be required to supply the necessary JAR files for your chosen configuration and connection. The **com.stc.jms.stcjms.jar** is required.

JCAPS messaging requires that several additional JAR files be added to your classpath. You can put them in the hot deploy directory. See the JCAPS documentation for more information about the JAR files you need. The JAR files can be found in the **lib** directory of the JCAPS installation.

Parameter Requirements: You need the connection parameters and the subject names used in the application under test. There may be other parameters required, depending on your environment. Get these from the developers of the application.

Detailed information about parameters and fields for JCAPS Messaging can be found in JMS Messaging (JNDI).

Oracle Steps

The following steps are available in this chapter.

Oracle OC4J (JNDI)
Oracle AQ Steps
Oracle AQ (JMS)
Oracle AQ (JPUB)
see **Message Consumer**
see **Read a File (Disk, URL or Class Path)**
see **Web Service Execution (XML)**
see **Raw Soap Request**
see **SQL Database Execution (JDBC)**
see **FTP Step**
see **Web Service Execution (Legacy)**

Oracle OC4J (JNDI)

The Oracle OC4J (JNDI) step lets you send messages to, and receive messages from topics and queues. You can also receive, modify and forward an existing message. Oracle OC4J (JNDI) supports all the common message types including Empty, Text, Object, Bytes, Message, and Mapped (Extended).

The Oracle OC4J (JNDI) step is configured using a single LISA editor regardless the messaging requirements. Input options will vary on the messaging requirements. The editor will only allow valid configurations, so when you enable certain features others may become inactive.

Prerequisites: You will be required to supply the necessary JAR files for your chosen configuration and connection.

OC4J messaging requires that the several JAR files be added to your classpath, including **dms.jar**, **oc4j.jar**, and **oc4jclient.jar**. You can put them in the hot deploy directory. See the OC4J documentation for more information about the JAR files you might need. The JAR files can be found in the lib directory of the OC4J installation.

Parameter Requirements: You need the connection parameters and the subject names used in the application under test. There may be other parameters required, depending on your environment. Get these from the developers of the application.

LISA, by default uses the `OC4J_SERVER` property in the JNDI Server URL. This property must be added to your configuration if you plan to use it.

Detailed information about the parameters and fields for this step can be found in JMS Messaging (JNDI).

Oracle AQ Steps

Oracle AQ is a messaging provider, like IBM WebSphere MQ, webMethods Broker, Tibco EMS, and others. It fits into LISA just like any of those other messaging providers.

The Oracle AQ step can be added to the test case to allow sending messages, receiving messages, receiving messages asynchronously, or some combination of those three.

In the case of AQ, there are two separate step types (JMS and JPUB). They have the same functionality but represent two different methods for communicating with the Oracle AQ messaging provider.

Both Oracle AQ steps have the standard configuration sections for messaging steps.

- A Connection section for configuring the connection information.
- An optional Subscriber section for configuring the location from which the step should receive its message and the type of message it will receive.
- An optional Publisher section for configuring the location to which the step should send its message and the type of message it will send. The contents of the message to send are configured in a separate tab.
- When the step is executed in a test case, it will send one message and/or receive one message. It can be executed multiple times in a loop in the same test case to send and/or receive multiple messages.
- When the step is used as an asynchronous subscriber then it is only run once in a test case, and an additional Consumer step is run to consume each message received from the provider.

There are two distinct ways to use AQ: JMS and JPUB.

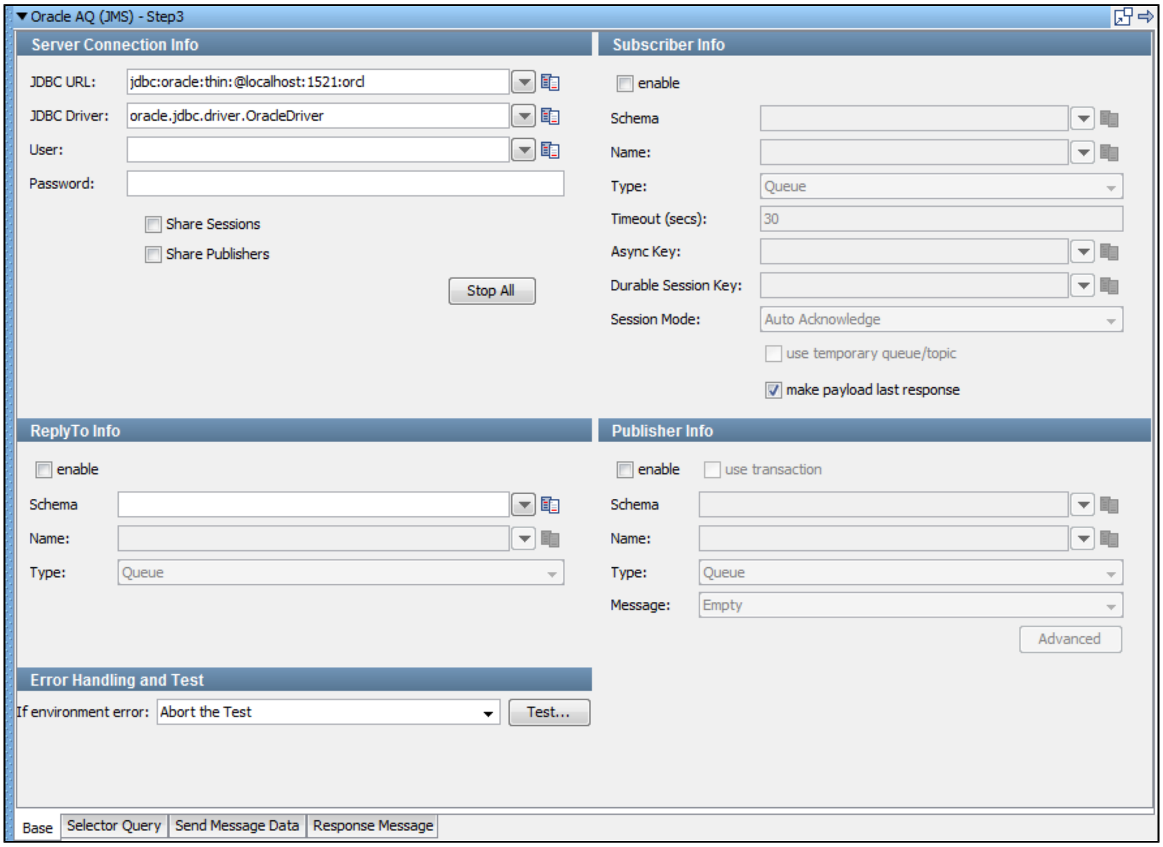
JMS: Using the JMS library it works like any other normal JMS provider, with a few notable differences:

- The JMS connection is not made through JNDI; it is made using a JDBC connection. It involves entering a JDBC URL, driver class name, username, and password.
- Queues and Topics are tied to schemas in the database. To send to or receive from a queue you need to give both the queue name and queue schema.
- Each Queue or Topic is restricted to a particular type of JMS Message. If a queue normally transports JMS Text Messages, then you cannot use that same queue to transport JMS Object Messages, or JMS Byte Messages, for example.
- Setting up JMS Queues and Topics in the Oracle DB involves running stored procedures.

Oracle AQ (JMS)

Oracle Advanced Queuing (AQ) is a messaging provider built into the Oracle database itself. It is used as the default JMS provider for many Oracle products, such as Oracle Enterprise Service Bus.

One of the two ways to utilize AQ is JMS.



Oracle AQ (JMS) - Step3

Server Connection Info

JDBC URL: jdbc:oracle:thin:@localhost:1521:orcl

JDBC Driver: oracle.jdbc.driver.OracleDriver

User:

Password:

□ Share Sessions

□ Share Publishers

Subscriber Info

□ enable

Schema

Name:

Type: Queue

Timeout (secs): 30

Async Key:

Durable Session Key:

Session Mode: Auto Acknowledge

□ use temporary queue/topic

☑ make payload last response

ReplyTo Info

□ enable

Schema

Name:

Type: Queue

Publisher Info

□ enable □ use transaction

Schema

Name:

Type: Queue

Message: Empty

Error Handling and Test

If environment error: Abort the Test

Using the JMS library, it works like any other normal JMS provider, with a few notable differences:

1. The JMS connection is not made through JNDI; it is made using a JDBC connection. It involves entering a JDBC URL, driver class name, user name, and password. This information goes under Server Connection Info of the LISA Oracle AQ(JMS) step.
2. Queues and topics are tied to schemas in the database. To send to or receive from a queue, you need to give both the queue name and queue schema.
3. Each queue or topic is restricted to a particular type of JMS Message. If a queue normally transports JMS Text Messages then you cannot use that same queue to transport JMS Object Messages, or JMS Byte Messages, for example.

There are four tabs available at the bottom of the editor.

- The Base tab is where you define your connection and messaging parameters.
- The Selector Query tab lets you specify a selector query to be run when listening for a message on a queue.
- The Send Message Data tab is where you will create your message content.
- The Response Message tab is where your response messages will be posted.

Base Information Tab

The Base tab view is shown in the previous example. It is divided into 5 major sections:

- Server Connection Info.
- Subscriber Info
- Publisher Info
- ReplyTo Info
- Error Handling and Test

The Server Connection Info and Error Handling and Test sections are always active. The Subscriber Info, Publisher Info, and, ReplyTo Info sections can be enabled or disabled using the enable check box in the top left corner of each section. Using these check boxes, you can configure the step to publish a step, subscribe a step, or both. You can also choose to include a **replyto** component in the step. When you have completely configured your test step, use the Test button in the Error Handling and Test section to test your configuration settings

Server Connection Info

Here you enter the JDBC related information.

These values should be parameterized with properties that are in your configuration, making it easy to change the application under test.

LISA by default uses the oracle.jdbc.driver.OracleDriver in the JDBC Driver location.

The five parameters should be available to you for the system under test. The pull-down menus contain common examples or templates for these values.

- **JDBC URL**: LISA pre-populates this field with default values.
- **JDBC Server**: LISA pre-populates this field with default values.
- **User**: Enter the user name.
- **Password**: Enter the password.
- **Share Sessions and Share Publishers** check boxes are used to share JMS Sessions and Publishers throughout the test case. This can lower overhead, but does not always provide a realistic simulation because typically JMS clients want to release resources. If you check **Share Publishers**, the **Share Sessions** check box is also selected for you as you cannot share publishers without sharing sessions.
- The Stop All button lets you stop any listeners at design time now. This is to resolve issues during test case creation where some listeners can get orphaned, but will still consume messages making it difficult or sometimes impossible to finish test case creation.

Publisher Info

Select the enable check box to set up the ability to send (publish) messages.

Select the **use transaction** check box to execute a commit when the message is sent.

Enter the following parameters:

- **Schema:** Enter the name of the schema to use.
- **Name**: Enter the name of the topic or queue to use.
- **Type**: Select whether you are using a topic or queue.
- **Message**: Select the type of message you are sending from the pull-down menu. The supported messages are: Empty, Text, Object, Bytes, Message, and Mapped (Extended).
- **Advanced** button: Displays a panel where you can add custom message properties to be sent with the message.

Subscriber Info

Select the enable check box to set up to enable the ability to receive (subscribe to) messages.

Enter the following parameters:

- **Schema**: Enter the name of the schema.
- **Name**: Enter the name of the topic or queue to use.
- **Type**: Select whether you are using a topic or queue, and whether you want to listen in synchronous or asynchronous mode. For asynchronous mode you will also have to have an entry in the **Async key** field. You can use the Browse icon, to the right of this field, to see what messages are waiting to be consumed from a queue (only).
- **Timeout (secs)**: Enter the period to wait before LISA interrupts waiting for a message (this field can be left blank for no timeout).
- **Async Key**: Enter the value needed to identify asynchronous messages. This is only needed in asynchronous mode. It will be used in a subsequent Message Consumer step to retrieve asynchronous messages.
- **Durable Session key**: By entering a name here you are requesting a durable session. You are also providing a key for that session. A

durable session lets you receive all of your messages from a topic even if you log off, and then log on again.
- **Session mode**: Select the appropriate mode from the available options by clicking the drop-down menu. Options are: Auto Acknowledge, Client Acknowledge, Use Transaction, Auto (Duplicates Okay).

ReplyTo Info

If your application needs a destination, it is set up in this section.

Select the enable check box to set up a destination queue/topic.

Enter the following parameters:

- **Schema**: Enter the name of the schema to be used.
- **Name**: Enter the name of the topic or queue to use.
- **Type**: Select from the drop[down, whether you want to use a topic or queue.

Error Handling and Test

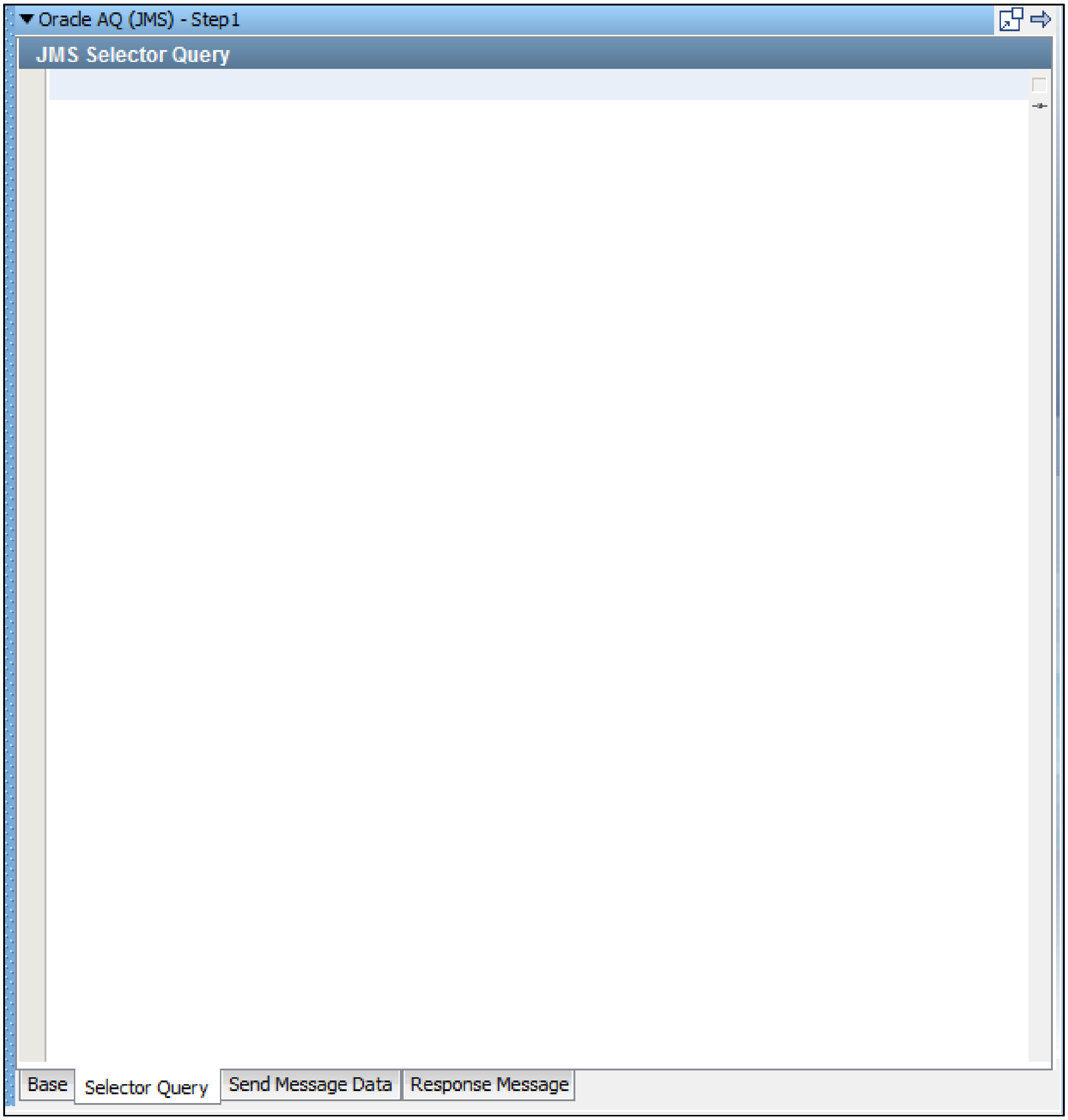
Error Handling and Test section lets you redirect to a step if an error occurs.

- **If environment error**: Select the step to redirect to if an error occurs.

Click the Test button to test your step configuration settings.

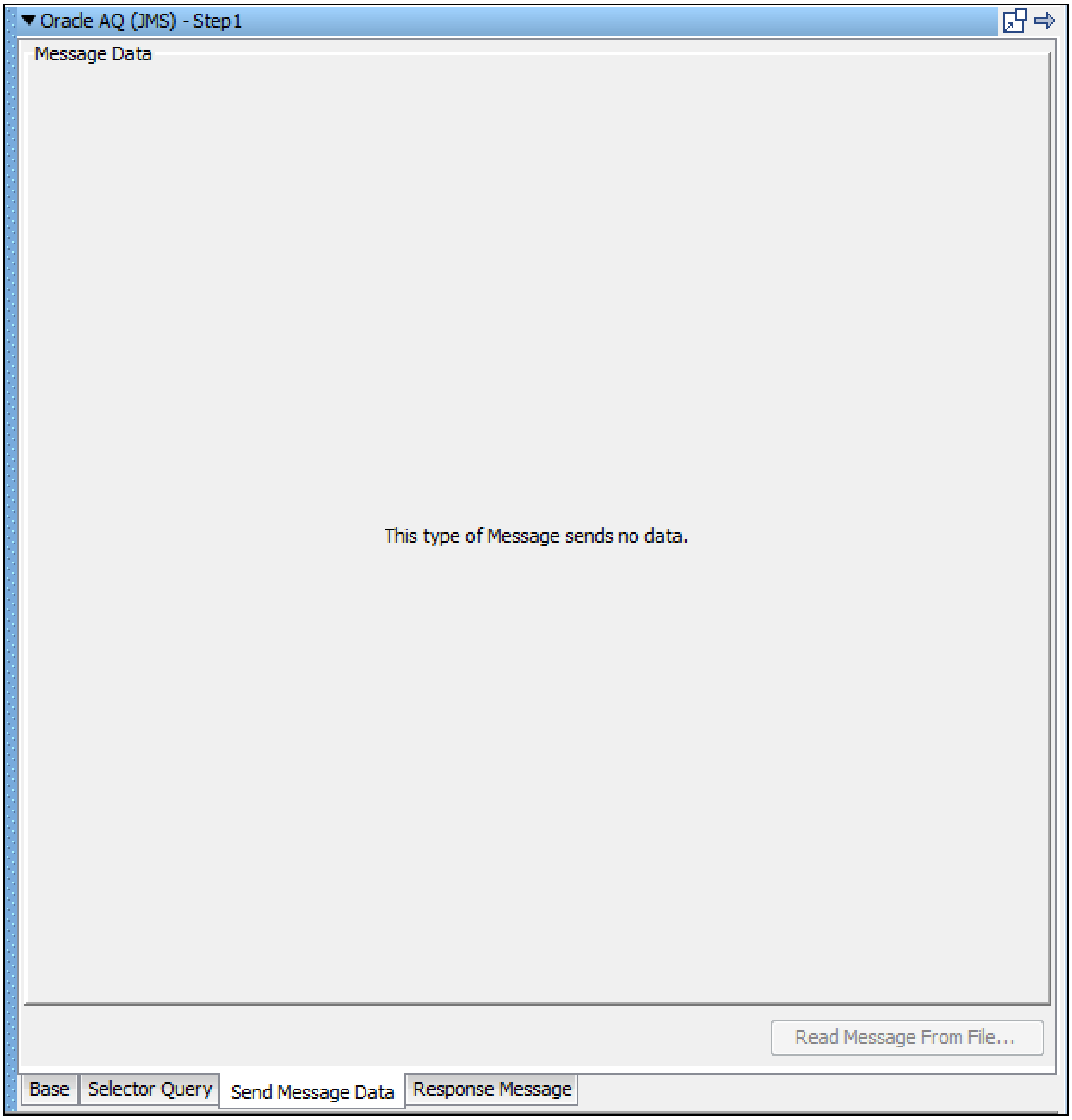
Selector Query Tab

TODO



Send Message Data Tab

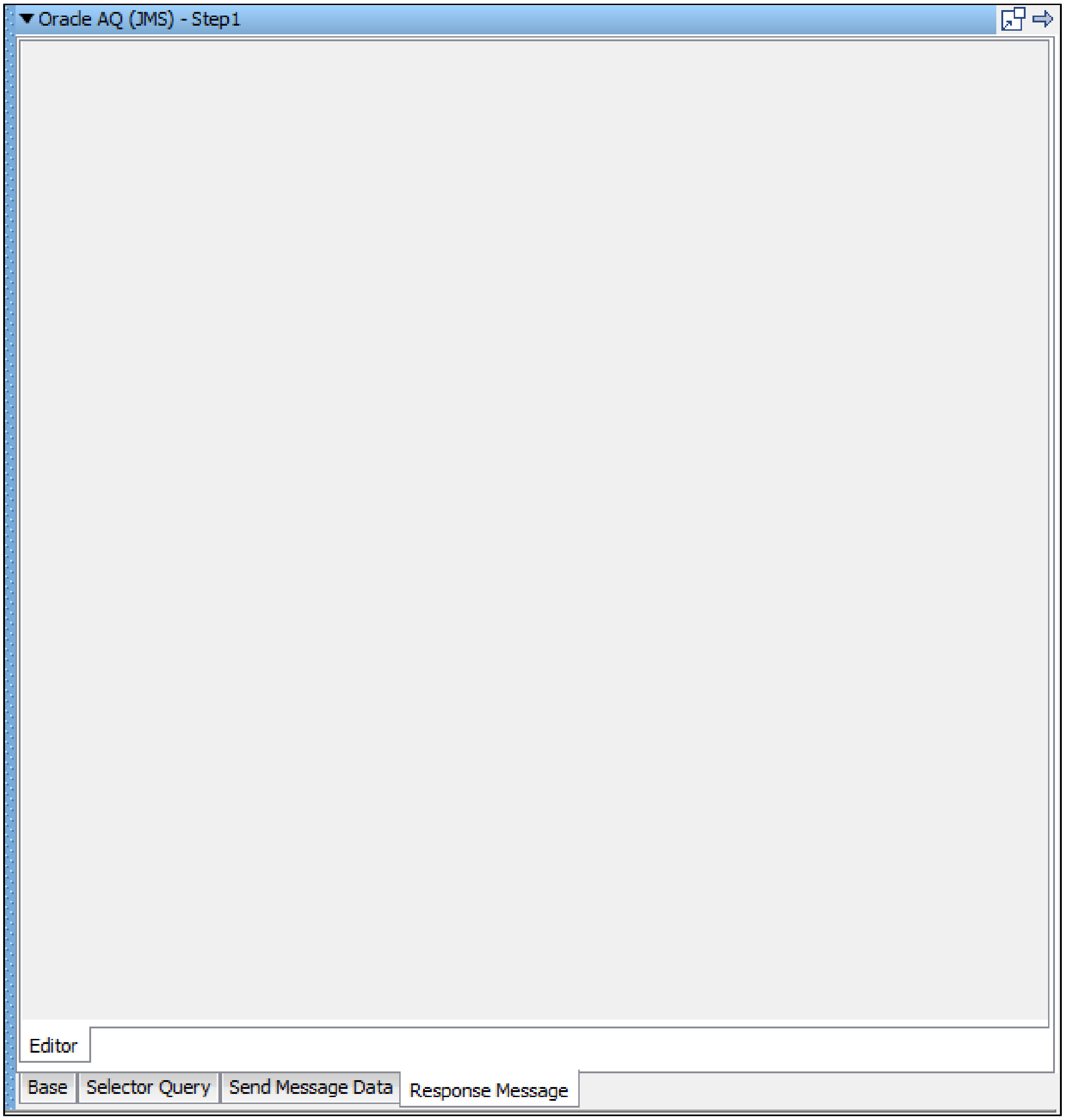
TODO



This type of Message sends no data.

Response Message Tab

TODO



Oracle AQ (JPUB)

There are two distinct ways to utilize AQ; one is JMS, the other is JPUB.

Oracle AQ's JMS API is a layer built on top of a lower-level AQ API. This lower-level API is much more difficult to deal with; it acts almost nothing like JMS.

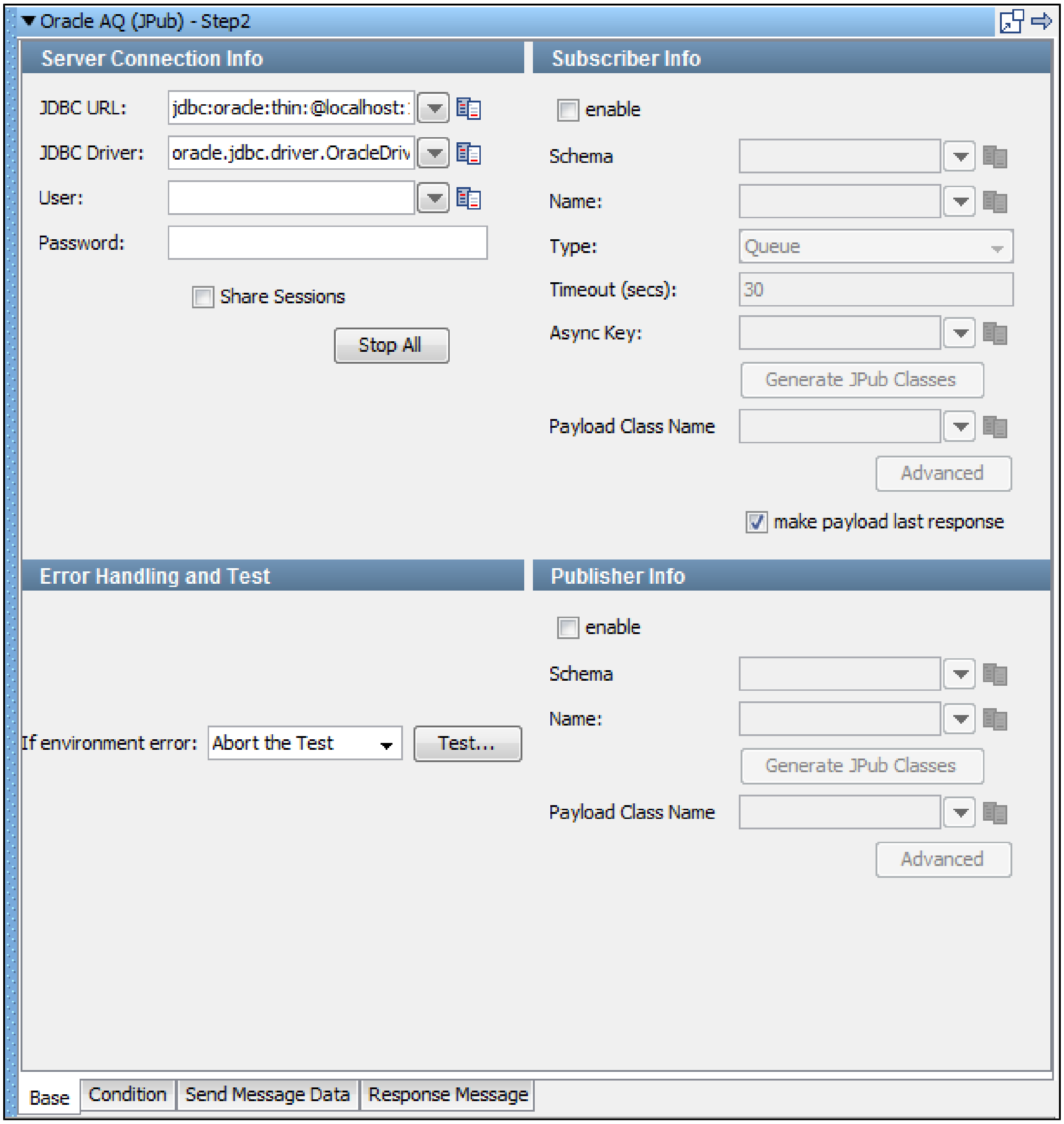
The principal distinction is the message format. Low-level AQ messages contain a payload, which can be any type defined in the database. It could be a varchar, or a clob, but usually it is a user-defined structured database type. Similar to AQ JMS Queues, each AQ low-level queue can only handle one payload type.

Oracle provides a utility, named JPUB, that can generate Java objects that can deal with these user-defined structured types, in the same way that Axis generates Java objects that utilize web services. Our low-level AQ step, named Oracle AQ JPUB, can automatically use this utility to generate the client classes based on the queue information. The user then fills in their payload object using a standard COE.

There is no distinction between queues or topics. A client can either remove the next message from the AQ queue, making it essentially a queue, or read the next message from the AQ queue without removing it, making it essentially a topic.

Setting up low-level AQ queues is again done through stored procedures. There is the possible additional step of creating your own user-defined structured type in the database before creating an AQ queue around it. Technically, you can interact with AQ JMS queues using the low-level API. The JMS queues simply have a specific payload type that's structured like a standard JMS message. However, you cannot use the AQ JMS API to interact with low-level AQ queues, that is, those that do not use a JMS payload type.

To add an Oracle AQ (JPUB) step to a test case, click the step to open its editor:



Server Connection Info

JDBC URL: jdbc:oracle:thin:@localhost:

JDBC Driver: oracle.jdbc.driver.OracleDriv

User:

Password:

□ Share Sessions

Subscriber Info

□ enable

Schema

Name:

Type: Queue

Timeout (secs): 30

Async Key:

Payload Class Name

☑ make payload last response

Error Handling and Test

If environment error: Abort the Test

Publisher Info

□ enable

Schema

Name:

Payload Class Name

There are four tabs available at the bottom of the editor.

- The **Base** tab is where you define your connection and messaging parameters.
- The **Condition** tab lets you specify a condition to be run.
- The **Send Message Data** tab is where you will create your message content.
- The **Response Message** tab is where your response messages will be posted.

Base Information Tab

The Base tab view is the default view and is shown in the previous illustration. It is divided into four major sections:

- Server Connection Info
- Subscriber Info
- Publisher Info
- Error Handling and Test

The Server Connection Info and Error Handling and Test sections are always active. The Subscriber Info and Publisher Info sections can be enabled or disabled using the enable check box in the top left corner of each section. Using these check boxes, you can configure the step to publish a step, subscribe a step, or both.

When you have completely configured your test step, use the Test button in the Error Handling and Test section to test your configuration settings.

Server Connection Info

These values should be parameterized with properties that are in your configuration, making it easy to change the application under test. By default, the oracle.jdbc.driver.OracleDriver in the JDBC Driver location is used.

- **JDBC URL**: This field is pre-populated with default values.
- **JDBC Server**: This field is pre-populated with default values.
- **User**: Enter the user name.
- **Password**: Enter the password.
- **Share Sessions and Share Publishers** check boxes: Used to share JMS Sessions and Publishers throughout the test case. This can lower overhead, but does not always provide a realistic simulation because typically JMS clients want to release resources. If you check Share Publishers, the Share Sessions check box is also selected for you as you cannot share publishers without sharing sessions.
- The Stop All button lets you stop any listeners at design time now. This is to resolve issues during test case creation where some listeners can get orphaned, but will still consume messages making it difficult or sometimes impossible to finish test case creation.

Publisher Info

Select the enable check box to set up the ability to send (publish) messages.

Enter the following parameters:

- **Schema:** Enter the name of the schema to use.
- **Name**: Enter the name of the topic or queue to use.
- **Generate JPub classes**: Click to generate the JPub classes.
- **Payload Class Name**: Enter the payload class name.
- **Advanced** button: Click to open the Publisher Advanced dialog to enter or select the Correlation and click OK**.**

Subscriber Info

Select the enable check box to set up to enable the ability to receive (subscribe to) messages.

Enter the following parameters:

- **Schema**: Enter the name of the schema.
- **Name**: Enter the name of the topic or queue to use.
- **Type**: Select whether you are using a topic or queue, and whether you want to listen in synchronous or asynchronous mode. For asynchronous mode you will also have to have an entry in the Async key field.
- **Timeout (secs)**: Enter the period to wait before LISA interrupts waiting for a message. This field can be left blank for no timeout.
- **Async Key**: Enter the value needed to identify asynchronous messages. This is only needed in asynchronous mode. It will be used in a subsequent Message Consumer step to retrieve asynchronous messages.
- **Generate JPub classes**: Click to generate the JPub classes.
- **Payload Class Name**: Enter the Payload class name.
- **Advanced**: Click to open the Subscriber Advanced dialog.

In the Advanced dialog, you can enter the Consumer Name, Correlation and Message ID.

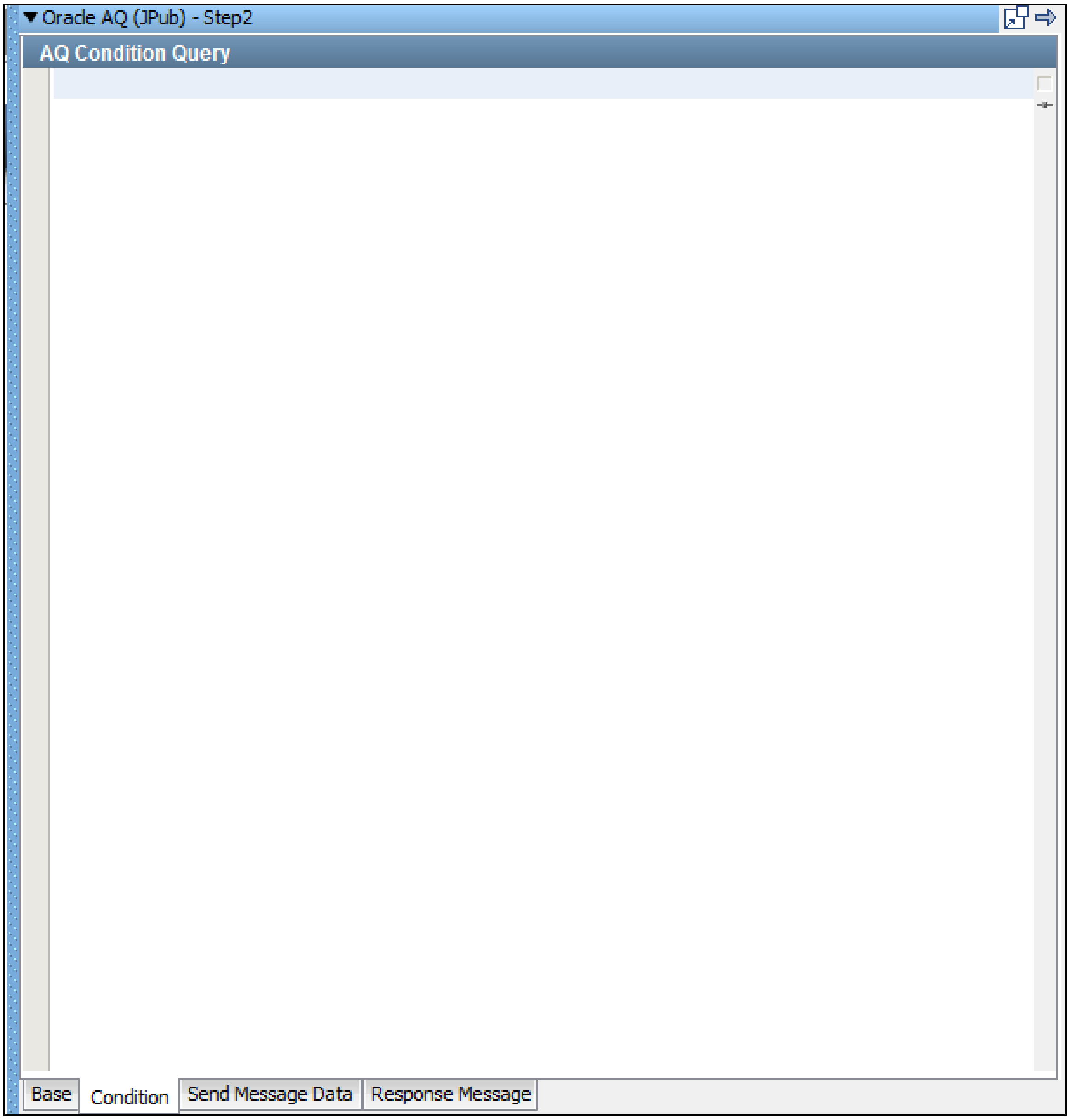
Error Handling and Test

Error Handling and Test section lets you redirect to a step if an error occurs.

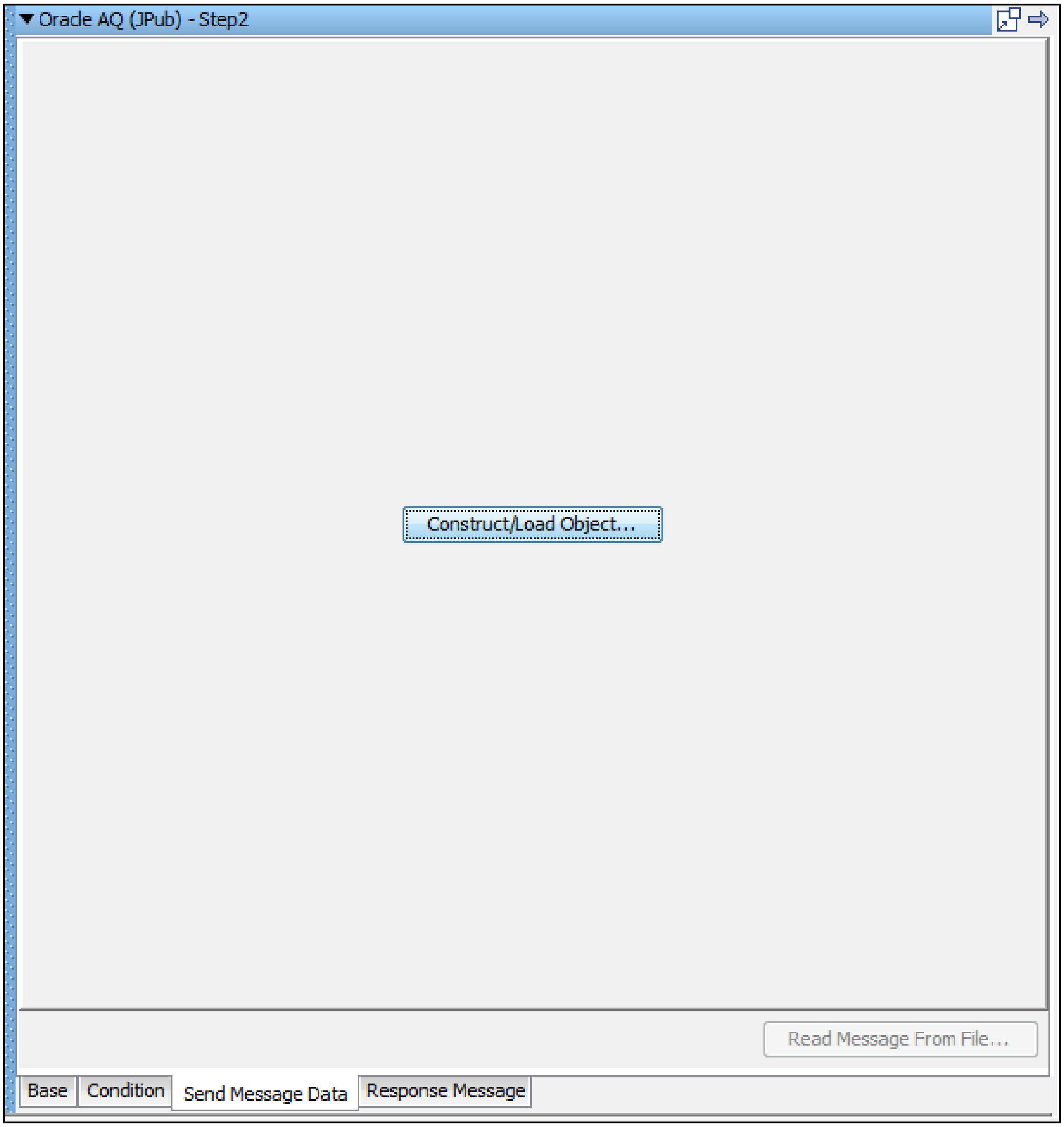
- If environment error: Select the step to redirect to if an error occurs.

Click the Test button to test your step configuration settings.

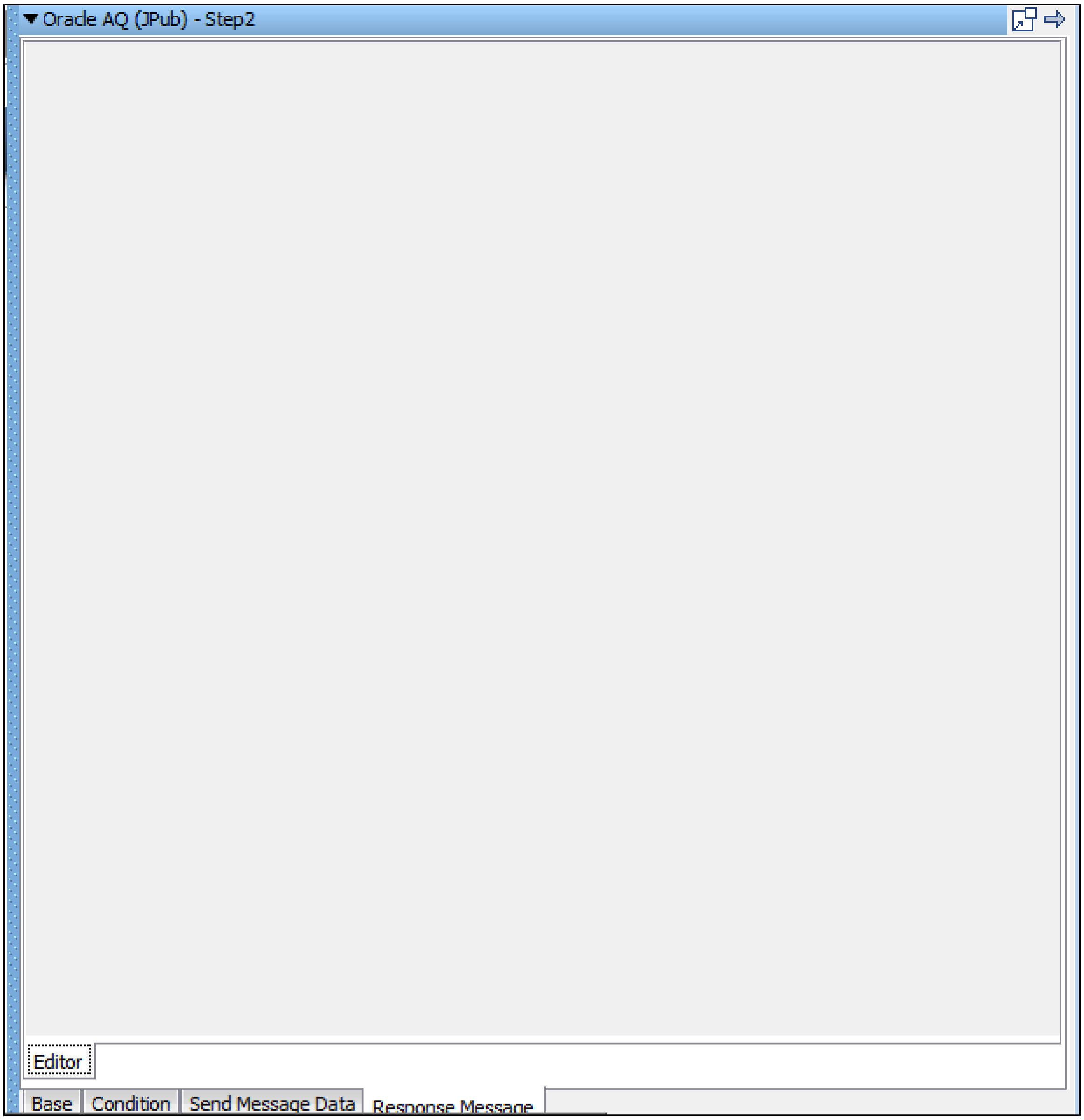
Condition tab



Send Message Data tab



Response Message tab



TIBCO Steps

The following steps are available in this chapter.

TIBCO Rendezvous Messaging
TIBCO Direct JMS
TIBCO EMS Messaging
see **Message Consumer**
see **Read a File (Disk, URL or Class Path)**
see **Web Service Execution (XML)**
see **Raw Soap Request**
see **SQL Database Execution (JDBC)**
see **FTP Step**
see **Web Service Execution (Legacy)**

TIBCO Rendezvous Messaging

The TIBCO Rendezvous Messaging step lets you send messages to, and receive messages from, Rendezvous "Subjects" using Native Rendezvous protocol. You can also receive, modify, and forward an existing message.

The TIBCO Rendezvous Messaging step is configured using a single editor regardless of the messaging requirements. Input options vary on the messaging requirements. The editor will only allow valid configurations, so when you enable certain features others may become inactive.

Prerequisites: You are required to supply the necessary JAR files for your chosen configuration and connection.

The TIBCO Rendezvous "bin" directory must be added to your PATH environment variable.

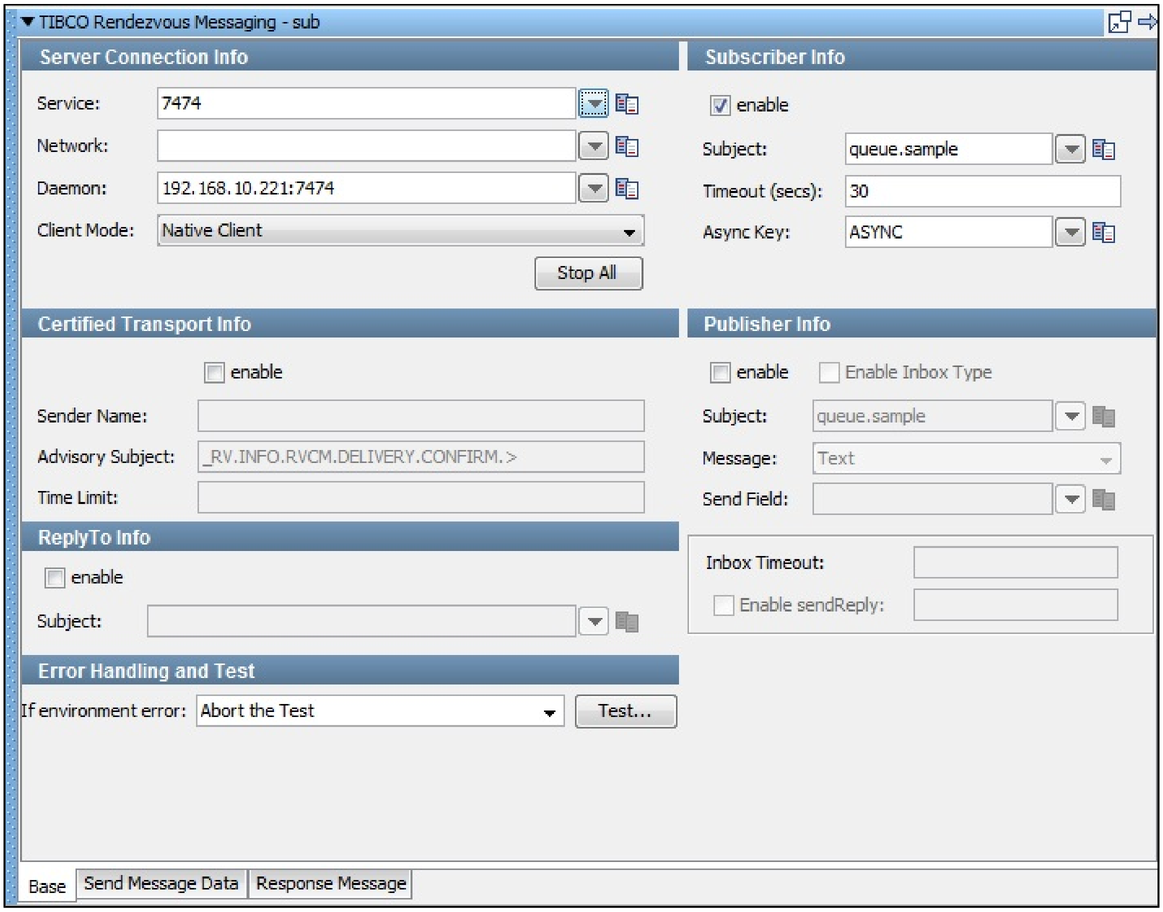
TIBCO Rendezvous requires that several TIBCO RV JAR files be in your classpath. The JAR files can be found in the **lib** directory of the TIBCO installation:

- **tibjms.jar**
- **tibjmsadmin.jar**
- **tibjmsapps.jar**
- **tibrvjms.jar**

TIBCO classes require access to the system class loader, so it is suggested that you create a LISA_PRE_CLASSPATH environment variable in your OS that lists the TIBCO RV JAR files.

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

Parameter Requirements: You need the connection parameters and the subject names used in the application under test. The following sections describe the parameters that you will need. There may be other parameters required, depending on your environment. Get these from the developers of the application.



TIBCO Rendezvous Messaging - sub

Server Connection Info

Service: 7474

Network:

Daemon: 192.168.10.221:7474

Client Mode: Native Client

Subscriber Info

☑ enable

Subject: queue.sample

Timeout (secs): 30

Async Key: ASYNC

Certified Transport Info

□ enable

Sender Name:

Advisory Subject: _RV.INFO.RVCM.DELIVERY.CONFIRM.>

Time Limit:

Publisher Info

□ enable □ Enable Inbox Type

Subject: queue.sample

Message: Text

Send Field:

Inbox Timeout:

□ Enable sendReply:

ReplyTo Info

□ enable

Subject:

Error Handling and Test

If environment error: Abort the Test

There are three tabs available at the bottom of the editor:

- **Base** tab: Where you define your connection and messaging parameters.
- **Send Message Data** tab: Where you create your message content.
- **Response Message** tab: Where your response messages will be posted.

Base Information (Base tab)

The Base tab is divided into six major sections:

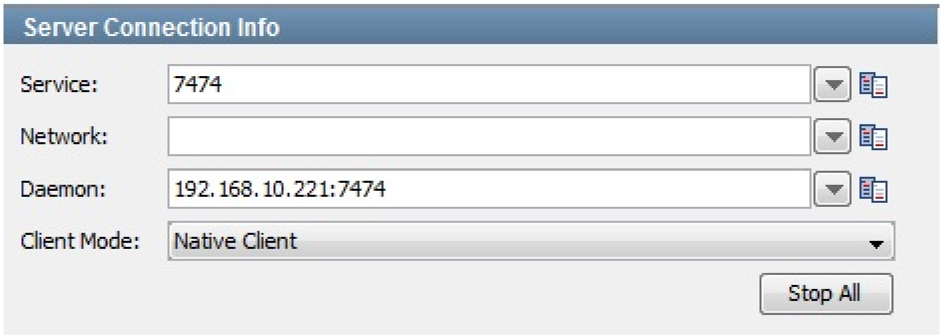
- Server Connection Info
- Subscriber Info
- Certified Transport Info
- Publisher Info
- ReplyTo Info

- Error Handling and Test

The Server Connection Info and Error Handling and Test sections are always active. The Subscriber Info, Publisher Info, and, ReplyTo Info sections can be enabled or disabled using the enable check box in the top left corner of each section. You can use these check boxes to configure the step to be a publish step, a subscribe step, or both. You can also choose to include a replyto component in the step. When you have completely configured your test step, use the Test button in the Error Handling and Test section to test your configuration settings.

Server Connection Info

Enter the connection information specific to Rendezvous information in the Server Connection Info area.



Service: 7474

Network:

Daemon: 192.168.10.221:7474

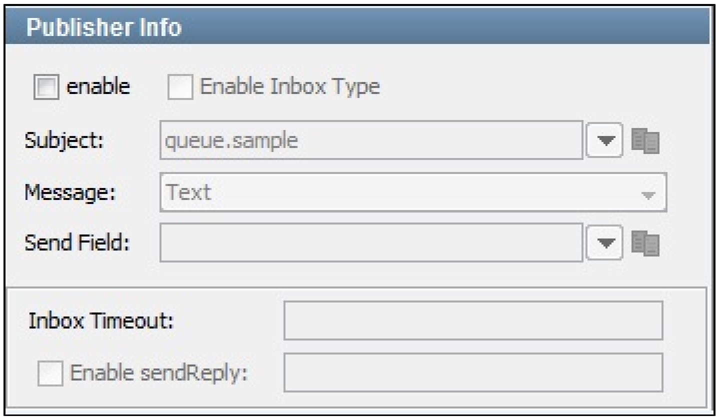
Client Mode: Native Client

Four parameters are available to you for the system under test.

- **Service, Network, and Daemon**: These are parameters to enable connection to the RV network you want to communicate on.
- **Client Mode**: Choose between the Rendezvous Native client and Java Client mode. Usually you will want to use the more versatile client mode.

These values should be parameterized with properties that are in your configuration, making it easy to change for a different system under test.

Publisher Info



□ enable □ Enable Inbox Type

Subject: queue.sample

Message: Text

Send Field:

Inbox Timeout:

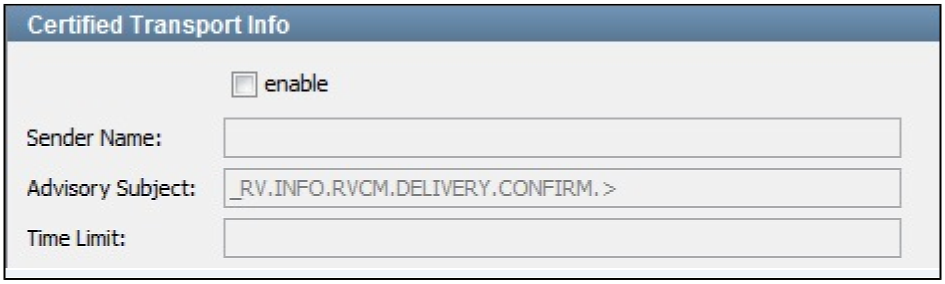
□ Enable sendReply:

Select the enable check box to set up the ability to send (publish) messages.

Enter the following parameters:

- **Subject**: Enter the name of the subject to use. You can define your own subjects. A valid subject name is: **queue.sample**. An invalid subject name is: **queue…..My_Samples** (null element) or **.My.Queue..** (three null elements).
- **Message**: Select the type of message you are sending from the pull-down menu. The supported messages are: Empty, Text, Object, Bytes, Message, and Mapped (Extended).
- **Send Field**: RV messages are actually maps of fields and values. This field is used to enable quick single field messages. When you enter a value here the Send Message data is put into the value of a field with this name. This is overridden with Mapped (Extended) type messages as they will let you do multiple fields and values in a single message.
- **Enable Inbox Type**: Select the Enable Inbox Type check box to enable the Inbox timeout and Enable SendReply options.
- **Enable sendReply**: Select the Enable Inbox Type to specify an Inbox Timeout or to enable sendReply functionality for the publisher.

Certified Transport Info



Certified Transport Info

□ enable

Sender Name:

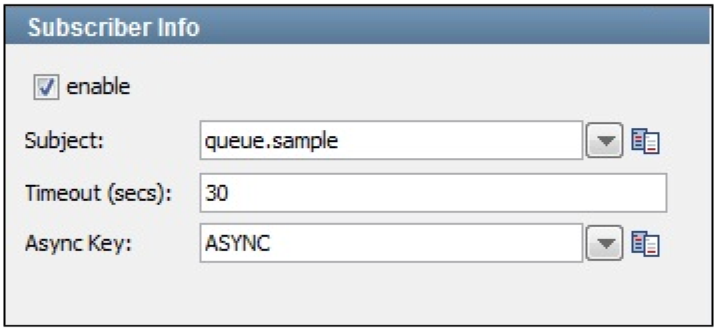
Advisory Subject: _RV.INFO.RVCM.DELIVERY.CONFIRM.>

Time Limit:

Select the **enable** check box to provide transport information

- **Sender Name**: The name that is the correspondent name of the CM transport.
- **Advisory Subject**: Rendezvous software constructs the subject names of system advisory messages using this template: **_RV.class.SYSTEM.name**. Rendezvous certified message delivery software constructs the subject names of advisory messages using these templates: **_RV.class.RVCM.category.condition.subject** and **_RV.class.RVCM.category.condition.name.** Distributed queue software constructs the subject names of advisory messages using this template: **_RV.class.RVCM.category.role.condition.name**. Rendezvous fault tolerance software constructs the subject names of advisory messages using this template: **_RV.class.RVFT.name.group**.
- **Time Limit**: Time limit in which a message will exist.

Subscriber Info



Subscriber Info

☑ enable

Subject: queue.sample

Timeout (secs): 30

Async Key: ASYNC

Selecting the **enable** box turns the subscriber function on. and lets you set up the ability to receive (subscribe to) messages.

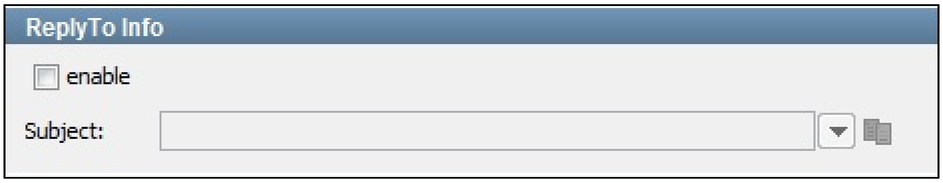
Enter the following parameters:

- **Subject**: Enter the name of the subject to use. You can define your own subjects.

- **Timeout (secs)**: Enter the period to wait before there is an interrupt waiting for a message (this field can be left blank for no timeout).

- **Async Key**: Enter the value needed to identify asynchronous messages. This is only needed in asynchronous mode. It will be used in a subsequent Message Consumer step to retrieve asynchronous messages.

ReplyTo Info



ReplyTo Info

□ enable

Subject:

Select the enable check box to set up a destination subject.

If your application needs a destination, it is set up in this section.

Enter the following parameters:

- **Subject**: Enter the name of the subject to use.

Error Handling and Test

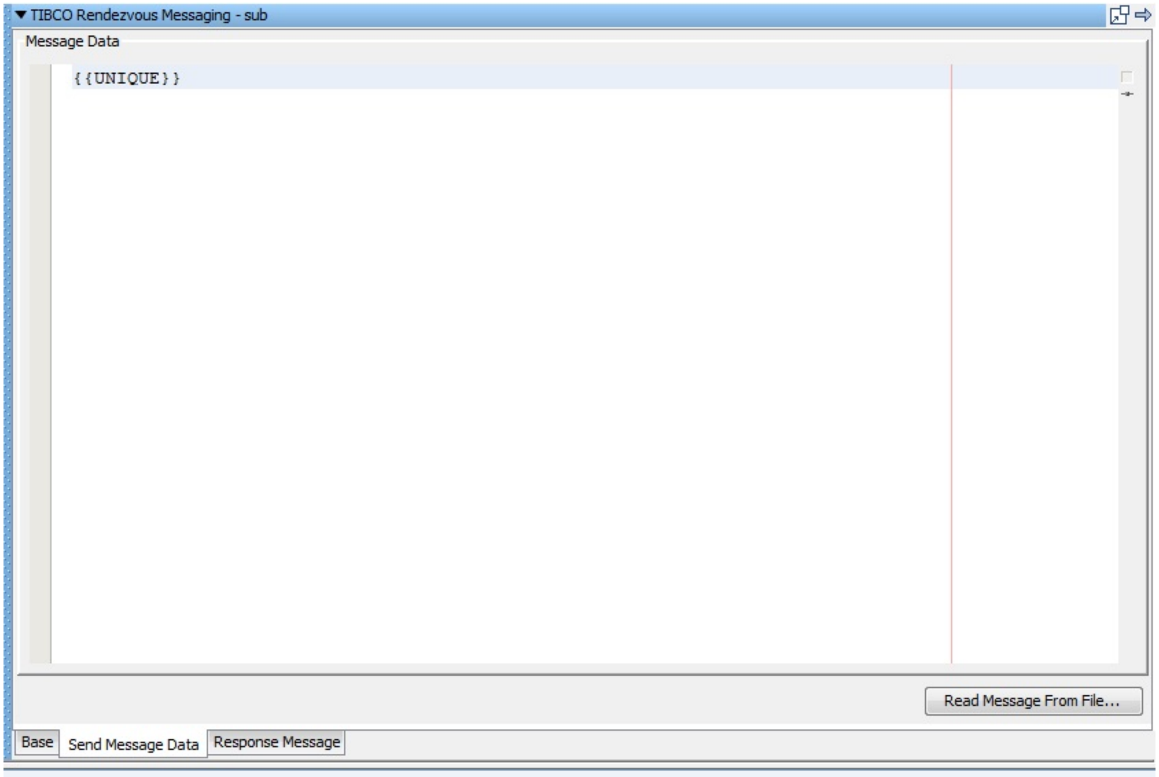
Error Handling and Test lets you redirect to a step if an error occurs.

- **If environment error**: Select the step to redirect to if an error occurs.

Click the Test button to test your step configuration settings.

Send Message Data

If your step is configured to publish, this is where you compose your message. The Send Message Data tab view in the following example shows a text message.



```
{{UNIQUE}}
```

This particular example shows an XML fragment with LISA properties being used. The text can be typed in or it can be read from a file using the Read Message from the File button in the bottom right corner, or it can be stored in a LISA property, in which case you would place the property in the editor: for example, **`LISA_PROP`**.

Notice that LISA properties are used in the message XML allowing the message to be created dynamically during the test run.

Response Message

If your step is configured to subscribe, your response will be shown. For more information, see JMS Messaging (JNDI).

TIBCO Direct JMS

The TIBCO Direct JMS step lets you send messages to, and receive messages from, topics and queues. You can also receive, modify, and forward an existing message.

TIBCO Direct JMS supports all the common message types including Empty, Text, Object, Bytes, Message, and Mapped (Extended).

The TIBCO Direct JMS step is configured using a single editor regardless of the messaging requirements. Input options will vary on the messaging requirements. The editor will only allow valid configurations, so when you enable certain features others may become inactive.

Prerequisites: You will be required to supply the necessary JAR files for your chosen configuration and connection.

TIBCO Direct JMS requires that several JAR files be added to your classpath. Check the TIBCO documentation for more information about the JAR files you might need. These are typically found in the TIBCO install directory in **lib** for the product you are using. It is suggested to create your LISA_PRE_CLASSPATH environment variable in your operating system and restart LISA. The TIBCO classes require access to the system class loader.

Parameter Requirements: You need the connection parameters and the subject names used in the application under test. LISA, by default uses the `TIBCO_SERVER` property in the JNDI Server URL. This property must be added to your configuration if you plan to use it. There may be other parameters required, depending on your environment. Get these from the developers of the application.

For more detailed information about parameters and fields, see JMS Messaging (JNDI).

TIBCO EMS Messaging

The TIBCO EMS Messaging step lets you send messages to, and receive messages from, topics and queues. You can also receive, modify, and forward an existing message.

LISA supports all the common message types including Empty, Text, Object, Bytes, Message, and Mapped (Extended).

Prerequisites: You will be required to supply the necessary JAR files for your chosen configuration and connection.

The TIBCO Rendezvous bin directory must be added to your PATH environment variable.

TIBCO Rendezvous requires that several TIBCO RV JAR files be in your classpath. The JAR files can be found in the lib directory of the TIBCO installation:

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

TIBCO classes require access to the system class loader, so it is suggested that you create a LISA_PRE_CLASSPATH environment variable in your OS that lists the TIBCO RV JAR files.

Parameter Requirements: You need the connection parameters and the subject names used in the application under test. The default is the `TIBCO_SERVER` property in the JNDI Server URL. This property must be added to your configuration if you plan to use it. There may be other parameters required, depending on your environment. Get these from the developers of the application.

For more detailed information about parameters and fields, see JMS Messaging (JNDI)

Sonic Steps

The following steps are available in this chapter.

SonicMQ Messaging (Native)

The SonicMQ Messaging (Native) step lets you send messages to, and receive messages from, topics and queues, using native Sonic protocol. You can also receive, modify, and forward an existing message.

SonicMQ Messaging (Native) supports all the common message types including Empty, Text, Object, Bytes, Message, and Mapped (Extended).

The SonicMQ Messaging (Native) step is configured using a single editor regardless of the messaging requirements. Input options will vary on the messaging requirements. The editor will only allow valid configurations, so when you enable certain features others may become inactive.

Prerequisites: You will be required to supply the necessary JAR files for your chosen configuration and connection.

Sonic requires several JAR files to be added to your classpath. You can put them in the **hotdeploy** directory. The JAR files can be found in the lib directory of the Sonic installation:

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

Parameter Requirements: You need the connection parameters and the subject names used in the application under test.

The four parameters should be available to you for the system under test.

- Broker Host
- Broker Port
- User
- Password

There may be other parameters required, depending on your environment. Get these from the developers of the application.

For more detailed information about parameters and fields, see JMS Messaging (JNDI)

SonicMQ Messaging (JNDI)

SonicMQ Messaging (JNDI) supports all the common message types including Empty, Text, Object, Bytes, Message, and Mapped (Extended).

The SonicMQ Messaging (JNDI) step lets you send messages to, and receive messages from topics and queues. You can also receive, modify and forward an existing message.

The SonicMQ Messaging (JNDI) step is configured using a single LISA editor regardless of the messaging requirements. Input options will vary on the messaging requirements. The editor will only allow valid configurations, so when you enable certain features others may become inactive.

Prerequisites: You will be required to supply the necessary JAR files for your chosen configuration and connection.

Sonic requires several JAR files be added to your classpath. You can put them in the hot deploy directory. The JAR files can be found in the lib directory of the Sonic installation:

- Sonic_XA.jar
- mfcontext.jar
- sonic_Client.jar

Parameter Requirements: You need the connection parameters and the subject names used in the application under test. LISA, by default uses the `SONICMQ_SERVER` property in the JNDI Server URL. This property must be added to your configuration if you plan to use it. There may be other parameters required, depending on your environment. Get these from the developers of the application.

For more detailed information about parameters and fields, see JMS Messaging (JNDI).

webMethods Steps

The following test steps are available in this chapter.

webMethods Broker

webMethods Broker supports Mapped (Extended) messages that will create Broker Events.

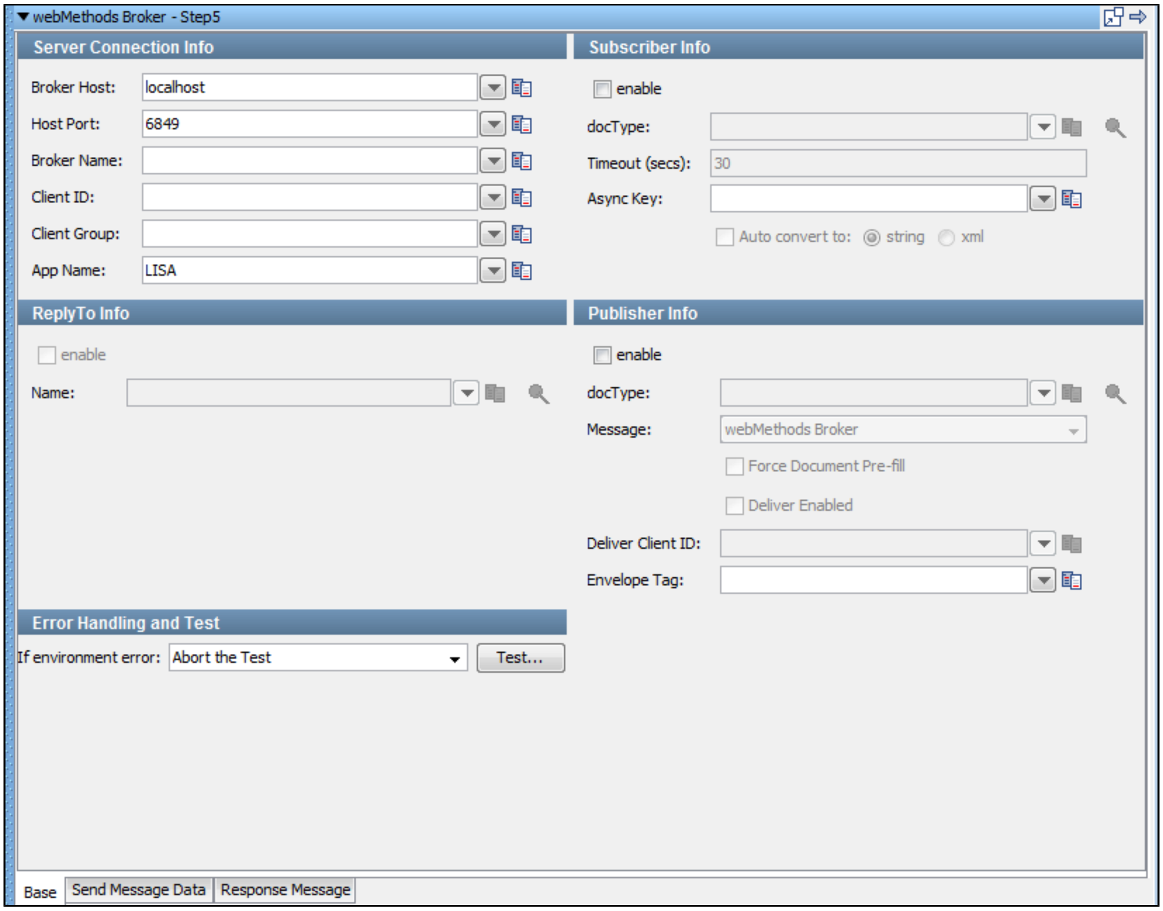
The webMethods Broker step lets you send messages to, and receive messages from the Broker. You can also receive, modify and forward an existing Broker Events/ Messages.

The webMethods Broker step is configured using a single LISA editor regardless of the messaging requirements. Input options will vary on the messaging requirements. The step editor will only allow valid configurations, so when you enable certain features others may become inactive.

Prerequisites: You will be required to supply the necessary JAR files for your chosen configuration and connection. The webMethods Broker requires that you put several JAR files in your classpath. The JAR files can be found in the **lib** directory of the webMethods installation.

- For webMethods Integration Server 7.1 and later, **installation_directory\common\lib\wm-isclient.jar**
  For Integration Server 7.0 and earlier, **installation_directory\lib\client.jar**
- **wm-enttoolkit.jar**
- **wmbrokerclient.jar**
- **wmjmsadmin.jar**
- **wmjmsclient.jar**
- **wmjmsnaming.jar**

Parameter Requirements: You need the connection parameters and the subject names used in the application under test. The following sections describe the parameters that you will need. There may be other parameters required, depending on your environment. Get these from the developers of the application.


webMethods Broker - Step5

Server Connection Info

Broker Host: localhost

Host Port: 6849

Broker Name:

Client ID:

Client Group:

App Name: LISA

Subscriber Info

□ enable

docType:

Timeout (secs): 30

Async Key:

□ Auto convert to: ● string ○ xml

ReplyTo Info

□ enable

Name:

Publisher Info

□ enable

docType:

Message: webMethods Broker

□ Force Document Pre-fill

□ Deliver Enabled

Deliver Client ID:

Envelope Tag:

Error Handling and Test

If environment error: Abort the Test

There are three tabs available at the bottom of the messaging step editor for webMethods Broker.

- The Base tab is where you define your connection and messaging parameters.
- The Send Message Data tab is where you will create your message content.
- The Response Message tab is where your response messages will be posted.

Base Tab

The Base tab view is shown in the previous illustration. It is divided into five major sections:

- Server Connection Info
- Subscriber Info
- Publisher Info
- ReplyTo Info
- Error Handling and Test

The Server Connection Info and Error Handling and Test sections are always active. The Subscriber Info, Publisher Info, and, ReplyTo Info sections can be enabled or disabled using the enable check box in the top left corner of each section. Using these check boxes you can configure the step to be a publish step, a subscribe step, or both. You can also include a **replyto** component in the step. When you have completely configured your test step, use the Test button in the Error Handling and Test section to test your configuration settings.

Server Connection Info

In the Server Connection Info section, enter the connection information specific to webMethods Broker.

The four parameters must be available to you for the system under test.

- **Broker Host**
- **Host Port**
- **Broker Name**
- **Client ID**
- **Client Group**: This is the Client group able to see the Broker destinations you wish to use.
- **App Name**: Specify the application using the Broker here. This is an optional parameter and mostly used in server logs for debugging. The default is "LISA". It is a good practice to do this, but if you must use something else for application logic you can.

We recommend that these values be parameterized with properties that are in your configuration, making it easy to change for a different system under test.

Publisher Info

Check the enable check box to set up the ability to send (publish) messages.

Enter the following parameters:

- **docType**: Enter the name of the docType to use.
- **Message**: Select the type of message you are sending from the pull-down menu. The supported messages are: webMethods Broker, Object, Message, and Mapped (Extended).
- **Force Document Pre-fill**: The selected docType is inspected and ithe message with the required fields is pre-loaded. This check box lets you have made modifications and decide you want any missing fields re-added. It will not write over existing fields with the same name. This property is only a design time effect and does nothing at test run time.
- **Deliver Enabled**: Select to enable the Deliver Client ID field.
- **Deliver Client ID**: Broker's Client Identification for the connection. If the value is null, the broker generates an identifier automatically. An error can be returned if the value is already in use by another connection.
- **Envelope Tag**: This lets the user set the env.tag property on a broker event message.

Subscriber Info

Select the enable check box to set up to enable the ability to receive (subscribe to) messages.

Enter the following parameters:

- **docType**: Enter the name of the docType to use.
- **Timeout (secs)**: Enter the period to wait before there is an interrupt waiting for a message (this field can be left blank for no timeout).
- **Async Key**: Enter the value needed to identify asynchronous messages. This is only needed in asynchronous mode. It will be used in a subsequent Message Consumer step to retrieve asynchronous messages.
- **Auto convert to**: Enter **string** to call the toString() function on the payload object to return its string representation; **xml** returns the payload in XML format.

ReplyTo Info

Select the enable check box to set up a destination queue/topic.

If your application needs a destination, it is set up in this section.

Enter the following parameters:

- **Name**: Enter the name of the topic or queue to use. The Search icon 🔍 can be used to browse the JNDI server for the topic or queue name.

Error Handling and Test

Error Handling and Test lets you redirect to a step if an error occurs.

- **If environment error**: Select the step to redirect to if an error occurs.

Click the Test button to test your step configuration settings.

Send Message Data Tab

This particular example shows how to select an object for the message. The message is stored in a LISA property.

Response Message Tab

If your step is configured to subscribe, your response will be shown here. For more information see JMS Messaging (JNDI)

webMethods Integration Server Services

The webMethods Integration Server Services step lets you execute Integration Server services through the native Java APIs. This is done using IData objects so it works with services not exposed through HTTP transports.

Prerequisites: You are required to supply the necessary JAR files for your chosen configuration and connection. See the list of JAR files in webMethods Broker.

Parameter Requirements: You need the connection parameters and the subject names used in the application under test. The following sections describe the parameters that you will need. There may be other parameters required, depending on your environment. Get these from the developers of the application.

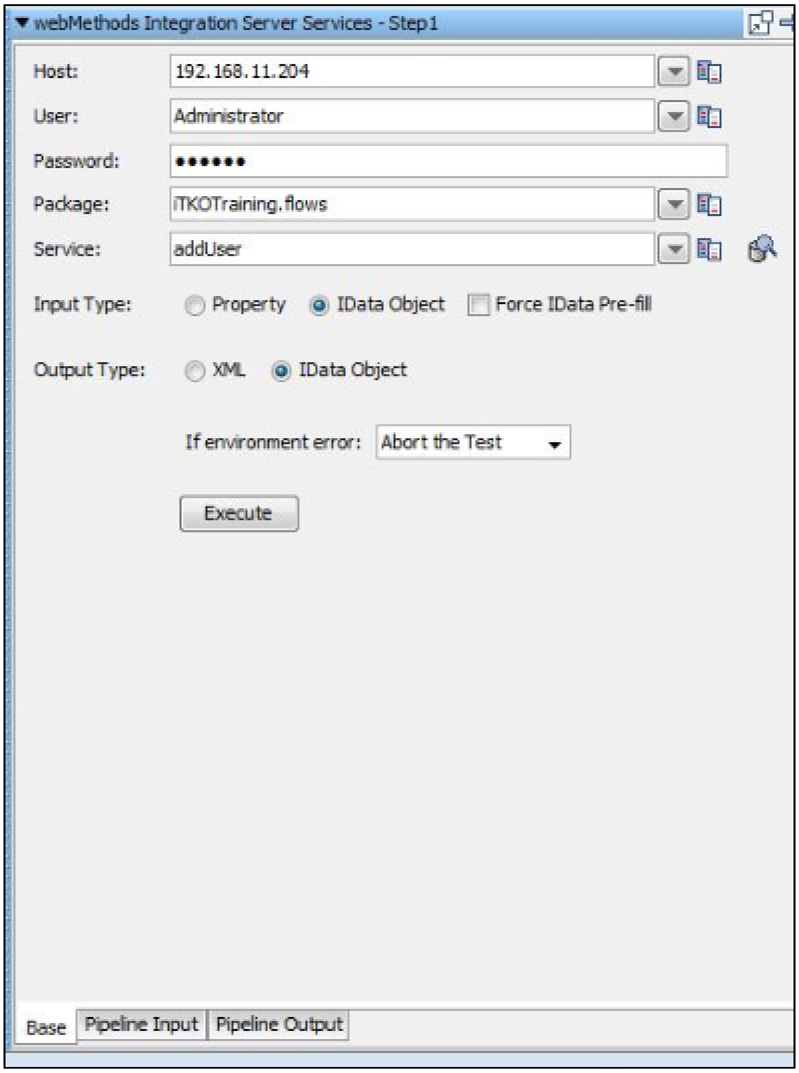
The messaging step editor for webMethods Integration Server Services has three tabs.

Base Tab: Server Connection Info

Enter the following parameters:

- **Host**: The host name.
- **User**: The userid.
- **Password**: The password.
- **Package**: The package the service is located in.
- **Service**: The name of the actual service you want to call.
- **Input Type**: Select type of input from **Property**, **IData Object** or **Force IData Pre-fill**.
- **Output Type**: Select the Output type from **XML** or **IData Object**.
- **If environment error**: Select the step to redirect to if an error occurs.

Click Execute to connect. You will see an object response. Export this object into a Java Execution Step to pull the payload or other properties from the response, which is an IData object itself. This can easily be done by creating a new Java Step in LISA and loading from property specifying the step name pattern for a last response. This is lisa.<stepName>.rsp.


webMethods Integration Server Services - Step1

Host: 192.168.11.204

User: Administrator

Password: ••••••

Package: iTKOTraining.flows

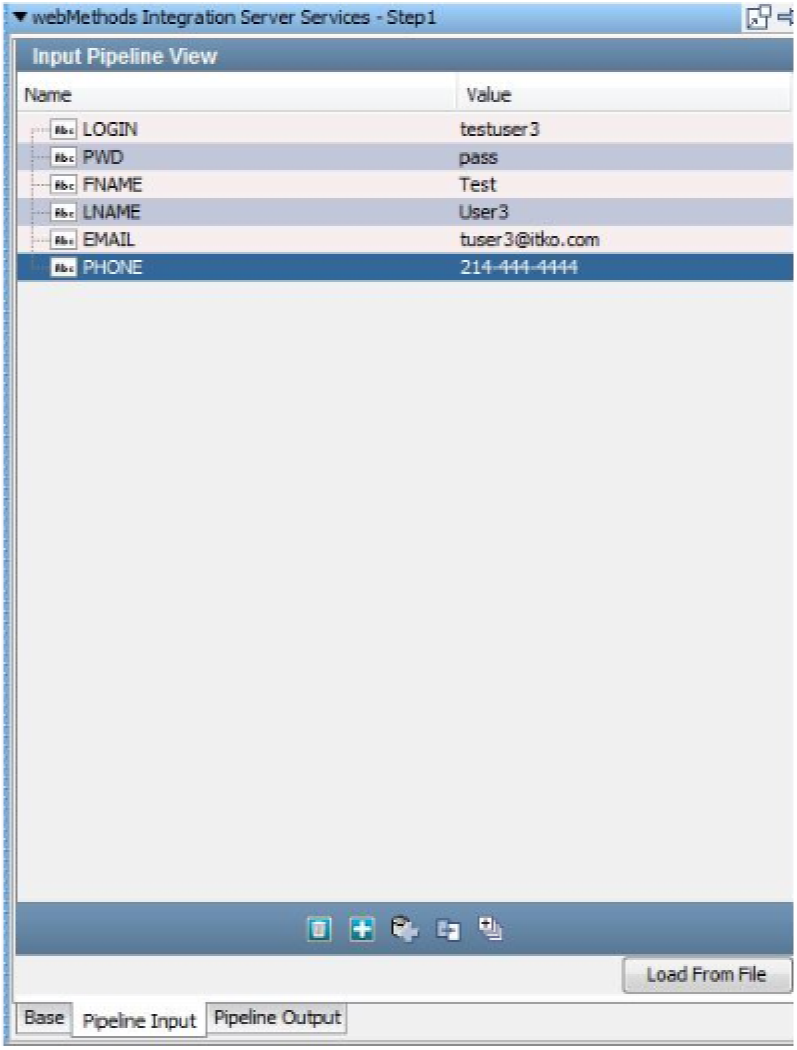
Service: addUser

Input Type: ○ Property ● IData Object □ Force IData Pre-fill

Output Type: ○ XML ● IData Object

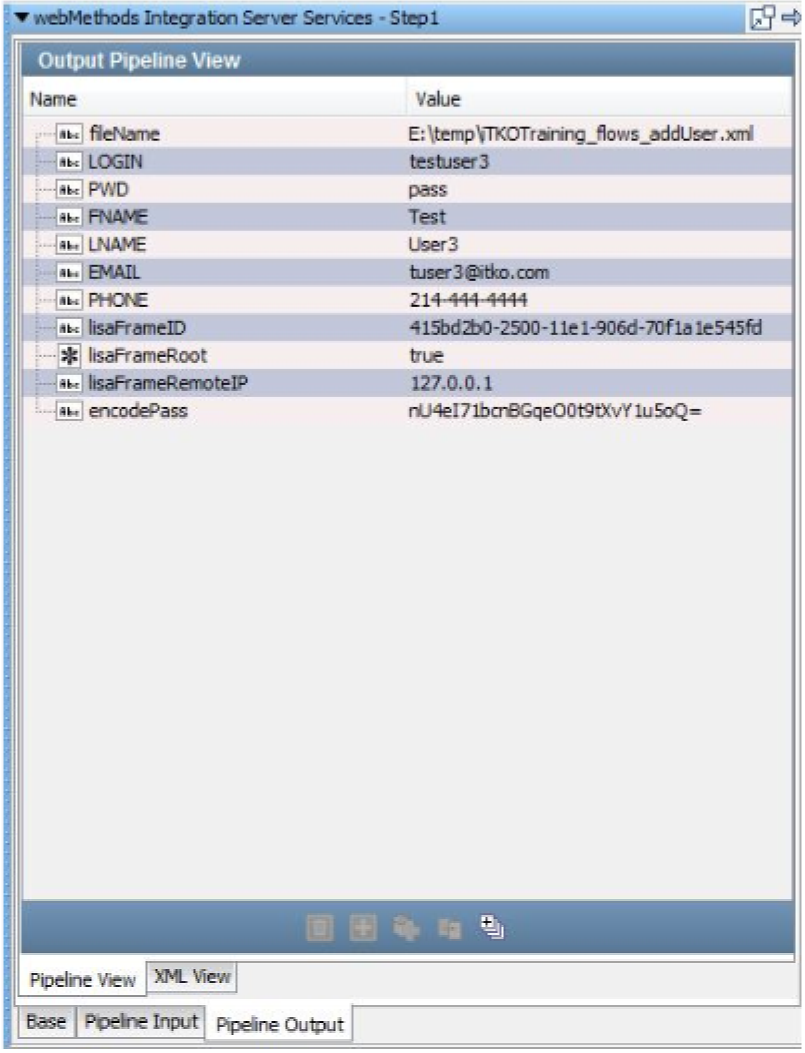
If environment error: Abort the Test

Pipeline Input Tab



Name	Value
LOGIN	testuser3
PWD	pass
FNAME	Test
LNAME	User3
EMAIL	tuser3@itko.com
PHONE	214-444-4444

Pipeline Output Tab



Name	Value
fileName	E:\temp\ITKOTraining_flows_addUser.xml
LOGIN	testuser3
PWD	pass
FNAME	Test
LNAME	User3
EMAIL	tuser3@itko.com
PHONE	214-444-4444
lisaFrameID	415bd2b0-2500-11e1-906d-70f1a1e545fd
lisaFrameRoot	true
lisaFrameRemoteIP	127.0.0.1
encodePass	nU4eI71bcnBGqeO0t9tXvY1u5oQ=

IBM Steps

The following steps are available in this chapter.

IBM WebSphere MQ
see **Message Consumer**

IBM WebSphere MQ

The IBM WebSphere MQ step lets you send messages to, and receive messages from, topics and queues. You can also receive, modify, and forward an existing message.

IBM WebSphere MQ supports all the common message types including Empty, Text, Object, Bytes and Message and Mapped (Extended).

The IBM WebSphere MQ step is configured using a single editor regardless of the messaging requirements. Input options will vary on the messaging requirements. The editor will only allow valid configurations, so when you enable certain features others may become inactive.

Prerequisites: To use WebSphere MQ, you must add several IBM JMS JAR files to the **LISA_HOME/lib** folder (or put them on the classpath some other way). The JAR files you need are:

- **WebSphere MQ Release 5.2**:

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

- **WebSphere MQ Release 6**:

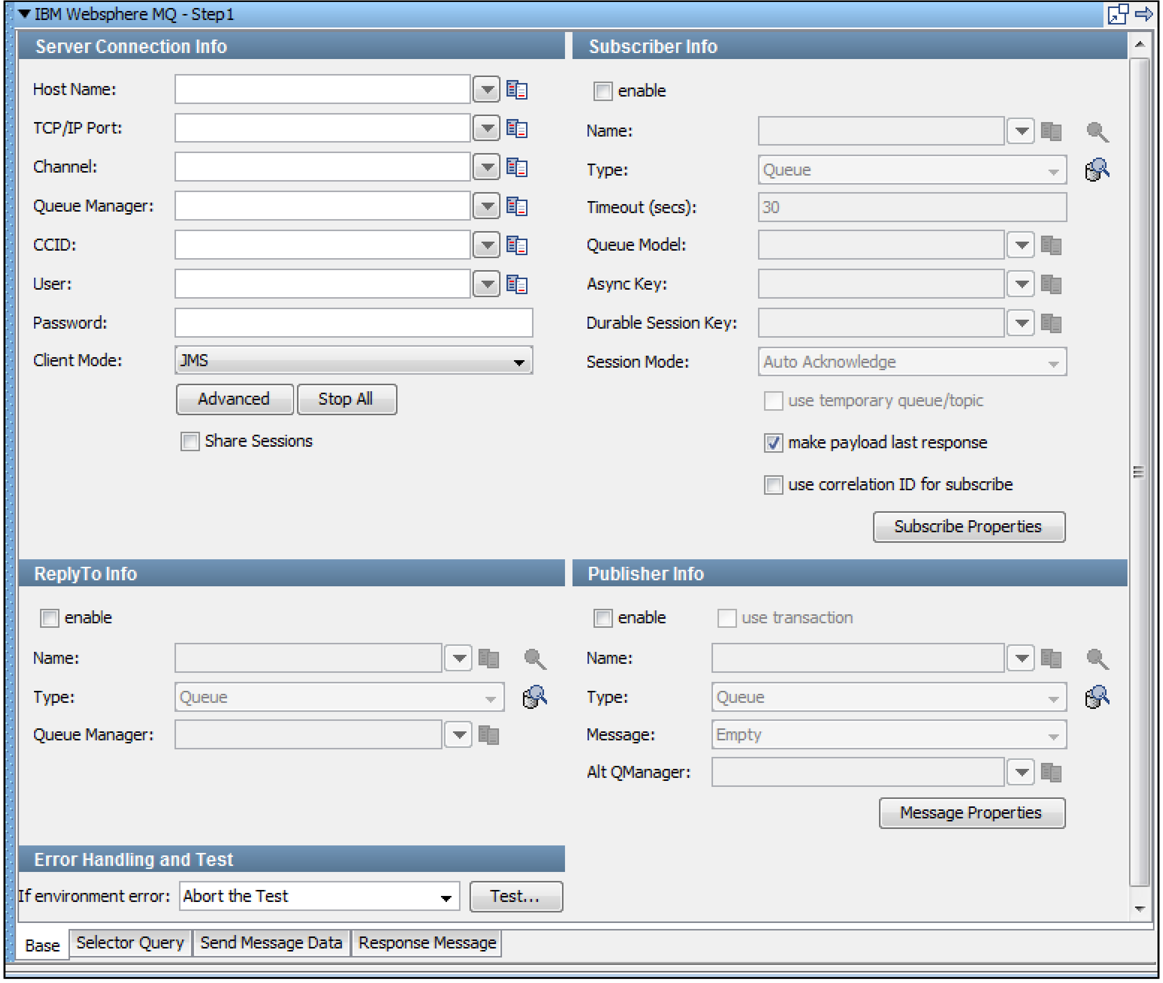
com.ibm.mq.jar
com.ibm.mq.pcf.jar
com.ibm.mqjms.jar
connector.jar
dhbcore.jar

- **WebSphere MQ Release 7**:

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

These can be found in your WebSphere MQ installation.

Parameter Requirements: You must have the connection parameters for your system under test. The following sections describe the parameters that you will need.



Server Connection Info
Client Mode: JMS
☐ Share Sessions

Subscriber Info
☐ enable
Type: Queue
Timeout (secs): 30
Session Mode: Auto Acknowledge
☐ use temporary queue/topic
☑ make payload last response
☐ use correlation ID for subscribe

ReplyTo Info
☐ enable
Type: Queue

Publisher Info
☐ enable ☐ use transaction
Type: Queue
Message: Empty

Error Handling and Test
If environment error: Abort the Test

The following tabs are available at the bottom of the messaging step editor for WebSphere MQ:

- The Base tab is where you define your connection and messaging parameters.
- The Selector Query tab lets you specify a selector query to be run when listening for a message on a queue.
- The Send Message Data tab is where you create your message content.
- The Response Message tab is where your response messages will be posted.

Base Information (Base tab)

The Base tab view is shown in the previous example. It is divided into five major sections: Server Connection Info, Subscriber Info, Publisher Info, ReplyTo Info, and Error Handling and Test.

The Server Connection Info and Error Handling and Test sections are always active.

The Subscriber Info, Publisher Info, and, ReplyTo Info sections can be enabled or disabled using the enable check box in the top left corner of each section. Using these check boxes, you can configure the step to be a publish step, a subscribe step, or both. You can also choose to include a replyto component in the step. When you have completely configured your test step, use the Test button in the Error Handling and Test section to test your configuration settings.

Server Connection Info

To connect to WebSphere MQ, enter the following information:

- **Host Name**
- **TCP/IP Port**
- **Channel**: Familiar to WebSphere MQ users, a connection property that is used for routing and management in the message bus.
- **Queue Manager**: Familiar to WebSphere MQ users, a connection property that is used for routing and management.
- **CCID**: Optional for connections and will only apply if character transformation needs to occur between the client (LISA) and server.
- **User**
- **Password**
- **Client Mode**: Lets you select how you want to interact with the WebSphere MQ server.
  - **JMS**: A pure Java implementation based on the JMS specification. We recommend you use the JMS Transport Protocol instead of MQ if you want this implementation.
  - **Native Client**: A pure Java implementation using IBM-specific APIs.
  - **Bindings**: Requires access to the native libraries from a WebSphere MQ client installation. You must make sure these libraries are accessible by the LISA application runtime. In most cases, having these available in the PATH environment will work.
- **Share Sessions**: Select to specify sharing everything in MQ Native Mode, including the connection.

Publisher Info

Check the enable check box to set up the ability to send (publish) messages. Click the **use transaction** check box to execute a commit when the message is sent.

Enter the following parameters:

- **Name**: Enter the name of the topic or queue to use. Use the Search icon to browse the JNDI server for the topic or queue name.
- **Type**: Select whether you are using a topic or queue. Use the Browse icon to the right of this field to see what messages are waiting to be consumed from a queue (only).
- **Message**: Select the type of message you are sending from the pull-down menu. Supported messages are: Empty, Text, Object, Bytes, Message, and Mapped (Extended).
- **Alt Qmanager**: TODO

Subscriber Info

Enable: Select the enable check box to set up to enable the ability to receive (subscribe to) messages.

Enter the following parameters:

- **Name**: Enter the name of the topic or queue to use. Use the Search icon to browse the JNDI server for the topic or queue name.
- **Type**: Select whether you are using a topic or queue, and whether you want to listen in synchronous or asynchronous mode. For asynchronous mode you also must have an entry in the Async key field. Use the Browse icon to the right of this field to see what messages are waiting to be consumed from a queue (only).
- **Timeout (secs)**: Enter the period to wait before LISA interrupts waiting for a message (this field can be left blank for no timeout).
- **Queue Model**: This is required by MQ to create temporary destinations. It is configured on the MQ server. It is only active when use temporary queue/topic is checked. In this case the ReplyTo Info section is disabled.
- **Async Key**: Enter the value needed to identify asynchronous messages. This is only needed in asynchronous mode. It will be used in a subsequent Message Consumer step to retrieve asynchronous messages.
- **Durable Session Key**: By entering a name here you are requesting a durable session. You are also providing a key for that session. A durable session lets you receive all your messages from a topic even if you log out, and then log in again.
- **Session Mode:** Using the drop-down list, select from:
  - **Auto Acknowledge**: TODO
  - **Client Acknowldge:** TODO
  - **Use Transaction**: To execute a Commit when a message is received.
  - **Auto (Duplicates Okay):** TODO
- **use temporary queue/topic** check box: Click the use temporary queue/topic check box if you want the JMS.provider to set up a temporary queue/topic on your behalf. When a temporary queue/topic is used, the JMS ReplyTo parameter of the message you send to the temporary queue/topic is automatically set. The temporary queue/topic feature must always be used with a publisher so that a reply can be sent. If you use a temporary queue/topic the ReplyTo section is disabled.
- **make payload last response**: Check this option if you want to make payload as the last response.

- **use correlation ID for subscribe**: TODO.

ReplyTo Info

Select the enable check box to set up a destination queue/topic.

If your application needs a destination, it is set up in this section.

Enter the following parameters:

- **Name**: Enter the name of the topic or queue to use. Use the Search icon to browse the JNDI server for the topic or queue name.
- **Type**: Select whether you are using a topic or queue. Use the Browse icon to the right of this field to see what messages are waiting to be consumed from a queue (only).
- **Queue Manager**: allows the replyTo to be on a different Queue Manager than the Publisher (in this step).

Error Handling and Test

Error Handling and Test lets you redirect to a step if an error occurs.

- **If environment error**: Select the step to redirect to if an error occurs.

Click the Test button to test your step configuration settings.

The other tabs contain fields and parameters that are thoroughly documented in JMS Messaging (JNDI).

Virtual Service Environment Steps

The Virtual Service Environment test steps are available in the *Virtualize Guide*.

Virtual Service Router Step
Virtual Service Tracker Step
Virtual Conversational_Stateless Response Selector Step
Virtual HTTP_S Listener
Virtual HTTP_S Live Invocation Step
Virtual HTTP_S Responder Step
Virtual JDBC Listener Step
Virtual JDBC Responder Step
Socket Server Emulator Step
Messaging Virtualization Marker Step
Compare Strings for Response Lookup Step
Compare Strings for Next Step Lookup Step
Virtual Java Listener Step
Virtual Java Live Invocation Step
Virtual Java Responder Step
Virtual TCP_IP Listener
Virtual TCP_IP Responder

Custom Extension Steps

The following steps are available in this chapter.

Custom Test Step Execution
Java Script Step
Pathfinder Agent Script Step
Swing Test Step
Create a Virtual Web Service

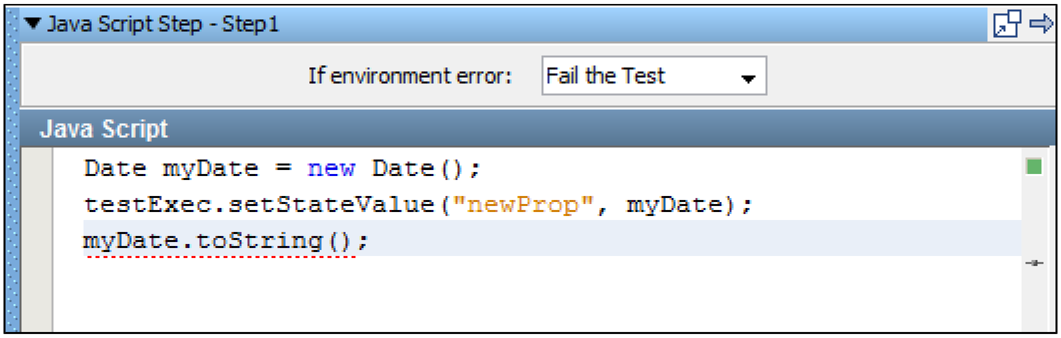
Custom Test Step Execution

The Custom Test Step executes a test step custom written by your team using the LISA SDK. This step is documented in the *Developer's Guide (SDK)*.

Java Script Step

The Java Script step gives you the flexibility of writing and executing a Java script to perform some function or procedure. Your script is executed using the BeanShell interpreter. You have access to all the LISA properties in the test case, including built-in objects.

Prerequisites: Some knowledge of BeanShell. For more information on BeanShell, see http://www.beanshell.org/.



If environment error: Fail the Test

```
Date myDate = new Date();
testExec.setStateValue("newProp", myDate);
myDate.toString();
```

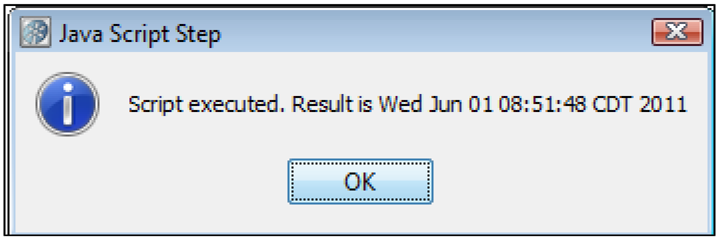
This is the script editor where you write your scripts. Double-clicking on an item in the list will paste that variable name into the editor. The last value exposed in the script will be saved as the response of this step.

Click the Test button to test your script. You will see the result from executing the script, or an error message describing the error that occurred.

The example in the preceding illustration shows that:

- A new Date object was created, initialized to the current date and time.
- This object was stored in a new LISA property, **myprop**, using one of LISA's exposed objects, **testExec**. For more information, see the *LISA Developer's Guide*.
- The toString() value of the Date object was set as the response of the step.

On testing this script, the following screen is displayed.



Script executed. Result is Wed Jun 01 08:51:48 CDT 2011

LISA property name syntax is very flexible and can include spaces. Property names that are not valid Java identifiers are converted for use in this step. Invalid characters are automatically replaced by an underscore (_).

Some things to remember:

- If you use LISA properties `{{{}`**`exampleprop}}`** in a script, it will be substituted for the actual value of the property at run time before the script is executed.

- If you need to get access to a property that has a "." in the name, LISA imports these into the script environment replacing "." with "_". So {{{}**`foo.bar}}`** in a script is the same as foo_bar.

- You can produce a LISA log event very easily inside a script step or assertion. There is a **testExec** object that is very useful. To produce a LISA log event, you can code

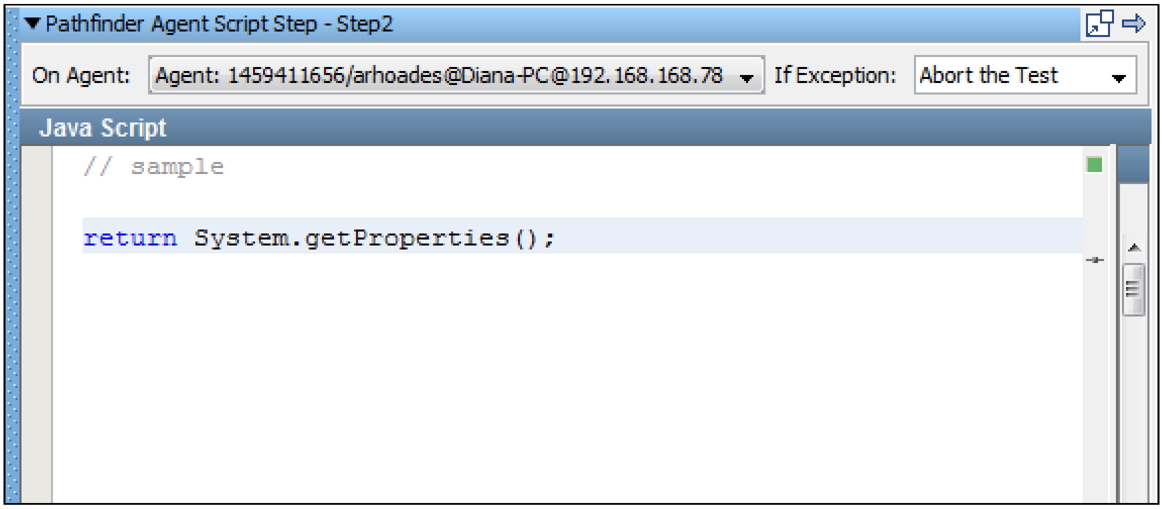
  ```
  testExec.log("Got here");
  ```

  as opposed to using the log4j logger. The **testExec.log()** object causes an actual LISA event to be raised and you can see it in the ITR.

Pathfinder Agent Script Step

The Pathfinder Agent Script step is exactly the same as the Java Script Step, but instead of executing the supplied code inside of LISA Workstation (or Simulators), it executes inside the JVM where the specified agent is installed.

The following window is the script editor.



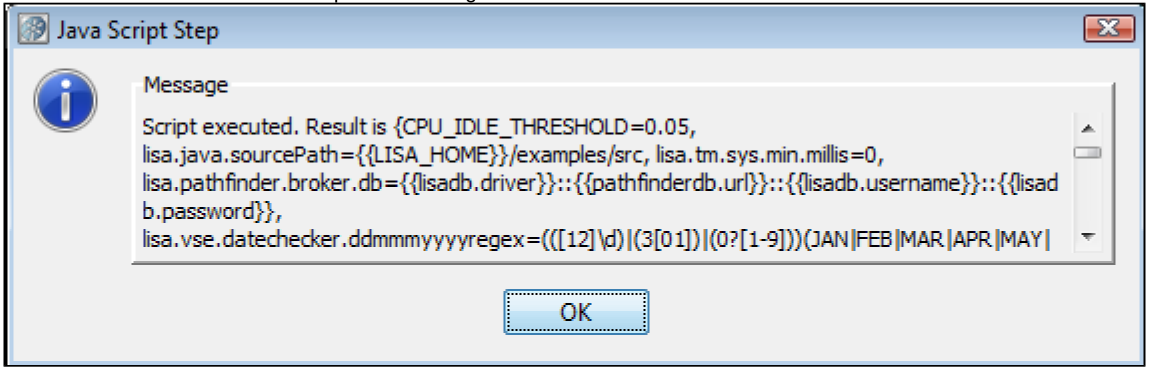
On Agent: Agent: 1459411656/arhoades@Diana-PC@192.168.168.78

If Exception: Abort the Test

```
// sample

return System.getProperties();
```

Click Test to execute the script.

Successful execution shows a completion message.



```
Script executed. Result is {CPU_IDLE_THRESHOLD=0.05,
lisa.java.sourcePath={{LISA_HOME}}/examples/src, lisa.tm.sys.min.millis=0,
lisa.pathfinder.broker.db={{lisadb.driver}}::{{pathfinderdb.url}}::{{lisadb.username}}::{{lisad
b.password}},
lisa.vse.datechecker.ddmmmyyyyregex=(([12]\d)|(3[01])|(0?[1-9]))(JAN|FEB|MAR|APR|MAY|
```

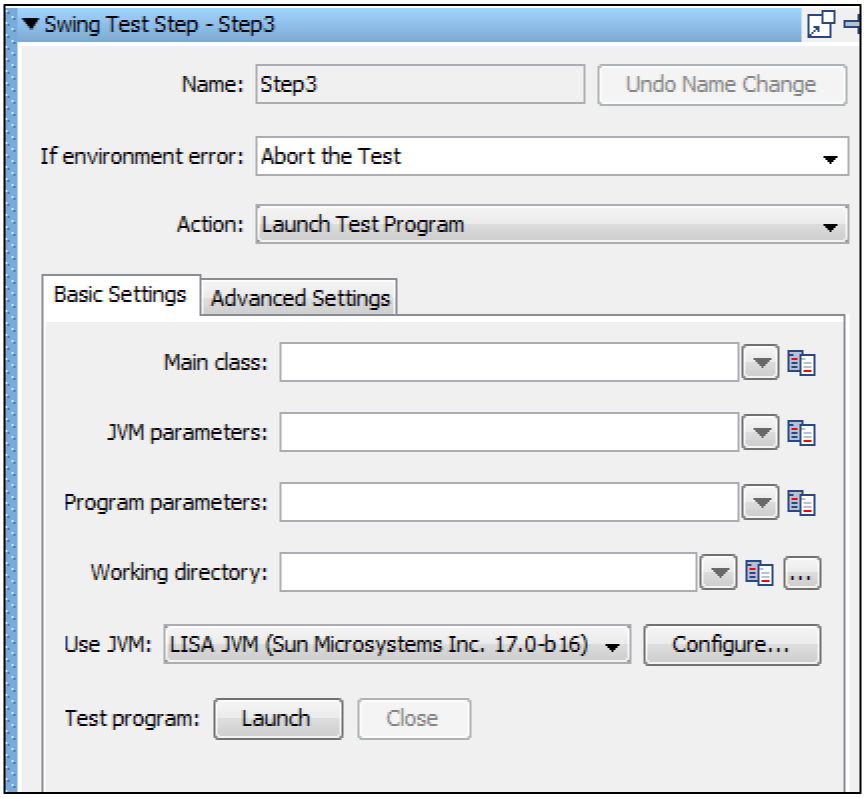
This step is used in EJB baseline test cases.

Swing Test Step

The Swing step lets LISA listen to mouse and keystrokes on a Java Swing application, record them and play them back.

To do this, LISA attaches to the AWT listener of the recorded application; the main class and JAR files for the application are required. The Swing recorder will remotely launch the application and listen to the AWT events and then send the events back to LISA for recording. After recording is completed the AWT events are played back to the application just as if the keyboard and mouse were being used.

A component browser is provided to look at the Java objects in the application and find information about the running object. From LISA every property and method provided by the object can be run, filtered and then asserted on. Assertions are created by attaching to the Swing Java object and running methods against it; for example, if a radio button isEnabled or isChecked can be validated.



Name: Step3

If environment error: Abort the Test

Action: Launch Test Program

Main class:

JVM parameters:

Program parameters:

Working directory:

Use JVM: LISA JVM (Sun Microsystems Inc. 17.0-b16)

Test program:

The top panel of the Swing Test Step Editor contains:

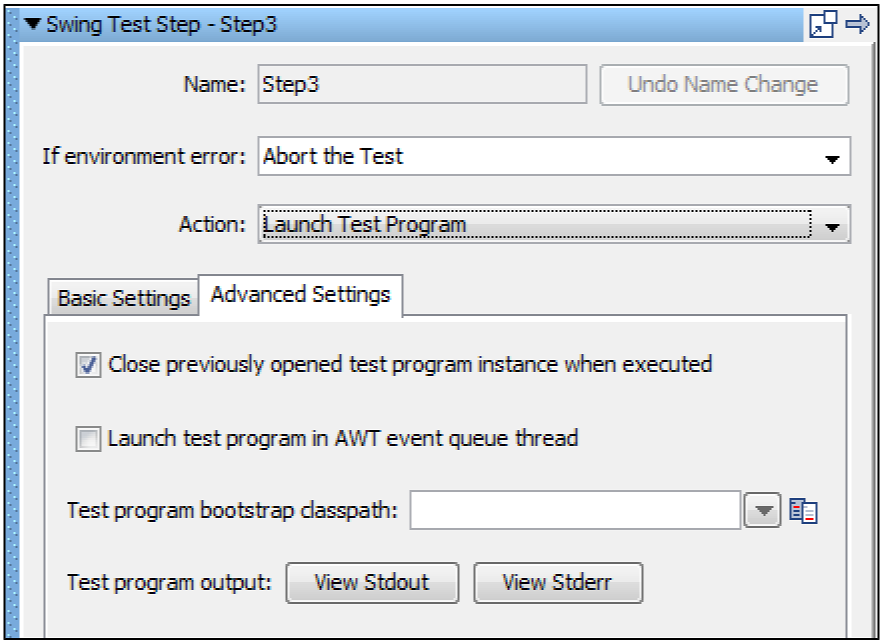
- **Name**: Enter the fame of the test step.
- **If Environment Error**: Select the necessary step in case of an error, from the drop-down options.
- **Action**: Select the necessary step to be taken from the drop-down options. **Mouse Input** is created by LISA when recording mouse movements from the AWT listener. **Key Input** is created by LISA when keys are pressed and recorded from the AWT listener. **Invoke Component Methods** are used to attach to components (buttons, text, frames, and so on) and run methods on them.

The Swing Test Step Editor has two tabs: Basic Settings and Advanced Settings. By default it opens in the Basic settings tab.

Basic Settings Tab

- **Main Class**: Contains the class name with the Java main method.
- **JVM parameters**: Any parameter to be sent to the JVM is entered here. The --classpath parameter contains all the .jar files to run the applications. If the application jars are in the hot deploy directory, this does not need to be set.
- **Program parameters**: Set any arguments that the application is expecting.
- **Working directory**: Points to the directory in which the application to be tested is installed. When the application is started, it is started from this directory.
- **Use JVM**: This will default to the JVM that LISA is running. An alternate JVM can be specified if the application requires a different JVM.
- **Test program**: Click the Launch button to open the application. The Close button closes the application.

Advanced Settings Tab



Name: Step3

If environment error: Abort the Test

Action: Launch Test Program

☑ Close previously opened test program instance when executed

☐ Launch test program in AWT event queue thread

Test program bootstrap classpath:

Test program output:

- **Close previously opened test program instances when executed**: Close any open applications and open a new one when playing back the steps recorded from the application. This makes sure the application always starts at the beginning of where the test case was recorded and prevents duplicate applications from being run.
- **Launch test program in AWT event queue thread:** The location of the AWT listener, so it can be attached to. If the Java application does not record in LISA, clear this box and LISA will try attaching to the AWT listener in the main of the program.
- **Test program bootstrap classpath**: This can be set for the application being tested. Only use the bootstrap if the application is getting errors when loading classes. The classes will need to be moved from the classpath (-classpath "myjar.jar" or LISA hot deploy) to the bootstrap.
- **Test program output**: This allows for viewing of the Java application messages that are sent to Stdout (standard out) and Stderr (standard error) from the launched application.

Mouse Input

A Mouse Input step is created when recording a Swing application.

The information about which button, any extra keys complementing the click and the location is recorded. The component that was clicked on is listed in the Component window.

Pressing the Select button will navigate the component browser to the object that was clicked. When a mouse click is played back, the component is found and the mouse is clicked in the middle of the component. This enables LISA to be tolerant to changes in a Swing User Interface.

Key Input

Key Input is recorded by LISA when a keystroke is done on the keyboard.

The recording can be an action like ALT + TAB to change between windows of the application or the typing of letters into a field. When a single letter is typed, the Key pressed, Key typed, and Key released steps are recorded to correspond to all actions. The component that the key action was taken on is listed and can be viewed in the component browser by pressing the Select button.

Invoke Component Methods

The Invoke Component Methods action is added by the tester when a component is available in the running application and methods of the component are to be run. For example, if there is a frame in the application and the title of the frame is wanted. To do this, Invoke Component Methods action is inserted into the workflow, the component is selected and the method of getTitle is available.

Component browser shows the active components of a Java application. The application must be launched from the Launch program step so that the component browser can attach to it. If the Component Browser does not navigate to the component, verify there is not an asterisk next to the title *Component Tree, indicating a refresh of the tree is required. The other reason a component will not show up could be that the application has not been launched and the component is not available in the application; for example, it could be on a folder tab that is not visible.

When the Select button is pressed, the component is sent back to the LISA step with information about the object. The data sheet contains information about the object. The call sheet will let you run methods on the object like getTitle().

Assertions can now be done on the results of the method calls just like any other object in LISA.

Create a Virtual Web Service

The Create Virtual Web Service test step has been deprecated in favor of VSE and Virtual Service from WSDL.

For more information, see this tech note.

Creating Test Cases

A test case is a complete specification of how to test a business component in the "system under test," or in some cases the complete "system under test".

A test case is persisted as an XML document, and contains all the information needed to test the given component or system. Test cases are created and maintained in LISA Workstation.

The first step in creating a LISA test case is to create a LISA project. Within this project, you can create single or multiple test cases.

The following topics are available in this chapter.

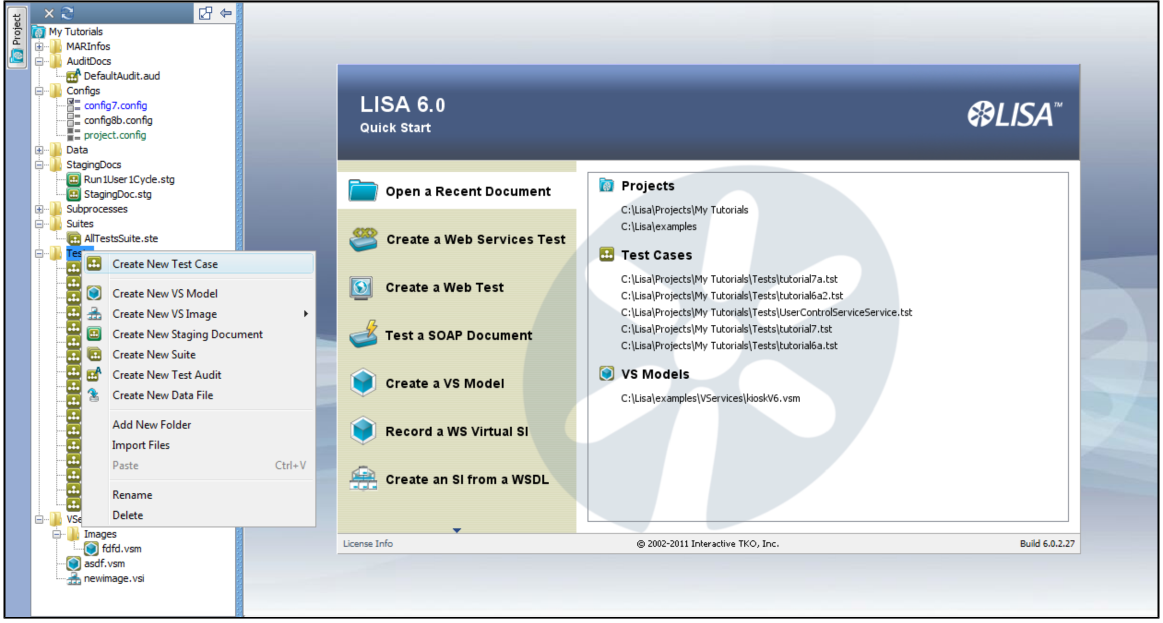
Creating a Test Case

You can create a test case by opening a project or creating a new project.

For example, if you open the default project "examples" within LISA, it opens with a tree structure that has folders for Configs, Data, Staging Doc, Suites, Tests, and others.

To create a test case

1. In the Project tree, right-click the Tests folder and click Create New Test Case.



2. In the dialog window, browse to the directory where you plan to store the test case and enter the name of the new test case.

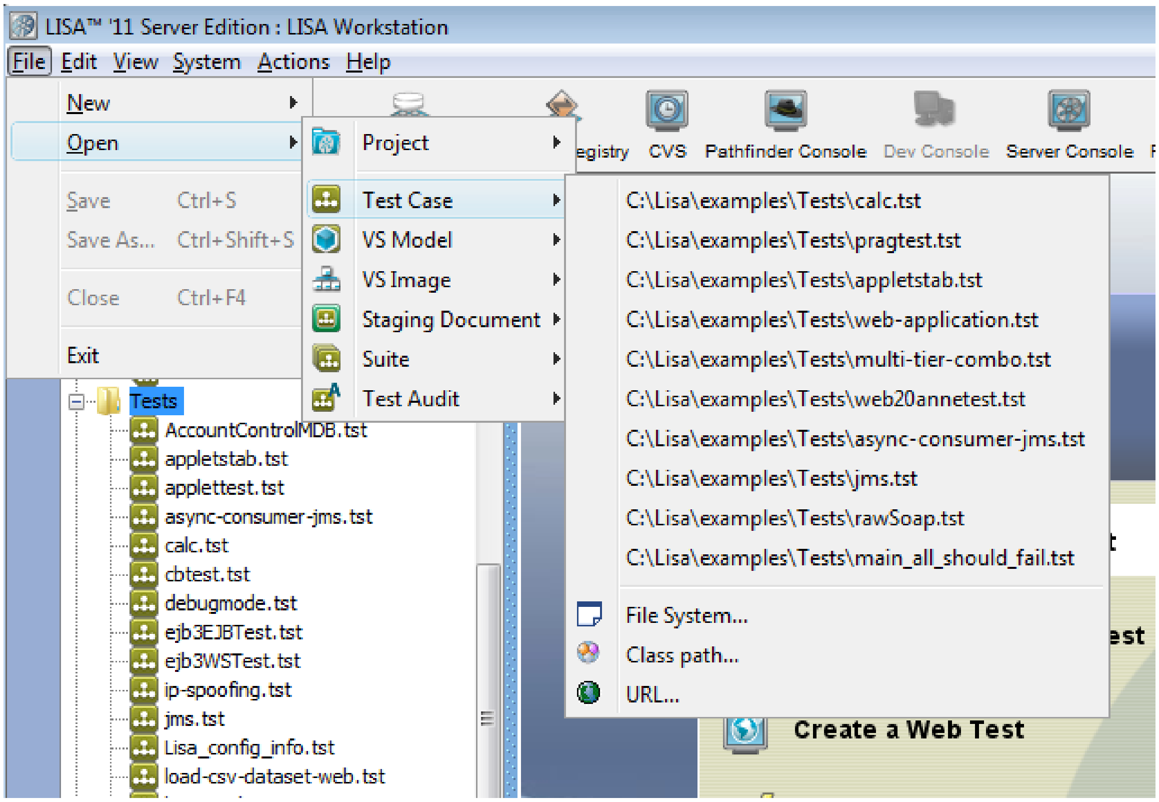
See Test Cases in Model Editor for more information.

Opening a Test Case

To open or view an existing test case

1. From the main menu, select File > Open > Test Case.

2. The recently-opened test cases are displayed. The test cases are listed in order of use; the most recently-opened is at the top of the list. If the target test case is in the list, select it. If it is not, then browse to it by clicking File System, Class path or URL.



3. Select the file you want to open, and click Open. The selected test case will be opened and displayed in the model editor. You can also open an existing test case by double-clicking on it from the project panel.

4. You are now ready to add new elements (filters, assertions), or modify the existing elements in the test case.

Saving a Test Case

When you save a test case, LISA also saves the results of each step in the test case to a Response document, with a suffix of ".rsp" in the same directory. For more information, see Response Documents.

If a test step element, filter, assertion, or data set has a required field and that field is not filled in, then you cannot save the test case.

To save a test case

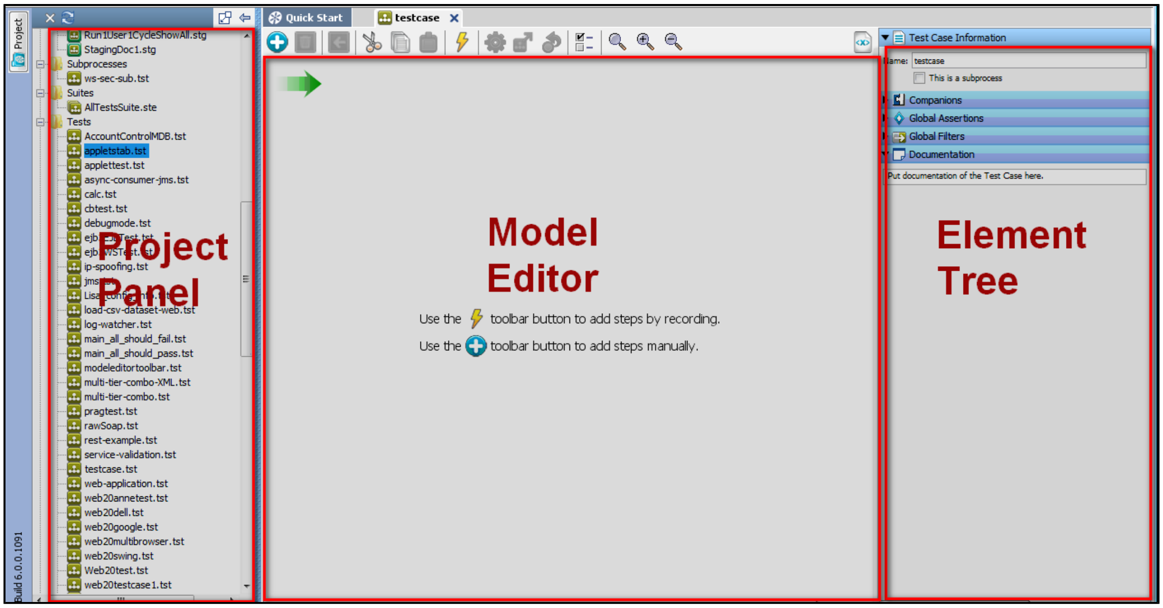
- Do one of the following:
  - Click the Save Save icon on the toolbar.
  - From the main menu, select File > Save (Sample). LISA displays the original test case name for you.

Test Cases in Model Editor

Test Cases in Model Editor

When you open a test case in LISA, its graphical user interface shows a model editor, which is used to create and manage test cases.

When you open an already existing test case, the view in the Model Editor provides a graphical view of the test case, with all the test steps and elements attached to it. For sample examples, you can open the test cases in the %LISA_HOME%/examples directory.



Use the toolbar button to add steps by recording.

Use the toolbar button to add steps manually.

Name: testcase

This is a subprocess

Put documentation of the Test Case here.

The model editor has three sections:

- **Project Panel**: To create a project and under which you create, view, edit test cases or other LISA documents.
- **Model Editor**: To add, edit, delete the test steps created for a test case.
- **Element Tree**: To apply filters, assertions, data sets, companions to a test case or test step.

Adding Test Steps

Steps can be added in several ways.

To add a test step

- From the main menu, select Commands > Create a New Step

- From the Test Case toolbar, click the Add Step icon

- In the workflow, right-click a step and click Add Step After. This adds a new step right after/below that step.

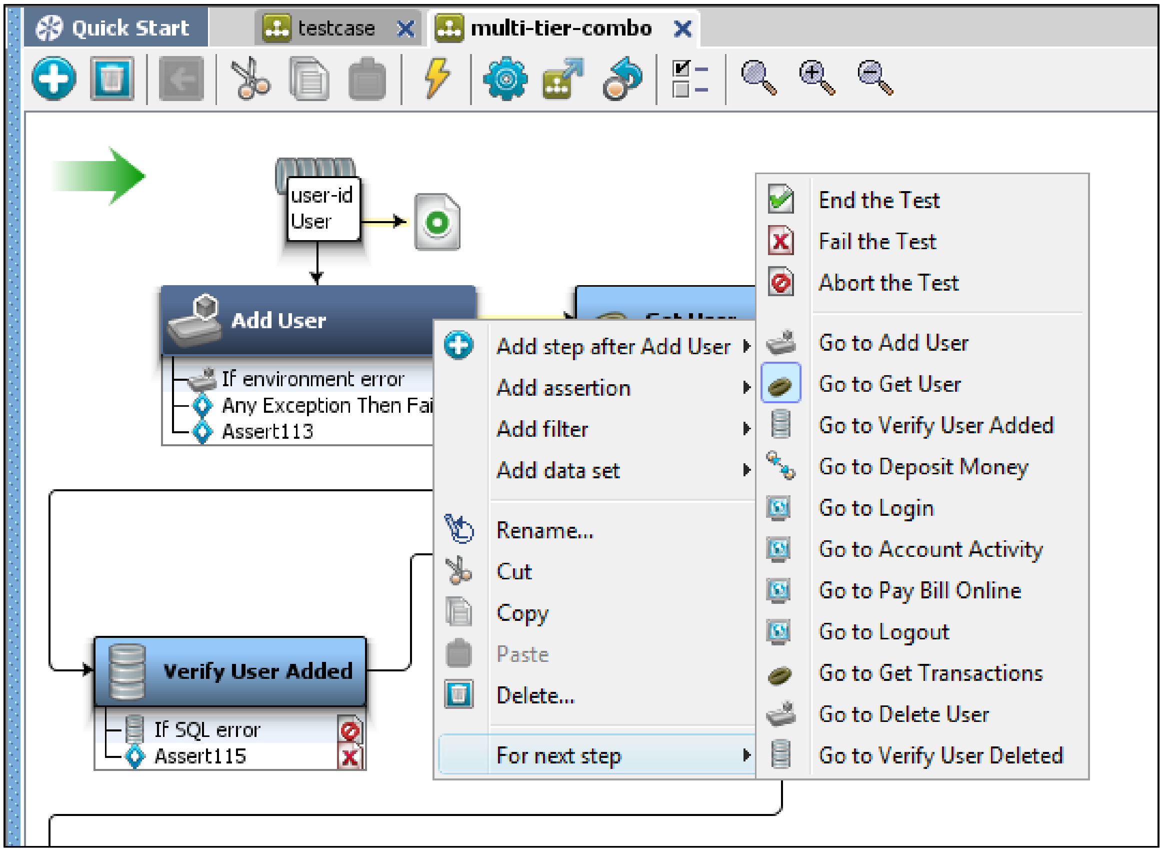
When a step is inserted into an existing workflow, and workflow is linear, the steps on either side of the inserted step will be updated automatically, keeping the workflow linear.

If the workflow contains branches or loops, the next step will not be set automatically.

Configuring the Next Step

To configure the Next step within the model editor

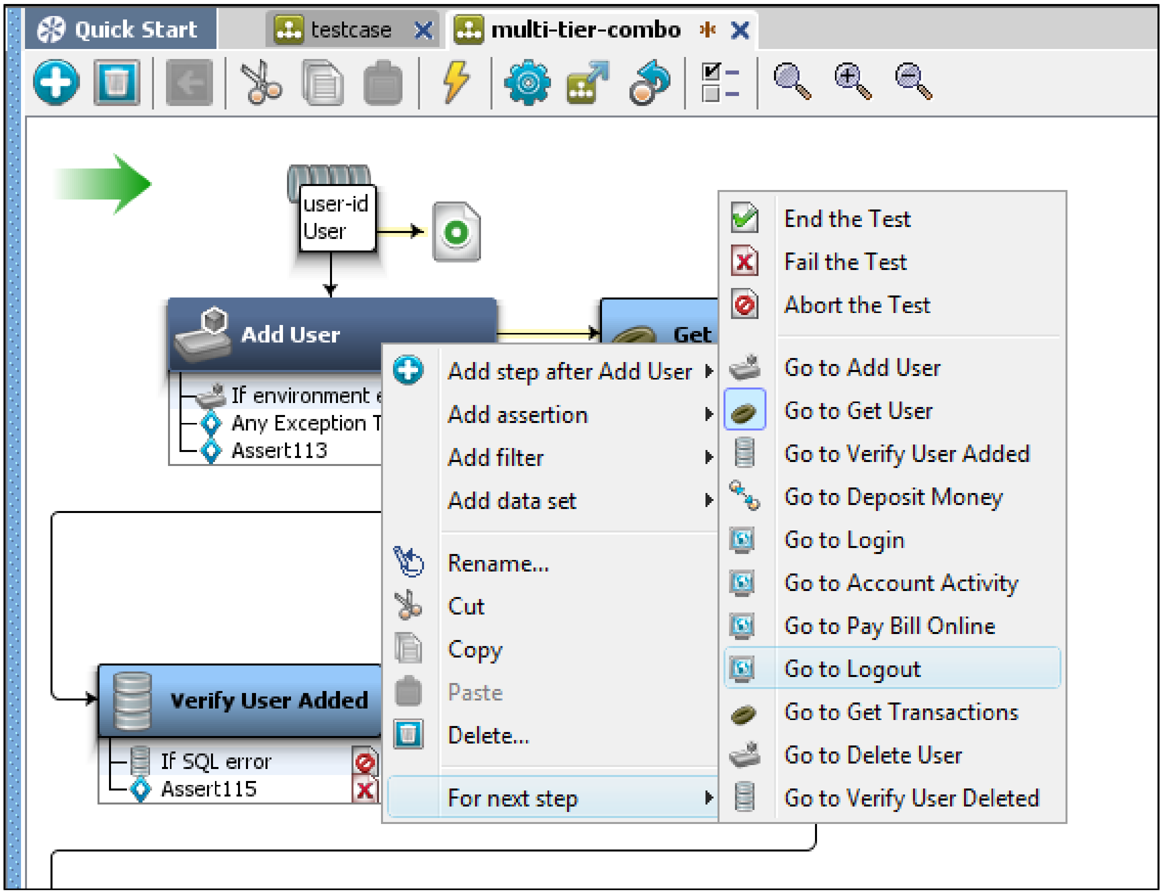
1. Select the target test step and right-click to open a menu. In the following example, the Add User step was selected.
2. Click For next step and select the target next step.



Reordering Test Steps Within a Test Case

To recorder the steps within the model editor

1. Select a step and right-click to reorder in a workflow.



2. For the previous example, select the Add User step and right-click to open a new menu.
3. Click For next step**,** and select the step to set as the next step.

You can also drag and drop in the steps within the model editor. All the steps will be updated automatically if the workflow is linear. When the workflow is non-linear, which means it contains branches and loops, the next steps will not be updated. In non-linear workflows the next step can be updated in the Step Information tab of that step.

Branching and Looping in a Test Case

Branching in a Test Case

Branching in a test case is done using assertions.

Any number of assertions can be applied to a test step. Every assertion has a condition that evaluates to true or false. The first assertion for which the assertion condition is satisfied gets fired, and that changes the path of the workflow. In many steps, a default error condition is automatically created as an assertion to failure. When creating assertions, it is best to be consistent, and always branch positive, or always negative.

Assertions are described in detail in Adding an Assertion.

Looping in a Test Case

Loops are created when a test step down the flow from a given test step sends control to the test step that started the flow, as in a loop. What is important is the ability to come out of the loops. As you may imagine, conditionally breaking out of a loop (equivalent to a "while" loop in programming) is achieved by setting assertions on a test step participating in the loop.

The other kind of loops that execute a given number of times or until a particular data is exhausted (equivalent to "for" or "foreach" loops in programming), are achieved with data sets. A data set is used to assign value(s) to one or more properties a finite number of times. The next step that needs to be executed after the data set is exhausted is specified with the data set definition, and this can be used to break the loop.

For example: if a data set contains 20 rows of users that need to be logged into a system, a loop can be created to run the login step for each row in the data set. Alternatively, a numeric counting data set can be used to cause a specific step to execute a fixed number of times.

A single step can call itself, and loop over a data set. Several steps can run in a loop, and use the data from a data set. This is accomplished when the final step in the group of steps points to the first step in the group.

Data sets are described in detail in Data Sets.

Importing Test Cases

You can import old test cases into the current working project directory.

To import test cases

1. Right-click the Tests folder in the Project tree
2. Select the Import Files option.
3. Select the files to be imported into that folder.
4. Click OK.

The process of importing starts and copies all the files (including the files within sub-directories) to the selected folder. It also merges all the configs from within the test cases and VS models into the project configs.

To import older versions of files (versions less than 5.0 or anything created before LISA 4.7),

1. From the main menu, select File > New > Project.
2. Select the option to Create Project from existing LISA documents directory.
3. Select a directory that has test cases from the older version.
4. Click Create.

This will create the project with all old version files converted to the new version. If a project with the same name exists, it will ask for another name for the new project.

With other project creation messages, you will also see the auto-convert message for all the test cases and VS models that were transformed to meet the new version's requirements.

After completion, you are able to see the following messages:

- Migration of the project
- Configuration change details
- Test case change details

All the imported files are selected.

Response (.rsp) Documents

When you record from a website or interact with a server, LISA saves the responses into a response document, so that it can refer to the information later.

LISA creates and maintains response documents automatically, with no effort on your part, and saves them as files with the same name as the test case file with an **.rsp** extension. Like the test case files, the response documents are XML files.

A response document maintains the HTTP response for each of the HTTP-based steps in a test case, the response information from web service calls, and JDBC results from a database query.

For example, if you have executed the HTTP-based steps by using the Interactive Test Run (ITR) utility, the saved response document contains the entire DOM tree for the result of each HTTP-based step in the test. You can use this response information to validate data, create simple filters or create simple assertions.

For more information on using HTTP responses to create filters, see Filters.

For more information on using the HTTP response to create assertions, see Assertions.

Not all steps have results that are amenable to storage in a response document.

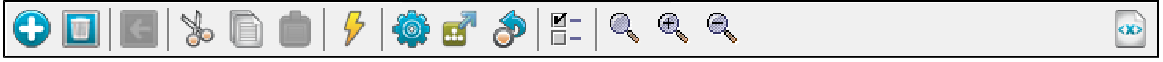
If you copy a test case file to another location, consider copying the associated response document also, so that you do not lose the saved responses.

Response documents are optional, in that they are not necessary for LISA to run tests. They exist to give you the ability to view results, view DOM Tree, view JDBC table, and more.

When you use the replay function in LISA, information is read from the response document.

Test Case Toolbar

This toolbar opens after you open a test case in the model editor. All the tasks here are specific to a test case.



Icon	Description
⊕	Create New step
🗑	Delete a test step
⬅	Set the currently selected step as the starting step in the workflow
✂	Cut the selected text
⧉	Copy the selected text
📋	Paste the selected text
⚡	Click to open a menu and create steps by recording a test case in the LISA browser. You can: • Record a test case via Proxy • Record a test case via DOM Events
⚙	Start a New Interactive Test Run
↗	Stage a Quick Test
↶	Replay Test To Specific Point
☑	Show Model Property
🔍	Reset zoom scale to 1:1
+	Zoom In
−	Zoom Out
<X>	View XML source

Building Subprocesses

A subprocess is a test case that is designed to be called from another test case, rather than run as a stand-alone test case.

Subprocesses can be used as modules in other test cases, hence increasing their re-usability. You can build a library of subprocesses that can be shared across many test cases.

In computer programming, a subprocess would be referred to as a function or a subroutine.

A test case must be self-contained; that is, the value for all the properties used in the test case must come from within the test case. A subprocess expects some property values to be provided by the test step that invokes it (input properties), and when the subprocess completes, it makes property values available to the calling step (return properties).

You can create the steps in the subprocess in the same way you would for a regular test case, with the following differences:

- Mark the test case as a subprocess in the Test Case Information tab of the test case (as explained in Creating a Subprocess Test Case).

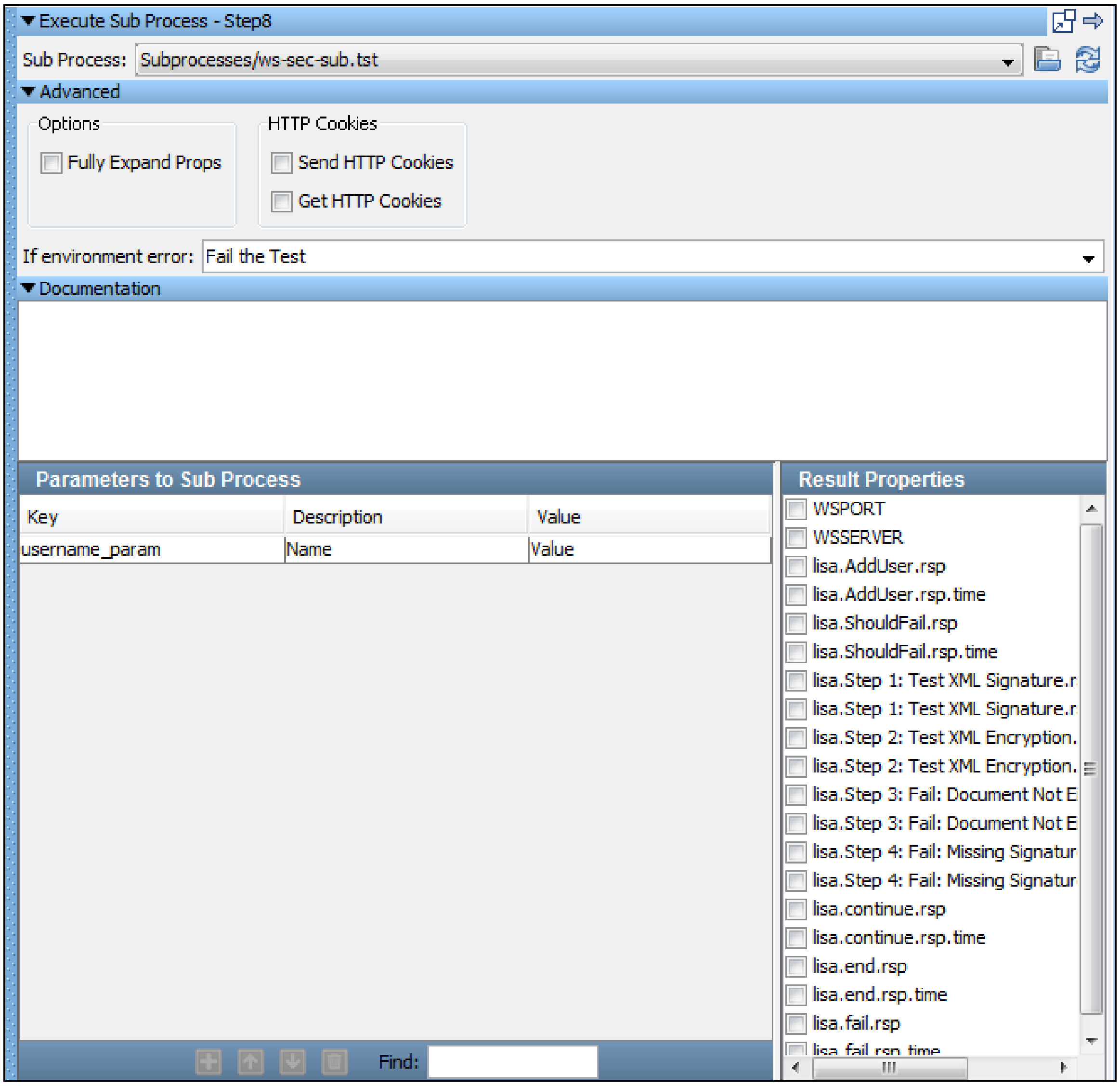
- Do not add data sets like you would in a test case. Instead, the data set should be part of the calling case, and the current values passed to the subprocess when it is invoked. The exception here is when a data set is an integral part of the subprocess logic itself. In that scenario, a local data set should be used.

- Do not use a configuration file or a companion inside the subprocess to initialize any parameters that you expect to be passed from the calling step. For testing purposes, we add these values elsewhere. When the subprocess is called, the calling step will pass these values.

> ⚠ The think time parameter in the parent test case (the test case that calls the subprocess) is propagated to the subprocess. If you need your subprocess think times to run independently of the calling process, ensure you have a testExec property named **lisa.subprocess.setThinkScaleFromParent** set to "false" to let you decide on a per-subprocess basis. For a global override, set **lisa.subprocess.setThinkScaleFromParent**=false in local.properties.

You can build subprocesses from scratch, or you can convert an existing test case into a subprocess.

The Execute Subprocess test step makes it easy to call a subprocess test case.



▼ Execute Sub Process - Step8

Sub Process: Subprocesses/ws-sec-sub.tst

▼ Advanced

Options

□ Fully Expand Props

HTTP Cookies

□ Send HTTP Cookies

□ Get HTTP Cookies

If environment error: Fail the Test

▼ Documentation

Parameters to Sub Process

Key	Description	Value
username_param	Name	Value

Find:

Result Properties

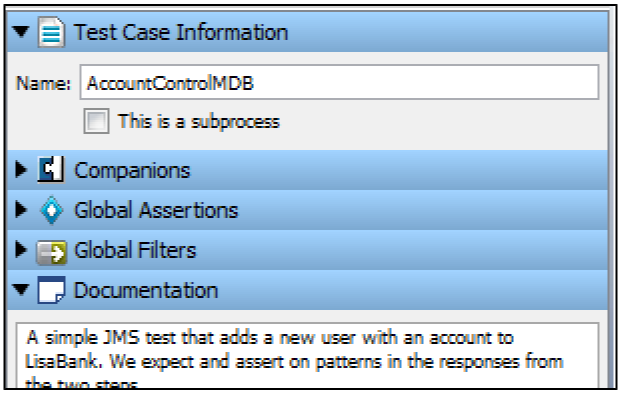
□ WSPORT
□ WSSERVER
□ lisa.AddUser.rsp
□ lisa.AddUser.rsp.time
□ lisa.ShouldFail.rsp
□ lisa.ShouldFail.rsp.time
□ lisa.Step 1: Test XML Signature.r
□ lisa.Step 1: Test XML Signature.r
□ lisa.Step 2: Test XML Encryption.
□ lisa.Step 2: Test XML Encryption.
□ lisa.Step 3: Fail: Document Not E
□ lisa.Step 3: Fail: Document Not E
□ lisa.Step 4: Fail: Missing Signatur
□ lisa.Step 4: Fail: Missing Signatur
□ lisa.continue.rsp
□ lisa.continue.rsp.time
□ lisa.end.rsp
□ lisa.end.rsp.time
□ lisa.fail.rsp
□ lisa.fail.rsp.time

The following topics are available in this chapter.

Creating a Subprocess Test Case

To create a subprocess test case

1. Create a new test case or open an existing one.
2. Open the Test Case Information tab of a test case. To open the Test Case Information tab, click anywhere in the empty space in the model editor (no steps should be selected). The Test Case Information tab will open in the right panel.

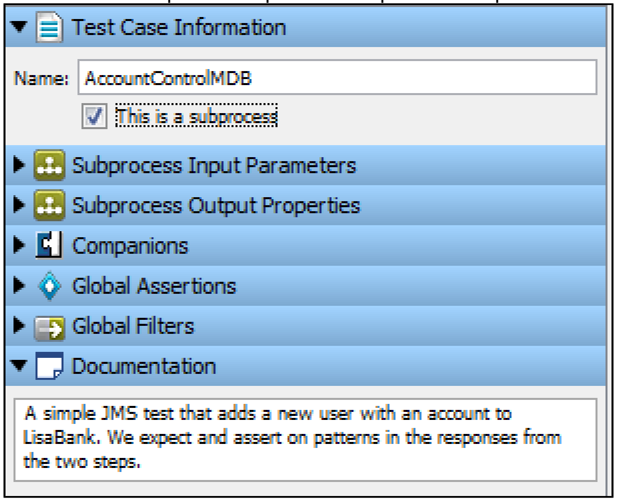


Name: AccountControlMDB

☐ This is a subprocess

A simple JMS test that adds a new user with an account to LisaBank. We expect and assert on patterns in the responses from the two steps

3. Select the **This is a subprocess** box to make this test case a subprocess. By default, a test case is not designated to be a subprocess.
4. New tabs for Subprocess Input and Subprocess Output Parameters are added.



Name: AccountControlMDB

☑ This is a subprocess

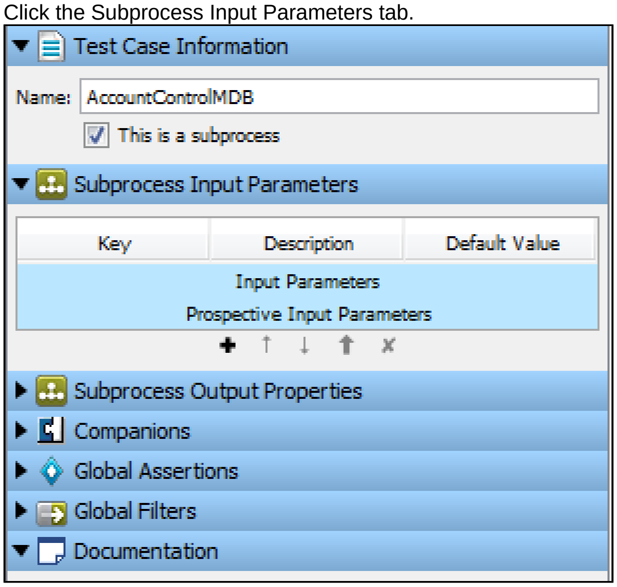
A simple JMS test that adds a new user with an account to LisaBank. We expect and assert on patterns in the responses from the two steps.

5. In the Documentation tab, provide detailed documentation of the subprocess. This text is visible in any test step that calls the subprocess.

When you have finished adding the subprocess steps, configure the input and output properties for the subprocess.

Subprocess Input Parameters

1. Click the Subprocess Input Parameters tab.

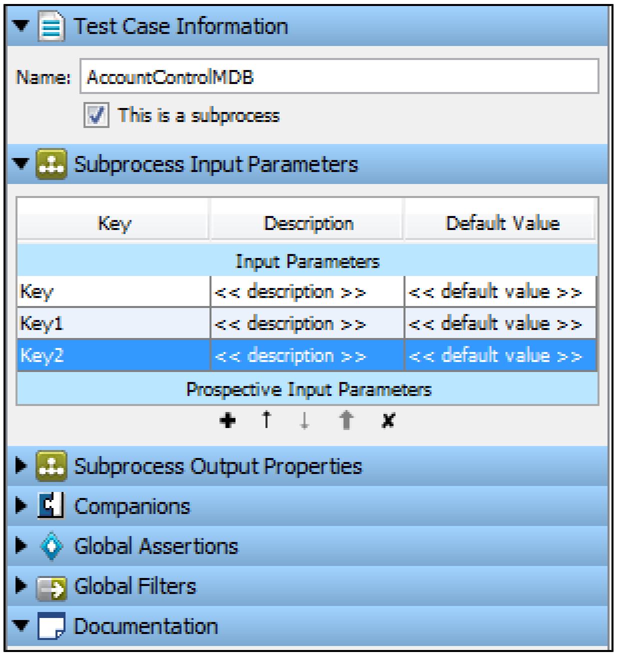


Name: AccountControlMDB

☑ This is a subprocess

Key	Description	Default Value
Input Parameters		
Prospective Input Parameters		

2. In the Subprocess Input Parameters tab, define the input parameters and the prospective input parameters. A list of the parameters needed by the subprocess appears in the Subprocess Input Parameters field. Some parameters will be added automatically, but you may need to add some yourself.



Test Case Information

Name: AccountControlMDB

☑ This is a subprocess

Subprocess Input Parameters

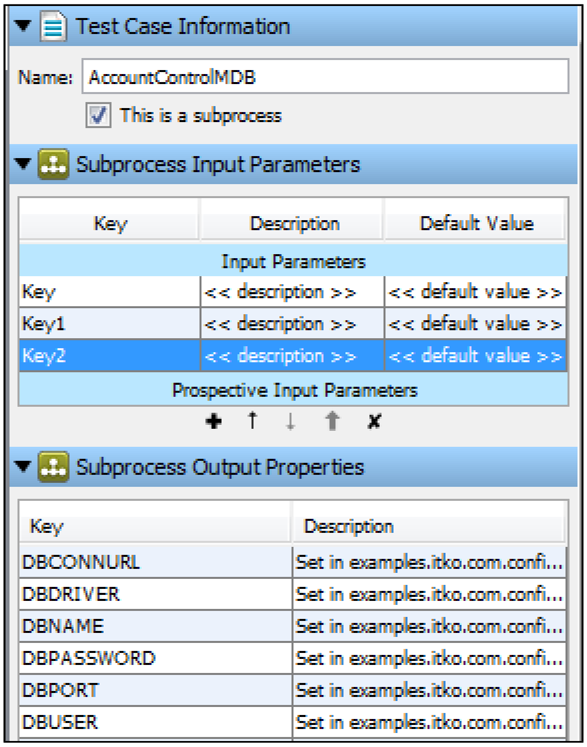
Key	Description	Default Value
Input Parameters		
Key	<< description >>	<< default value >>
Key1	<< description >>	<< default value >>
Key2	<< description >>	<< default value >>
Prospective Input Parameters		

3. Use the Add button ✚ at the bottom of this panel to add a new property.
4. Enter values for Key (property name), Description, and Default Value. The default value here is used when you run the subprocess in the Interactive Test Run facility (ITR). This lets you test the subprocess as though it was a regular test case. These default values are ignored when the subprocess is invoked by another test step.
5. Remove properties using the Delete button ✘.

- **Prospective Input Parameters (may be required parameters)**: If LISA finds a property in the subprocess that may be an input property, but there is some doubt, the property is listed here. If it is a valid input property, use the Add button to promote this property to the Subprocess Input Parameters list.

Subprocess Output Properties

- **Subprocess Output (Result) Properties:** A list of all the properties set by the subprocess. A list of the parameters needed by the subprocess appears in the Subprocess Output Parameters field. Some parameters will be added automatically, but you may need to add some yourself.



Test Case Information

Name: AccountControlMDB

☑ This is a subprocess

Subprocess Input Parameters

Key	Description	Default Value
Input Parameters		
Key	<< description >>	<< default value >>
Key1	<< description >>	<< default value >>
Key2	<< description >>	<< default value >>
Prospective Input Parameters		

Subprocess Output Properties

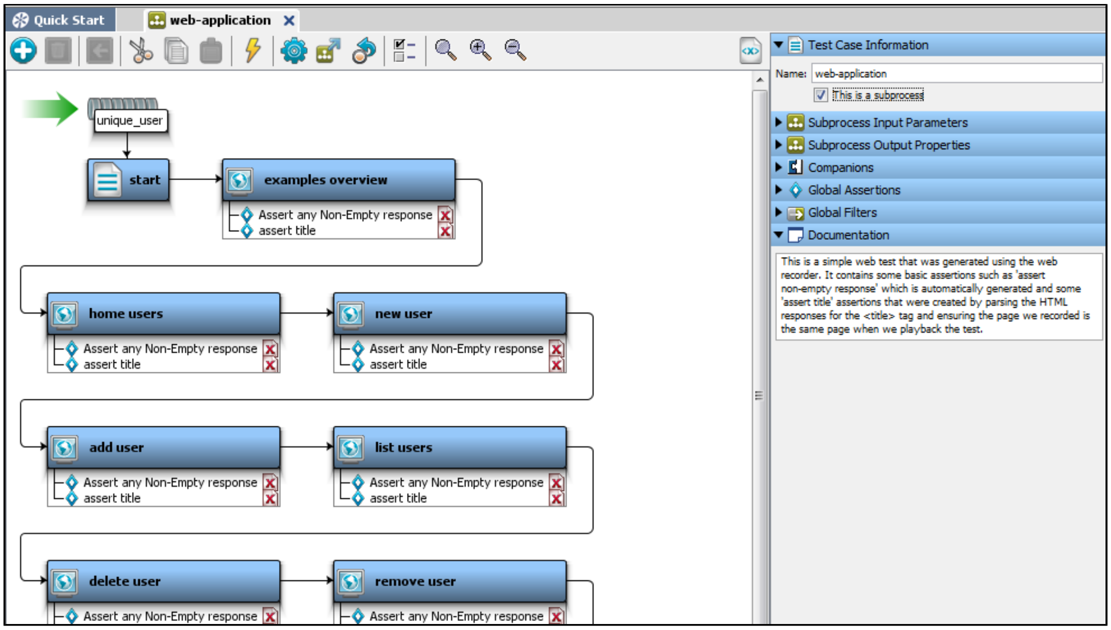
Key	Description
DBCONNURL	Set in examples.itko.com.confi...
DBDRIVER	Set in examples.itko.com.confi...
DBNAME	Set in examples.itko.com.confi...
DBPASSWORD	Set in examples.itko.com.confi...
DBPORT	Set in examples.itko.com.confi...
DBUSER	Set in examples.itko.com.confi...

1. Use the Add button ✚ at the bottom of this panel to add a new property.
2. If there are properties here that you do not want to be made available to the calling step, remove them using the Delete ✘ button.

After the input and return properties have been checked, and default values have been given to all the input parameters, the subprocess can be run in the ITR.

Converting an Existing Test Case into a Subprocess

To convert an existing test case into a subprocess

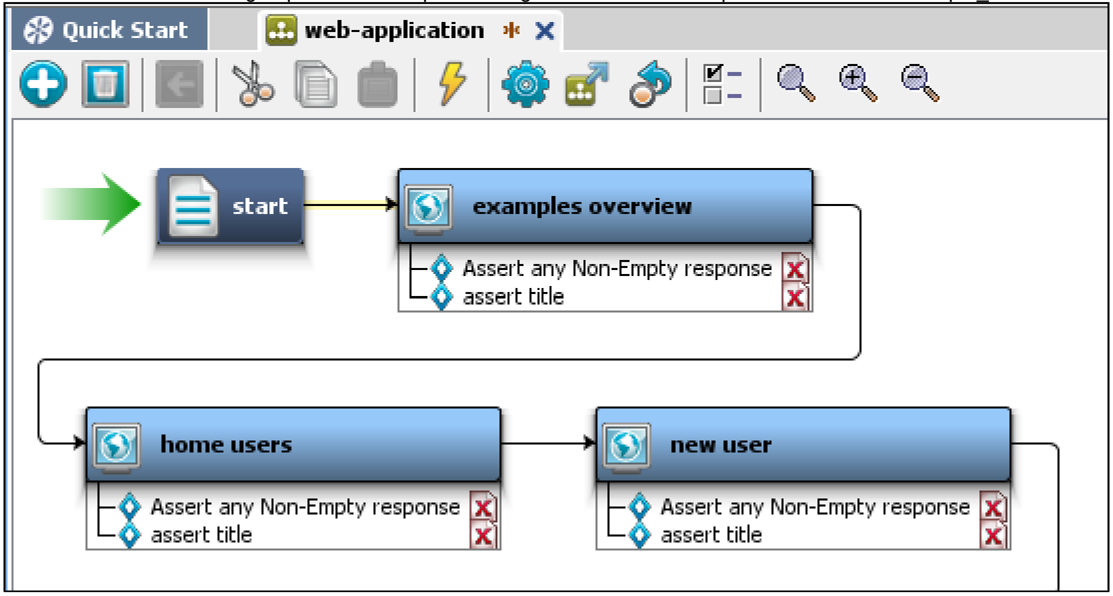


Name: web-application

☑ This is a subprocess

This is a simple web test that was generated using the web recorder. It contains some basic assertions such as 'assert non-empty response' which is automatically generated and some 'assert title' assertions that were created by parsing the HTML responses for the <title> tag and ensuring the page we recorded is the same page when we playback the test.

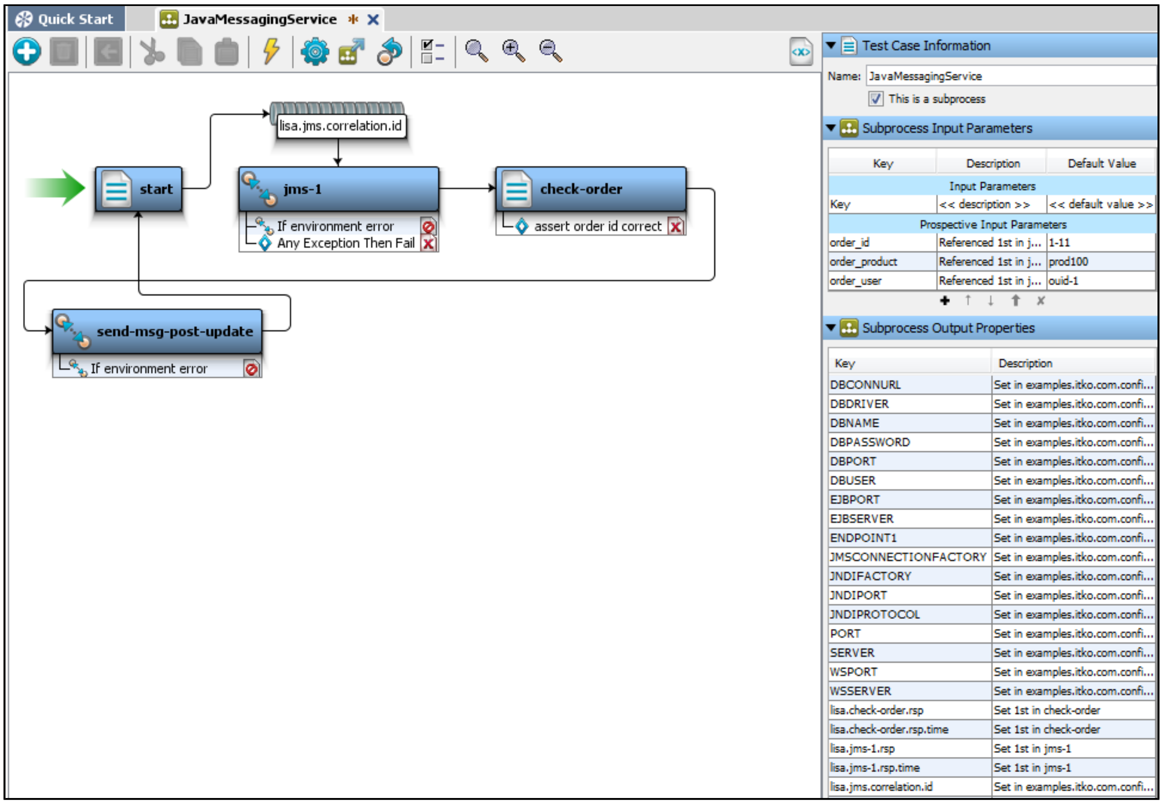
1. Open the test case and rename it appropriately. This example converts the web-application test case from the LISA examples directory.
2. Mark the test case as a subprocess in the Test Case Information tab of the test case.
3. Remove any data sets that provide the values of properties that are to be input properties of the new subprocess. The exception here is when a data set is an integral part of the subprocess logic itself. In our example we removed the unique_user data set.



4. Remove any properties from configuration files, or a companion that initializes parameters that are to be input properties of the new subprocess.
5. After the input and output properties have been checked, and default values have been given to all the input parameters, the subprocess can be run in the ITR.

Subprocess Example

The following example shows a subprocess that was derived from the jms (jms.tst) test case in the LISA examples directory, and a test case that calls this subprocess.



Test Case Information

Name: JavaMessagingService

☑ This is a subprocess

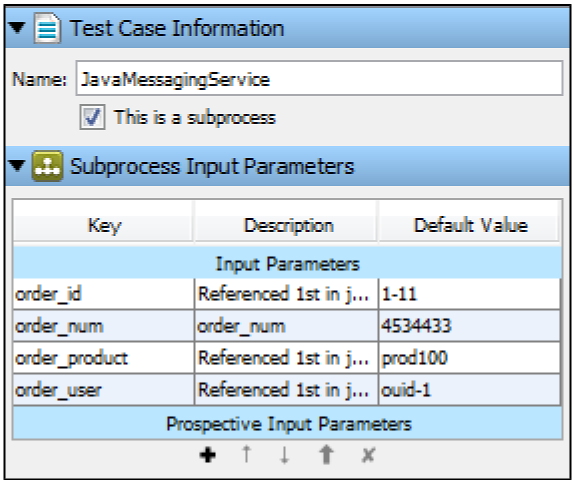
Subprocess Input Parameters

Key	Description	Default Value
Input Parameters		
Key	<< description >>	<< default value >>
Prospective Input Parameters		
order_id	Referenced 1st in j...	1-11
order_product	Referenced 1st in j...	prod100
order_user	Referenced 1st in j...	ouid-1

Subprocess Output Properties

Key	Description
DBCONNURL	Set in examples.itko.com.confi...
DBDRIVER	Set in examples.itko.com.confi...
DBNAME	Set in examples.itko.com.confi...
DBPASSWORD	Set in examples.itko.com.confi...
DBPORT	Set in examples.itko.com.confi...
DBUSER	Set in examples.itko.com.confi...
EJBPORT	Set in examples.itko.com.confi...
EJBSERVER	Set in examples.itko.com.confi...
ENDPOINT1	Set in examples.itko.com.confi...
JMSCONNECTIONFACTORY	Set in examples.itko.com.confi...
JNDIFACTORY	Set in examples.itko.com.confi...
JNDIPORT	Set in examples.itko.com.confi...
JNDIPROTOCOL	Set in examples.itko.com.confi...
PORT	Set in examples.itko.com.confi...
SERVER	Set in examples.itko.com.confi...
WSPORT	Set in examples.itko.com.confi...
WSSERVER	Set in examples.itko.com.confi...
lisa.check-order.rsp	Set 1st in check-order
lisa.check-order.rsp.time	Set 1st in check-order
lisa.jms-1.rsp	Set 1st in jms-1
lisa.jms-1.rsp.time	Set 1st in jms-1
lisa.jms.correlation.id	Set in examples.itko.com.confi...

One data set, order_data, was removed from the original test case. That data set provided the values for **order_num** in the original test case. The property **order_num** is now an input property.



Test Case Information

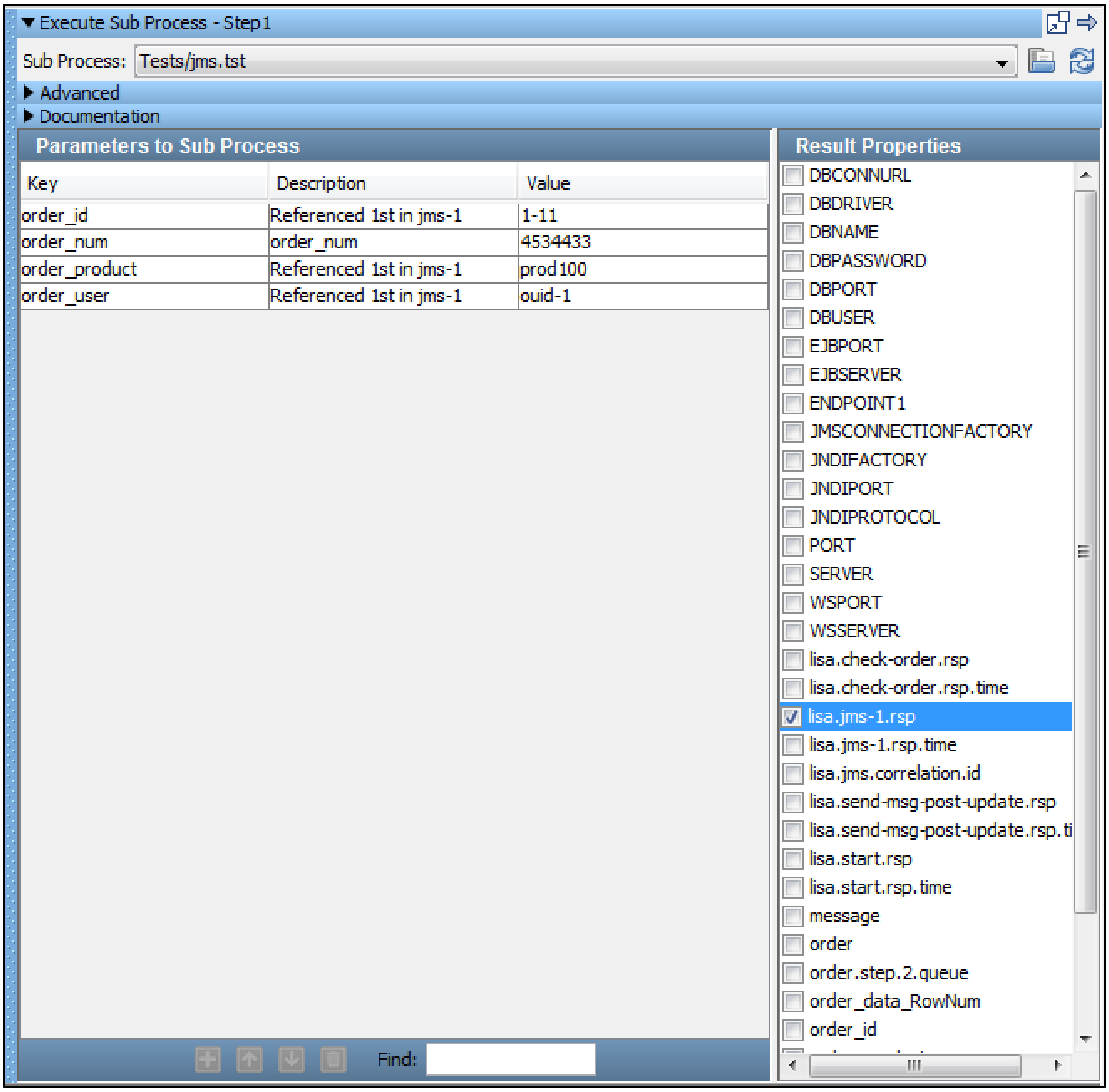
Name: JavaMessagingService

☑ This is a subprocess

Subprocess Input Parameters

Key	Description	Default Value
Input Parameters		
order_id	Referenced 1st in j...	1-11
order_num	order_num	4534433
order_product	Referenced 1st in j...	prod100
order_user	Referenced 1st in j...	ouid-1
Prospective Input Parameters		

The following illustration shows a test case with one test step: Execute Subprocess.



Execute Sub Process - Step1

Sub Process: Tests/jms.tst

Advanced

Documentation

Parameters to Sub Process

Key	Description	Value
order_id	Referenced 1st in jms-1	1-11
order_num	order_num	4534433
order_product	Referenced 1st in jms-1	prod100
order_user	Referenced 1st in jms-1	ouid-1

Find:

Result Properties

- □ DBCONNURL
- □ DBDRIVER
- □ DBNAME
- □ DBPASSWORD
- □ DBPORT
- □ DBUSER
- □ EJBPORT
- □ EJBSERVER
- □ ENDPOINT1
- □ JMSCONNECTIONFACTORY
- □ JNDIFACTORY
- □ JNDIPORT
- □ JNDIPROTOCOL
- □ PORT
- □ SERVER
- □ WSPORT
- □ WSSERVER
- □ lisa.check-order.rsp
- □ lisa.check-order.rsp.time
- ☑ lisa.jms-1.rsp
- □ lisa.jms-1.rsp.time
- □ lisa.jms.correlation.id
- □ lisa.send-msg-post-update.rsp
- □ lisa.send-msg-post-update.rsp.ti
- □ lisa.start.rsp
- □ lisa.start.rsp.time
- □ message
- □ order
- □ order.step.2.queue
- □ order_data_RowNum
- □ order_id

Step1 is of type Execute Sub Process, and it invokes the subprocess **jms** (which is shown previously).

The input property matches, and the calling step has asked for the **lisa.jms-1.rsp** property to be made available after the subprocess has finished executing.

Building Documents

A staging document contains the information about how to run a test case.

An audit document lets you set success criteria for a test case within a suite.

You can apply metrics and add events, which are used for monitoring the test case after the test run.

A suite document can be used to run a collection of test cases.

In this section, the following topics are covered:

Building Staging Documents

A staging document contains the information about how to run a test case.

Creating a Staging Document

You create staging documents from within LISA Workstation.

If a test case contains a global data set on the first step and the data set is set to end the test when the data set is drained, then all instances of the test will end for a staged run, overriding any other staging parameters such as steady state time.

Local data sets will not end the staged run in this fashion nor will data sets on steps other than the first test.

To create a staging document

1. From the main menu, select File > New > Staging Document.
   The New Staging Doc dialog appears.
2. Enter the name of the new staging document.
3. Click OK.
   The staging document editor appears.
4. In the Base tab, specify basic information about the staging document. This information includes the staging document name, the load pattern, and the distribution pattern.
5. In the Reports tab, specify the type of report that you want to create at runtime.
6. In the Metrics tab, specify the metrics that you want to record at runtime.
7. In the Documentation tab, enter descriptive information about the staging document.
8. (Optional) In the IP Spoofing tab, enable and configure IP spoofing.
9. (Optional) In the Source View tab, review the XML version of the staging document.
10. From the main menu, select File > Save.

Staging Document Editor

The Staging Document Editor is where you specify the criteria for running test cases.

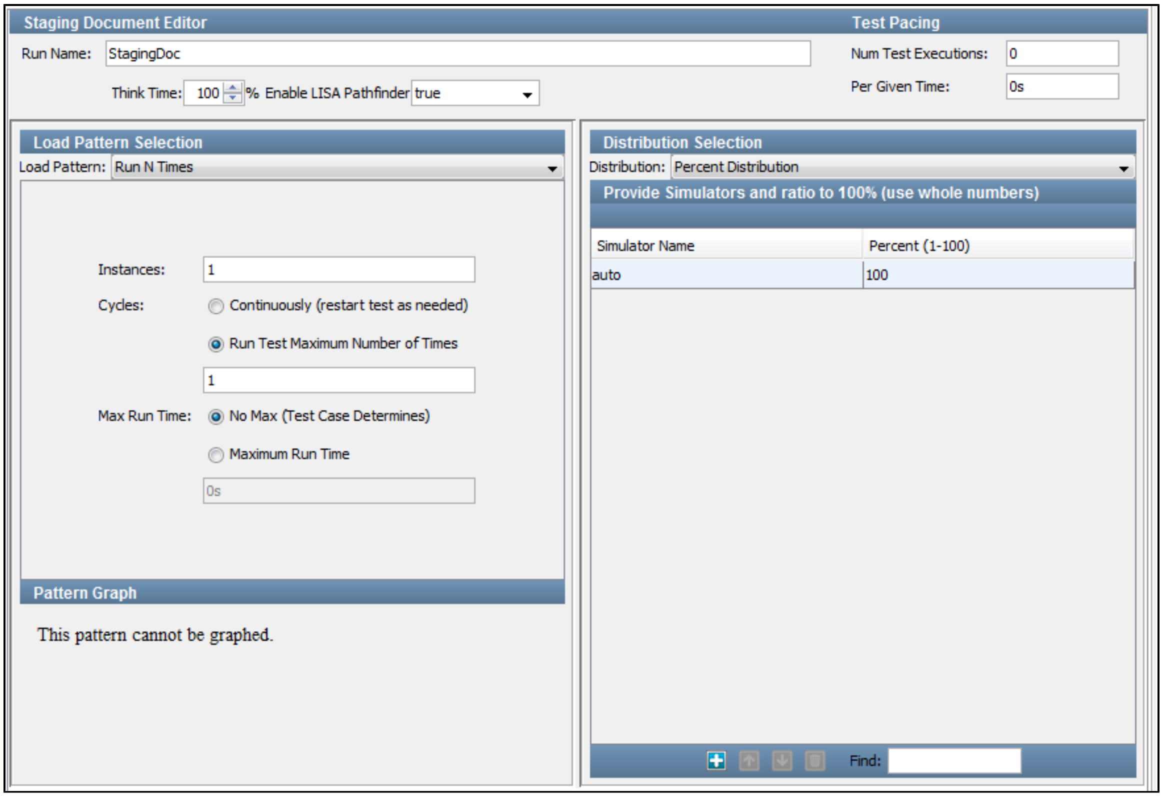
The Staging Document Editor contains the following tabs:

- **Base**: To specify the basic parameters.
- **Reports**: To select and add reports.
- **Metrics**: To select metrics, and specify your sampling intervals.
- **Documentation**: To enter descriptive information about the staging document.
- **IP Spoofing**: To enter the IP spoofing details. IP spoofing allows multiple IP addresses on a network interface to be used when making network requests.
- **Source View**: To view the XML source of the staging document.

Staging Document Editor - Base Tab

The Base tab of the Staging Document Editor describes the basic parameters of a test case:

- Global adjustments to think times
- Duration of the test (elapsed time or number of runs)
- Information to pace tests, such that a given number of tests complete in a specified time period
- Number of virtual users
- Load patterns for the virtual users
- Distribution patterns for the virtual users



Staging Document Editor

Run Name: StagingDoc

Think Time: 100 % Enable LISA Pathfinder true

Test Pacing

Num Test Executions: 0

Per Given Time: 0s

Load Pattern Selection

Load Pattern: Run N Times

Instances: 1

Cycles: ○ Continuously (restart test as needed)

● Run Test Maximum Number of Times

1

Max Run Time: ● No Max (Test Case Determines)

○ Maximum Run Time

0s

Pattern Graph

This pattern cannot be graphed.

Distribution Selection

Distribution: Percent Distribution

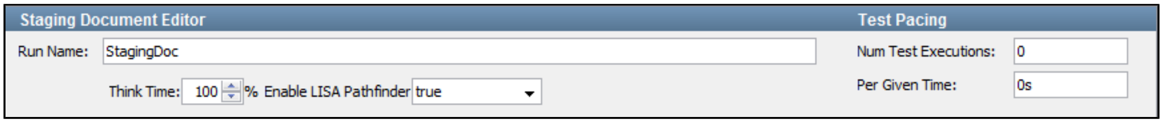
Provide Simulators and ratio to 100% (use whole numbers)

Simulator Name	Percent (1-100)
auto	100

The Base tab is divided into the following panels:

- Upper Panel
- Load Pattern Selection Panel
- Distribution Selection Panel

Upper Panel



Staging Document Editor

Run Name: StagingDoc

Think Time: 100 % Enable LISA Pathfinder true

Test Pacing

Num Test Executions: 0

Per Given Time: 0s

The upper panel lets you set the following parameters:

- **Run Name:** The name of the staging document.
- **Think Time:** The think time in percentage, for all the test steps in the test case. Each test step can declare a think time in the step information section. Here, you can apply a global change to these think times, as a percentage of their values. Think times can be eliminated by setting the percentage to 0%, halved by setting it to 50%, or doubled by setting it to 200%. For example, if the percentage is 50%, then a step's think time of 4-6 seconds will be reduced to 2-3 seconds.
- **Enable LISA Pathfinder:** You can choose to enable or disable LISA Pathfinder.
- **Num Test Executions:** The number of test executions that you want to complete in a given time.
- **Per Given Time:** The time period (wall-clock) in which you want the tests to run. You can specify **h** for hours, **m** for minutes, or **s** for seconds. For example, you can specify that you want LISA to adjust the time such that 2500 tests complete in 8 minutes.

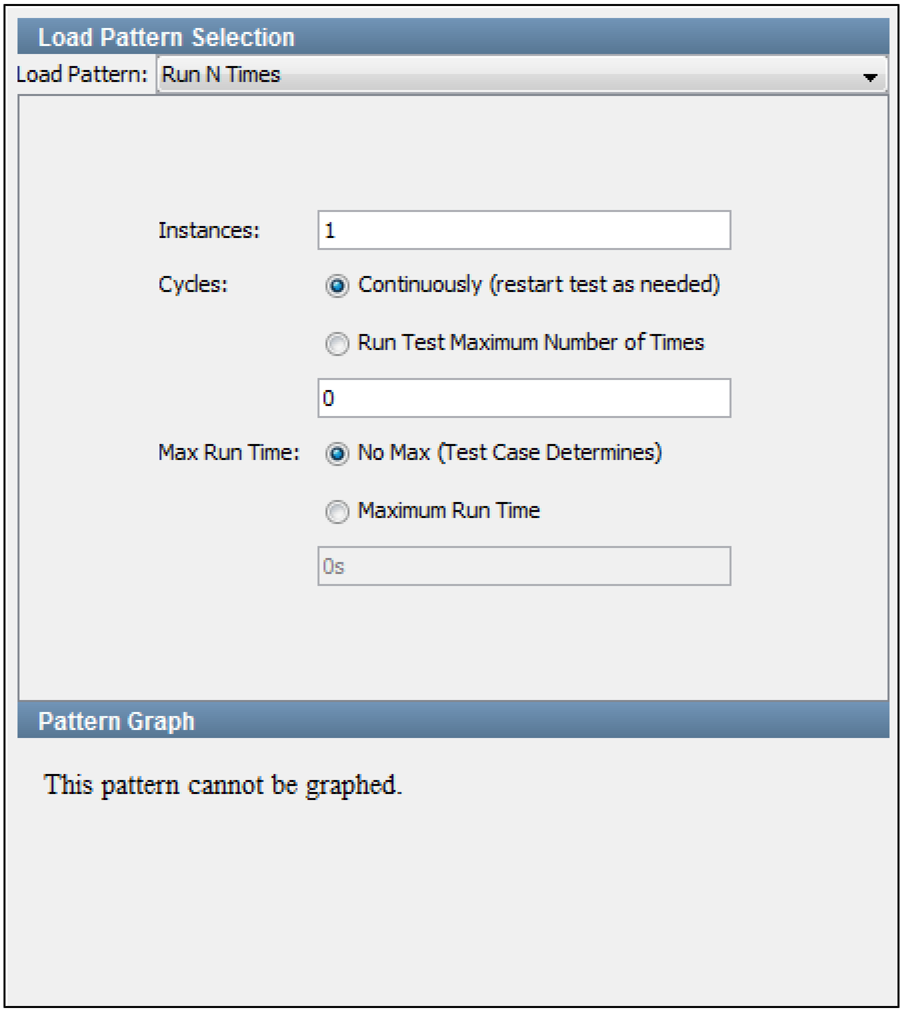
LISA does not change think times to achieve the required pace.

If the test pace cannot be achieved because it is too high, LISA will run without any pause between tests. LISA will report in the log that the test is running at a lower pace than requested.

If the test pace cannot be achieved because too few virtual users have been specified, LISA will not add more users. To estimate the number of virtual users needed, see the Optimizer Utility.

Load Pattern Selection Panel

The Load Pattern Selection panel lets you set the duration of the test, the number of virtual users (instances), and the load pattern for those virtual users (if you have more than one virtual user). For more information, see Load Pattern Selection.



Load Pattern: Run N Times

Instances: 1

Cycles: ● Continuously (restart test as needed)

○ Run Test Maximum Number of Times

0

Max Run Time: ● No Max (Test Case Determines)

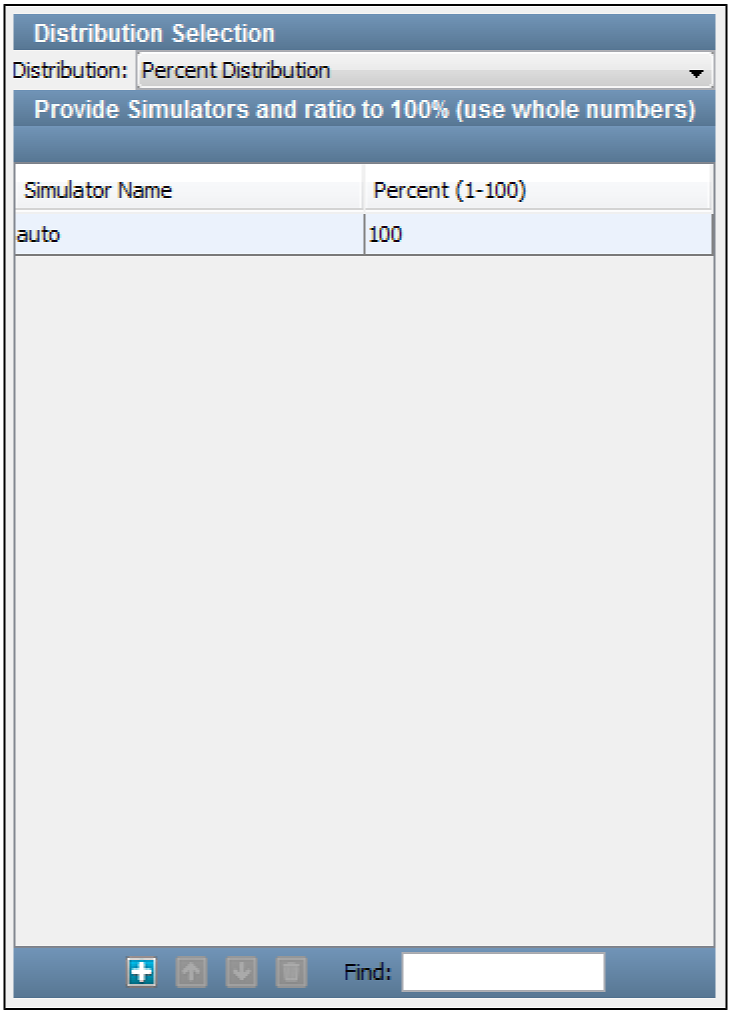
○ Maximum Run Time

0s

This pattern cannot be graphed.

Distribution Selection Panel

The Distribution Selection panel lets you distribute virtual users (instances) over your running simulators. For more information, see Distribution Selection.



Distribution Selection

Distribution: Percent Distribution

Provide Simulators and ratio to 100% (use whole numbers)

Simulator Name	Percent (1-100)
auto	100

Find:

Load Pattern Selection

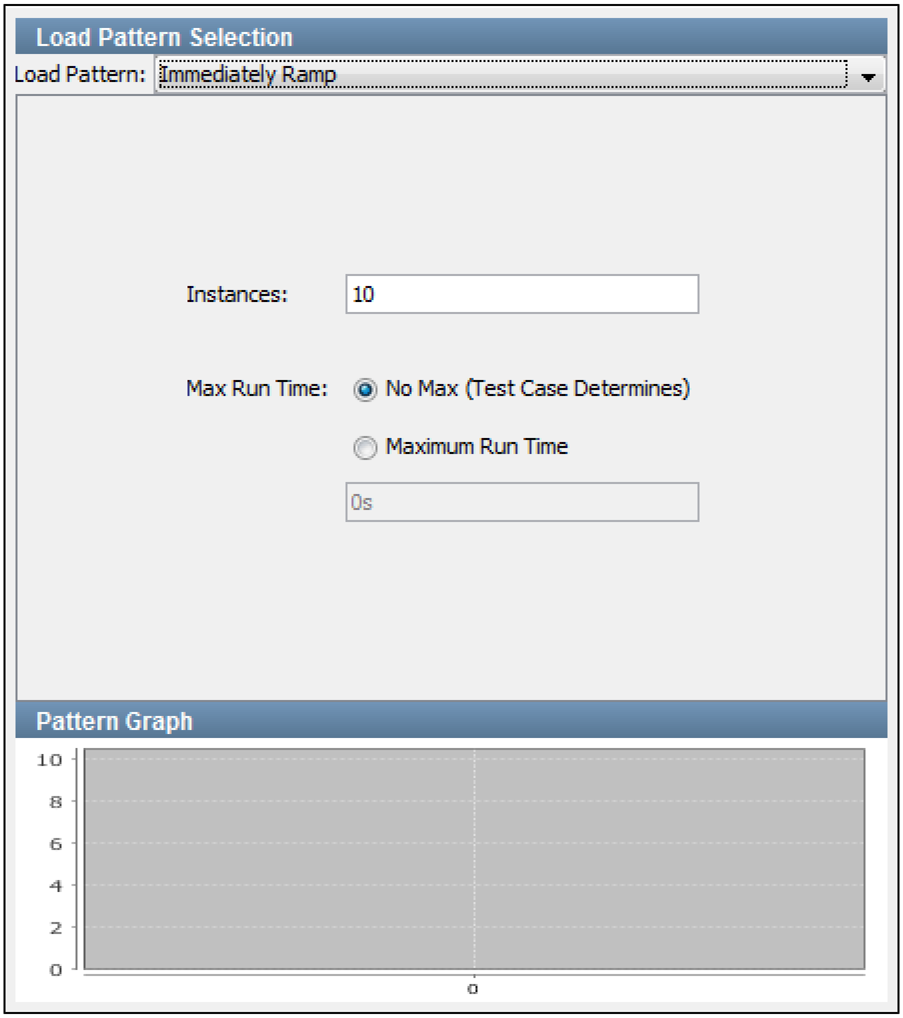
The Base tab of the Staging Document Editor includes a Load Pattern Selection panel.

The load pattern options are:

- Immediately Ramp
- Manual Load Pattern
- Run N Times
- Stair Steps
- Weighted Average Pattern

Immediately Ramp

The Immediately Ramp pattern is applicable when you are running just a few virtual users (instances), but you want to specify how long to run the test. You are not concerned with any loading pattern. This pattern will start all virtual users at the same time.



Load Pattern Selection

Load Pattern: Immediately Ramp

Instances: 10

Max Run Time: ● No Max (Test Case Determines)

○ Maximum Run Time

0s

Pattern Graph

To configure this pattern, enter the following parameters:

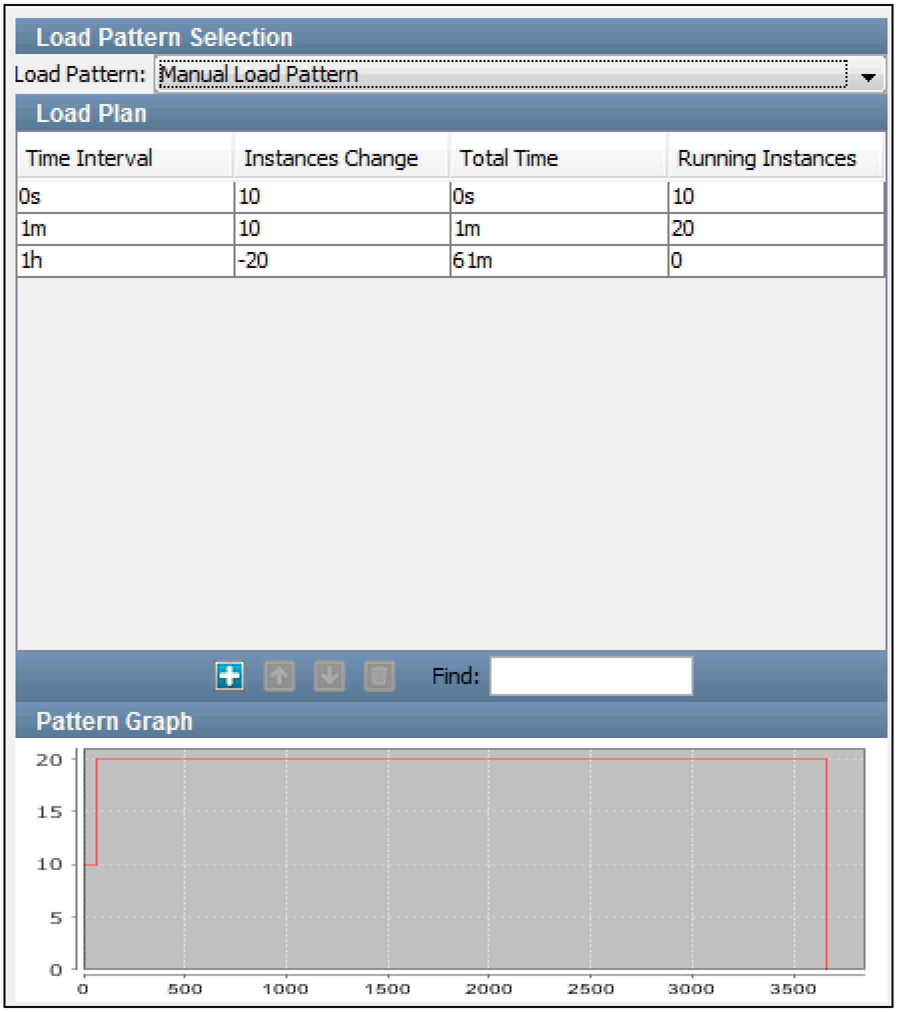
- **Instances**: The number of virtual users.
- **Max Run Time**: Choose between **No Max** (the time is determined by the test case) and **Maximum Run Time**. In the latter case, specify the **Maximum Run Time**. This will overwrite any value entered for Cycles. You can specify **h** for hours, **m** for minutes, **s** (or no letter) for seconds (default).

In the preceding image, there are 10 virtual users running the test concurrently for one hour.

The graph at the bottom provides a graphical view of the pattern.

Manual Load Pattern

The Manual Load pattern gives you the most control over the loading and unloading of virtual users (instances). This pattern is similar to the Stair Steps pattern, but it lets you specify both the number of virtual users to add, and the time interval for each step in the pattern, individually.



Load Pattern: Manual Load Pattern

Time Interval	Instances Change	Total Time	Running Instances
0s	10	0s	10
1m	10	1m	20
1h	-20	61m	0

To configure this pattern, you define each step as a row in a table. In each row, you define the time interval and the number of virtual users to add or remove (Instances Change column) for that step. The elapsed time (Total Time column) and the total number of virtual users (Running Instances column) are automatically calculated.

In the **Time Interval** column, enter a time followed by **h** for hours, **m** for minutes, **s** (or no letter) for seconds (default).

Use the standard icons to add, remove, or change the current order of the steps from the toolbar at the bottom of the table.

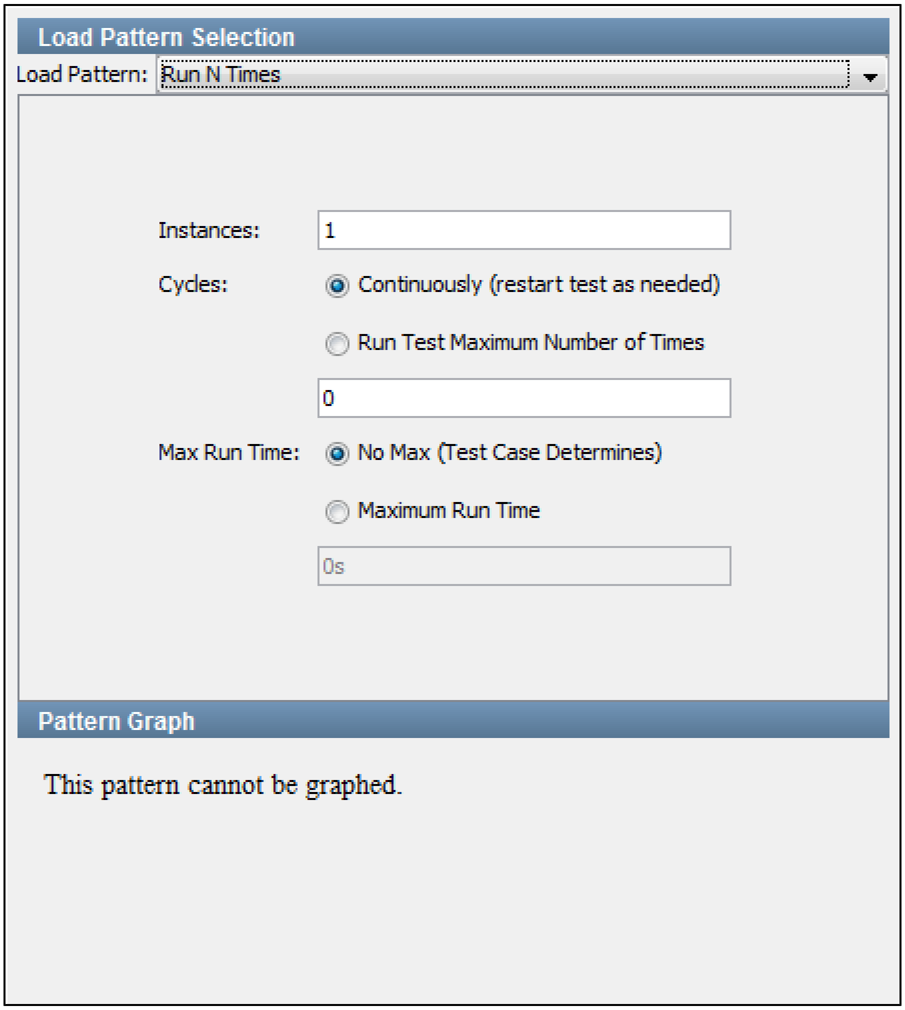
You may have to scroll to see these icons. Alternatively, you can select a row and use the following short-cuts:

- Add a line: Ctrl+Shift+A
- Delete a line: Ctrl+Shift+D
- Move line up: Ctrl+Shift+Up Arrow
- Move line down: Ctrl+Shift+Down Arrow
- Extended view: Ctrl+Shift+L

The graph at the bottom provides a graphical view of the pattern.

Run N Times

The Run N Times pattern is applicable when you are running only one or just a few virtual users (instances), but you want to specify how many times the test will run. You are not concerned with any loading pattern. This pattern will start all virtual users at the same time.



Load Pattern Selection

Load Pattern: Run N Times

Instances: 1

Cycles: ● Continuously (restart test as needed)

○ Run Test Maximum Number of Times

0

Max Run Time: ● No Max (Test Case Determines)

○ Maximum Run Time

0s

Pattern Graph

This pattern cannot be graphed.

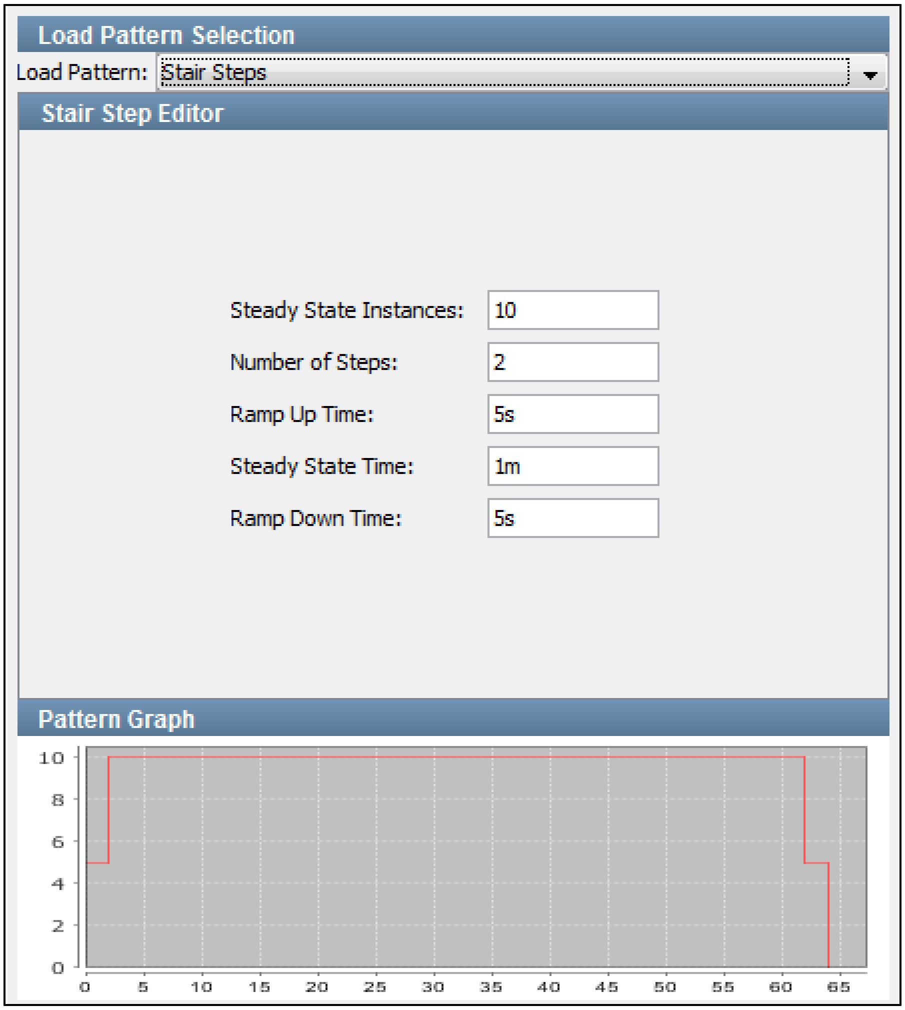
To configure this pattern, enter the following parameters:

- **Instances**: The number of virtual users.
- **Cycles**: Choose between running continuously until the time in the **Max Run Time** setting has been reached, or running a maximum number of times.
- **Max Run Time**: Choose between No Max (the time is determined by the test case) and Maximum Run Time. In the latter case, specify the Maximum Run Time. This will overwrite any value entered for Cycles. You can specify **h** for hours, **m** for minutes, **s** (or no letter) for seconds (default).

In the preceding image, there is a single user running the test case one time to conclusion.

Stair Steps

The Stair Steps pattern introduces virtual users (instances) to the system in well-defined steps, rather than all at once. You specify the total number of steady state users, the ramp up and ramp down times, and the number of steps for the ramp.



Load Pattern Selection

Load Pattern: Stair Steps

Stair Step Editor

Steady State Instances: 10

Number of Steps: 2

Ramp Up Time: 5s

Steady State Time: 1m

Ramp Down Time: 5s

Pattern Graph

To configure this pattern, enter the following parameters:

- **Steady State Instances**: The maximum number of virtual users to run at steady state.
- **Number of Steps**: The number of steps to use to reach the maximum number of virtual users. The number of virtual users to introduce at each step is: **Steady State Instances** divided by **Number of Steps**.
- **Ramp Up Time**: The time period over which virtual users are added to reach the maximum number of virtual users. The time interval between steps is: **Ramp Up Time** divided by **Number of Steps**.
- **Steady State Time**: The time period of the meaningful test run.
- **Ramp Down Time**: The time period over which virtual users are removed. Because the tests will run to completion after a "stop" request, the ramp down times are approximate.

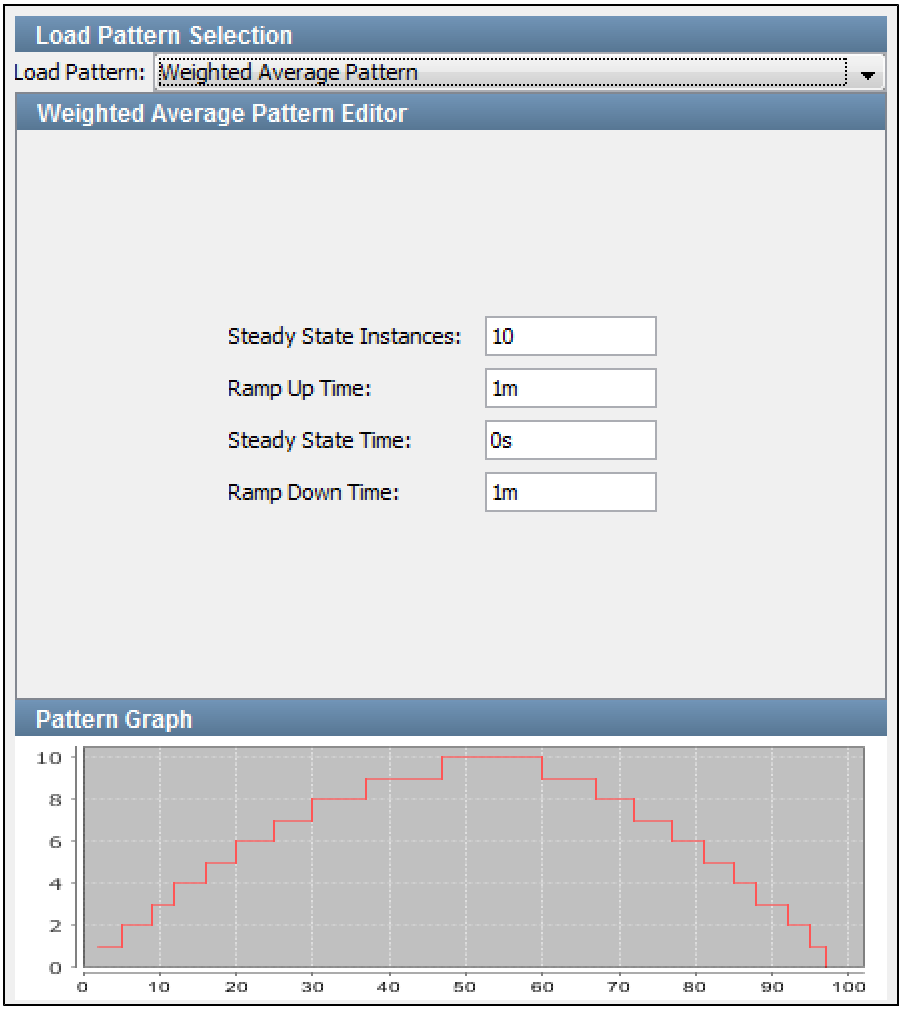
You can enter a time followed by **h** for hours, **m** for minutes, **s** (or no letter) for seconds (default).

In the preceding image, there are 10 instances and 2 steps. Therefore, 5 virtual users will be added in each step. The ramp up time is 5 seconds. The test will run in steady state for 1 minute. Two virtual users will be removed approximately every 5 seconds.

The graph at the bottom provides a graphical view of the pattern.

Weighted Average Pattern

The Weighted Average pattern will add and remove virtual users (instances) based on a statistical calculation. LISA will calculate the number of steps, and the time period between steps, as frequently as every second, by honoring a moving weighted average. This will approximate a bell curve distribution, with most virtual users being added within 2 standard deviations of the mid-point of the load, or unload ramp time.



Load Pattern: Weighted Average Pattern

Steady State Instances: 10

Ramp Up Time: 1m

Steady State Time: 0s

Ramp Down Time: 1m

To configure this pattern, enter the following parameters:

- **Steady State Instances**: The maximum number of virtual users to run at steady state.
- **Ramp Up Time**: The time period over which virtual users are added to reach the maximum number of virtual users.
- **Steady State Time**: The time period of the meaningful test run.
- **Ramp Down Time**: The time period over which virtual users are removed. Because the tests will run to completion after a "stop" request, the ramp down times are approximate.

You can enter a time followed by **h** for hours, **m** for minutes, **s** (or no letter) for seconds (default).

In the preceding image, there are 10 virtual users ramping up over 1 minute, running in a steady state for 3 minutes, before ramping down over approximately 1 minute.

The graph at the bottom provides a graphical view of the pattern.

Distribution Selection

The Base tab of the Staging Document Editor includes a Distribution Selection panel.

If you are using LISA Workstation, you will not have use for this panel because your virtual users will be running locally (using a simulator that is part of LISA Workstation).

If you are using LISA Server, with several simulator servers active, this panel lets you specify how to distribute your virtual users over these simulators.

The distribution options are:

- Balanced Based on Instance Capacity
- Dynamic Simulator Scaling with DCM
- Percent Distribution
- Round Robin Distribution

Balanced Based on Instance Capacity

The Balanced Based on Instance Capacity distribution requests that LISA control the allocation of virtual users (instances), based on an assessment of current loading (percent load on each simulator).

Each simulator is initiated with a defined number of virtual users that can be allocated. The default is 255. LISA dynamically tracks the percent load and adds virtual users to specific simulators to try to keep the load percentage as even as possible. Therefore, the simulator with the lowest percentage load is a candidate for the next virtual user introduced into the system.

This distribution is useful when the system is already running tests from several other testers, and you want to optimize your load distribution.

You do not need to specify any parameters.

Dynamic Simulator Scaling with DCM

The Dynamic Simulator Scaling with DCM distribution requests that LISA determine when more capacity is required to meet the needs of a running test and then automatically expand the lab.

This distribution pattern includes the following parameters:

- **Checkpoint time:** The interval at which LISA will evaluate whether more simulators are needed. You can enter the value in seconds (for example, **300s**) or minutes (for example, **5m**).
- **DCM Dynamic Lab Name:** If you want to stage to an existing coordinator and the test determines that it needs more capacity to run the test, then this parameter indicates what lab it should spin up to run more simulators. You must include the VLM prefix and the fully qualified lab name (for example, **AGL:Root/MyLab**).
- **Maximum Expansion:** The maximum number of simulators that will be created at the time of expansion. If you have a policy that limits the number of simulators per user, you can use this parameter to enforce the policy. The default value of 0 means unlimited.

During the checkpoint evaluation, LISA examines the performance of the simulators that are currently running and how many more virtual users need to be brought online. If more simulators are needed, LISA begins the process of expanding the lab. When the simulators come online, LISA starts directing traffic to them.

Percent Distribution

The Percent Distribution distributes virtual users (instances) over the simulators based on percentages that you choose.

The names of the running simulators (plus local) are available in a drop-down menu in the **Simulator Name** column.

Select a simulator and specify a percentage of virtual users for the simulator. Repeat this action for all the simulators that you want to include until you have 100 percent. You must use integer values for the percentages.

> ⚠ Although "Auto" appears as a choice in the drop-down menu, it is not recommended. It will result in confusing allocations of virtual users, as it will compete with your explicit distribution choices.

Round Robin Distribution

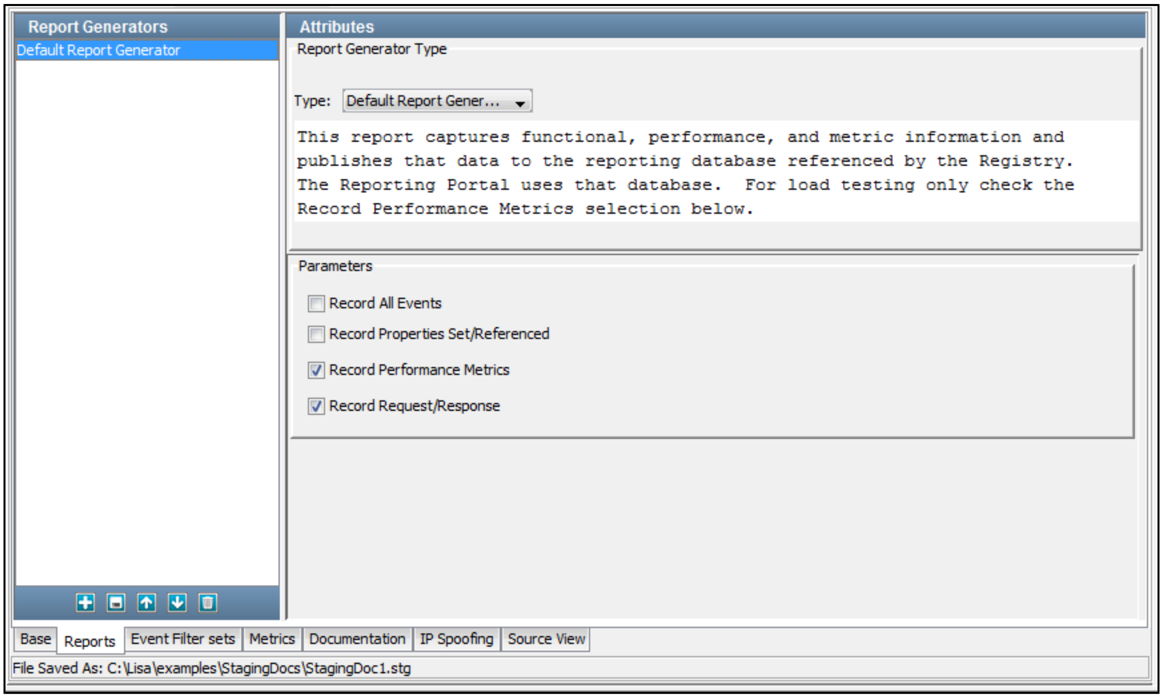
The Round Robin Distribution requests that LISA control the allocation of virtual users (instances), based on a simple round-robin distribution pattern.

LISA picks an arbitrary simulator and adds a user, then goes to the next simulator and adds a user, continuing to add virtual users as needed. After all simulators have been used, the process continues with the first simulator. This distribution is useful when staging a single, large load test on your system.

You do not need to specify any parameters.

Staging Document Editor - Reports Tab

The Reports tab of the Staging Document Editor lets you specify the report generator that you want to invoke for every test case or test suite run.



Default Report Generator

Report Generator Type

Type: Default Report Gener...

```
This report captures functional, performance, and metric information and
publishes that data to the reporting database referenced by the Registry.
The Reporting Portal uses that database.  For load testing only check the
Record Performance Metrics selection below.
```

Parameters

- □ Record All Events
- □ Record Properties Set/Referenced
- ☑ Record Performance Metrics
- ☑ Record Request/Response

File Saved As: C:\Lisa\examples\StagingDocs\StagingDoc1.stg

The following report generators are available:

- **Default Report Generator**: Captures functional, performance, and metric information and publishes that data to the reporting database referenced by the registry. The Reporting Portal uses that database.
- **Load Test Report Generator**: Designed for load tests with thousands of virtual users. This report captures load metrics but not step-level metrics (there would be too much data and the reporting database would slow down the test).
- **XML Report Generator**: Creates an XML file with all the possible data that can be captured. The captured data can be limited using the report options. To view this report, import the file into the Reporting Portal.

Details of the data available in each report generator, viewing reports, report contents and output options are discussed in detail in Reports.

The metrics to capture for the reports are discussed more fully in Generating Metrics.

Reports can be selected in the following modules and can be seen in the Report Viewer:

- Staging Quick Tests
- Building Staging Documents
- Running Test Suites

After they are requested, they can be viewed and managed later.

When you select a report from the drop-down, a summary of the report appears in the area below.

According to the selected report, the parameters will change. You must set the parameters as and where necessary.

The Reports tab is divided into the following areas:

Right panel: This is where you select the report to be added.

Attributes: This contains the "Report Generator Type" and the required parameters for the report.

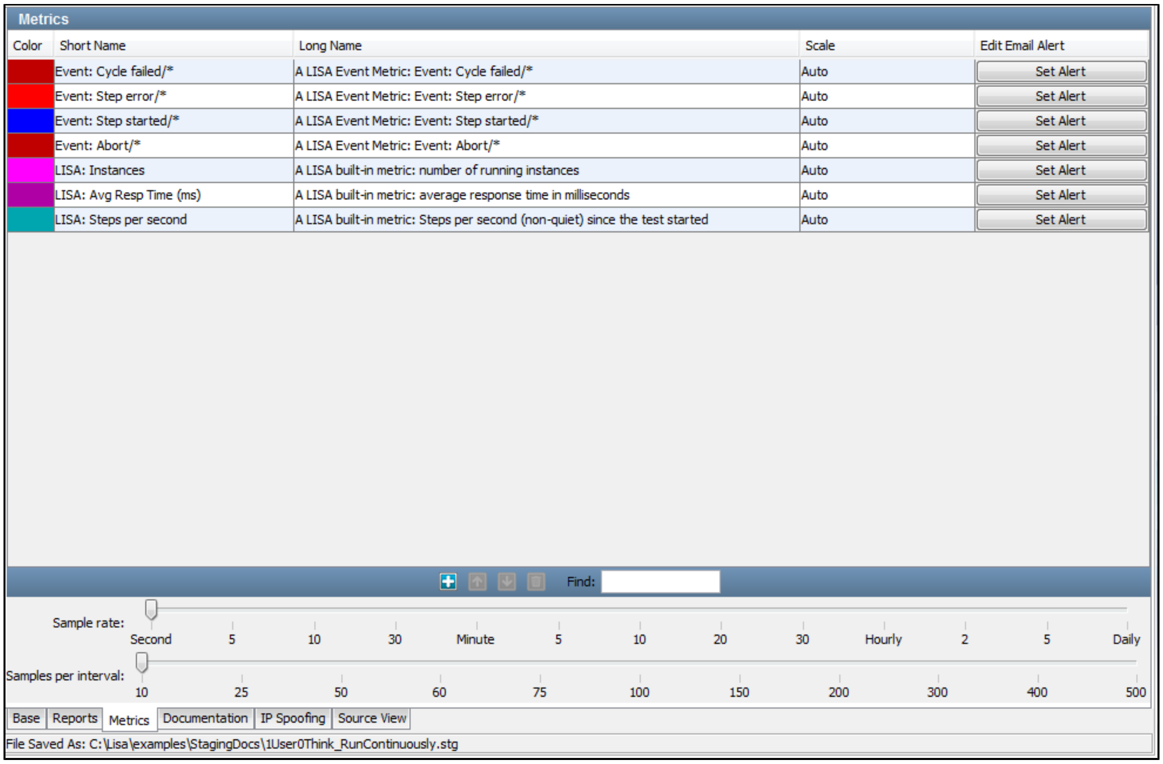
- **Report Generator Type**: A pull-down menu lists the available report types.
- **Parameters**: These are the parameters required to set the selected report generator.

Left panel: This shows the list of added reports. You can add, save, move, and delete reports by using the toolbar at the bottom of the panel.

Staging Document Editor - Metrics Tab

The Metrics tab of the Staging Document Editor lets you select the test metrics that you want to record.

You can also set the sampling specifications for the collection of the metrics and set an email alert on any metric that you have selected.



Metrics

Color	Short Name	Long Name	Scale	Edit Email Alert
🟥	Event: Cycle failed/*	A LISA Event Metric: Event: Cycle failed/*	Auto	Set Alert
🟥	Event: Step error/*	A LISA Event Metric: Event: Step error/*	Auto	Set Alert
🟦	Event: Step started/*	A LISA Event Metric: Event: Step started/*	Auto	Set Alert
🟥	Event: Abort/*	A LISA Event Metric: Event: Abort/*	Auto	Set Alert
🟪	LISA: Instances	A LISA built-in metric: number of running instances	Auto	Set Alert
🟪	LISA: Avg Resp Time (ms)	A LISA built-in metric: average response time in milliseconds	Auto	Set Alert
🟦	LISA: Steps per second	A LISA built-in metric: Steps per second (non-quiet) since the test started	Auto	Set Alert

File Saved As: C:\Lisa\examples\StagingDocs\1User0Think_RunContinuously.stg

The Metrics tab is divided into two sections. The top panel has the default metrics listed, differentiated by color code. The bottom panel has the sampling parameters.

You can add or delete metrics and set email alerts in the top panel.

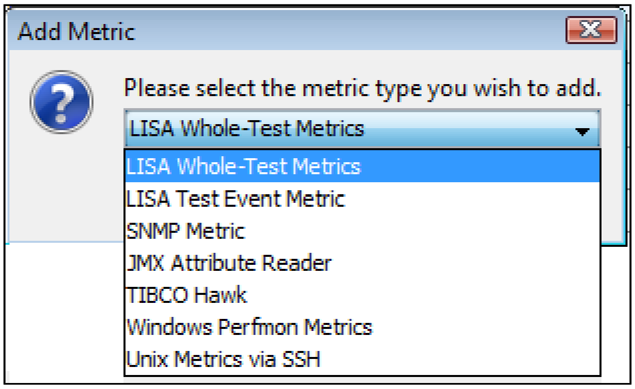
The following types of metrics are available:

- LISA Whole-Test Metrics
- LISA Test Event Metrics
- SNMP Metrics
- JMX Attribute Reader (JMX metrics)
- TIBCO Hawk
- Windows Perfmon Metrics
- UNIX Metrics via SSH

For more information about each metric, see Types of Metrics.

Add a Metric

To add a metric, click the Add icon on the toolbar. A dialog to add metrics will open up with a list of metrics that can be added.



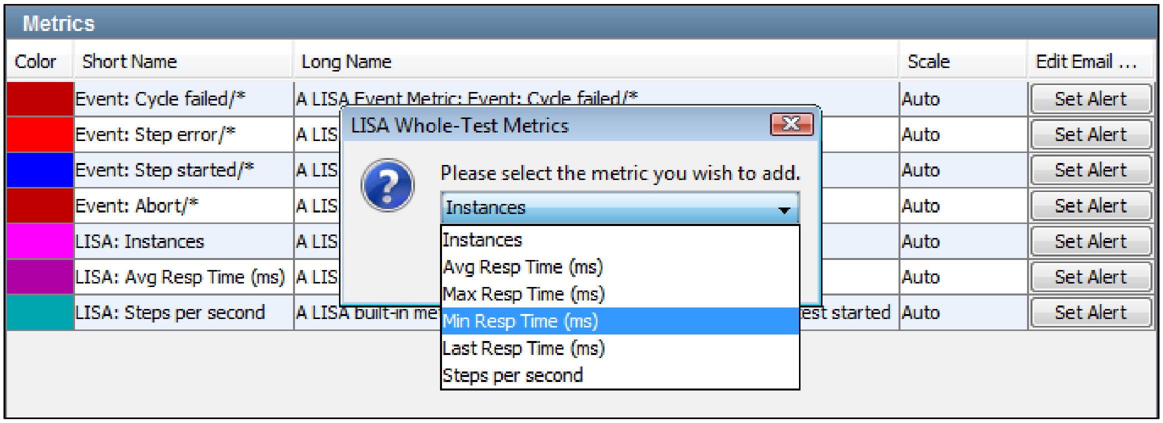
Add Metric

Please select the metric type you wish to add.

LISA Whole-Test Metrics

Select the target metric type and click OK.

The Metric Selection dialog opens.



Metrics

Color	Short Name	Long Name	Scale	Edit Email ...
🟥	Event: Cycle failed/*	A LISA Event Metric: Event: Cycle failed/*	Auto	Set Alert
🟥	Event: Step error/*	A LIS	Auto	Set Alert
🟦	Event: Step started/*	A LIS	Auto	Set Alert
🟥	Event: Abort/*	A LIS	Auto	Set Alert
🟪	LISA: Instances	A LIS	Auto	Set Alert
🟪	LISA: Avg Resp Time (ms)	A LIS	Auto	Set Alert
🟩	LISA: Steps per second	A LISA built-in me est started	Auto	Set Alert

LISA Whole-Test Metrics

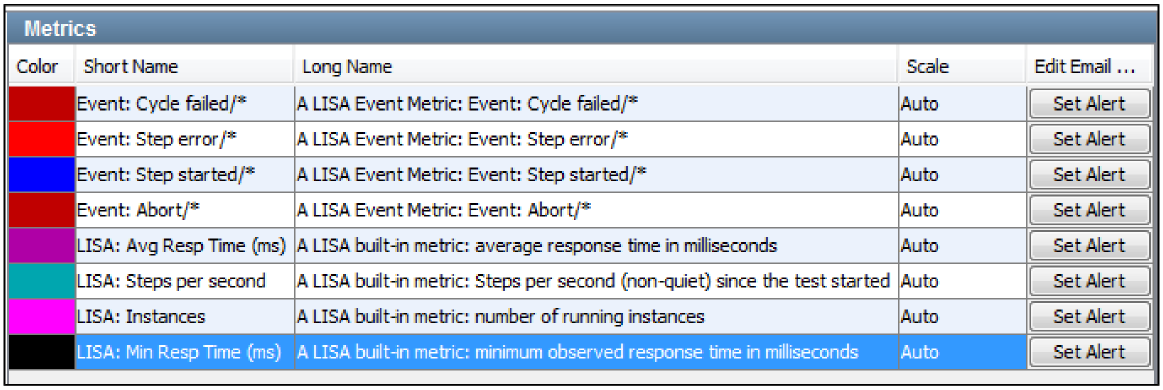
Please select the metric you wish to add.

Instances

- Instances
- Avg Resp Time (ms)
- Max Resp Time (ms)
- Min Resp Time (ms)
- Last Resp Time (ms)
- Steps per second

Select the target sub-category for this metric and click OK. Sub-categories are different for each metric.

The newly-added metric appears in the list of existing metrics in a different color.



Metrics

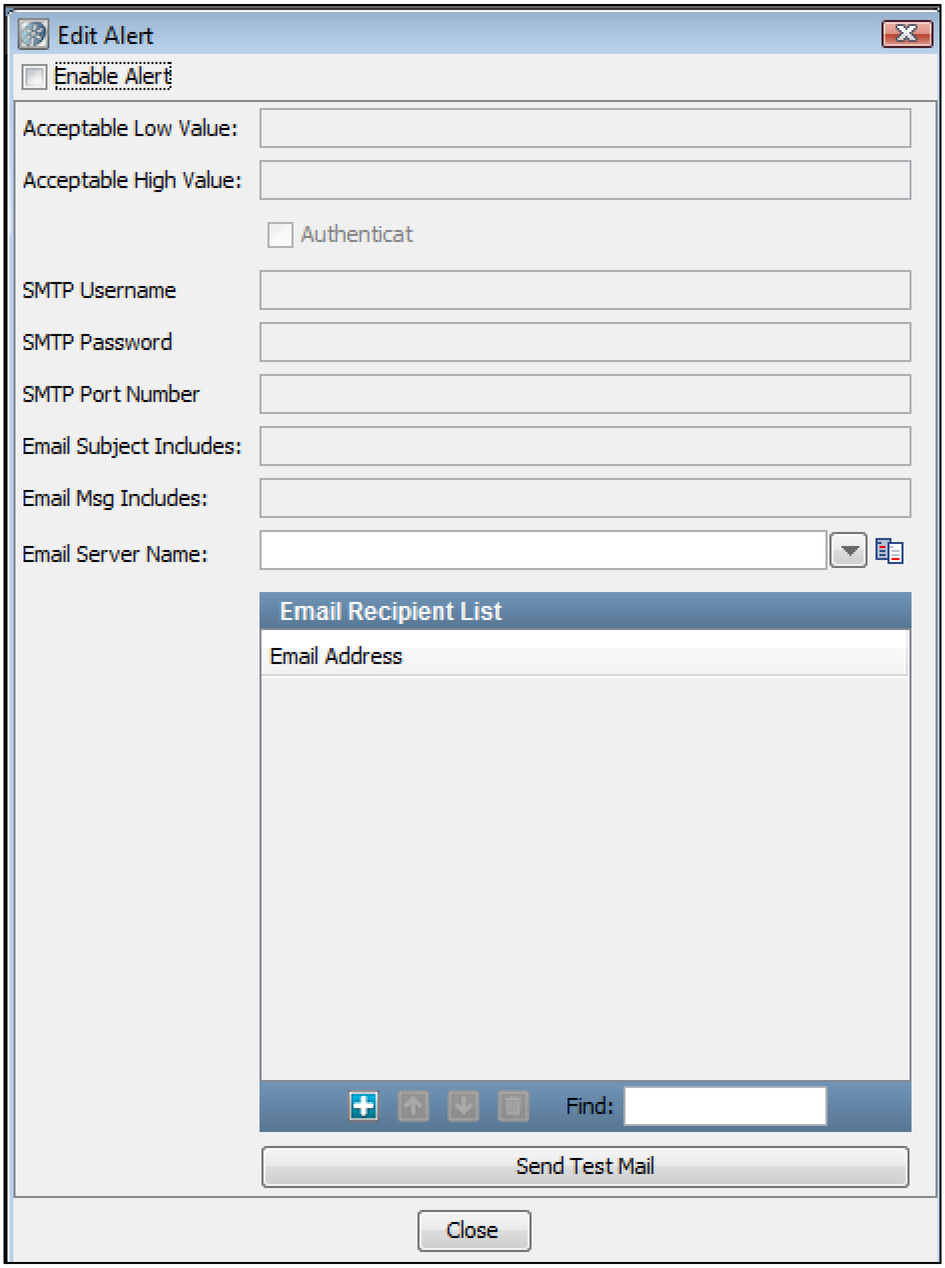
Color	Short Name	Long Name	Scale	Edit Email ...
🟥	Event: Cycle failed/*	A LISA Event Metric: Event: Cycle failed/*	Auto	Set Alert
🟥	Event: Step error/*	A LISA Event Metric: Event: Step error/*	Auto	Set Alert
🟦	Event: Step started/*	A LISA Event Metric: Event: Step started/*	Auto	Set Alert
🟥	Event: Abort/*	A LISA Event Metric: Event: Abort/*	Auto	Set Alert
🟪	LISA: Avg Resp Time (ms)	A LISA built-in metric: average response time in milliseconds	Auto	Set Alert
🟩	LISA: Steps per second	A LISA built-in metric: Steps per second (non-quiet) since the test started	Auto	Set Alert
🟪	LISA: Instances	A LISA built-in metric: number of running instances	Auto	Set Alert
⬛	LISA: Min Resp Time (ms)	A LISA built-in metric: minimum observed response time in milliseconds	Auto	Set Alert

Descriptions of all the metrics in all categories, and details on how to configure them for inclusion in reports, can be found in Generating Metrics.

Setting an Email Alert

To set an email alert on an individual metric, click the Set Alert button corresponding to that metric.

The Edit Alert dialog appears.



You can set the acceptable limits for the metric (low and high value), and the details to be sent in an email. You must provide the name of your email server and a list of email addresses.

Email alerts added to staging documents store the SMTP password as an encrypted value. Also, if you open an existing staging document and re-save it, the SMTP password is saved as an encrypted value.

Email addresses are added and removed by using the Add and Delete buttons at the bottom of the window.

When you are finished, click Close.

Notice that the Set Alert button is now an Edit Alert button on the Metrics tab.

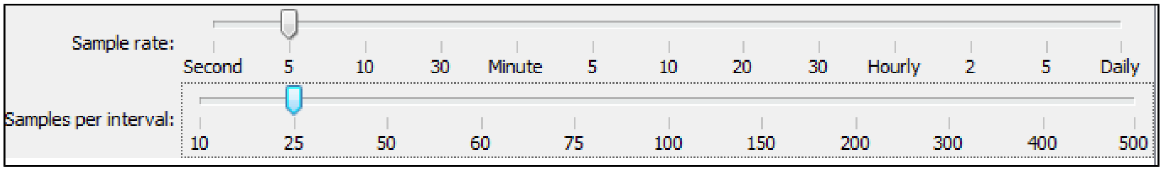
⚠ For the email alert to work, you also need to set the SMTP server-related paths in the **lisa.properties** file.

Sampling Parameters

The bottom panel has two slider bars that let you set sampling parameters:

- **Sample rate**: Specifies how often to take a sample; that is, record the value of a metric. It is specified as a time period, and is the reciprocal of the sampling rate.

- **Samples per interval**: Specifies how many samples are used to create an interval for calculating summary values for the metric.



Taking sample values every minute (Sample rate=1 minute) and averaging every 60 samples (Samples per interval=60) will produce a metric value every minute and a summary value (average) every hour.

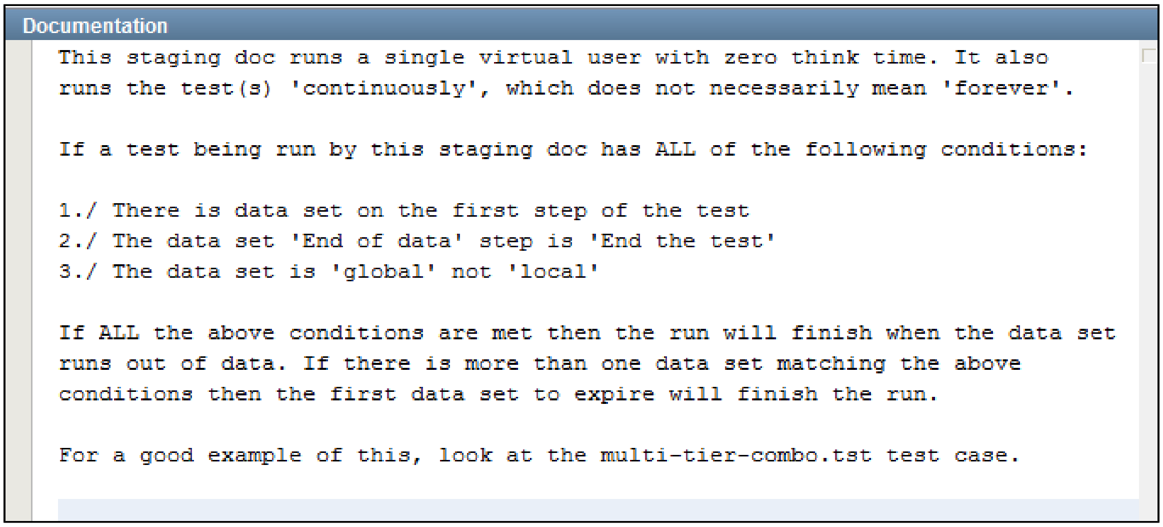
For example, the default is a 1-second sample rate, and 10 samples per interval (making an interval 10 seconds).

The preceding example uses 5 seconds per sample, and 25 samples per interval, producing a metric value every 5 seconds and a summary value every 125 seconds.

Metric values are stored in XML files or database tables for inclusion in reports (see the previous section on reports).

Staging Document Editor - Documentation Tab

The Documentation tab of the Staging Document Editor lets you enter descriptive information about the staging document.

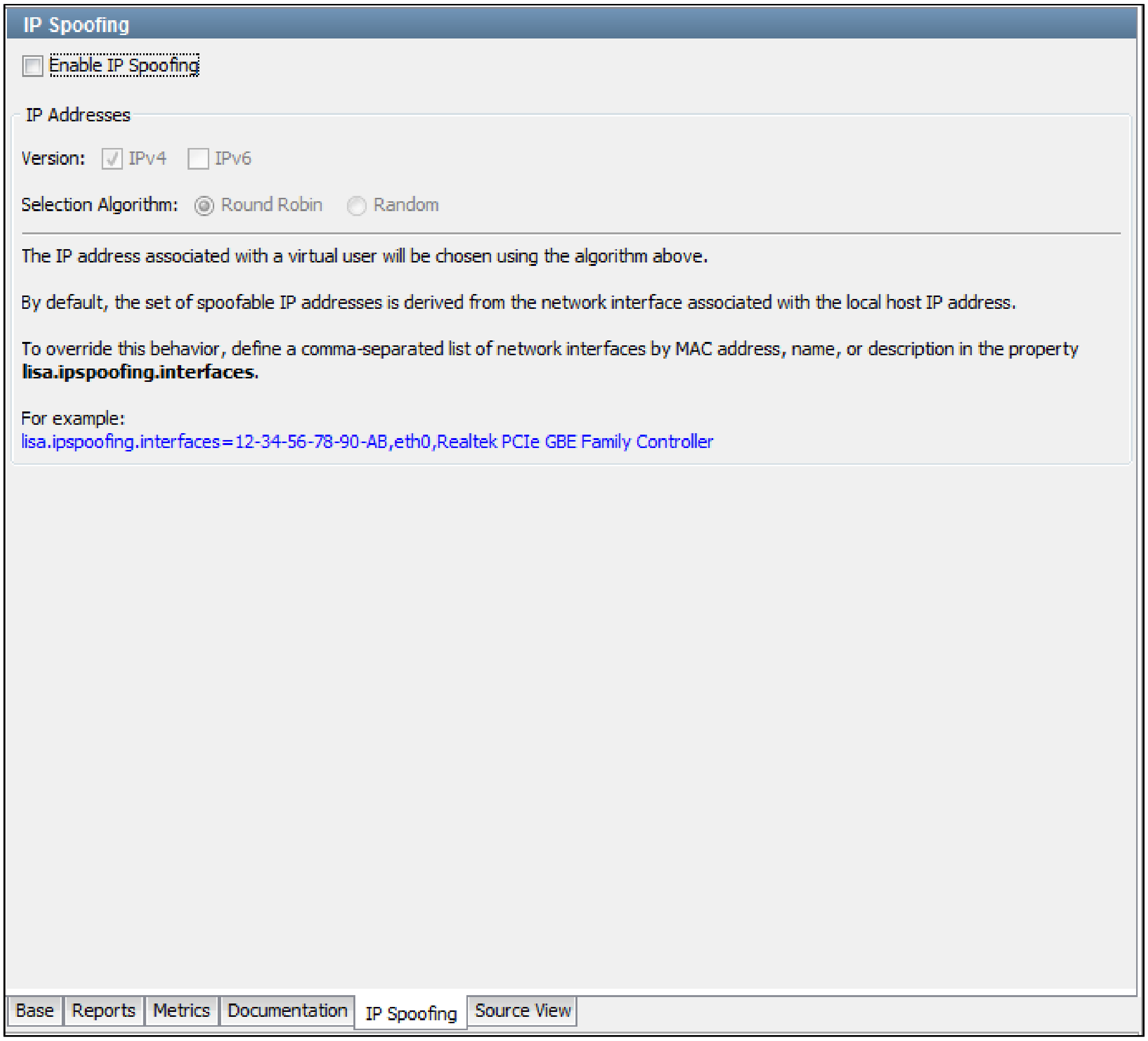


```
This staging doc runs a single virtual user with zero think time. It also
runs the test(s) 'continuously', which does not necessarily mean 'forever'.

If a test being run by this staging doc has ALL of the following conditions:

1./ There is data set on the first step of the test
2./ The data set 'End of data' step is 'End the test'
3./ The data set is 'global' not 'local'

If ALL the above conditions are met then the run will finish when the data set
runs out of data. If there is more than one data set matching the above
conditions then the first data set to expire will finish the run.

For a good example of this, look at the multi-tier-combo.tst test case.
```

Staging Document Editor - IP Spoofing Tab

The IP Spoofing tab of the Staging Document Editor lets multiple IP addresses on a network interface to be used when making network requests.

In a performance testing scenario, enabling IP spoofing against a system under test gives the appearance that requests are originating from many different virtual users. For systems such as web applications, this outcome will often more closely resemble "real-world" behavior.


IP Spoofing

□ Enable IP Spoofing

IP Addresses

Version: ☑ IPv4 □ IPv6

Selection Algorithm: ● Round Robin ○ Random

The IP address associated with a virtual user will be chosen using the algorithm above.

By default, the set of spoofable IP addresses is derived from the network interface associated with the local host IP address.

To override this behavior, define a comma-separated list of network interfaces by MAC address, name, or description in the property **lisa.ipspoofing.interfaces**.

For example:
lisa.ipspoofing.interfaces=12-34-56-78-90-AB,eth0,Realtek PCIe GBE Family Controller

- **Enable IP Spoofing:** If selected, IP addresses will be spoofed for all supported LISA steps in a test case.

Version

- **IPv4:** If selected, IPv4 addresses will be spoofed.
- **IPv6:** If selected, IPv6 addresses will be spoofed.

Either **IPv4** or **IPv6** must be selected.

Selection Algorithm

- **Round Robin:** Spoofed IP addresses will be selected in a round-robin format.
- **Random:** Spoofed IP addresses will be selected in a random format.

Support

Currently, IP spoofing is supported only for HTTP.

You can configure the following steps to use IP spoofing:

- HTTP/HTML Request
- REST Step
- Web Service Execution (XML)
- Raw SOAP Request

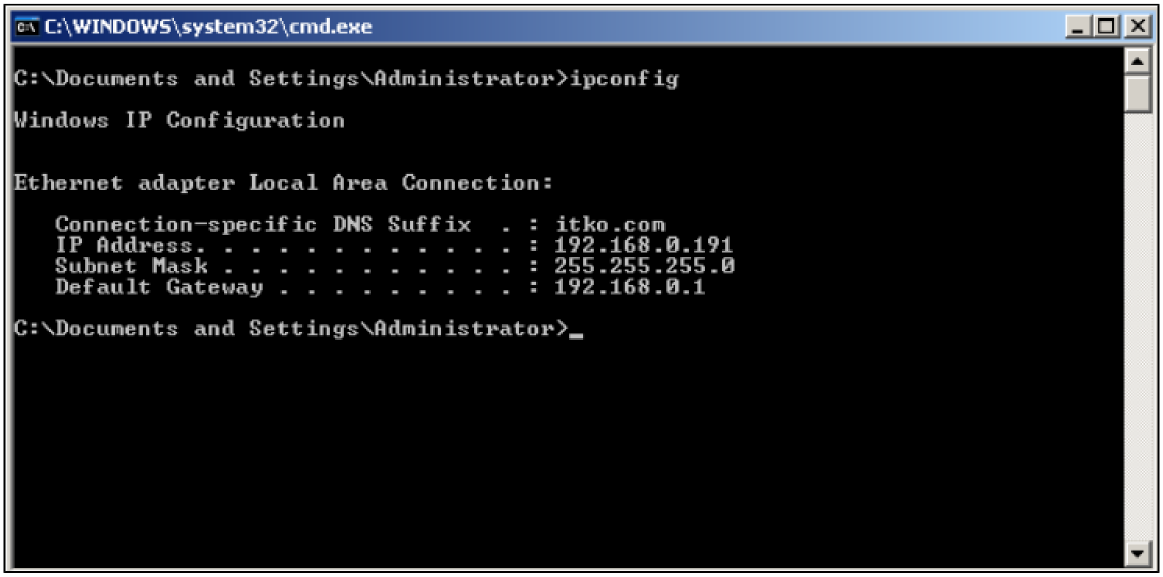
Configuring IP Addresses in Windows

This section describes how to add IP addresses to a network interface for IP spoofing.

⚠ Before adding IP addresses, make sure that you are an Administrator.

On Windows, IP address information can be obtained by using the command-line utility **ipconfig**.

For example, the following screenshot shows the output of **ipconfig** for a server with a single network interface card. It has a single IP address, **192.168.0.191** and is named **Local Area Connection**.



```
C:\Documents and Settings\Administrator>ipconfig

Windows IP Configuration


Ethernet adapter Local Area Connection:

   Connection-specific DNS Suffix  . : itko.com
   IP Address. . . . . . . . . . . . : 192.168.0.191
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.0.1

C:\Documents and Settings\Administrator>_
```

To add a single IP address, 192.168.0.201, the **netsh** command-line utility can be used:

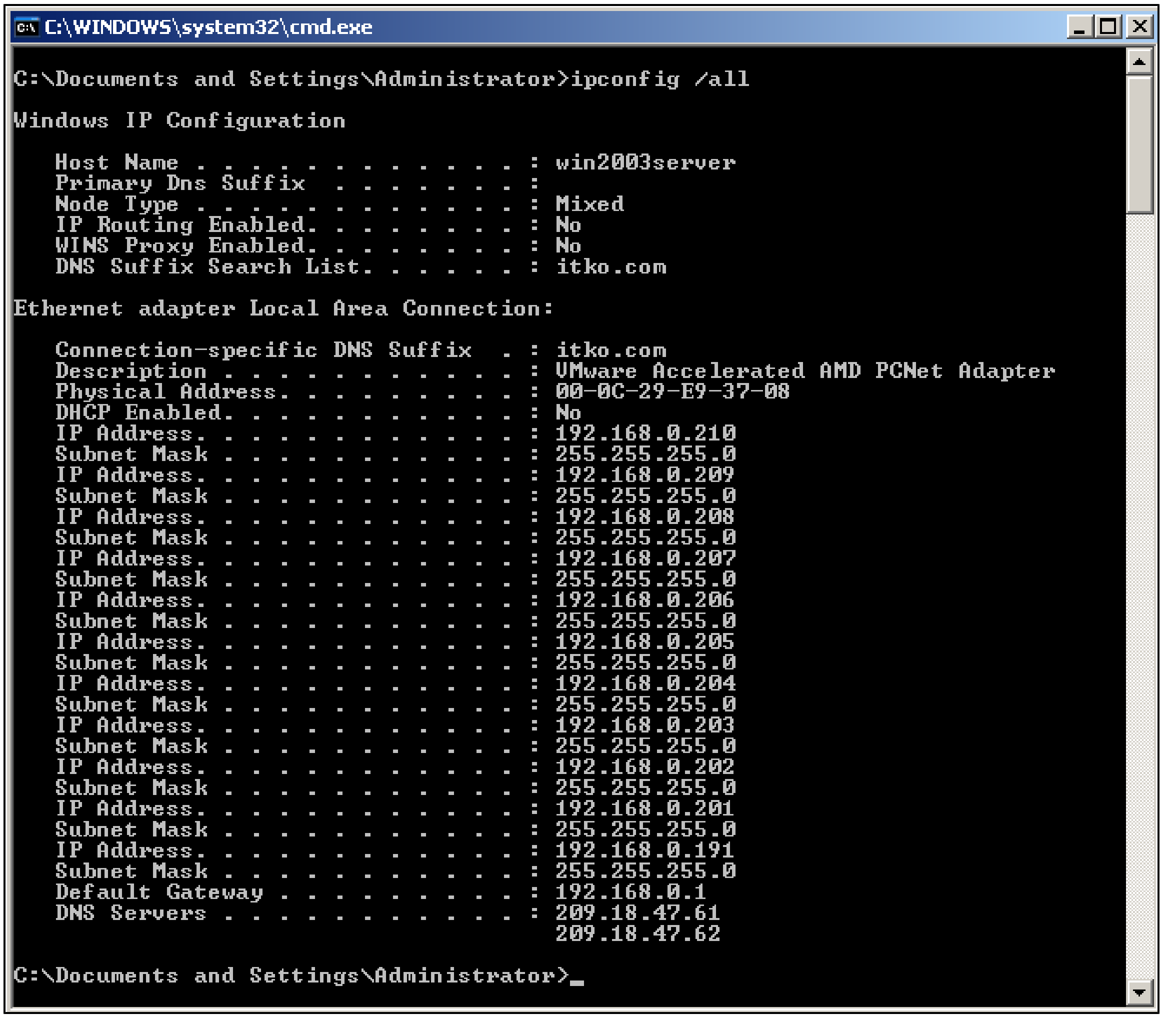
```
netsh in ip add address "Local Area Connection" 192.168.0.201 255.255.255.0
```

If many IP addresses are to be added, **netsh** can be used in a loop.

For example, the following command will add 9 more IP addresses between 192.168.0.202 and 192.168.0.210:

```
for /L %i in (202, 1, 210) do netsh in ip add address "Local Area Connection" 192.168.0.%i
255.255.255.0
```

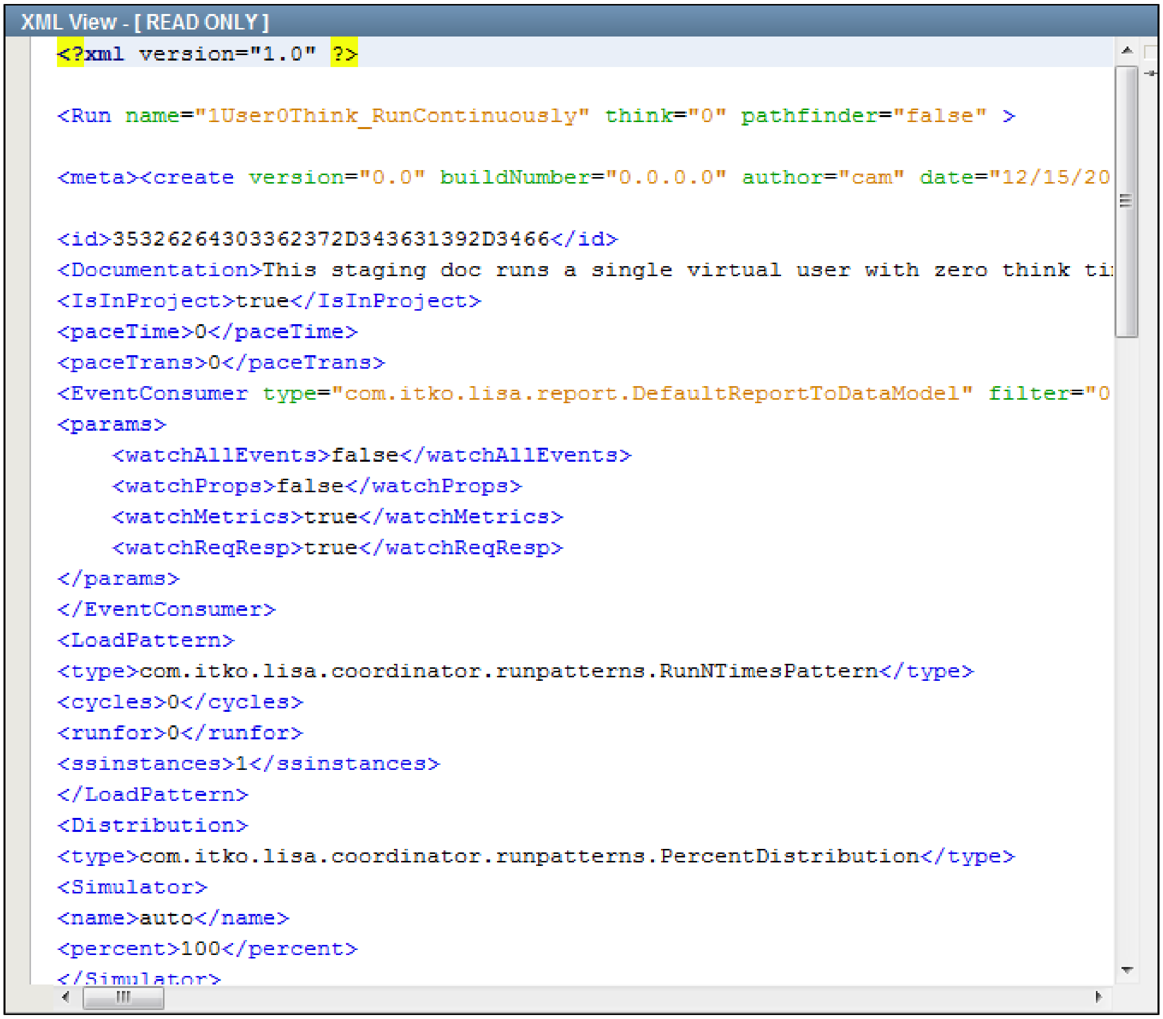
If these commands are successful, the new IP addresses can be verified by using the command-line utility **ipconfig /all**.



```
C:\Documents and Settings\Administrator>ipconfig /all

Windows IP Configuration

   Host Name . . . . . . . . . . . . : win2003server
   Primary Dns Suffix  . . . . . . . :
   Node Type . . . . . . . . . . . . : Mixed
   IP Routing Enabled. . . . . . . . : No
   WINS Proxy Enabled. . . . . . . . : No
   DNS Suffix Search List. . . . . . : itko.com

Ethernet adapter Local Area Connection:

   Connection-specific DNS Suffix  . : itko.com
   Description . . . . . . . . . . . : VMware Accelerated AMD PCNet Adapter
   Physical Address. . . . . . . . . : 00-0C-29-E9-37-08
   DHCP Enabled. . . . . . . . . . . : No
   IP Address. . . . . . . . . . . . : 192.168.0.210
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   IP Address. . . . . . . . . . . . : 192.168.0.209
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   IP Address. . . . . . . . . . . . : 192.168.0.208
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   IP Address. . . . . . . . . . . . : 192.168.0.207
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   IP Address. . . . . . . . . . . . : 192.168.0.206
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   IP Address. . . . . . . . . . . . : 192.168.0.205
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   IP Address. . . . . . . . . . . . : 192.168.0.204
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   IP Address. . . . . . . . . . . . : 192.168.0.203
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   IP Address. . . . . . . . . . . . : 192.168.0.202
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   IP Address. . . . . . . . . . . . : 192.168.0.201
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   IP Address. . . . . . . . . . . . : 192.168.0.191
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.0.1
   DNS Servers . . . . . . . . . . . : 209.18.47.61
                                       209.18.47.62

C:\Documents and Settings\Administrator>_
```

Staging Document Editor - Source View Tab

The Source View tab of the Staging Document Editor shows the XML source of the staging document.



```
<?xml version="1.0" ?>

<Run name="1User0Think_RunContinuously" think="0" pathfinder="false" >

<meta><create version="0.0" buildNumber="0.0.0.0" author="cam" date="12/15/20

<id>35326264303362372D343631392D3466</id>
<Documentation>This staging doc runs a single virtual user with zero think ti
<IsInProject>true</IsInProject>
<paceTime>0</paceTime>
<paceTrans>0</paceTrans>
<EventConsumer type="com.itko.lisa.report.DefaultReportToDataModel" filter="0
<params>
    <watchAllEvents>false</watchAllEvents>
    <watchProps>false</watchProps>
    <watchMetrics>true</watchMetrics>
    <watchReqResp>true</watchReqResp>
</params>
</EventConsumer>
<LoadPattern>
<type>com.itko.lisa.coordinator.runpatterns.RunNTimesPattern</type>
<cycles>0</cycles>
<runfor>0</runfor>
<ssinstances>1</ssinstances>
</LoadPattern>
<Distribution>
<type>com.itko.lisa.coordinator.runpatterns.PercentDistribution</type>
<Simulator>
<name>auto</name>
<percent>100</percent>
</Simulator>
```

Staging Document Examples

The following staging documents are provided in the StagingDocs folder of the examples project. All of them use the Run N Times load pattern and the Percent Distribution pattern.

You can use these documents as the starting point of your new staging document and rename it to save it under a different name.

- **1User0Think_RunContinuously**: Runs a single virtual user with zero think time. Runs the test continuously, which does not necessarily mean forever. LISA Pathfinder is not enabled.
- **1user1cycle0think**: Runs a single virtual user one time with zero think time. LISA Pathfinder is enabled.
- **ip-spoofing**: Lets you test IP spoofing support. LISA Pathfinder is not enabled.
- **jboss-jmx-metrics-localhost**: Runs three concurrent virtual users one time for 440 seconds. LISA Pathfinder is not enabled.
- **Run1User1Cycle**: Runs a single virtual user one time with 100 percent think time. LISA Pathfinder is not enabled.
- **Run1User1CycleShowAll**: Runs a single virtual user one time with 100 percent think time. LISA Pathfinder is not enabled.

Building Audit Documents

An audit document lets you set success criteria for a test case within a suite.

An audit document can track the following:

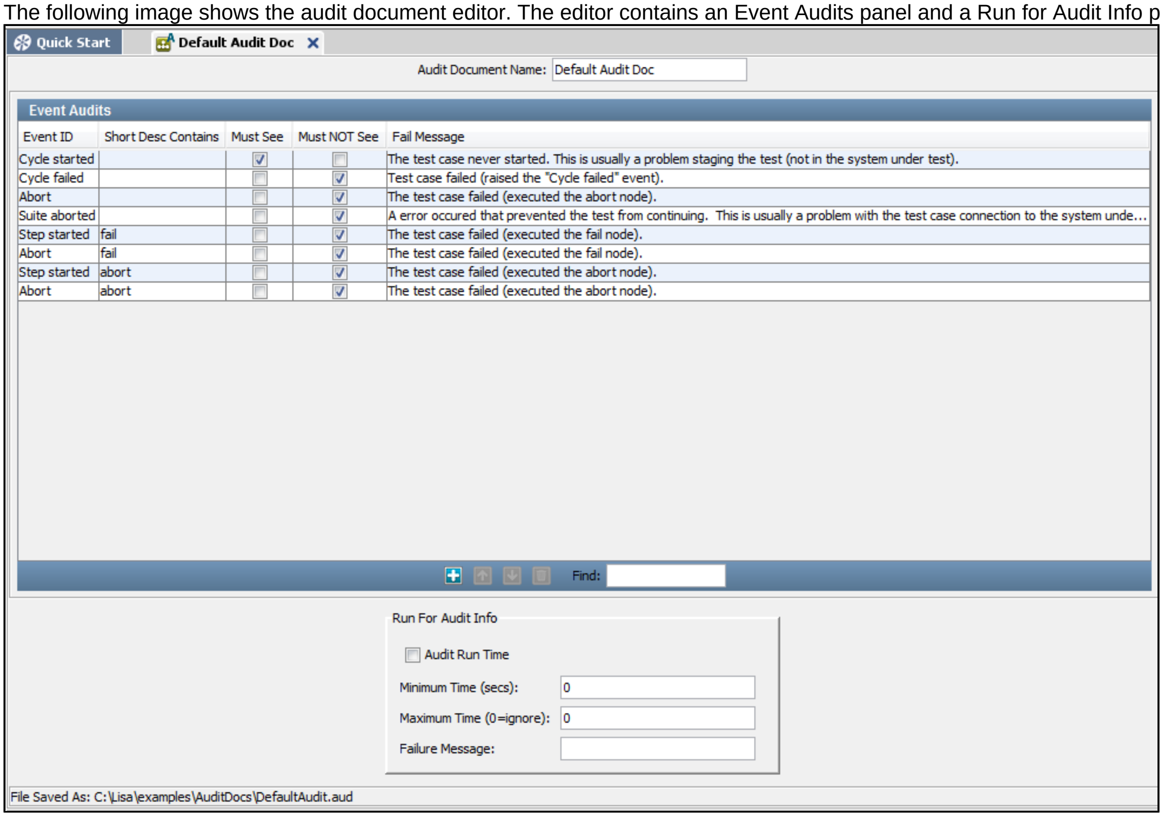
- Events that must occur or must not occur during a test
- Whether a test takes too little or too much time to complete

You can specify audit documents in the Base tab of the suite document editor. Be sure to select the Record All Events check box in the Reports tab.

Audit documents are located in the **AuditDocs** folder of a project. The file extension is .aud.

When you create a project, the **AuditDocs** folder contains a default audit document named **DefaultAudit.aud**.

The following image shows the audit document editor. The editor contains an Event Audits panel and a Run for Audit Info panel.



Audit Document Name: Default Audit Doc

Event ID	Short Desc Contains	Must See	Must NOT See	Fail Message
Cycle started		☑	☐	The test case never started. This is usually a problem staging the test (not in the system under test).
Cycle failed		☐	☑	Test case failed (raised the "Cycle failed" event).
Abort		☐	☑	The test case failed (executed the abort node).
Suite aborted		☐	☑	A error occured that prevented the test from continuing. This is usually a problem with the test case connection to the system unde…
Step started	fail	☐	☑	The test case failed (executed the fail node).
Abort	fail	☐	☑	The test case failed (executed the fail node).
Step started	abort	☐	☑	The test case failed (executed the abort node).
Abort	abort	☐	☑	The test case failed (executed the abort node).

☐ Audit Run Time

Minimum Time (secs): 0

Maximum Time (0=ignore): 0

Failure Message:

File Saved As: C:\Lisa\examples\AuditDocs\DefaultAudit.aud

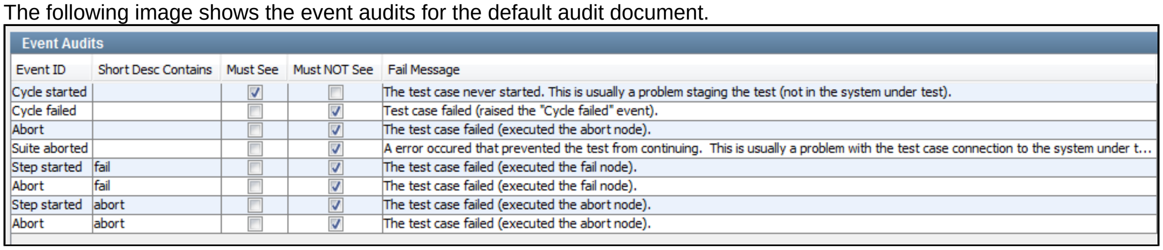
To create an audit document:

1. From the main menu, select File > New > Test Audit.
2. Enter the name of the audit document, and click OK.
   The audit document editor opens.
3. If you want to audit events, then configure the parameters in the Event Audits panel.
4. If you want to audit the execution time, then configure the parameters in the Run for Audit Info panel.
5. From the main menu, select File > Save.

Event Audits Panel

The Event Audits panel lets you specify events that must occur or must not occur during a test.

The following image shows the event audits for the default audit document.



Event ID	Short Desc Contains	Must See	Must NOT See	Fail Message
Cycle started		☑	☐	The test case never started. This is usually a problem staging the test (not in the system under test).
Cycle failed		☐	☑	Test case failed (raised the "Cycle failed" event).
Abort		☐	☑	The test case failed (executed the abort node).
Suite aborted		☐	☑	A error occured that prevented the test from continuing. This is usually a problem with the test case connection to the system under t…
Step started	fail	☐	☑	The test case failed (executed the fail node).
Abort	fail	☐	☑	The test case failed (executed the fail node).
Step started	abort	☐	☑	The test case failed (executed the abort node).
Abort	abort	☐	☑	The test case failed (executed the abort node).

To add an event, click the Add ⊞ icon.

In the new row, select the event name from the drop-down list in the Event ID column.

Each row has following parameters:

- **Short Desc Contains**: If you want to filter an event based on the value of the event's short description, then enter the keyword(s) from the short description in this column.
- **Must See**: Select this check box if the event must occur during the test.
- **Must NOT See**: Select this check box if the event must not occur during the test.
- **Fail Message**: A message to be logged if this audit fails.

Add additional rows for each event that you want to include.

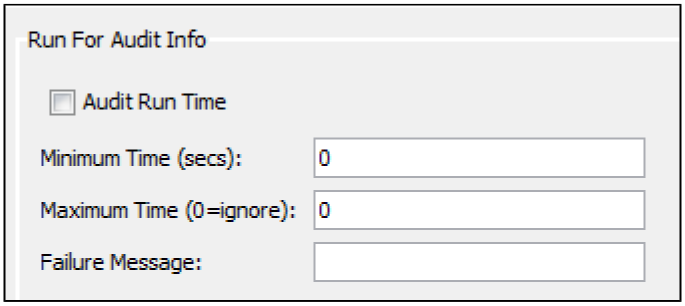
If you try to add event audits that logically conflict with each other, then the editor displays a warning message.

You can rearrange rows by using the Up ⬆ and Down ⬇ icons. You can delete rows by using the Delete 🗑 icon.

Run for Audit Info Panel

The Run for Audit Info panel lets you audit the execution time.

The following image shows the Run for Audit Info panel.



Run For Audit Info

□ Audit Run Time

Minimum Time (secs): 0

Maximum Time (0=ignore): 0

Failure Message:

This panel has following parameters:

- **Audit Run Time**: Select this check box to enable the execution time audit.
- **Minimum Time**: The minimum time (in seconds) that the test must run.
- **Maximum Time**: The maximum time (in seconds) that the test can run and still be considered a successful audit. If there is no maximum time constraint, then enter 0.
- **Failure Message**: A message to be logged if this audit fails.

Understanding Events

An event is a message broadcast from LISA informing any interested parties that an action has occurred.

The following topics are available.

Events Overview
Adding and Viewing Events
Types of Events

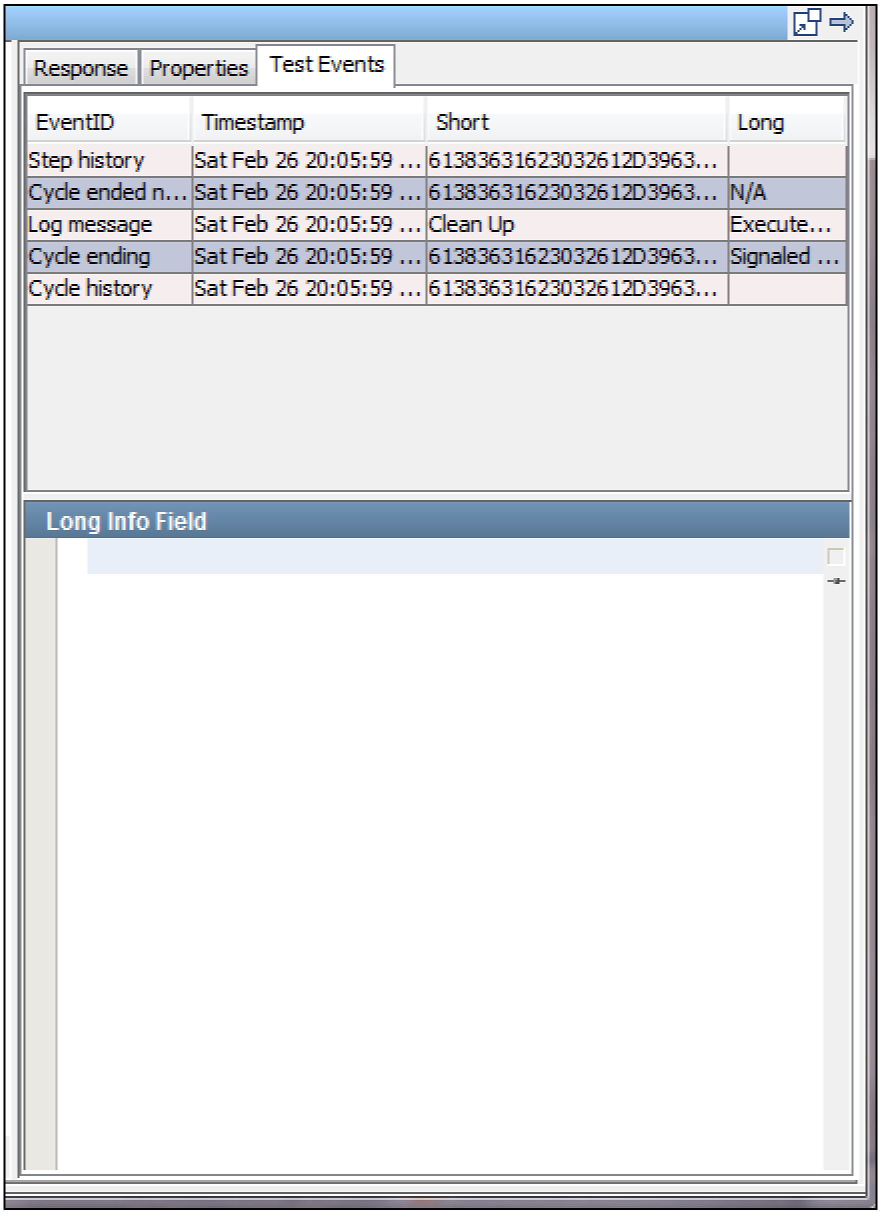
Events Overview

An event is a message broadcast from LISA informing any interested parties that an action has occurred. Events are created for every major action that occurs in a test case.

An event is created every time a step is executed, a property is set, a response time is reported, a result is returned, a test succeeds or fails, and more. LISA provides you the ability to see every event or to filter out the events that are not of interest.

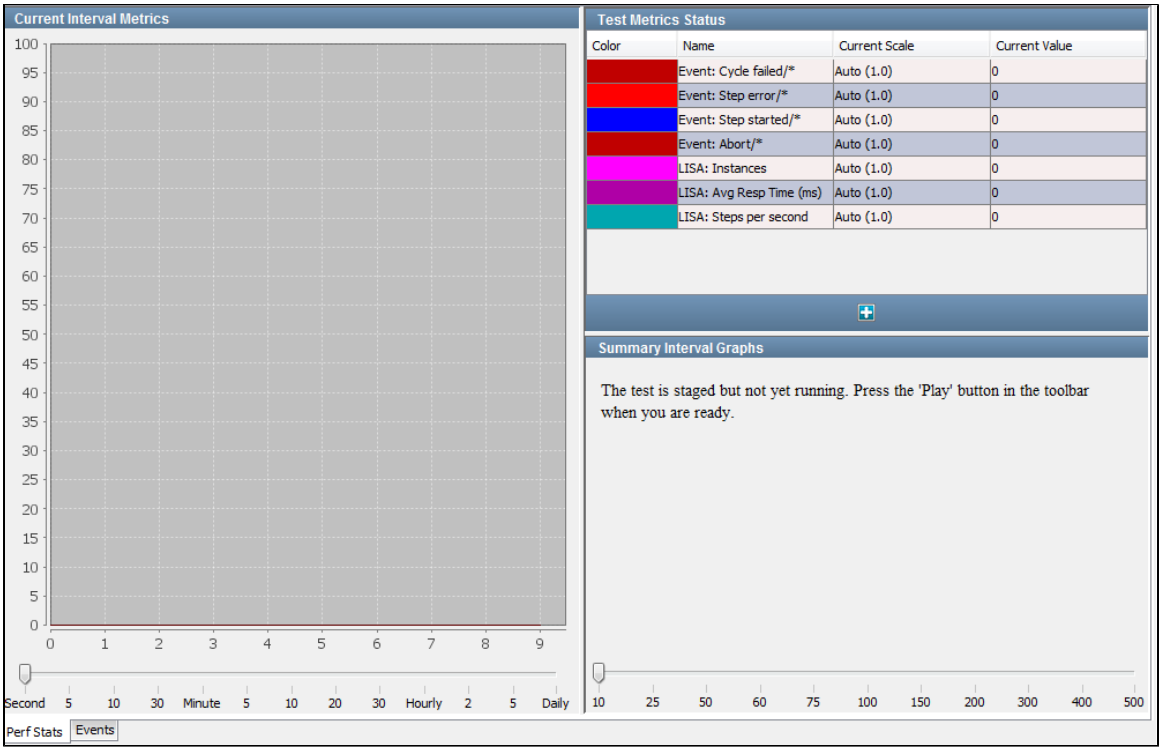
Events are important when you are monitoring tests or analyzing test results.

You can observe events during an Interactive Test Run (ITR) by clicking the Test Events tab in the ITR. The following image shows the Test Events tab.



EventID	Timestamp	Short	Long
Step history	Sat Feb 26 20:05:59 ...	61383631623032612D3963...	
Cycle ended n...	Sat Feb 26 20:05:59 ...	61383631623032612D3963...	N/A
Log message	Sat Feb 26 20:05:59 ...	Clean Up	Execute...
Cycle ending	Sat Feb 26 20:05:59 ...	61383631623032612D3963...	Signaled ...
Cycle history	Sat Feb 26 20:05:59 ...	61383631623032612D3963...	

You can observe and filter events in a Quick Test.



Color	Name	Current Scale	Current Value
🟥	Event: Cycle failed/*	Auto (1.0)	0
🟥	Event: Step error/*	Auto (1.0)	0
🟦	Event: Step started/*	Auto (1.0)	0
🟥	Event: Abort/*	Auto (1.0)	0
🟪	LISA: Instances	Auto (1.0)	0
🟪	LISA: Avg Resp Time (ms)	Auto (1.0)	0
🟦	LISA: Steps per second	Auto (1.0)	0

The test is staged but not yet running. Press the 'Play' button in the toolbar when you are ready.

The same can be done while monitoring a staged test or a test suite.

You can select events to be included in reports, and select events to be used as metrics that can be monitored and included in reports.

For more information about reports, see Reports. For more information about metrics, see Generating Metrics.

> ⚠ Internal to LISA, a step can also be referred to as a node, explaining why some events have *"node"* in the EventID.

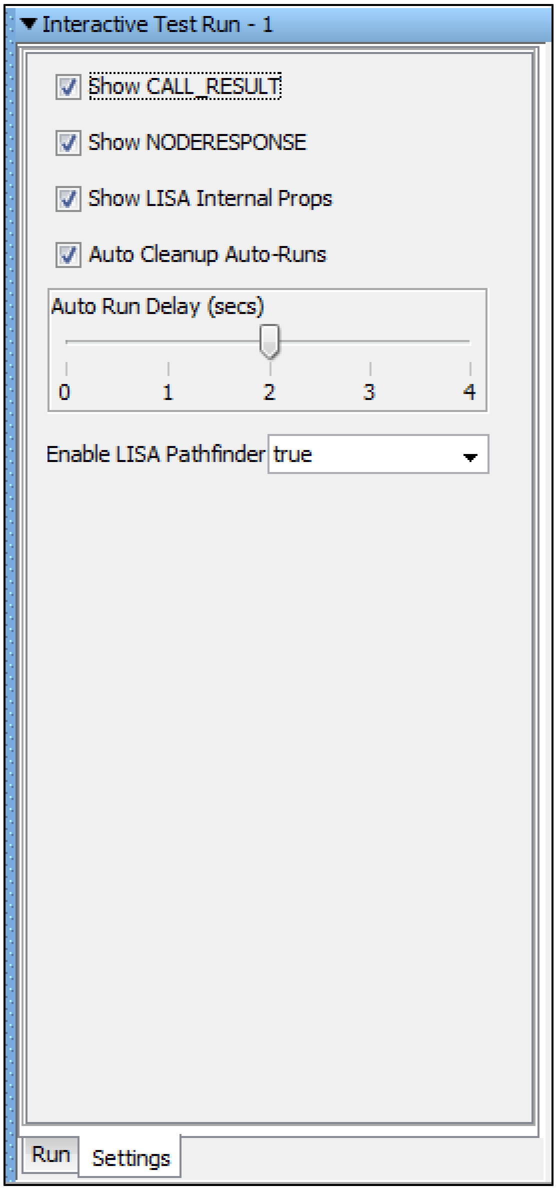
An event is characterized by an EventID, a short value, and a long value. EventIDs are keywords that indicate the type of event. The short values and long values contain information about the event. Their contents vary with the type of event.

Adding and Viewing Events

You can add events through the ITR and through a quick test or staging document.

Adding and Viewing Events through the ITR

You can enable some events in the Settings tab of the Interactive Test Run (ITR) utility. Select the Show CALL_RESULT and Show NODERESPONSE check boxes to view those events.



☑ Show CALL_RESULT

☑ Show NODERESPONSE

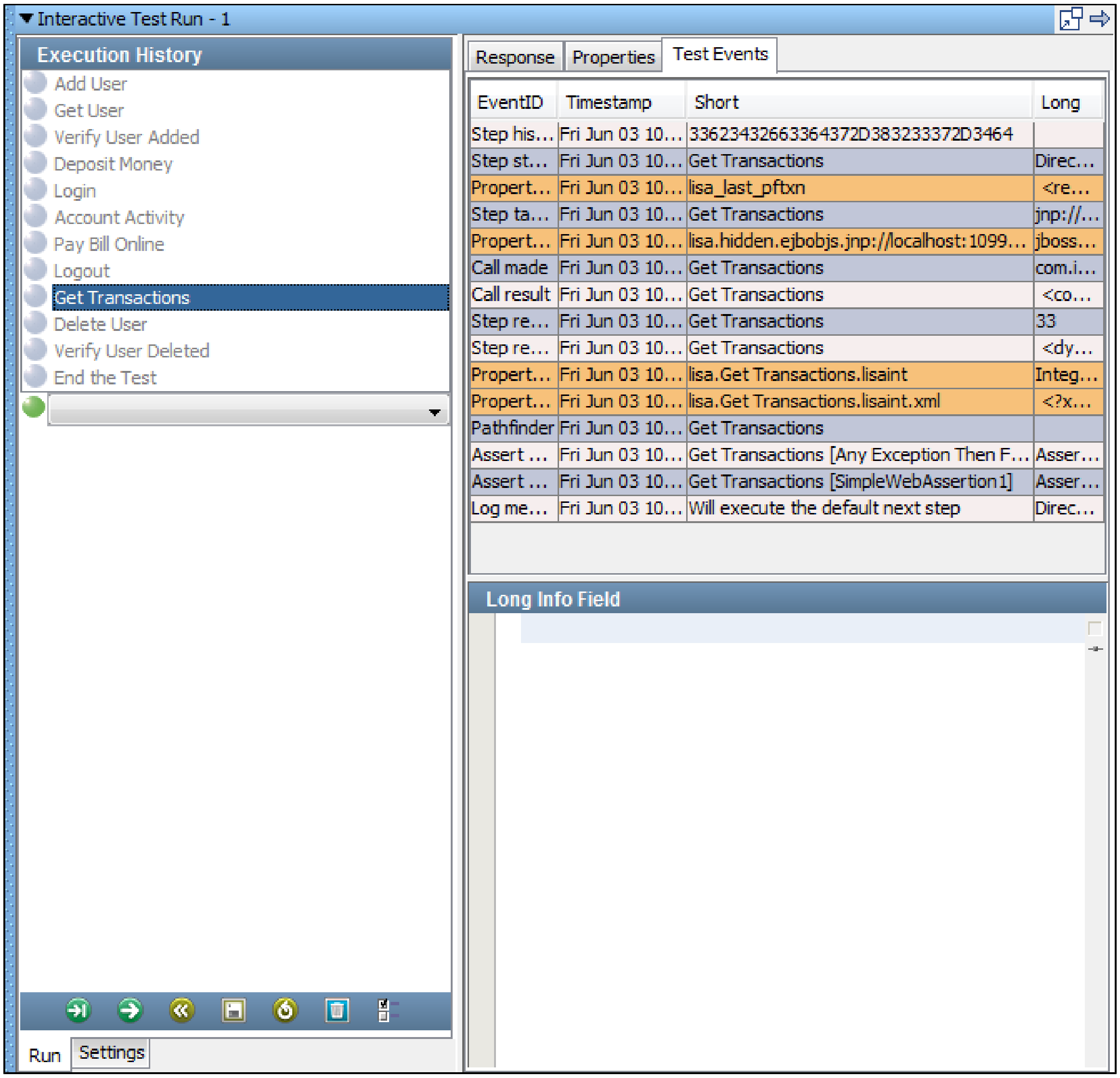
☑ Show LISA Internal Props

☑ Auto Cleanup Auto-Runs

Auto Run Delay (secs): 2

Enable LISA Pathfinder: true

You can view test events in the Interactive Test Run (ITR) by clicking the Test Events tab. The following image shows the Test Events tab.



Interactive Test Run - 1

Execution History

- Add User
- Get User
- Verify User Added
- Deposit Money
- Login
- Account Activity
- Pay Bill Online
- Logout
- Get Transactions
- Delete User
- Verify User Deleted
- End the Test

EventID	Timestamp	Short	Long
Step his…	Fri Jun 03 10…	33623432663364372D383233372D3464	
Step st…	Fri Jun 03 10…	Get Transactions	Direc…
Propert…	Fri Jun 03 10…	lisa_last_pftxn	<re…
Step ta…	Fri Jun 03 10…	Get Transactions	jnp://…
Propert…	Fri Jun 03 10…	lisa.hidden.ejbobjs.jnp://localhost:1099…	jboss…
Call made	Fri Jun 03 10…	Get Transactions	com.i…
Call result	Fri Jun 03 10…	Get Transactions	<co…
Step re…	Fri Jun 03 10…	Get Transactions	33
Step re…	Fri Jun 03 10…	Get Transactions	<dy…
Propert…	Fri Jun 03 10…	lisa.Get Transactions.lisaint	Integ…
Propert…	Fri Jun 03 10…	lisa.Get Transactions.lisaint.xml	<?x…
Pathfinder	Fri Jun 03 10…	Get Transactions	
Assert …	Fri Jun 03 10…	Get Transactions [Any Exception Then F…	Asser…
Assert …	Fri Jun 03 10…	Get Transactions [SimpleWebAssertion1]	Asser…
Log me…	Fri Jun 03 10…	Will execute the default next step	Direc…

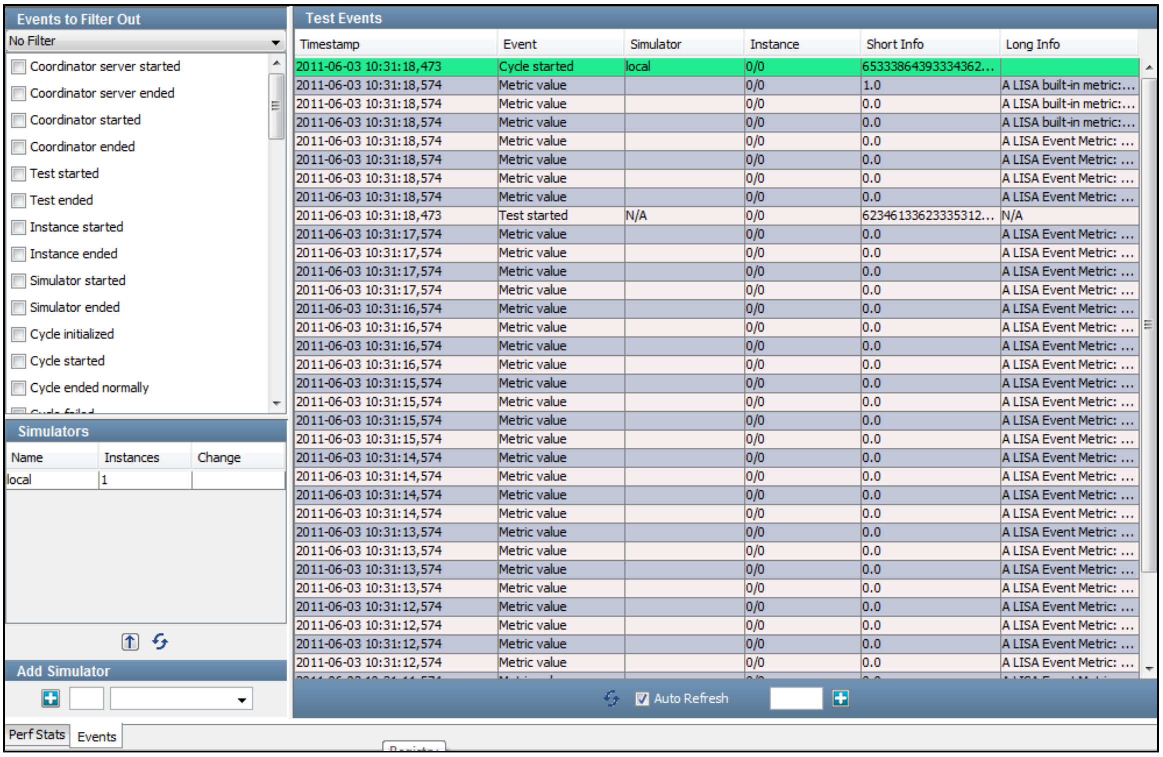
Adding and Viewing Events through a Quick Test/Staging Document

When you start a test case execution through the Start Test option or the Start Suite option, the test will be staged and relevant graphs will appear in the Perf Stats tab.

To view the test events, click the Events tab.

You can select the Events to Filter Out in the left panel or select from the predefined filter sets.

Select the Auto Refresh check box in the Events tab to refresh the test events list. As the test runs, you will see the events that you designated on the Events tab.



Events to Filter Out

No Filter

- ☐ Coordinator server started
- ☐ Coordinator server ended
- ☐ Coordinator started
- ☐ Coordinator ended
- ☐ Test started
- ☐ Test ended
- ☐ Instance started
- ☐ Instance ended
- ☐ Simulator started
- ☐ Simulator ended
- ☐ Cycle initialized
- ☐ Cycle started
- ☐ Cycle ended normally

Simulators

Name	Instances	Change
local	1	

Add Simulator

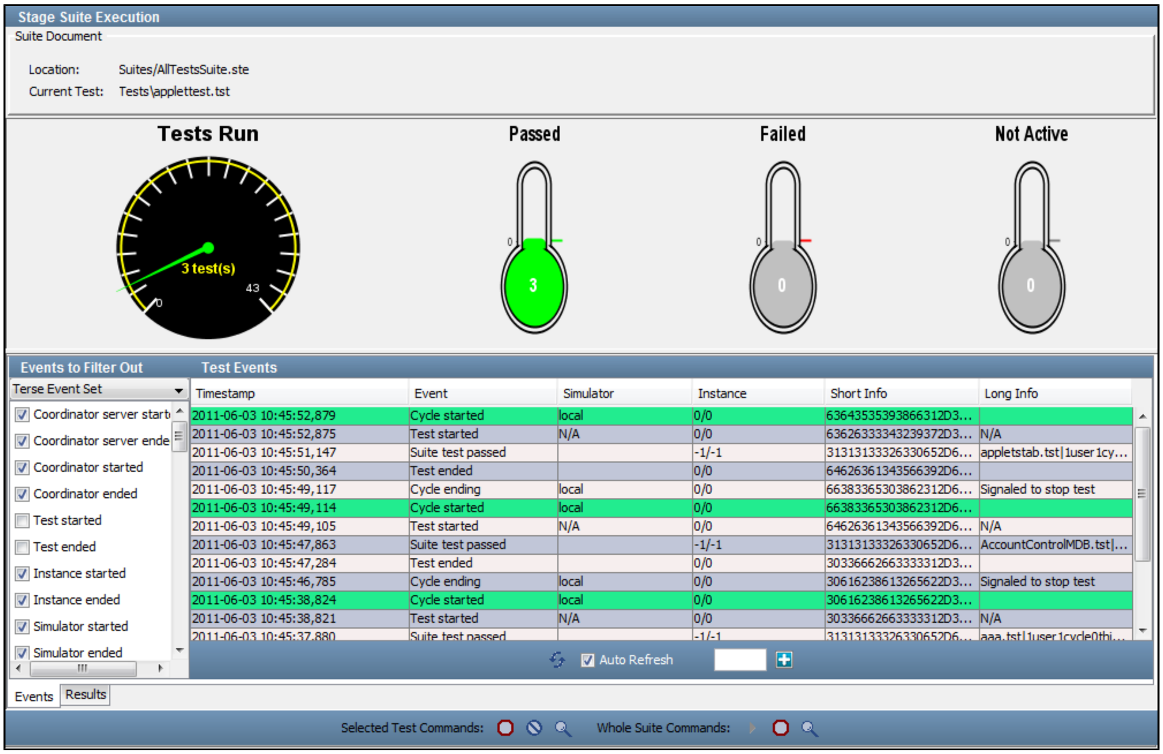
Test Events

Timestamp	Event	Simulator	Instance	Short Info	Long Info
2011-06-03 10:31:18,473	Cycle started	local	0/0	65333864393334362...	
2011-06-03 10:31:18,574	Metric value		0/0	1.0	A LISA built-in metric:...
2011-06-03 10:31:18,574	Metric value		0/0	0.0	A LISA built-in metric:...
2011-06-03 10:31:18,574	Metric value		0/0	0.0	A LISA built-in metric:...
2011-06-03 10:31:18,574	Metric value		0/0	0.0	A LISA Event Metric: ...
2011-06-03 10:31:18,574	Metric value		0/0	0.0	A LISA Event Metric: ...
2011-06-03 10:31:18,574	Metric value		0/0	0.0	A LISA Event Metric: ...
2011-06-03 10:31:18,574	Metric value		0/0	0.0	A LISA Event Metric: ...
2011-06-03 10:31:18,473	Test started	N/A	0/0	62346133623335312...	N/A
2011-06-03 10:31:17,574	Metric value		0/0	0.0	A LISA Event Metric: ...
2011-06-03 10:31:17,574	Metric value		0/0	0.0	A LISA Event Metric: ...
2011-06-03 10:31:17,574	Metric value		0/0	0.0	A LISA Event Metric: ...
2011-06-03 10:31:17,574	Metric value		0/0	0.0	A LISA Event Metric: ...
2011-06-03 10:31:16,574	Metric value		0/0	0.0	A LISA Event Metric: ...
2011-06-03 10:31:16,574	Metric value		0/0	0.0	A LISA Event Metric: ...
2011-06-03 10:31:16,574	Metric value		0/0	0.0	A LISA Event Metric: ...
2011-06-03 10:31:16,574	Metric value		0/0	0.0	A LISA Event Metric: ...
2011-06-03 10:31:15,574	Metric value		0/0	0.0	A LISA Event Metric: ...
2011-06-03 10:31:15,574	Metric value		0/0	0.0	A LISA Event Metric: ...
2011-06-03 10:31:15,574	Metric value		0/0	0.0	A LISA Event Metric: ...
2011-06-03 10:31:15,574	Metric value		0/0	0.0	A LISA Event Metric: ...
2011-06-03 10:31:14,574	Metric value		0/0	0.0	A LISA Event Metric: ...
2011-06-03 10:31:14,574	Metric value		0/0	0.0	A LISA Event Metric: ...
2011-06-03 10:31:14,574	Metric value		0/0	0.0	A LISA Event Metric: ...
2011-06-03 10:31:14,574	Metric value		0/0	0.0	A LISA Event Metric: ...
2011-06-03 10:31:13,574	Metric value		0/0	0.0	A LISA Event Metric: ...
2011-06-03 10:31:13,574	Metric value		0/0	0.0	A LISA Event Metric: ...
2011-06-03 10:31:13,574	Metric value		0/0	0.0	A LISA Event Metric: ...
2011-06-03 10:31:13,574	Metric value		0/0	0.0	A LISA Event Metric: ...
2011-06-03 10:31:12,574	Metric value		0/0	0.0	A LISA Event Metric: ...
2011-06-03 10:31:12,574	Metric value		0/0	0.0	A LISA Event Metric: ...
2011-06-03 10:31:12,574	Metric value		0/0	0.0	A LISA Event Metric: ...
2011-06-03 10:31:12,574	Metric value		0/0	0.0	A LISA Event Metric: ...

☑ Auto Refresh

View Events in Test Suites

You can also observe the events when you stage a test suite.



Stage Suite Execution

Suite Document

Location: Suites/AllTestsSuite.ste

Current Test: Tests\applettest.tst

Events to Filter Out

Terse Event Set

- ☑ Coordinator server starte
- ☑ Coordinator server ende
- ☑ Coordinator started
- ☑ Coordinator ended
- ☐ Test started
- ☐ Test ended
- ☑ Instance started
- ☑ Instance ended
- ☑ Simulator started
- ☑ Simulator ended

Test Events

Timestamp	Event	Simulator	Instance	Short Info	Long Info
2011-06-03 10:45:52,879	Cycle started	local	0/0	63643535393866312D3...	
2011-06-03 10:45:52,875	Test started	N/A	0/0	63626333343239372D3...	N/A
2011-06-03 10:45:51,147	Suite test passed		-1/-1	31313133326330652D6...	appletstab.tst|1user1cy...
2011-06-03 10:45:50,364	Test ended		0/0	64626361343566392D6...	
2011-06-03 10:45:49,117	Cycle ending	local	0/0	66383365303862312D6...	Signaled to stop test
2011-06-03 10:45:49,114	Cycle started	local	0/0	66383365303862312D6...	
2011-06-03 10:45:49,105	Test started	N/A	0/0	64626361343566392D6...	N/A
2011-06-03 10:45:47,863	Suite test passed		-1/-1	31313133326330652D6...	AccountControlMDB.tst|...
2011-06-03 10:45:47,284	Test ended		0/0	30336662663333312D3...	
2011-06-03 10:45:46,785	Cycle ending	local	0/0	30616238613265622D3...	Signaled to stop test
2011-06-03 10:45:38,824	Cycle started	local	0/0	30616238613265622D3...	
2011-06-03 10:45:38,821	Test started	N/A	0/0	30336662663333312D3...	N/A
2011-06-03 10:45:37,880	Suite test passed		-1/-1	31313133326330652D6...	aaa.tst|1user1cycle0thi...

☑ Auto Refresh

You can select events to be included in reports, and select events to be used as metrics that can be monitored and included in reports.

For more information about reports, see Reports. For more information about metrics, see Applying Metrics.

An event is characterized by an EventID, a short value, and a long value. EventIDs are keywords that indicate the type of event. The short values and long values contain information about the event. Their contents vary with the type of event.

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

Types of Events

The following table describes the standard events. The table is sorted by event name.

Event Name	Old Event Name	Description	Short Info	Long Info
Abort	EVENT_ABORT	This event ends a test in a "cannot finish" state. It is a failing type of end event.	Event Aborted	
Assert evaluated	EVENT_ASSERT_EVALUATED	Every non-embedded assert will generate either this event if it did not fire, or the EVENT_ASSERT if it did fire. Firing means it is true and its consequence will be followed. These events are the asserts that did not get to execute their consequence.	Assert Evaluated	
Assertion fired	EVENT_ASSERT	An assertion on a step "fired".	The name of the assertion	The log message of the assertion, or a LISA-generated message if there is no log message set
Call made	EVENT_CALL	Steps that perform object calls like web services or EJBs use this event to report each call that is made on the object.	The name of the step	The call as a string, for example, **void setName(java.lang.String name[Basic Checking])**
Call result	EVENT_CALLRESULT	Steps that perform object calls like web services or EJBs use this event to report the response they get from calls.	The name of the step	The call response as a string. If the response is an object, then an XML view of the object is shown.
Coordinator ended	EVENT_REMOVECOORDINATOR	A coordinator has been removed.	The name of the coordinator server	
Coordinator server ended	EVENT_REMOVECOORDSERVER	The coordinator server was ended.	The name of the coordinator server	
Coordinator server started	EVENT_COORDSERVERCREATED	The coordinator server was created.	The name of the coordinator server	

Coordinator started	EVENT_NEWCOORDINATOR	A new coordinator was created.	The name of the coordinator server	
Cycle ended normally	EVENT_NORMALEND	Indicates that the test execution completed in a successful state	The name of the test case	
Cycle ending	EVENT_STOPTESTSIGNAL	The instances have been instructed to stop testing.		
Cycle failed	EVENT_TESTFAILED	Indicates that the test execution failed. That there are no exceptions in the test case, but either logic errors in the test case or in the system under test caused the test case not to complete as expected.	The name of the test case	
Cycle history	EVENT_CYCLE_HISTORY	Every model that runs will generate one of these events. It is the final trace of all the details of its run.	Cycle History	
Cycle initialized	EVENT_INIT	Indicates that the test has been initialized (loaded).	The name of the test case	
Cycle runtime error	EVENT_TESTRUNERROR	An abnormal LISA error occurred. For example, the coordinator lost its connection to a simulator while running a test.	Varies but it is usually the name of the test element (step, data set, or filter) that has the error	Usually a message that explains the error
Cycle started	EVENT_START	Indicates that this test instance and cycle have just started.	The name of the test case	
Data set read	EVENT_DATA_SET_READ	Data sets generate an event that makes it clear what values are about to be used.	Data set read	
HTTP performance	EVENT_HTTP_PERFSTATS	This event is generated for every HTTP transaction that LISA executes to capture the performance statistics.	HTTP performance statistics	
Info message	EVENT_NODEMSG	Basic logging data. For example, the HTTP/HTML request step will send this message with the step name in the short field and the URL being sent to the server in the long field.	Usually the name of the step during which this message was generated	Usually a message that LISA generated
Instance ended	EVENT_VUSEREND	Sent when a simulator instance has finished.	The name of the simulator	
Instance started	EVENT_VUSERSTART	Sent when a simulator creates an instance.	The name of the simulator	
Log message	EVENT_LOGMSG	Basic logging data. Can be turned off, when filtering events, to minimize overhead.	The message sent to the log	

Metric alert	EVENT_METRICALERT	A metric has been collected and is reporting its value.	The short name of the metric, for example, **LISA: Avg Response Time**	The value of the metric collected
Metric started	EVENT_METRIC_START	Metrics that are collected generate real-time events of their values.	Metric Started	
Metric value	EVENT_METRIC_VALUE	Metrics that are collected generate real-time events of their values.	Metric value	
Model definition error	EVENT_TESTDEFERROR	A test case error was discovered during execution of the test. For example, the name is constructed from a property that does not exist.	Varies but it is usually the name of the test element (step, data set, or filter) that has the error	Usually a message that explains the error
Pathfinder	EVENT_INTEGRATION	The system being tested has LISA integration enabled. This event contains the LISA integration XML data that was captured.	The name of the step	The XML representation of the LISA Integration data captured
Property set	EVENT_SETPROP	A property was set.	The property key	The property value
Simulator ended	EVENT_SIMEND	Sent when a simulator has ended.	The name of the simulator	
Simulator started	EVENT_SIMSTARTED	Sent when a simulator has started.	The name of the simulator	
Step bandwidth consumed	EVENT_BANDWIDTH	Approximate amount of data sent and received from the system under test for the step execution	The name of the step	Actual number of bytes read/received
Step error	EVENT_TRANSFAILED	An error has occurred in the system under test. For example, there was no response from a web server. This event is on a per-step basis. The EVENT_TESTFAILED refers to the complete test case.	The name of the step	If available, a message to help determine the cause of the failure
Step history	EVENT_NODE_HISTORY	Every node has a history event that fires of type **com.itko.lisa.test.NodeExecHistory** in its long info.	Step History	
Step request	EVENT_REQUEST	Steps that support this event use it to report the actual request made to the system under test.	The name of the step	The request data as a string
Step request bandwidth	EVENT_REQUEST_BANDWIDTH			

Step response	EVENT_NODERESPONSE	Indicates that a transaction has been completed against the system under test.	The name of the step	The response data as a string
Step response bandwidth	EVENT_RESPONSE_BANDWIDTH			
Step response time	EVENT_RESPTIME	The amount of time a transaction took to execute against the system under test.	The name of the step	The number of milliseconds to execute the step
Step started	EVENT_TRANSACTION	A step is being executed. Transaction is a synonym for step.	The name of the step	
Step target	EVENT_NODE_TARGET	Every step has a target, such as the URL for a web request or the JNDI name of an EJB.		
Step warning	EVENT_WARNING	A warning was recorded. For example, a filter took the default value because it could not find the current value.	Step warning	
Subprocess finished	EVENT_SUBPROCESS_ENDED	A subprocess has finished executing.		
Subprocess ran	EVENT_SUBPROCESS	A subprocess (subtest) was started.		
Suite aborted	EVENT_SUITE_FINISHED	When a setup test (defined in a suite document) has failed, then the suite will not run the tests defined in the suite. This event informs you that this has happened.	The name of the suite	
Suite ended	EVENT_SUITE_SKIPPED	All of the tests running as part of a suite have finished.	The name of the suite	
Suite history	EVENT_SUITE_HISTORY	Every suite that runs will generate one of these events. It is the final trace of all the details of its run.	Suite History	
Suite setup/teardown	EVENT_SUITE_SETUPTEARDOWN	The suite has a setup or teardown test defined and that test has just started to run.	The name of the suite	The name of the test, path to the test, and other information
Suite started	EVENT_SUITE_STARTING	A suite is starting to run.	The name of the suite	
Suite test failed	EVENT_SUITE_TESTFAILED	A test running as part of a suite ended in failure.	The name of the suite	
Suite test passed	EVENT_SUITE_TESTPASSED	A test running as part of a suite ended successfully.	The name of the suite	
Suite test staged	EVENT_SUITE_TESTSTAGED	A test is staged to run as part of a suite.	The name of the suite	The name of the test, path to the test, and other information

Test ended	EVENT_TESTEND	Sent when the coordinator stops the test.	The name of the test case	
Test not active	EVENT_TEST_NOT_ACTIVE	A test in a suite is marked as inactive.	Event Test Not Active	
Test started	EVENT_TESTSTARTED	Sent when the coordinator starts the test.	The name of the test case	
VS log message	EVENT_VSE_LOG	VSE internal logging	VS log message	
VS no transaction match	EVENT_VSE_NO_TRANS_MATCH	A virtual service did not match a transaction request to at least one recorded response.	VS transaction no match	
VS service ended	EVENT_VSE_SERVICE_STOP	A virtual service was stopped.	VS service stopped	
VS service started	EVENT_VSE_SERVICE_START	A virtual service was started.	VS service started	
VS transaction finished	EVENT_VS_TRANSACTION_FINISHED	A virtual service finished processing a transaction request.		
VS transaction match	EVENT_VSE_TRANS_MATCH	A virtual service matched a transaction request to at least one recorded response.	VS transaction match	
VSE server reset	EVENT_VSE_SERVER_RESET	A service reset request was made of a server.	VSE server resets	
VSE server shutdown	EVENT_VSE_SERVER_SHUTDOWN	A virtual service environment was asked to shut down.	VSE server shutdown	
VSE server stop	EVENT_VSE_SERVER_STOP	A service stop request was made of a server.	VSE server stops	

Generating Metrics

There are a wealth of features related to the generation and capture of data for the purpose of reporting.

Metric collection, our own metric calculation method, is an extensible reporting mechanism, and provides the ability to generate a variety of reports to a variety of outputs.

Metrics lets you apply quantitative methods and measurements to the performance and functional aspects of your tests, and the system under test.

The software metric is a measure of some property of a piece of a software, a hardware system, or their specifications. Quantitative methods using metrics have proved to be very powerful in several areas of computing, and testing is no exception.

Metrics fall into two broad groups:

- **Gauge:** A gauge provides an instantaneous reading of a value, such as response time or CPU utilization.
- **Counter:** A counter provides a continuous count of a property, such as the number of failed tests.

For most metrics, the type of metric, gauge or counter, is already known. When a metric could be used as either, you can specify the type you want.

Metrics can be added to the following staging documents and test suite documents, and can be generated by test monitors and quick tests. Information about specifying and generating metrics for each document or test is available in sections about each editor.

- Quick Tests: Use to monitor the test.

- Staging Documents: Use to include in reports.
- Test Suite Documents: Use to include in reports.
- Test Monitors: Use to monitor tests.

Types of Metrics

Types of Metrics

LISA Whole Test Metrics

Whole Test Metrics, as the name suggests, collect all the basic information about the test case and provide six sub-metrics.

The following sub-metrics are collected. The response time metrics are reported in milliseconds.

- **Instances** (the number of virtual users)
- **Avg Resp Time**
- **Max Resp Time**
- **Min Resp Time**
- **Last Resp Time**
- **Steps per second**

⚠ The number of virtual users (Instances), average response times (Avg Resp Time) and Steps Per Second sub-metrics are added by default to a staging document's metric list.

LISA Test Event Metrics

Test Event Metrics provide metrics based on LISA events. These metrics include both counters and gauges.

Event metrics let you filter events of a given type by including a regular expression to match the short description field of the event.

There are eight sub-metrics in the Test Event Metrics category.

- **Coordinator server started**
- **Coordinator server ended**
- **Coordinator started**
- **Coordinator ended**
- **Test started**
- **Test ended**
- **Instance started**
- **Instance ended**

To add test event metrics

In addition to selecting the metric, you can specify:

- **Key Expression Match**: Enter an expression to say to sample the chosen event only if it has this expression in its short description field. If you leave this blank, or enter *, then every event of this type will be reported.
- **Metric is a Counter**: If this box is selected, the value counts over time are recorded. If the box is cleared, the absolute value is recorded (metric is functioning as a gauge).
- Click OK to add this metric to the list of metrics.

SNMP Metrics

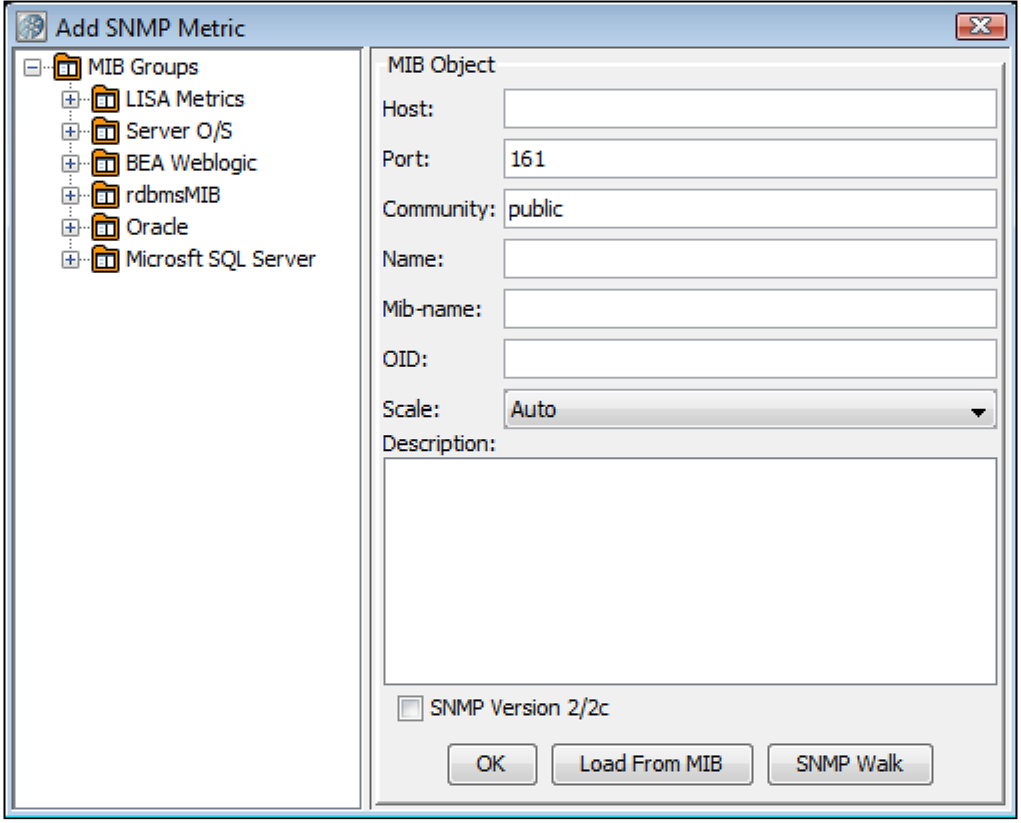
The **SNMP** metrics use the Simple Network Management Protocol (SNMP) to monitor system performance.

Setting up SNMP Support

Before you can collect SNMP metrics, you must configure SNMP. For details on setting up SNMP on UNIX and Windows, see Installing and Configuring SNMP.

Adding SNMP Metrics

Select **SNMP Metric** from the dialog and click OK. The Add SNMP Metric dialog is displayed.



Host:

Port: 161

Community: public

Name:

Mib-name:

OID:

Scale: Auto

☐ SNMP Version 2/2c

The **MIB Groups** tree displays all the SNMP metrics that come standard with LISA Workstation.

Easy setup for the following is provided by MIBs. A MIB is a predefined database of a set of metrics on a given domain.

- **Host**: Provides system information about a server hosting a coordinator server. Host metrics include **hrProcessorLoad** for CPU utilization and **hrSystemUptime** for the amount of time since this host was last initialized.
- **Server O/S**: Provides information related to the system, like up time, date, number of users, and so forth.
- **BEA WebLogic**: Provides JDBC, JMS, JVM, socket, servlet and web application information about a Server running BEA WebLogic. BEA WebLogic metrics include **jvmRuntime-HeapFreeCurrent** for the current amount of free memory in the JVM heap in bytes, and **webAppComponentRuntimeOpenSessionsCurrentCount** for the current total number of open sessions in this component.
- **RDBMS**: Provides information about a server running a generic relational database management system. RDBMS metrics include **rdbmsSrvInfoPageReads** for the number of physical page reads completed since the RDBMS was last restarted, and **rdbmsSrvInfoPageWrites** for the number of single page writes completed since the RDBMS was last restarted.
- **Oracle**: Provides information about a server running Oracle. Oracle metrics include **oraDbSysUserCommits** for the number of user commits, and **oraDbSysUserRollbacks** for the number of times data has rolled back.
- **Microsoft SQL Server**: Provides information about a server running Microsoft SQL Server. Microsoft SQL Server metrics include **mssqlSrvInfoCacheHitRatio** for the percentage of time that a requested data page was found in the data cache (instead of being read from disk), and **mssqlSrvInfoUserConnections** for the number of open user connections.

You can browse through these to select metrics.

When you click the target metric in the left panel, the required parameters for this metric are completed in the **MIB Object** form in the right panel. A description of the metric is displayed in the text box. The other information can be ignored or accepted.

You must enter the domain name or IP address of the host computer (Host) where you are collecting the metrics. Click OK to add this metric.

Repeat until you have added all the target SNMP metrics.

To add SNMP metrics that are not in the MIB Group tree, you must manually enter the data (OID) into the **MIB Object** form. An OID is the unique identifier of a particular metric, using a tree-structured naming scheme. The domain root OID for ITKO is ".1.3.6.1.4.1.12841.1.1", so all SNMP metrics will start from there.

The **Load From MIB** button lets you browse the file system for MIBs.

The **SNMP Walk** button lets you browse an SNMP tree.

⚠ All SNMP MIBs are supported. The set of MIBs displayed in the Add SNMP Metric dialog are only a sample of those supported. The Add SNMP Metric dialog makes it easier to understand the Object IDs (OIDs) in those MIBs. However, any valid OID works, not only those displayed in the Add SNMP Metric dialog. A set of all the standard MIBs from IETF and IANA are provided. Those MIBs are stored in the **LISA_HOME/snmp/ietf** and **LISA_HOME/snmp/iana** directories.

JMX Metrics

The JMX metrics uses the Java Management Extension (JMX) API to provide metrics.

JMX connectors are provided for easy set-up:

- Any JSR 160 RMI connection
- JBoss 3.2-4.0
- JSE 5 Connector
- Oracle AS (OCJ4)
- Tomcat 5.0.28
- WebLogic 6.1-8.1
- WebLogic 9.x
- WebSphere 5.x
- iTKO JMX Agents

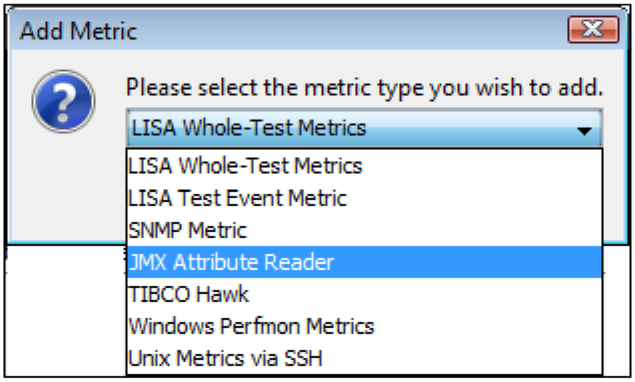
Each of these requires slightly different connection parameters. The values that you require for your particular server should be available from your server administrator. There is no agreement on standard metrics, so each provider provides slightly different metrics. To use other JMX features, you can invoke them as RMI steps.

Only numerical attributes are supported.

Here we will use JBoss as our example.

To add JMX metrics

1. Select **JMX Attribute Reader** from the dialog:

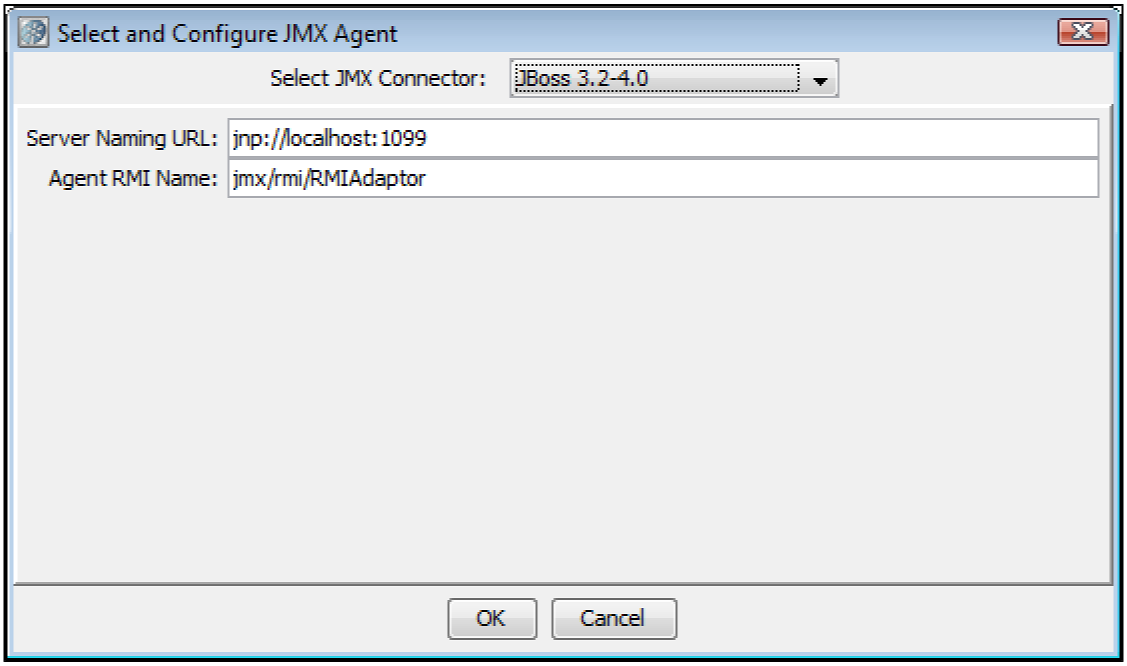


Add Metric

Please select the metric type you wish to add.

LISA Whole-Test Metrics

2. Click OK to display a second dialog, **Select and Configure JMX Agent**, to configure the JMX agent.



Select JMX Connector: JBoss 3.2-4.0

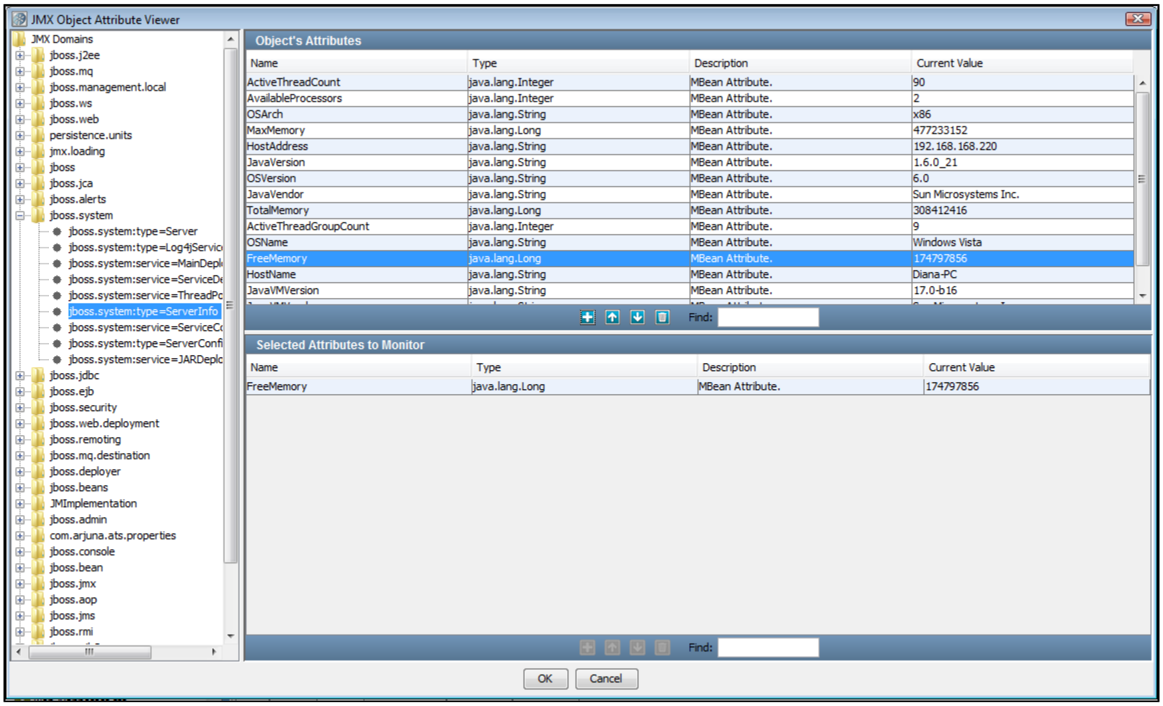
Server Naming URL: jnp://localhost:1099

Agent RMI Name: jmx/rmi/RMIAdaptor

3. Select the target connector, **JBoss**, and enter the connection information **Server Naming URL** and **Agent RMI name**, for your particular installation.

4. Click OK.

The **JMX Object Attribute Viewer** is now displayed:



Object's Attributes

Name	Type	Description	Current Value
ActiveThreadCount	java.lang.Integer	MBean Attribute.	90
AvailableProcessors	java.lang.Integer	MBean Attribute.	2
OSArch	java.lang.String	MBean Attribute.	x86
MaxMemory	java.lang.Long	MBean Attribute.	477233152
HostAddress	java.lang.String	MBean Attribute.	192.168.168.220
JavaVersion	java.lang.String	MBean Attribute.	1.6.0_21
OSVersion	java.lang.String	MBean Attribute.	6.0
JavaVendor	java.lang.String	MBean Attribute.	Sun Microsystems Inc.
TotalMemory	java.lang.Long	MBean Attribute.	308412416
ActiveThreadGroupCount	java.lang.Integer	MBean Attribute.	9
OSName	java.lang.String	MBean Attribute.	Windows Vista
FreeMemory	java.lang.Long	MBean Attribute.	174797856
HostName	java.lang.String	MBean Attribute.	Diana-PC
JavaVMVersion	java.lang.String	MBean Attribute.	17.0-b16

Selected Attributes to Monitor

Name	Type	Description	Current Value
FreeMemory	java.lang.Long	MBean Attribute.	174797856

5. On the left is the **JMX Domain hierarchy** for JBoss. JMX metrics use an object-attribute model where domain objects are an area published by the particular application server, (for instance "system"), and attributes are name/value pairs within the object.

When you select an object from this tree, the base attributes of that object are also displayed in the tree. After you select a base attribute, the rest of the attribute name appears in the list in the top right **Object Attribute panel**.

In the previous example, we selected the domain object to be **jboss.system**, the base attribute to be **ServerInfo**, and the rest of the attribute name to be **FreeMemory**. The attributes for ServerInfo are displayed in the **Object Attribute panel**.

To select one of these attributes (metrics), select the metric, and click the Add ➕ icon. This metric will be added to the list of selected metrics in the bottom panel – **Selected Attributes to Monitor panel**.

To remove an attribute from this panel, click the Delete icon.

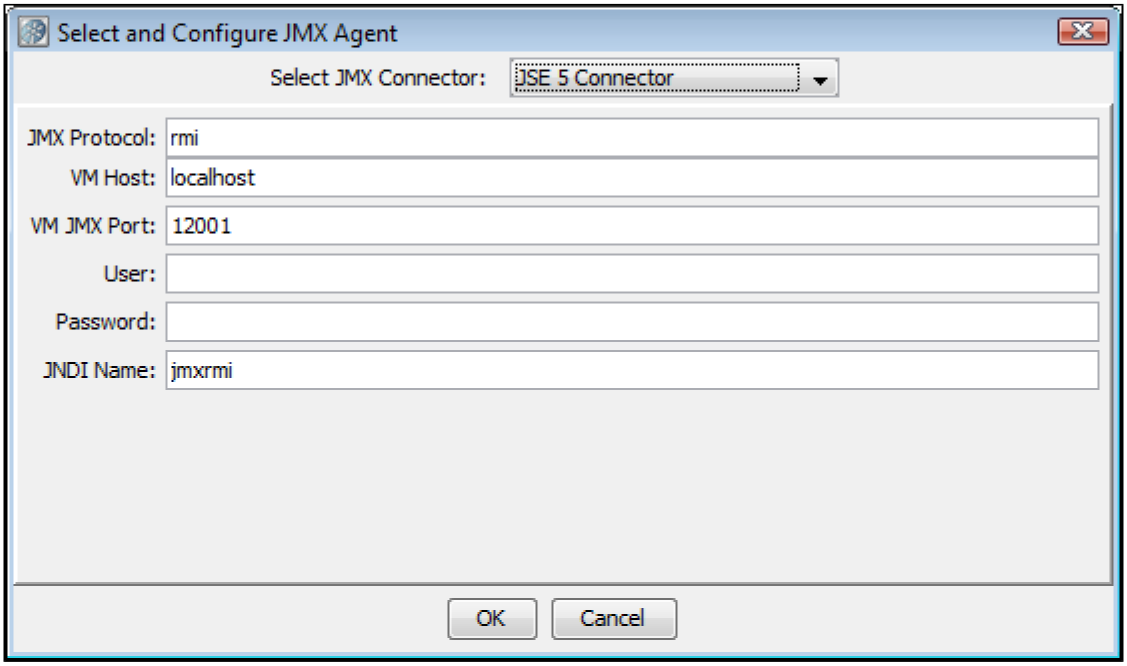
Repeat this process until all of your chosen metrics appear in your list (bottom panel).

6. Click OK to return to the main metrics panel.

Notice that the JMX metrics we added are now on our list of metrics.

Depending on the application server, vendor-specific JARs may be required to enable JMX communication with that application server. Visit ITKO Forums for specific information.

7. Similarly, you can also select the **JSE 5 Connector** from the JMX Connector list, with the following settings:



Select JMX Connector: JSE 5 Connector

JMX Protocol: rmi

VM Host: localhost

VM JMX Port: 12001

User:

Password:

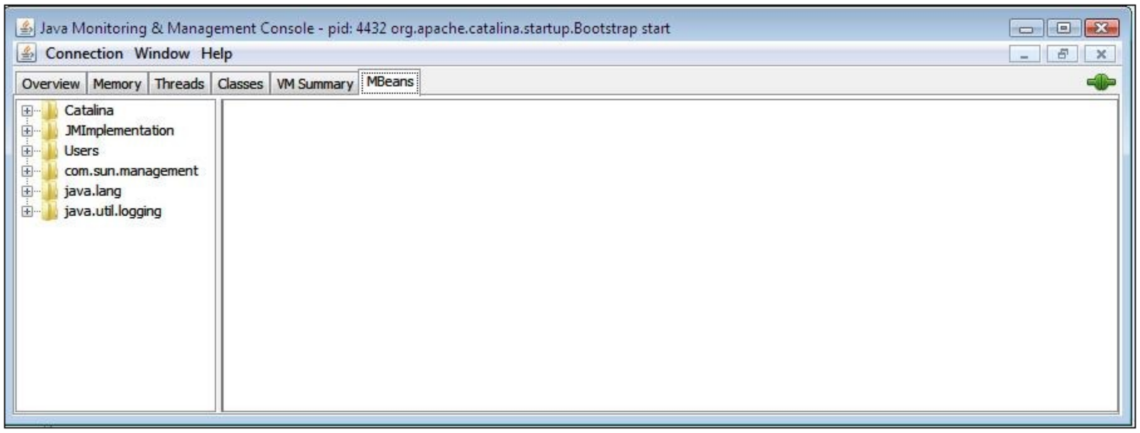
JNDI Name: jmxrmi

8. Click OK to connect to the JMX Agent.

Enabling JMX Metrics for Tomcat

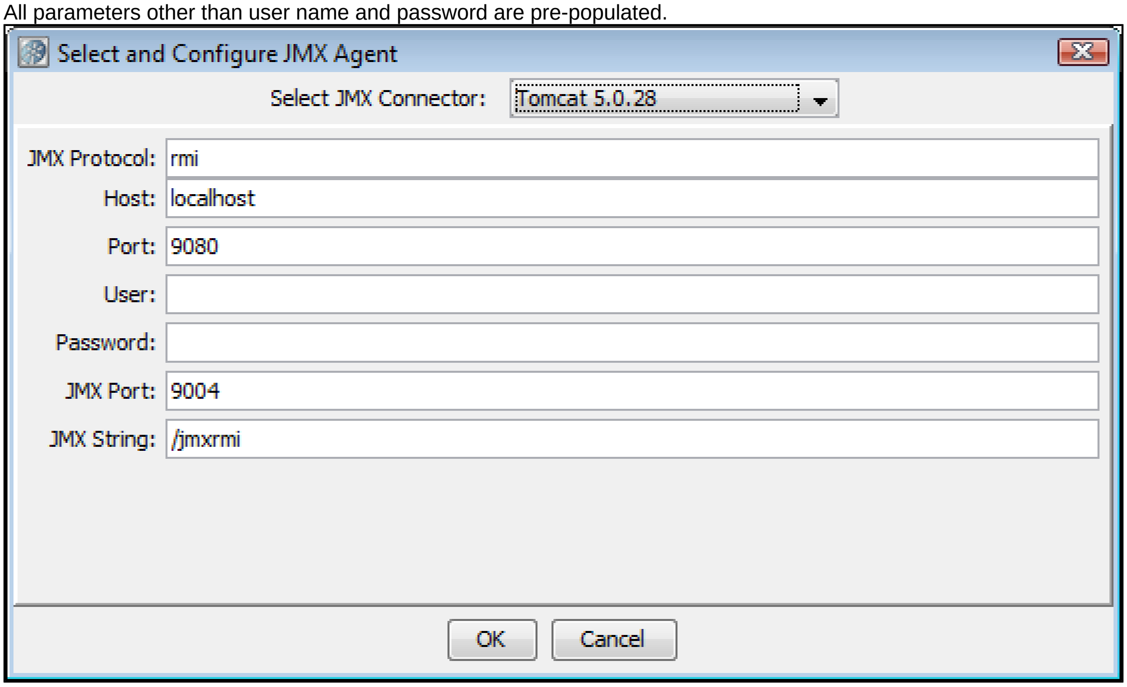
To enable JMX metrics for Tomcat

1. Modify the **catalina.bat** file and add **CATALINA_OPTS.** You can use the embedded Jakarta-Tomcat packaged with LISA.
2. Connect to Tomcat through JConsole.



3. Launch LISA, open a staging document, and go to the **Metrics** tab.
4. Click the Add icon and select **JMX Attribute Reader**.
5. Click OK.
6. Select **Tomcat 5.0.28**.

7. All parameters other than user name and password are pre-populated.

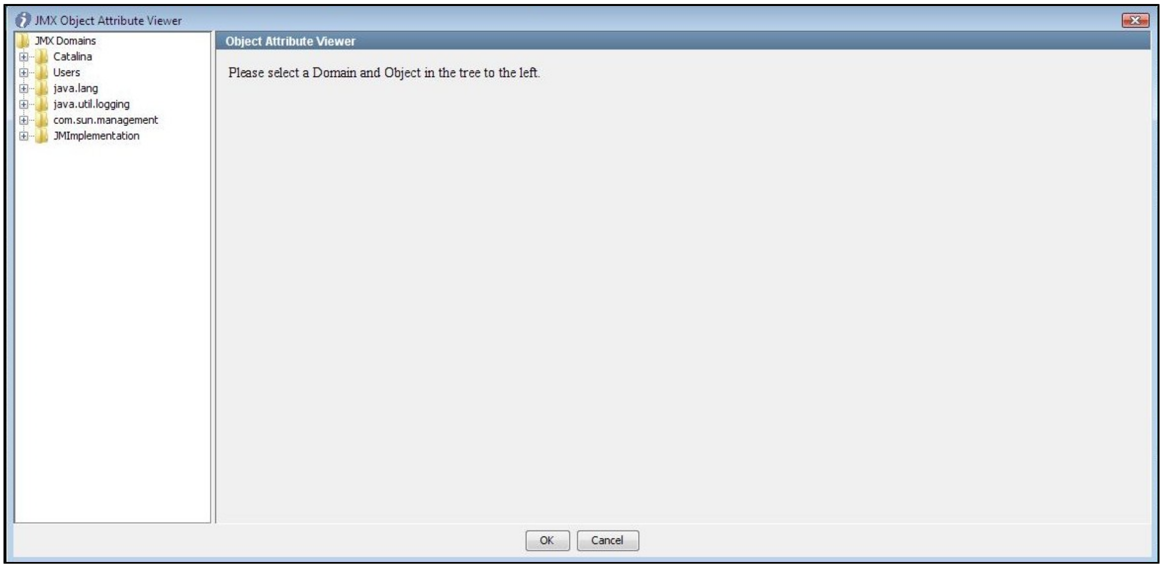


Select and Configure JMX Agent

Select JMX Connector: Tomcat 5.0.28

JMX Protocol: rmi
Host: localhost
Port: 9080
User:
Password:
JMX Port: 9004
JMX String: /jmxrmi

8. Click OK and connect to the JMX console.



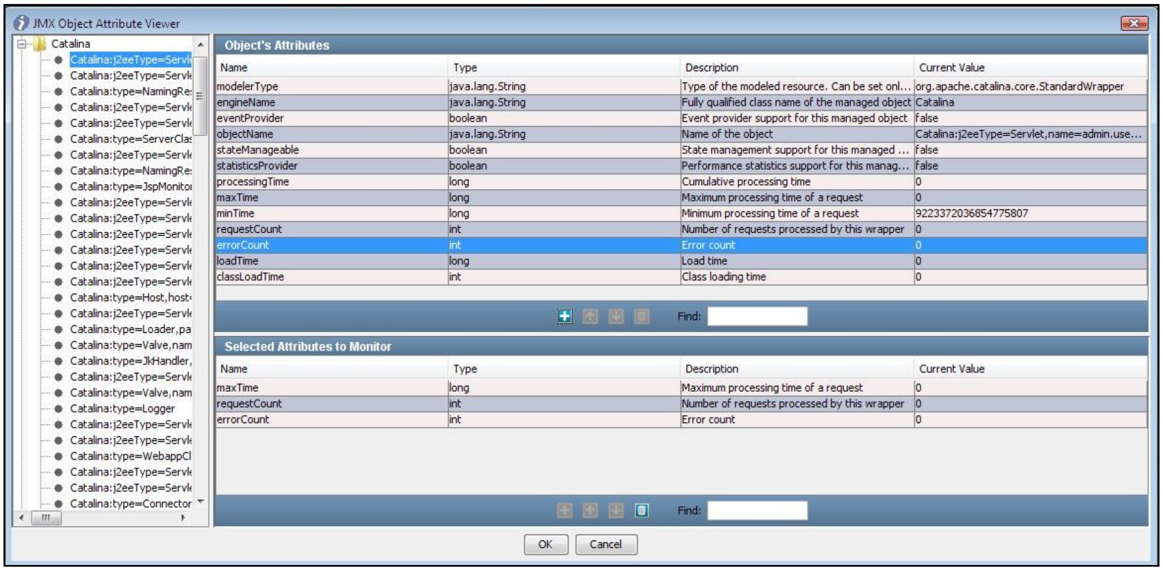
JMX Object Attribute Viewer

JMX Domains
- Catalina
- Users
- java.lang
- java.util.logging
- com.sun.management
- JMImplementation

Object Attribute Viewer

Please select a Domain and Object in the tree to the left.

9. Select a few objects specific to Catalina and add them in the list.



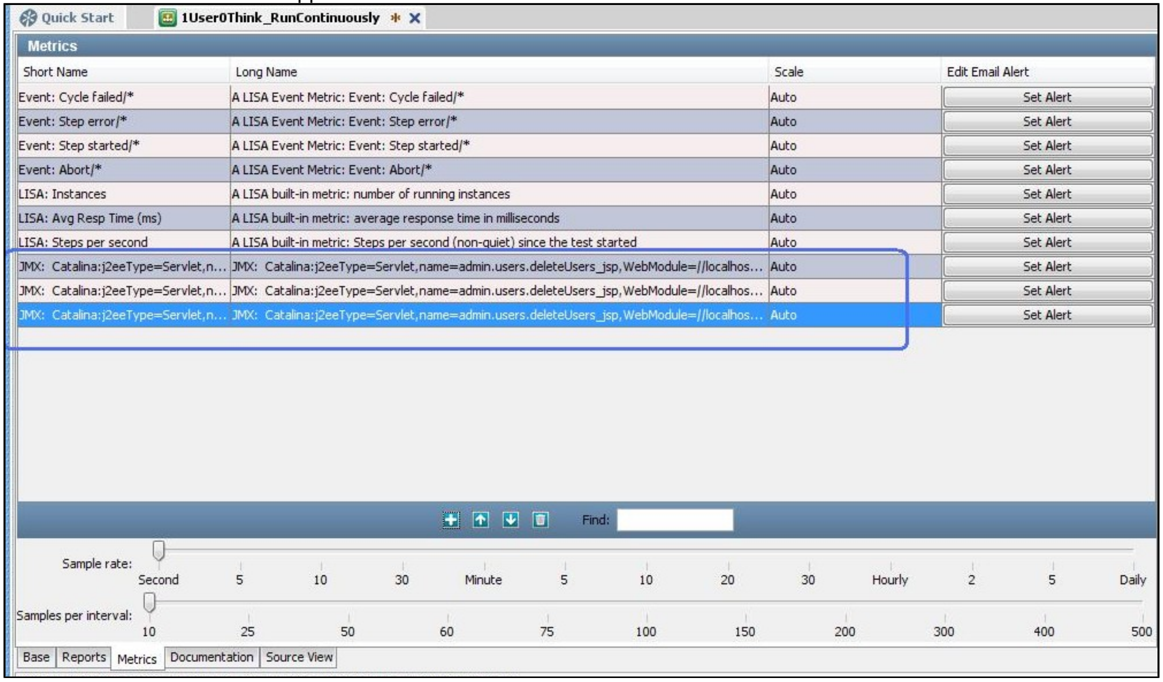
Object's Attributes

Name	Type	Description	Current Value
modelerType	java.lang.String	Type of the modeled resource. Can be set onl…	org.apache.catalina.core.StandardWrapper
engineName	java.lang.String	Fully qualified class name of the managed object	Catalina
eventProvider	boolean	Event provider support for this managed object	false
objectName	java.lang.String	Name of the object	Catalina:j2eeType=Servlet,name=admin.use…
stateManageable	boolean	State management support for this managed …	false
statisticsProvider	boolean	Performance statistics support for this manag…	false
processingTime	long	Cumulative processing time	0
maxTime	long	Maximum processing time of a request	0
minTime	long	Minimum processing time of a request	9223372036854775807
requestCount	int	Number of requests processed by this wrapper	0
errorCount	int	Error count	0
loadTime	long	Load time	0
classLoadTime	int	Class loading time	0

Selected Attributes to Monitor

Name	Type	Description	Current Value
maxTime	long	Maximum processing time of a request	0
requestCount	int	Number of requests processed by this wrapper	0
errorCount	int	Error count	0

10. Click OK and these attributes appear in the metrics list.



Metrics

Short Name	Long Name	Scale	Edit Email Alert
Event: Cycle failed/*	A LISA Event Metric: Event: Cycle failed/*	Auto	Set Alert
Event: Step error/*	A LISA Event Metric: Event: Step error/*	Auto	Set Alert
Event: Step started/*	A LISA Event Metric: Event: Step started/*	Auto	Set Alert
Event: Abort/*	A LISA Event Metric: Event: Abort/*	Auto	Set Alert
LISA: Instances	A LISA built-in metric: number of running instances	Auto	Set Alert
LISA: Avg Resp Time (ms)	A LISA built-in metric: average response time in milliseconds	Auto	Set Alert
LISA: Steps per second	A LISA built-in metric: Steps per second (non-quiet) since the test started	Auto	Set Alert
JMX: Catalina:j2eeType=Servlet,n…	JMX: Catalina:j2eeType=Servlet,name=admin.users.deleteUsers_jsp,WebModule=//localhos…	Auto	Set Alert
JMX: Catalina:j2eeType=Servlet,n…	JMX: Catalina:j2eeType=Servlet,name=admin.users.deleteUsers_jsp,WebModule=//localhos…	Auto	Set Alert
JMX: Catalina:j2eeType=Servlet,n…	JMX: Catalina:j2eeType=Servlet,name=admin.users.deleteUsers_jsp,WebModule=//localhos…	Auto	Set Alert

TIBCO Hawk Metrics

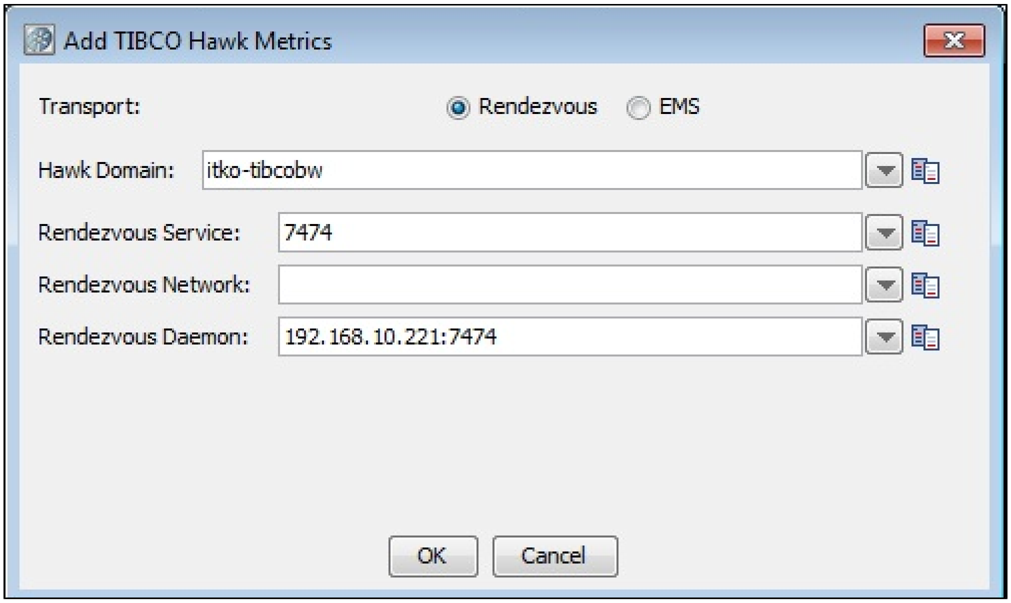
TIBCO Hawk is a tool for monitoring and managing distributed applications and operating systems. Unlike other monitoring solutions, TIBCO Hawk software uses TIBCO Messaging software for communication and inherits many of its benefits. These benefits include a flexible architecture, enterprise-wide scalability, and location-transparent product components that are simple to configure.

LISA has out-of-the-box integration with TIBCO Hawk for monitoring distributed applications and operating systems metrics within the context of testing. TIBCO Hawk provides in-container metrics for TIBCO BusinessWorks process archives. By using TIBCO Hawk, LISA can monitor metrics of all activities within any TIBCO BusinessWorks process that is deployed. This facilitates peering inside of a process to understand where bottlenecks are occurring.

Prerequisites:

- TIBCO Hawk jar files need to be copied to **<LISA_INSTALL_HOME>/lib.**
- TIBCO Rendezvous and/or TIBCO EMS JAR files, depending on which transport is being used by TIBCO Hawk, need to be copied to **<LISA_INSTALL_HOME>/lib**.

1. Create a staging document.
2. On the **Metrics** tab of the staging document, click the Add icon and select **TIBCO Hawk** from the drop-down list.



Transport: ● Rendezvous ○ EMS

Hawk Domain: itko-tibcobw

Rendezvous Service: 7474

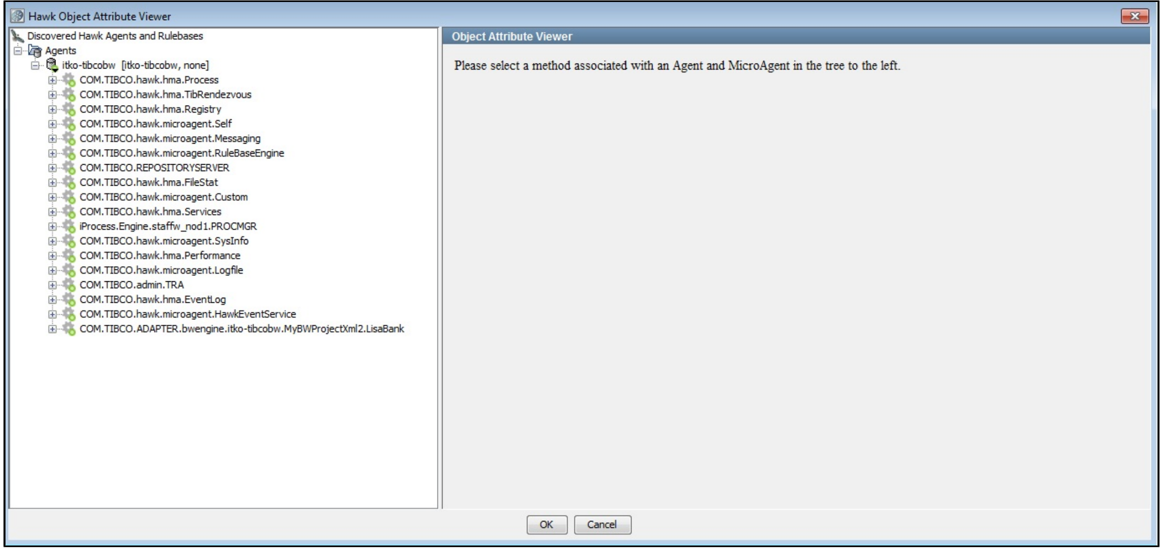
Rendezvous Network:

Rendezvous Daemon: 192.168.10.221:7474

The following parameters are available:

- **Transport**: Select **Rendezvous** or **EMS**.
- **Hawk Domain**
- **Rendezvous Service**
- **Rendezvous Network**
- **Rendezvous Daemon**

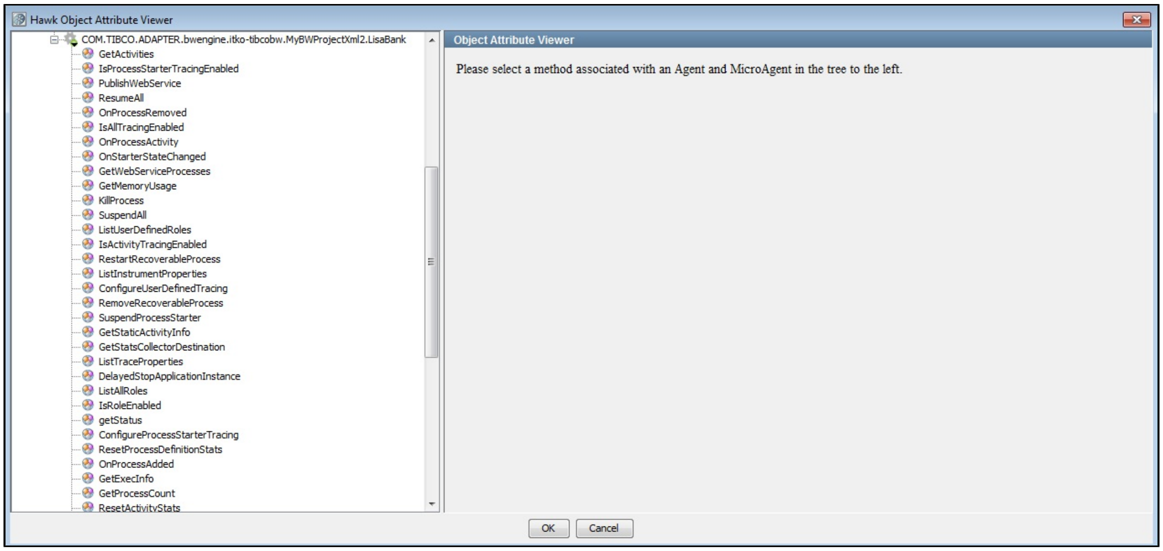
1. Click OK to display the Hawk Object Attribute Viewer.



Discovered Hawk Agents and Rulebases
Agents
itko-tibcobw [itko-tibcobw, none]
COM.TIBCO.hawk.hma.Process
COM.TIBCO.hawk.hma.TibRendezvous
COM.TIBCO.hawk.hma.Registry
COM.TIBCO.hawk.microagent.Self
COM.TIBCO.hawk.microagent.Messaging
COM.TIBCO.hawk.microagent.RuleBaseEngine
COM.TIBCO.REPOSITORYSERVER
COM.TIBCO.hawk.hma.FileStat
COM.TIBCO.hawk.microagent.Custom
COM.TIBCO.hawk.hma.Services
iProcess.Engine.staffw_nod1.PROCMGR
COM.TIBCO.hawk.microagent.SysInfo
COM.TIBCO.hawk.hma.Performance
COM.TIBCO.hawk.microagent.Logfile
COM.TIBCO.admin.TRA
COM.TIBCO.hawk.hma.EventLog
COM.TIBCO.hawk.microagent.HawkEventService
COM.TIBCO.ADAPTER.bwengine.itko-tibcobw.MyBWProjectXml2.LisaBank

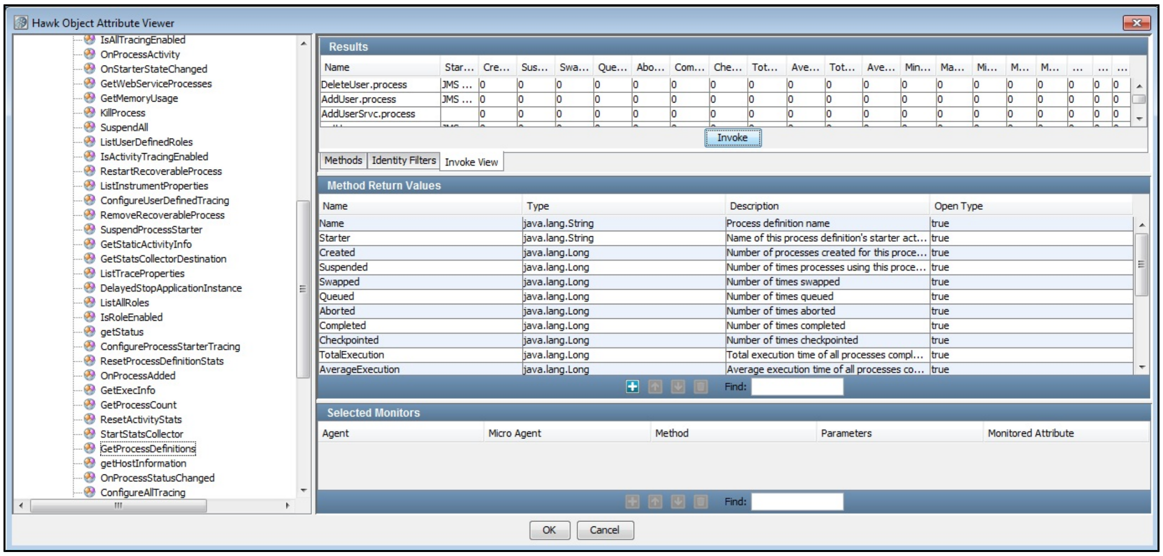
Please select a method associated with an Agent and MicroAgent in the tree to the left.

2. Select the Process Archive and expand beneath it.



Please select a method associated with an Agent and MicroAgent in the tree to the left.

3. **GetProcessDefinitions** will retrieve the Process Definitions defined in the Process Archive.

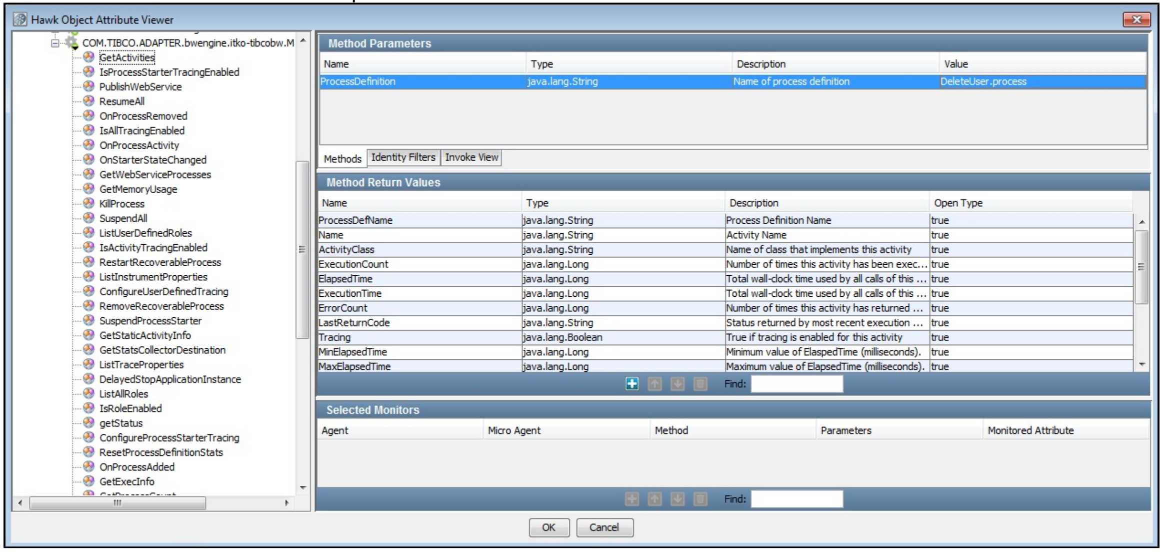


Name	Star…	Cre…	Sus…	Swa…	Que…	Abo…	Com…	Che…	Tot…	Ave…	Tot…	Ave…	Min…	Ma…	Mi…	M…	M…	…	…	…
DeleteUser.process	JMS …	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AddUser.process	JMS …	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AddUserSrvc.process		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Name	Type	Description	Open Type
Name	java.lang.String	Process definition name	true
Starter	java.lang.String	Name of this process definition's starter act…	true
Created	java.lang.Long	Number of processes created for this proce…	true
Suspended	java.lang.Long	Number of times processes using this proce…	true
Swapped	java.lang.Long	Number of times swapped	true
Queued	java.lang.Long	Number of times queued	true
Aborted	java.lang.Long	Number of times aborted	true
Completed	java.lang.Long	Number of times completed	true
Checkpointed	java.lang.Long	Number of times checkpointed	true
TotalExecution	java.lang.Long	Total execution time of all processes compl…	true
AverageExecution	java.lang.Long	Average execution time of all processes co…	true

Agent	Micro Agent	Method	Parameters	Monitored Attribute

4. Use the Process Definition Name as parameter for **GetActivities**.

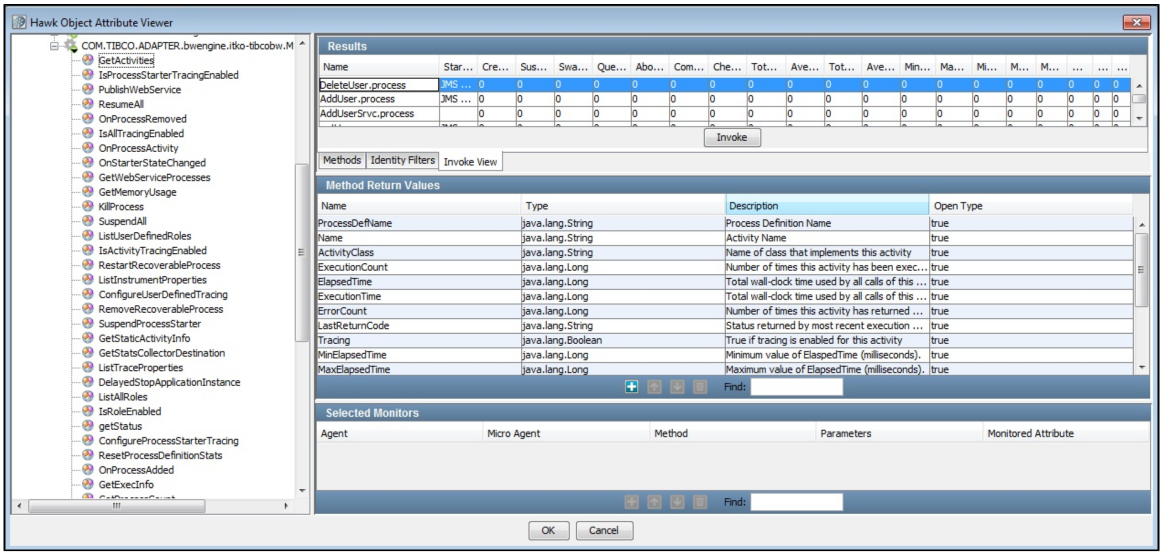


Name	Type	Description	Value
ProcessDefinition	java.lang.String	Name of process definition	DeleteUser.process

Name	Type	Description	Open Type
ProcessDefName	java.lang.String	Process Definition Name	true
Name	java.lang.String	Activity Name	true
ActivityClass	java.lang.String	Name of class that implements this activity	true
ExecutionCount	java.lang.Long	Number of times this activity has been exec…	true
ElapsedTime	java.lang.Long	Total wall-clock time used by all calls of this …	true
ExecutionTime	java.lang.Long	Total wall-clock time used by all calls of this …	true
ErrorCount	java.lang.Long	Number of times this activity has returned …	true
LastReturnCode	java.lang.String	Status returned by most recent execution …	true
Tracing	java.lang.Boolean	True if tracing is enabled for this activity	true
MinElapsedTime	java.lang.Long	Minimum value of ElaspedTime (milliseconds).	true
MaxElapsedTime	java.lang.Long	Maximum value of ElapsedTime (milliseconds).	true

Agent	Micro Agent	Method	Parameters	Monitored Attribute

5. TIBCO Hawk provides a number of metrics for Process Activities.



Results

Name	Star…	Cre…	Sus…	Swa…	Que…	Abo…	Com…	Che…	Tot…	Ave…	Tot…	Ave…	Min…	Ma…	Mi…	M…	M…	…	…	…
DeleteUser.process	JMS …	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AddUser.process	JMS …	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AddUserSrvc.process		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

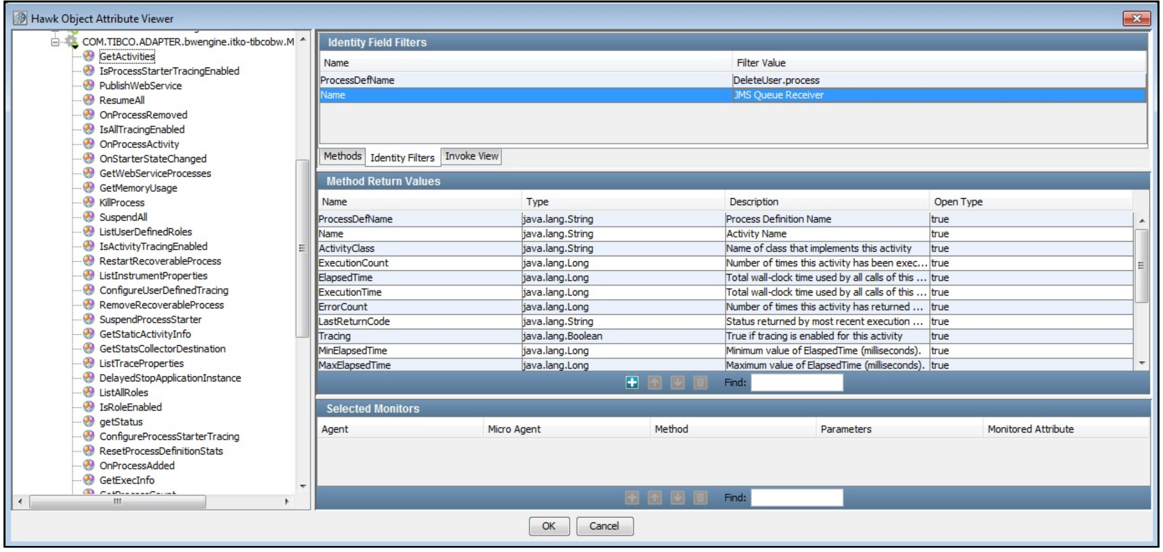
Method Return Values

Name	Type	Description	Open Type
ProcessDefName	java.lang.String	Process Definition Name	true
Name	java.lang.String	Activity Name	true
ActivityClass	java.lang.String	Name of class that implements this activity	true
ExecutionCount	java.lang.Long	Number of times this activity has been exec…	true
ElapsedTime	java.lang.Long	Total wall-clock time used by all calls of this …	true
ExecutionTime	java.lang.Long	Total wall-clock time used by all calls of this …	true
ErrorCount	java.lang.Long	Number of times this activity has returned …	true
LastReturnCode	java.lang.String	Status returned by most recent execution …	true
Tracing	java.lang.Boolean	True if tracing is enabled for this activity	true
MinElapsedTime	java.lang.Long	Minimum value of ElaspedTime (milliseconds).	true
MaxElapsedTime	java.lang.Long	Maximum value of ElapsedTime (milliseconds).	true

Selected Monitors

Agent	Micro Agent	Method	Parameters	Monitored Attribute

6. Provide a Filter with the Process Definition Name and the Activity Name.



Identity Field Filters

Name	Filter Value
ProcessDefName	DeleteUser.process
Name	JMS Queue Receiver

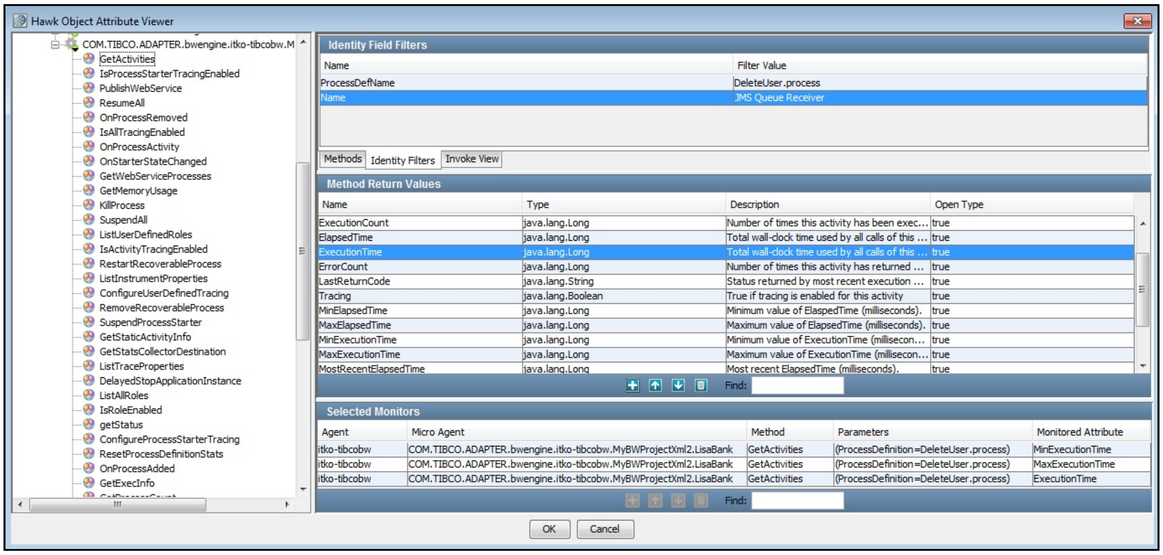
Method Return Values

Name	Type	Description	Open Type
ProcessDefName	java.lang.String	Process Definition Name	true
Name	java.lang.String	Activity Name	true
ActivityClass	java.lang.String	Name of class that implements this activity	true
ExecutionCount	java.lang.Long	Number of times this activity has been exec…	true
ElapsedTime	java.lang.Long	Total wall-clock time used by all calls of this …	true
ExecutionTime	java.lang.Long	Total wall-clock time used by all calls of this …	true
ErrorCount	java.lang.Long	Number of times this activity has returned …	true
LastReturnCode	java.lang.String	Status returned by most recent execution …	true
Tracing	java.lang.Boolean	True if tracing is enabled for this activity	true
MinElapsedTime	java.lang.Long	Minimum value of ElaspedTime (milliseconds).	true
MaxElapsedTime	java.lang.Long	Maximum value of ElapsedTime (milliseconds).	true

Selected Monitors

Agent	Micro Agent	Method	Parameters	Monitored Attribute

7. Select Method Return Values and click OK.



Identity Field Filters

Name	Filter Value
ProcessDefName	DeleteUser.process
Name	JMS Queue Receiver

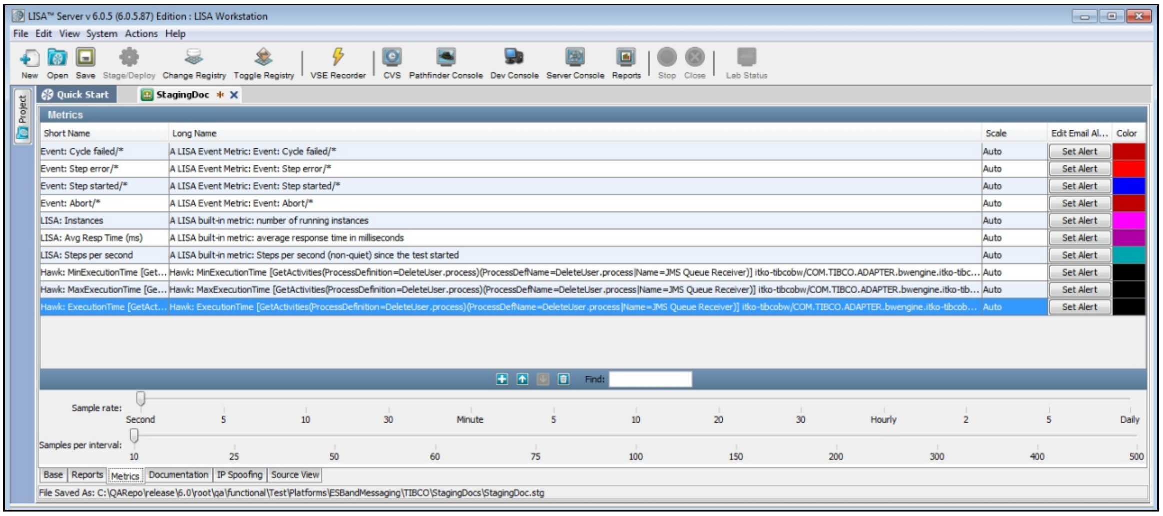
Method Return Values

Name	Type	Description	Open Type
ExecutionCount	java.lang.Long	Number of times this activity has been exec…	true
ElapsedTime	java.lang.Long	Total wall-clock time used by all calls of this …	true
ExecutionTime	java.lang.Long	Total wall-clock time used by all calls of this …	true
ErrorCount	java.lang.Long	Number of times this activity has returned …	true
LastReturnCode	java.lang.String	Status returned by most recent execution …	true
Tracing	java.lang.Boolean	True if tracing is enabled for this activity	true
MinElapsedTime	java.lang.Long	Minimum value of ElaspedTime (milliseconds).	true
MaxElapsedTime	java.lang.Long	Maximum value of ElapsedTime (milliseconds).	true
MinExecutionTime	java.lang.Long	Minimum value of ExecutionTime (millisecon…	true
MaxExecutionTime	java.lang.Long	Maximum value of ExecutionTime (millisecon…	true
MostRecentElapsedTime	java.lang.Long	Most recent ElapsedTime (milliseconds).	true

Selected Monitors

Agent	Micro Agent	Method	Parameters	Monitored Attribute
itko-tibcobw	COM.TIBCO.ADAPTER.bwengine.itko-tibcobw.MyBWProjectXml2.LisaBank	GetActivities	(ProcessDefinition=DeleteUser.process)	MinExecutionTime
itko-tibcobw	COM.TIBCO.ADAPTER.bwengine.itko-tibcobw.MyBWProjectXml2.LisaBank	GetActivities	(ProcessDefinition=DeleteUser.process)	MaxExecutionTime
itko-tibcobw	COM.TIBCO.ADAPTER.bwengine.itko-tibcobw.MyBWProjectXml2.LisaBank	GetActivities	(ProcessDefinition=DeleteUser.process)	ExecutionTime

8. Activity Metrics will now be monitored for test cases staged with this staging document.



Short Name	Long Name	Scale	Edit Email Al...	Color
Event: Cycle failed/*	A LISA Event Metric: Event: Cycle failed/*	Auto	Set Alert	🟥
Event: Step error/*	A LISA Event Metric: Event: Step error/*	Auto	Set Alert	🟥
Event: Step started/*	A LISA Event Metric: Event: Step started/*	Auto	Set Alert	🟦
Event: Abort/*	A LISA Event Metric: Event: Abort/*	Auto	Set Alert	🟥
LISA: Instances	A LISA built-in metric: number of running instances	Auto	Set Alert	🟪
LISA: Avg Resp Time (ms)	A LISA built-in metric: average response time in milliseconds	Auto	Set Alert	🟪
LISA: Steps per second	A LISA built-in metric: Steps per second (non-quiet) since the test started	Auto	Set Alert	🟦
Hawk: MinExecutionTime [Get...	Hawk: MinExecutionTime [GetActivities(ProcessDefinition=DeleteUser.process)(ProcessDefName=DeleteUser.process|Name=JMS Queue Receiver)] itko-tibcobw/COM.TIBCO.ADAPTER.bwengine.itko-tibc...	Auto	Set Alert	⬛
Hawk: MaxExecutionTime [Ge...	Hawk: MaxExecutionTime [GetActivities(ProcessDefinition=DeleteUser.process)(ProcessDefName=DeleteUser.process|Name=JMS Queue Receiver)] itko-tibcobw/COM.TIBCO.ADAPTER.bwengine.itko-tib...	Auto	Set Alert	⬛
Hawk: ExecutionTime [GetAct...	Hawk: ExecutionTime [GetActivities(ProcessDefinition=DeleteUser.process)(ProcessDefName=DeleteUser.process|Name=JMS Queue Receiver)] itko-tibcobw/COM.TIBCO.ADAPTER.bwengine.itko-tibcob...	Auto	Set Alert	⬛

File Saved As: C:\QARepo\release\6.0\root\qa\functional\Test\Platforms\ESBandMessaging\TIBCO\StagingDocs\StagingDoc.stg

TIBCO Hawk can be also be invoked within a test case, through TIBCO Hawk APIs.

Windows Perfmon Metrics

The Perfmon Metrics use Microsoft Windows Perfmon to provide metrics to monitor system performance on a Windows operating system. These metrics are similar to the SNMP metrics.

Setting up Perfmon

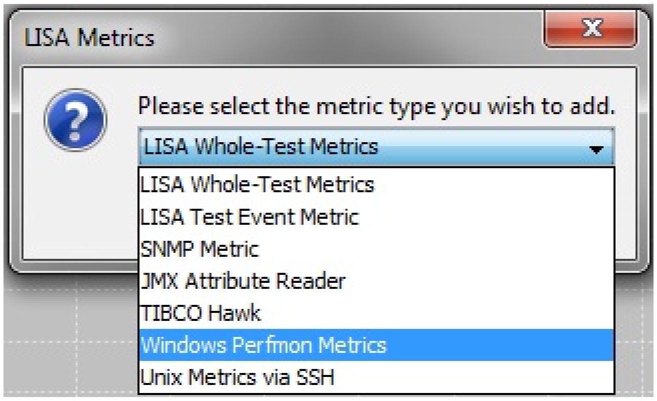
Before you can collect Perfmon metrics you must configure Perfmon on your Windows computer.

You must have Microsoft .NET framework 2.0 or higher installed and you need to run the **setup-wperfmon.bat** file located in the LISA_HOME/bin directory.

For more details on setting up Perfmon, see Installing Performance Monitor (Perfmon).

To add Perfmon metrics

1. Select **Windows Perfmon Metrics** from the LISA Metrics dialog and click OK.

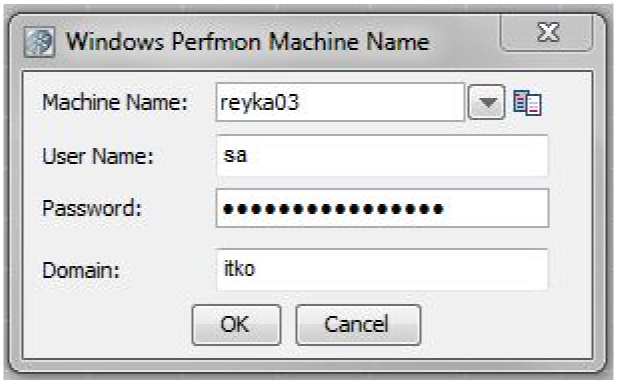


Please select the metric type you wish to add.

LISA Whole-Test Metrics

LISA Whole-Test Metrics
LISA Test Event Metric
SNMP Metric
JMX Attribute Reader
TIBCO Hawk
Windows Perfmon Metrics
Unix Metrics via SSH

2. The Windows Perfmon Machine Name dialog is displayed.



Windows Perfmon Machine Name

Machine Name: reyka03

User Name: sa

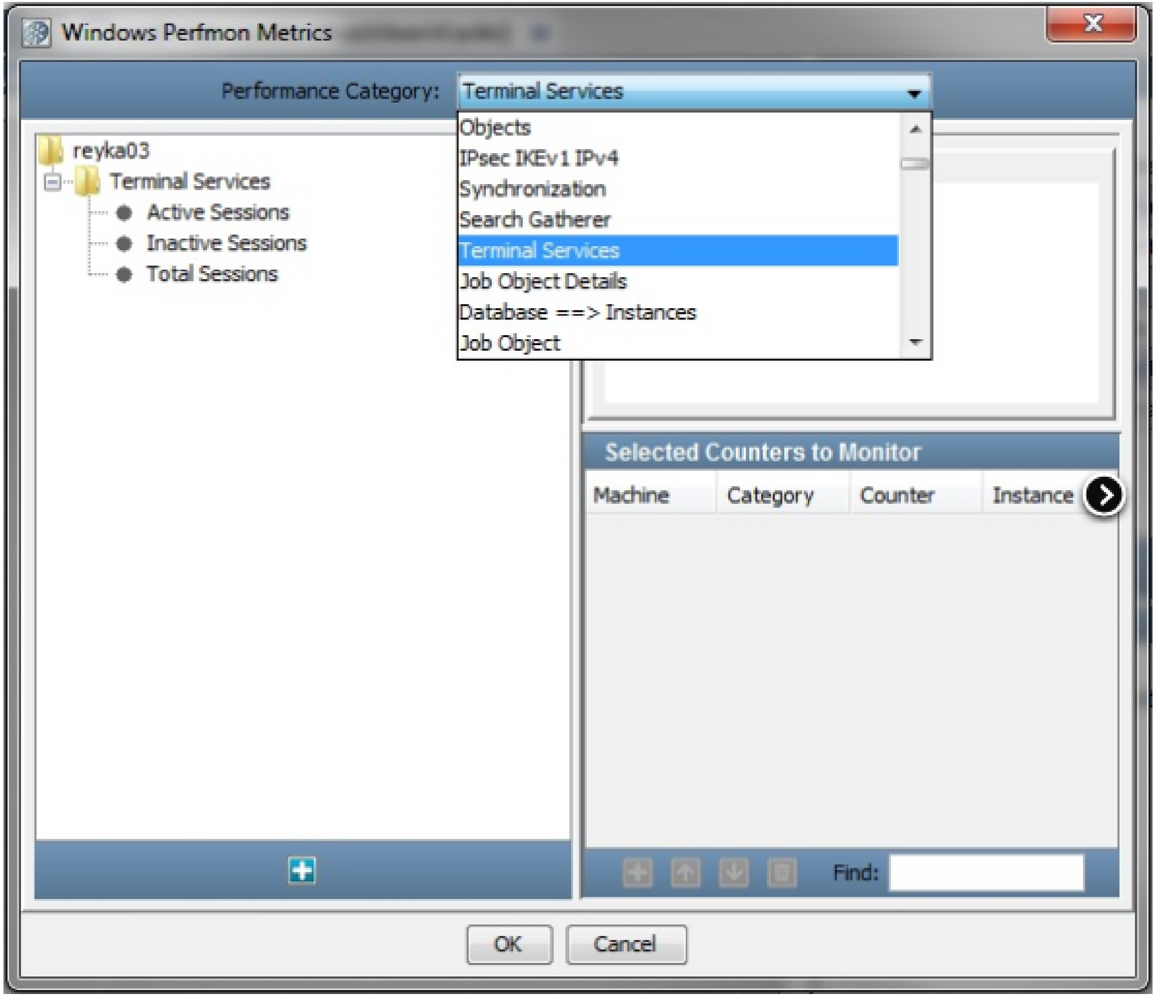
Password: ●●●●●●●●●●●●●●●●

Domain: itko

- **Machine Name**: Enter the **Machine Name** of your Windows installation. This can be found by right-clicking My Computer, selecting Properties, and going to the Computer Name tab.
- **User Name**: Enter the user name of the remote computer.
- **Password**: Enter the password of the remote computer.
- **Domain**: Enter the domain name. A domain is optional.

3. Click OK to proceed.

4. After you enter valid login credentials, a window appears showing all the available metric types. The Windows Perfmon Metrics screen is displayed.

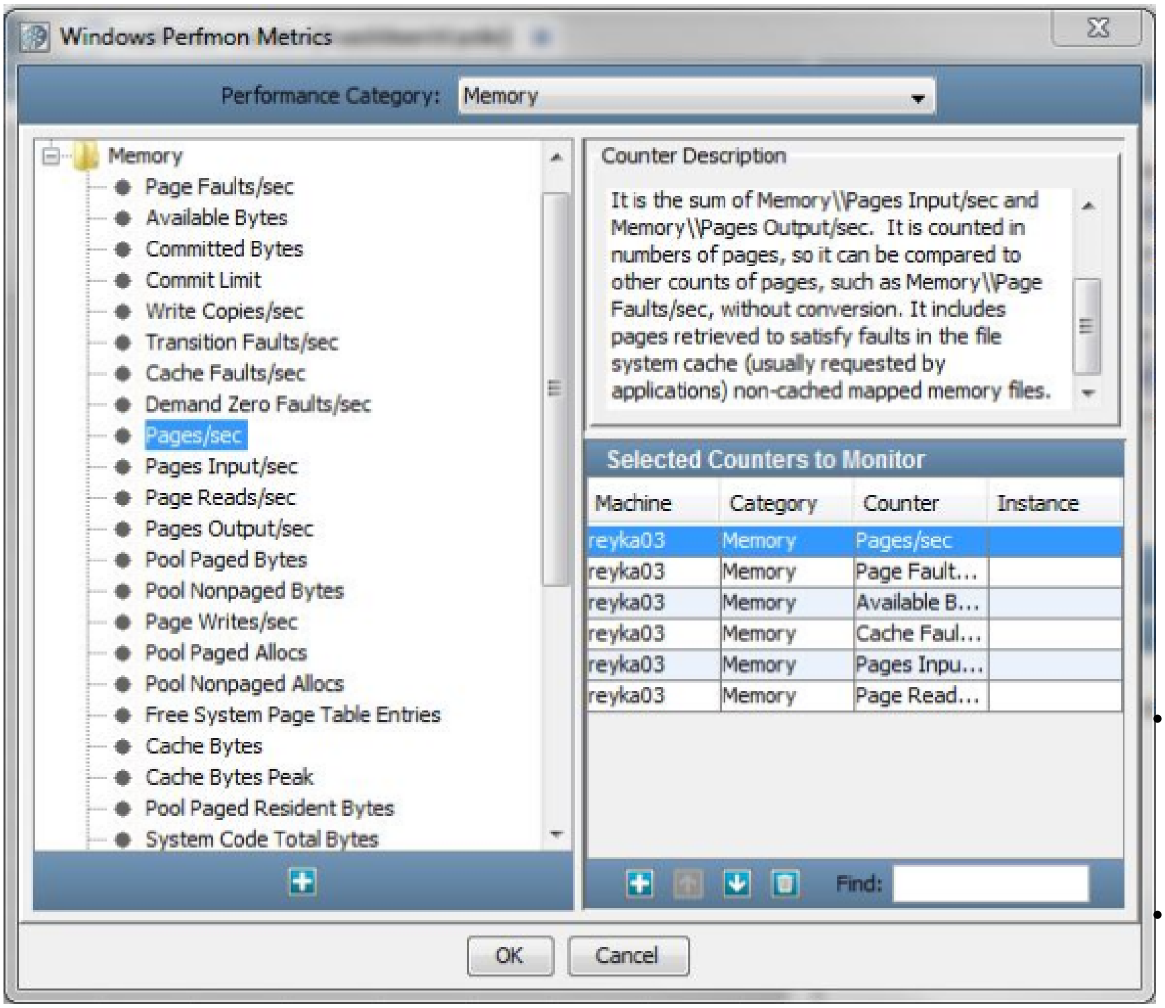


Windows Perfmon Metrics

Performance Category: Terminal Services

5. The Perfmon Metric window has three panels.

- The left panel displays the **Selected Performance** Category.
- The top right panel displays the description of the highlighted category in the **Counter Description** section.



Performance Category: Memory

Counter Description

It is the sum of Memory\\Pages Input/sec and Memory\\Pages Output/sec. It is counted in numbers of pages, so it can be compared to other counts of pages, such as Memory\\Page Faults/sec, without conversion. It includes pages retrieved to satisfy faults in the file system cache (usually requested by applications) non-cached mapped memory files.

Selected Counters to Monitor

Machine	Category	Counter	Instance
reyka03	Memory	Pages/sec	
reyka03	Memory	Page Fault...	
reyka03	Memory	Available B...	
reyka03	Memory	Cache Faul...	
reyka03	Memory	Pages Inpu...	
reyka03	Memory	Page Read...	

- The bottom right panel lists the metric that is highlighted in the left panel.

  To select a particular metric:

- **Performance Category**: Select a performance category from the pull-down menu.

  This lists various categories which can be monitored. To name a few:
  - .NET CLR Remoting/ LocksandThreads
  - .NET CLR Data/Networking
  - Job Objects/Job Object Details
  - Performance/RSVP Service
  - Memory/Print Queue
  - ICMP/Process
  - Outlook/Logical disk
- IP/Server/Cache

6. After you select the category, it will be added to the left panel.

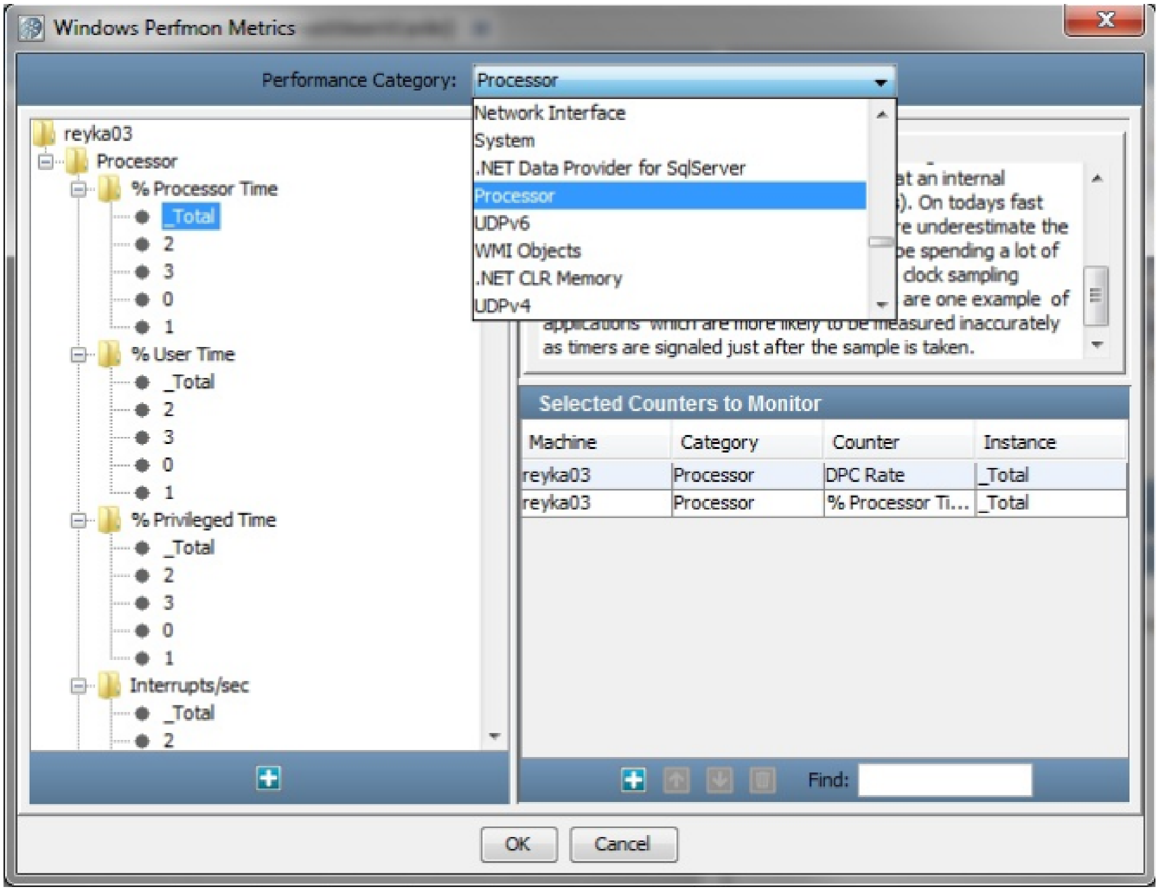
7. Double-click the target metric in the left panel to add it to the **Selected Counters to Monitor** table on the right panel.

8. Repeat until you have added all the target Perfmon metrics.

9. Click OK.

Example

We have added the following metric:



Windows Perfmon Metrics

Performance Category: Processor

- Network Interface
- System
- .NET Data Provider for SqlServer
- Processor
- UDPv6
- WMI Objects
- .NET CLR Memory
- UDPv4

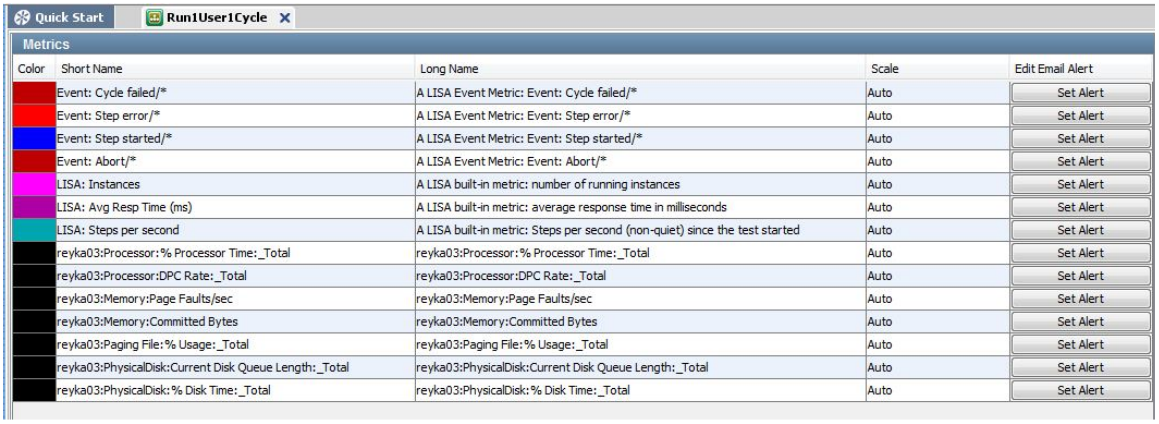
```
reyka03
  Processor
    % Processor Time
      _Total
      2
      3
      0
      1
    % User Time
      _Total
      2
      3
      0
      1
    % Privileged Time
      _Total
      2
      3
      0
      1
    Interrupts/sec
      _Total
      2
```

at an internal). On todays fast e underestimate the be spending a lot of clock sampling are one example of applications which are more likely to be measured inaccurately as timers are signaled just after the sample is taken.

Selected Counters to Monitor

Machine	Category	Counter	Instance
reyka03	Processor	DPC Rate	_Total
reyka03	Processor	% Processor Ti...	_Total

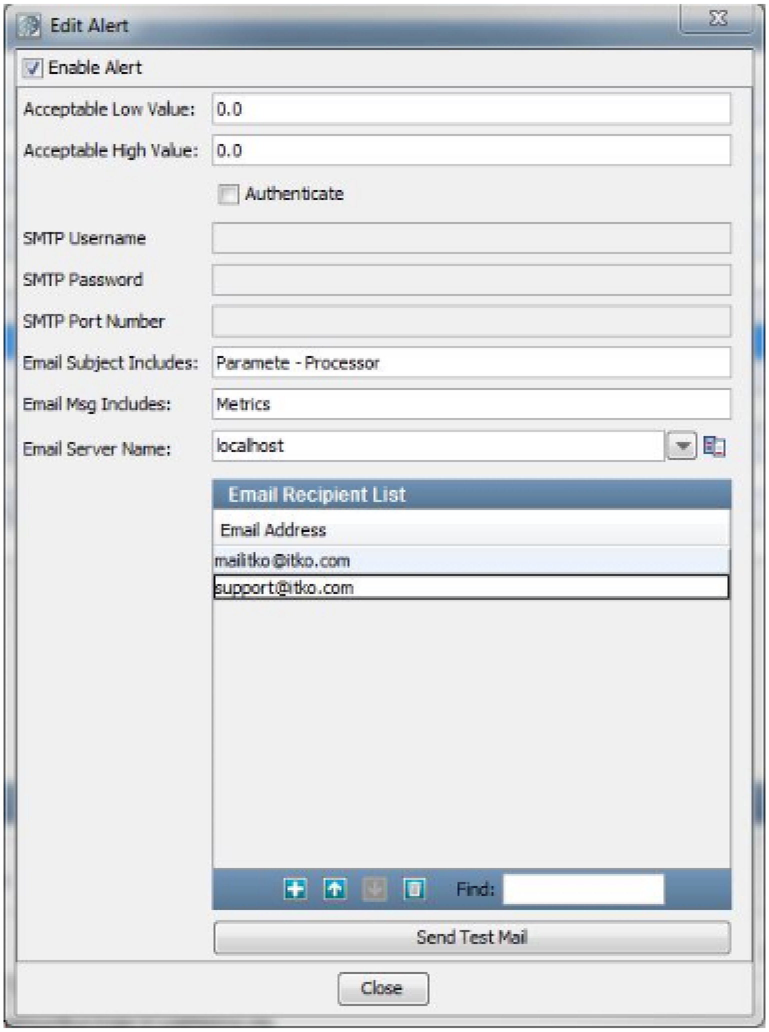
Click OK to add these to the Metric list on the Metric tab of the staging document.



Metrics

Color	Short Name	Long Name	Scale	Edit Email Alert
🟥	Event: Cycle failed/*	A LISA Event Metric: Event: Cycle failed/*	Auto	Set Alert
🟥	Event: Step error/*	A LISA Event Metric: Event: Step error/*	Auto	Set Alert
🟦	Event: Step started/*	A LISA Event Metric: Event: Step started/*	Auto	Set Alert
🟥	Event: Abort/*	A LISA Event Metric: Event: Abort/*	Auto	Set Alert
🟪	LISA: Instances	A LISA built-in metric: number of running instances	Auto	Set Alert
🟪	LISA: Avg Resp Time (ms)	A LISA built-in metric: average response time in milliseconds	Auto	Set Alert
🟦	LISA: Steps per second	A LISA built-in metric: Steps per second (non-quiet) since the test started	Auto	Set Alert
⬛	reyka03:Processor:% Processor Time:_Total	reyka03:Processor:% Processor Time:_Total	Auto	Set Alert
⬛	reyka03:Processor:DPC Rate:_Total	reyka03:Processor:DPC Rate:_Total	Auto	Set Alert
⬛	reyka03:Memory:Page Faults/sec	reyka03:Memory:Page Faults/sec	Auto	Set Alert
⬛	reyka03:Memory:Committed Bytes	reyka03:Memory:Committed Bytes	Auto	Set Alert
⬛	reyka03:Paging File:% Usage:_Total	reyka03:Paging File:% Usage:_Total	Auto	Set Alert
⬛	reyka03:PhysicalDisk:Current Disk Queue Length:_Total	reyka03:PhysicalDisk:Current Disk Queue Length:_Total	Auto	Set Alert
⬛	reyka03:PhysicalDisk:% Disk Time:_Total	reyka03:PhysicalDisk:% Disk Time:_Total	Auto	Set Alert

You can also set an email alert for this metric by clicking on the Set Alert button.



Edit Alert

☑ Enable Alert

Acceptable Low Value: 0.0

Acceptable High Value: 0.0

□ Authenticate

SMTP Username

SMTP Password

SMTP Port Number

Email Subject Includes: Paramete - Processor

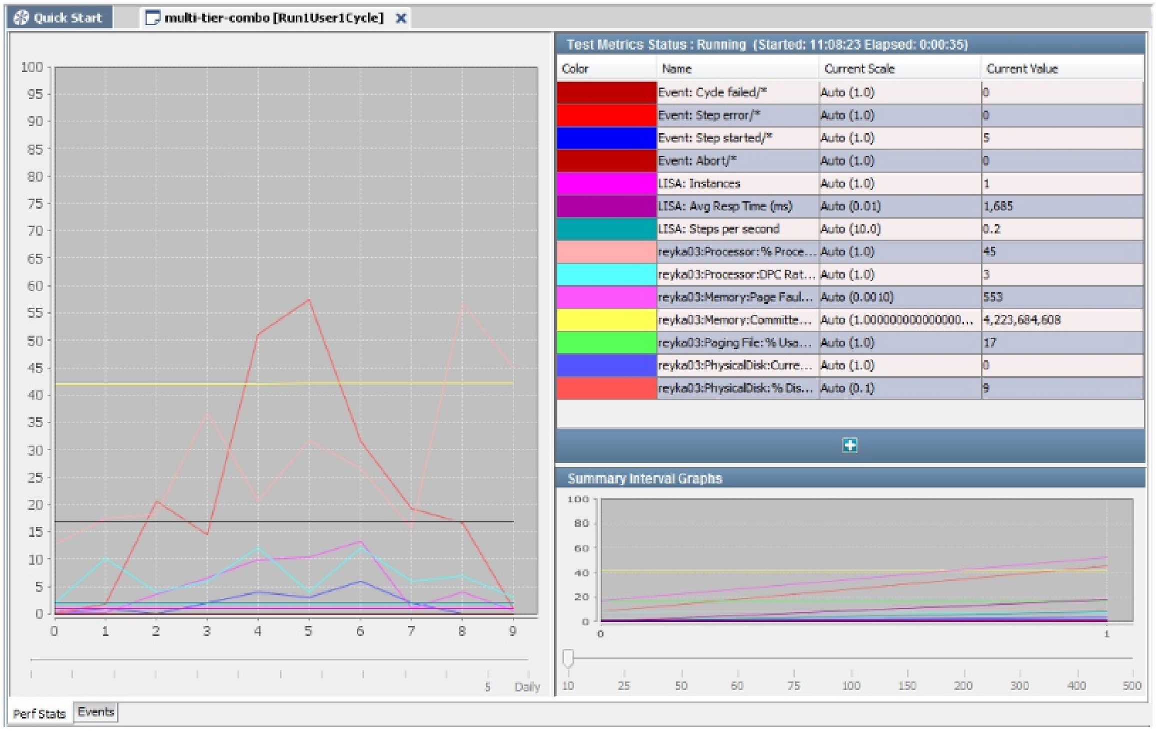
Email Msg Includes: Metrics

Email Server Name: localhost

Email Recipient List

Email Address
mailitko@itko.com
support@itko.com

The following graph shows the Perfmon metric being collected while a test is executing.



Test Metrics Status : Running (Started: 11:08:23 Elapsed: 0:00:35)

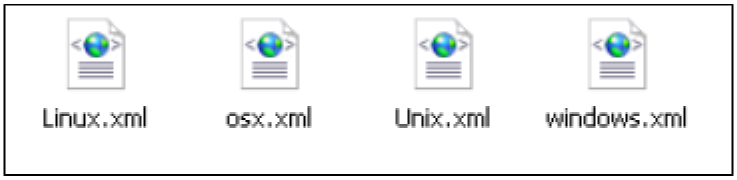
Color	Name	Current Scale	Current Value
🟥	Event: Cycle failed/*	Auto (1.0)	0
🟥	Event: Step error/*	Auto (1.0)	0
🟦	Event: Step started/*	Auto (1.0)	5
🟥	Event: Abort/*	Auto (1.0)	0
🟪	LISA: Instances	Auto (1.0)	1
🟪	LISA: Avg Resp Time (ms)	Auto (0.01)	1,685
🟦	LISA: Steps per second	Auto (10.0)	0.2
🟥	reyka03:Processor:% Proce…	Auto (1.0)	45
🟦	reyka03:Processor:DPC Rat…	Auto (1.0)	3
🟪	reyka03:Memory:Page Faul…	Auto (0.0010)	553
🟨	reyka03:Memory:Committe…	Auto (1.000000000000000…	4,223,684,608
🟩	reyka03:Paging File:% Usa…	Auto (1.0)	17
🟦	reyka03:PhysicalDisk:Curre…	Auto (1.0)	0
🟥	reyka03:PhysicalDisk:% Dis…	Auto (0.1)	9

UNIX Metrics Via SSH

This metric is used to collect command line metrics. This metric will gather inputs like authentication details, host name and the selection of metrics that need to be collected.

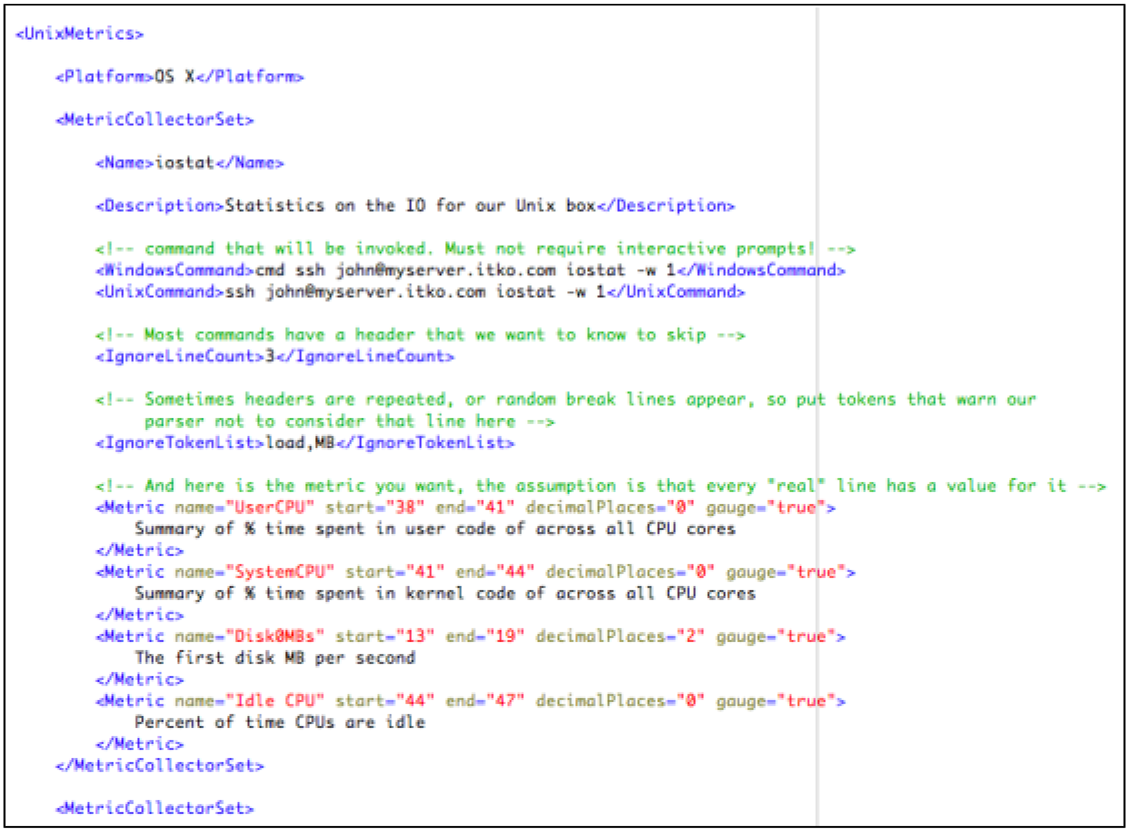
The metric data is stored in an XML file that is located in the **LISA_HOME/umetrics** directory by default, but you can define a new location by updating the key **stats.unix.xml.folder** in **local.properties**.

Each file in that directory has a unique file name based on operating system.



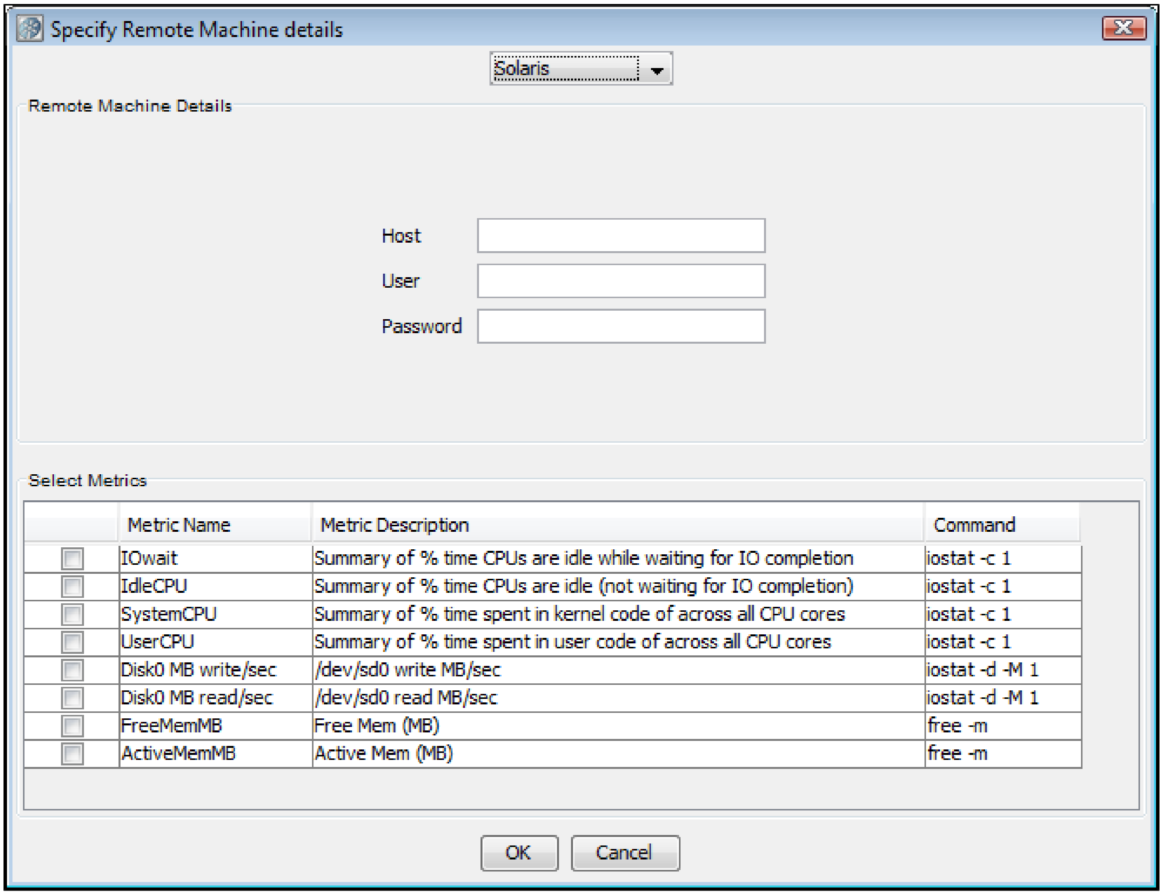
The XML file is used to feed the information about the command and metrics that needs to be collected on a particular operating system. For example, to make the command and metrics are known on a Linux operating system, a linux.xml file should be in the LISA_HOME/umetrics folder.

Following is an example of an OSX command parser for **iostat** that collects CPU and disk0 metrics.



```
<UnixMetrics>

    <Platform>OS X</Platform>

    <MetricCollectorSet>

        <Name>iostat</Name>

        <Description>Statistics on the IO for our Unix box</Description>

        <!-- command that will be invoked. Must not require interactive prompts! -->
        <WindowsCommand>cmd ssh john@myserver.itko.com iostat -w 1</WindowsCommand>
        <UnixCommand>ssh john@myserver.itko.com iostat -w 1</UnixCommand>

        <!-- Most commands have a header that we want to know to skip -->
        <IgnoreLineCount>3</IgnoreLineCount>

        <!-- Sometimes headers are repeated, or random break lines appear, so put tokens that warn our
             parser not to consider that line here -->
        <IgnoreTokenList>load,MB</IgnoreTokenList>

        <!-- And here is the metric you want, the assumption is that every "real" line has a value for it -->
        <Metric name="UserCPU" start="38" end="41" decimalPlaces="0" gauge="true">
            Summary of % time spent in user code of across all CPU cores
        </Metric>
        <Metric name="SystemCPU" start="41" end="44" decimalPlaces="0" gauge="true">
            Summary of % time spent in kernel code of across all CPU cores
        </Metric>
        <Metric name="Disk0MBs" start="13" end="19" decimalPlaces="2" gauge="true">
            The first disk MB per second
        </Metric>
        <Metric name="Idle CPU" start="44" end="47" decimalPlaces="0" gauge="true">
            Percent of time CPUs are idle
        </Metric>
    </MetricCollectorSet>

    <MetricCollectorSet>
```

In the **Specify Remote Machine details** screen, enter the operating system: Linux, OS X, or Solaris.



Host

User

Password

	Metric Name	Metric Description	Command
☐	IOwait	Summary of % time CPUs are idle while waiting for IO completion	iostat -c 1
☐	IdleCPU	Summary of % time CPUs are idle (not waiting for IO completion)	iostat -c 1
☐	SystemCPU	Summary of % time spent in kernel code of across all CPU cores	iostat -c 1
☐	UserCPU	Summary of % time spent in user code of across all CPU cores	iostat -c 1
☐	Disk0 MB write/sec	/dev/sd0 write MB/sec	iostat -d -M 1
☐	Disk0 MB read/sec	/dev/sd0 read MB/sec	iostat -d -M 1
☐	FreeMemMB	Free Mem (MB)	free -m
☐	ActiveMemMB	Active Mem (MB)	free -m

Enter the remote machine details:

- **Host**
- **User**
- **Password**

Each operating system has a slightly different set of metrics from which to choose. Use the check box in the left column to select the metrics you would like to collect.

Building Test Suites

You can run/stage a combination of test cases together in LISA. The collection of test cases run together are in a form of a test suite.

The following topics are available:

Creating a Test Suite
Test Suite Editor

Creating a Test Suite

You create test suites from within LISA Workstation.

To create a test suite

1. From the main menu, select File > New > Suite.
   The suite document editor appears.
2. In the Base tab, specify basic information about the suite. This information includes the suite name and the suite entries.
3. In the Reports tab, specify the type of report that you want to create at runtime.
4. In the Metrics tab, specify the metrics that you want to record at runtime.
5. In the Documentation tab, enter descriptive information about the suite.
6. From the main menu, select File > Save.

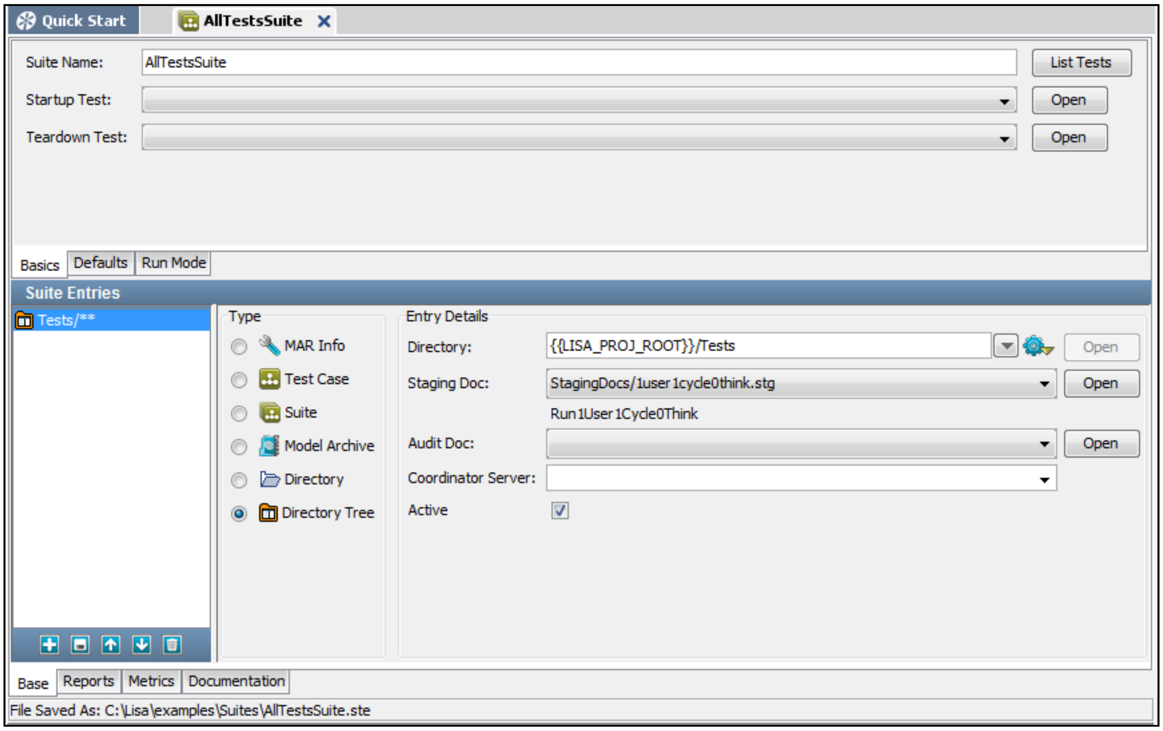
Test Suite Editor

When you create a new test suite, or open an existing test suite, the Test Suite Editor is displayed.

In this section, the following topics are covered:

Test Suite Editor - Base Tab

The Base tab of the Suite Document Editor contains basic information about a test suite.



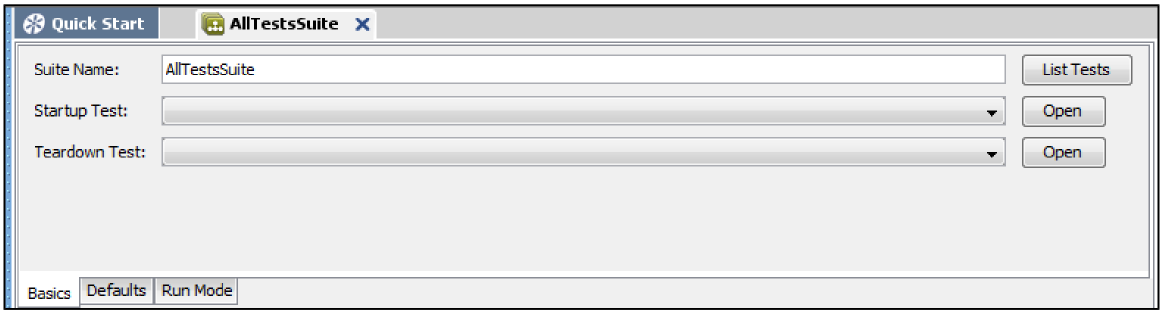
Suite Name: AllTestsSuite
Startup Test:
Teardown Test:
Tests/**
Type: ● Directory Tree
Directory: {{LISA_PROJ_ROOT}}/Tests
Staging Doc: StagingDocs/1user1cycle0think.stg
Run1User1Cycle0Think
Audit Doc:
Coordinator Server:
Active ☑
File Saved As: C:\Lisa\examples\Suites\AllTestsSuite.ste

Top Panel

The top panel of the Base tab has basic information about the test suite as a whole.

To configure a suite, you also need to enter parameters in all the sub-tabs.

Basics Tab



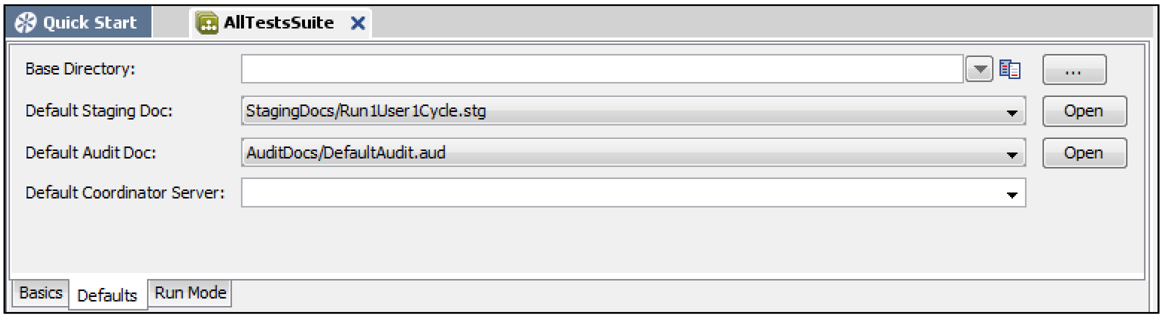
Suite Name: AllTestsSuite
Startup Test:
Teardown Test:

This tab has the following parameters:

- **Suite Name**: The name of the suite.
- **Startup Test/Teardown Test**: You can specify a startup test that will run before the suite has begun, and a teardown test that will run after the suite has completed. Neither the startup test nor the teardown test is included in any test statistics. Events in these tests will appear in the reports. Test suite testing will not continue if the startup test fails. As of release 6.0.5, you can use a MAR info file as a startup or teardown test. The primary asset of the MAR info file must be a test case.
- **List Tests**: Displays a dialog window with a list of the tests currently included in your suite. You must save your suite for this list to be

current.

Defaults Tab



Base Directory:

Default Staging Doc: StagingDocs/Run1User1Cycle.stg

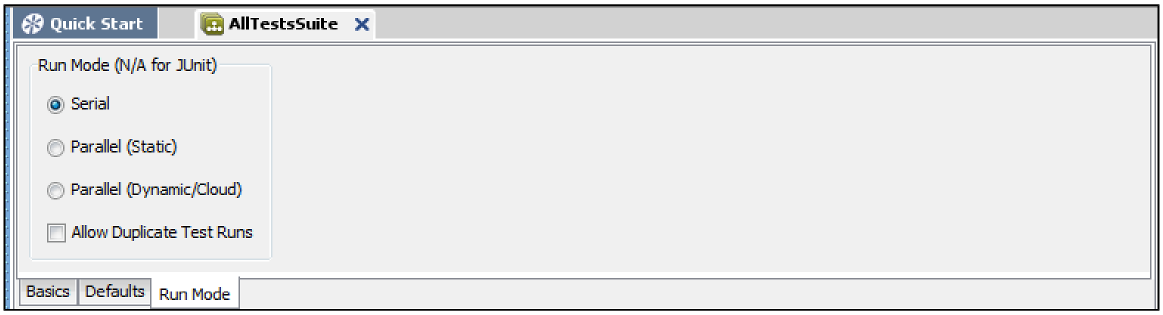
Default Audit Doc: AuditDocs/DefaultAudit.aud

Default Coordinator Server:

This tab has the following parameters:

- **Base Directory**: The name of the directory that will be assumed to be the base directory for any individual test that does not contain a complete path, or a test cannot be found elsewhere it will look in this directory. If you do not specify a base directory, a default will be created for you when the suite document is saved.
- **Default Staging Doc**: The staging document to use if one is not specified for an individual test.
- **Default Audit Doc**: The audit document to use if one is not specified for an individual test.
- **Default Coordinator Server**: The coordinator server to use if one is not specified for an individual test. The pull-down menu lists the available coordinator servers. This parameter is needed only if you are running LISA Server.

Run Mode Tab



Run Mode (N/A for JUnit)

● Serial

○ Parallel (Static)

○ Parallel (Dynamic/Cloud)

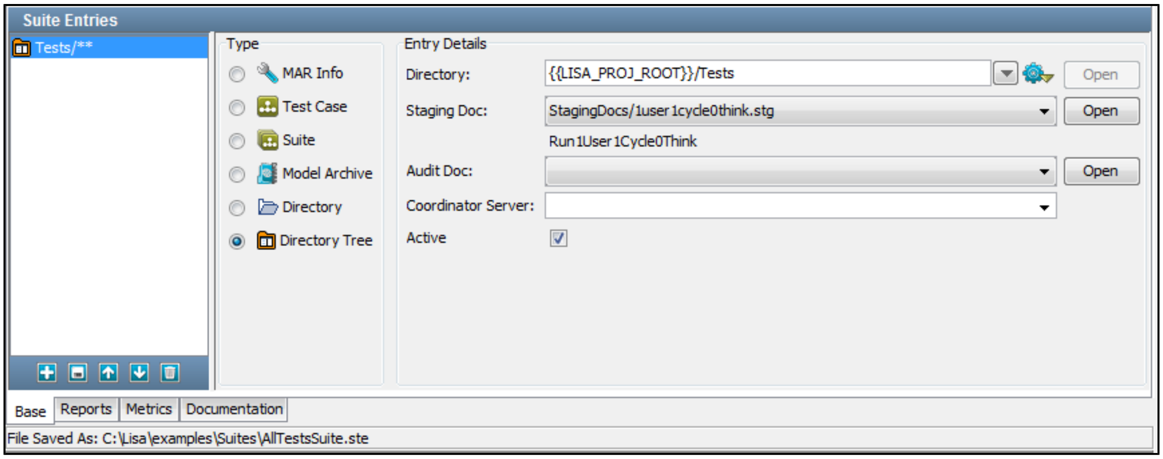
□ Allow Duplicate Test Runs

This tab has the following parameters:

- **Serial**: Tests will be run one after another in the order they show up in the suite document. This is the setting you would use for functional and regression testing.
- **Parallel**: Tests will be run at the same time. This is the setting you would use for load and performance testing. You must have enough virtual users to be able to run all tests concurrently.
- **Allow Duplicate Test Runs**: Lets you choose if you want the same test to run more than once. Duplicate tests can occur in a suite if the same test appears in an included suite, a directory, or a directory tree.

Bottom Panel

The bottom panel of the Base tab shows the types of documents that can be added in the suite and the associated document details.


Suite Entries

Tests/**

Type: ○ MAR Info ○ Test Case ○ Suite ○ Model Archive ○ Directory ● Directory Tree

Entry Details

Directory: {{LISA_PROJ_ROOT}}/Tests

Staging Doc: StagingDocs/1user1cycle0think.stg

Run1User1Cycle0Think

Audit Doc:

Coordinator Server:

Active ☑

Base | Reports | Metrics | Documentation

File Saved As: C:\Lisa\examples\Suites\AllTestsSuite.ste

This panel is where you build your suite by adding individual tests, existing suites, and directories and directory trees that contain tests.

A list of the suite entries is displayed in the left panel, after all of the tests have been added and saved to the suite document.

Suite entries are added individually by selecting from the following types. Depending on the selection, the Entry Details area will change.

Type: Select the suite entry type as one of the following:

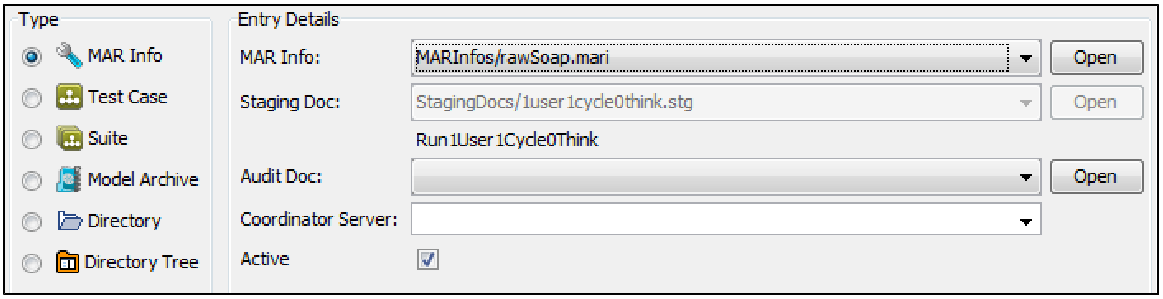
- **MAR Info**: The name of a Model Archive (MAR) info file.
- **Test Case**: The name of a test case.
- **Suite**: The name of a suite document. All tests will be extracted from this suite and run as individual tests.
- **Model Archive**: The name of a Model Archive (MAR) file.
- **Directory**: The name of a directory that contains tests to be included in the suite. The same staging document, audit document, and coordinator server will be used for all tests in this directory. If new test cases are added to the directory, they will automatically be included in this suite.
- **Directory Tree**: The name of a directory that contains tests to be included in the suite. LISA will look in the named directory and recursively through all sub-directories in the tree. The same staging document, audit document, and coordinator server will be used for all tests in this directory tree. If new test cases are added to the directory tree, they will automatically be included in this suite.

The Active check box lets you control whether the selected entry will run. For example, you can clear the Active check box for one of the test cases in a suite.

Adding a Document Type in a Suite

Depending on the document type selected, the fields for entry details change.

As an example, we will add a MAR info document. The entry details for the same have changed from the ones shown for a directory tree.


Type: ● MAR Info ○ Test Case ○ Suite ○ Model Archive ○ Directory ○ Directory Tree

Entry Details

MAR Info: MARInfos/rawSoap.mari

Staging Doc: StagingDocs/1user1cycle0think.stg

Run1User1Cycle0Think

Audit Doc:

Coordinator Server:

Active ☑

- **MAR Info**: The MAR info name. Enter the name or select from the pull-down menu or browse to the file or directory. When it is selected, click Open to open the respective file.
- **Staging Doc**: The name of the staging document for this entry. Enter the name, select from the pull-down menu, or browse to the document. If you leave this entry blank, the default staging document will be used. After it is selected, click Open to open the staging document. When you have selected a staging document to use in this suite, the name of the staging document (the same Run Name as you see on the editor tab when you are editing the document) appears below the Staging Doc field.
- **Audit Doc**: The name of the audit document for this entry. Enter the name, select from the pull-down menu, or browse to the document. If you leave this entry blank, the default audit document will be used. After it is selected, click Open to open the audit document.
- **Coordinator Server**: The name of the coordinator server to use with this entry. Select from the list of available coordinator servers listed in the pull-down menu. This parameter is needed only if you are running a LISA Server.

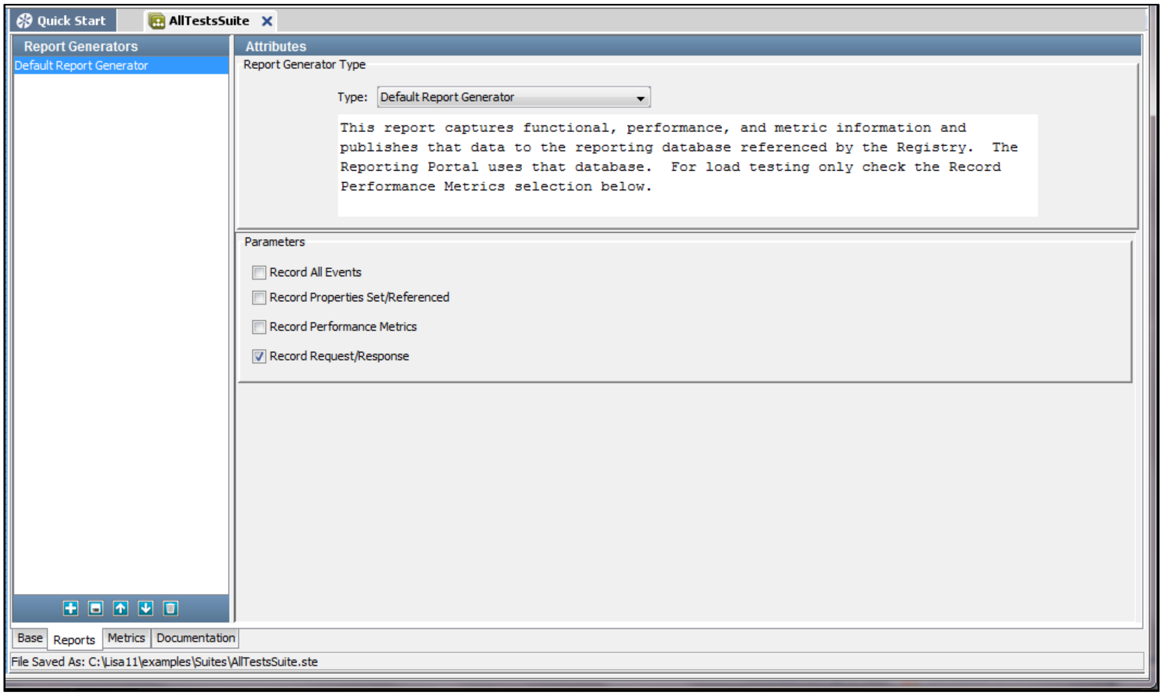
Click the Add icon on the toolbar to add this entry to the list shown in the left panel. You can delete entries by using the Delete icon. You can re-arrange entries by using the Move Up and Move Down icons.

After you have entered all your test entries, save your test suite document.

Test Suite Editor - Reports Tab

The Reports tab of the Test Suite Editor is where you specify the LISA test reports you want to produce, and the events you want to capture at the test suite level.

Details of the reports available in each report generator, viewing reports, report contents and output options are discussed in detail in Reports. The metrics to capture for the reports are discussed more fully in Metrics.



Default Report Generator

Report Generator Type

Type: Default Report Generator

```
This report captures functional, performance, and metric information and
publishes that data to the reporting database referenced by the Registry.  The
Reporting Portal uses that database.  For load testing only check the Record
Performance Metrics selection below.
```

Parameters

☐ Record All Events

☐ Record Properties Set/Referenced

☐ Record Performance Metrics

☑ Record Request/Response

File Saved As: C:\Lisa11\examples\Suites\AllTestsSuite.ste

The Reports tab contains the following panels:

- Report Generators
- Attributes

Report Generators

The Report Generators panel has a list of the reports that have been selected.

To add a report, click the Add icon ⊞ at the bottom of the list panel and select the report type from the Type pull-down menu in the center of the Reports panel.

To delete a report, select the report in the list and click the Delete 🗑 icon.

Attributes

The Attributes panel lists the available report types.

The following report types are available:

- Default Report Generator
- XML Report Generator

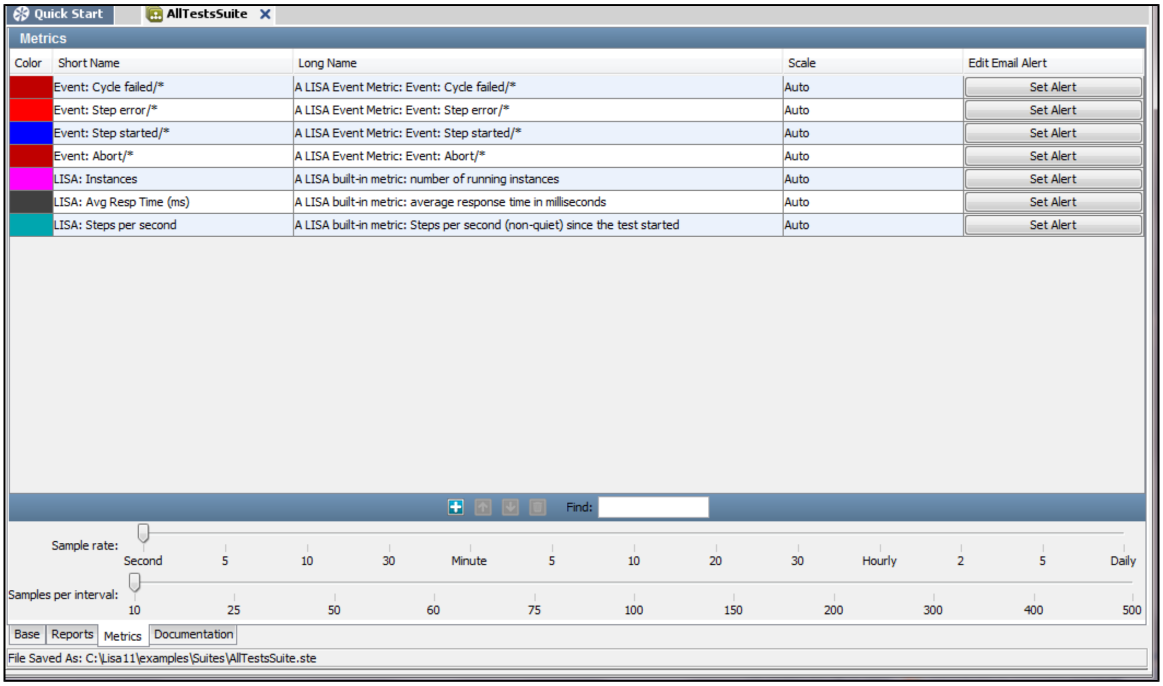
Parameters

- **Record All Events**: If selected, will record all events.
- **Record Properties Set/Referenced**: If selected, will record the set or referenced properties.
- **Record Performance Metrics**: If selected, will record the performance metrics. Use this for load testing.
- **Record Request/Response**: If selected, will record the request and response.

Test Suite Editor - Metrics Tab

The Metrics tab of the Test Suite Editor is where you select the test metrics you want to record, and set the sampling specifications for the collection of the metrics.

You can also set an email alert on any metric that you have selected.



Color	Short Name	Long Name	Scale	Edit Email Alert
🟥	Event: Cycle failed/*	A LISA Event Metric: Event: Cycle failed/*	Auto	Set Alert
🟥	Event: Step error/*	A LISA Event Metric: Event: Step error/*	Auto	Set Alert
🟦	Event: Step started/*	A LISA Event Metric: Event: Step started/*	Auto	Set Alert
🟥	Event: Abort/*	A LISA Event Metric: Event: Abort/*	Auto	Set Alert
🟪	LISA: Instances	A LISA built-in metric: number of running instances	Auto	Set Alert
⬛	LISA: Avg Resp Time (ms)	A LISA built-in metric: average response time in milliseconds	Auto	Set Alert
🟦	LISA: Steps per second	A LISA built-in metric: Steps per second (non-quiet) since the test started	Auto	Set Alert

File Saved As: C:\Lisa11\examples\Suites\AllTestsSuite.ste

The Metrics tab is divided into two sections.

The top panel shows the default metrics that are added to the suite. You can add more metrics by clicking the Add button on the toolbar.

In the bottom panel, two slider bars enable you to set sampling parameters:

- **Sample rate**: Specifies how often to take a sample, that is, record the value of a metric. This parameter is specified as a time period, and is the reciprocal of the sampling rate.
- **Samples per interval**: Specifies how many samples are used to create an interval for calculating summary values for the metric.

So, taking sample values every minute (Sample Rate Per Interval=1 minute), and averaging every 60 samples (Samples Per Interval=60), will produce a metric value every minute, and a summary value (average) every hour.

For example, the default is a 1 second Sample Rate, and 10 samples per interval (making an interval 10 seconds).

Metric values are stored in XML files or database tables for inclusion in reports (see Reports).

Adding Metrics

The following metrics categories are available:

- LISA Whole Test Metrics
- LISA Test Event Metrics
- JMX Metrics
- SNMP Metrics
- TIBCO Hawk Metrics
- Windows Perfmon Metrics
- UNIX Metrics Via SSH

To add metrics

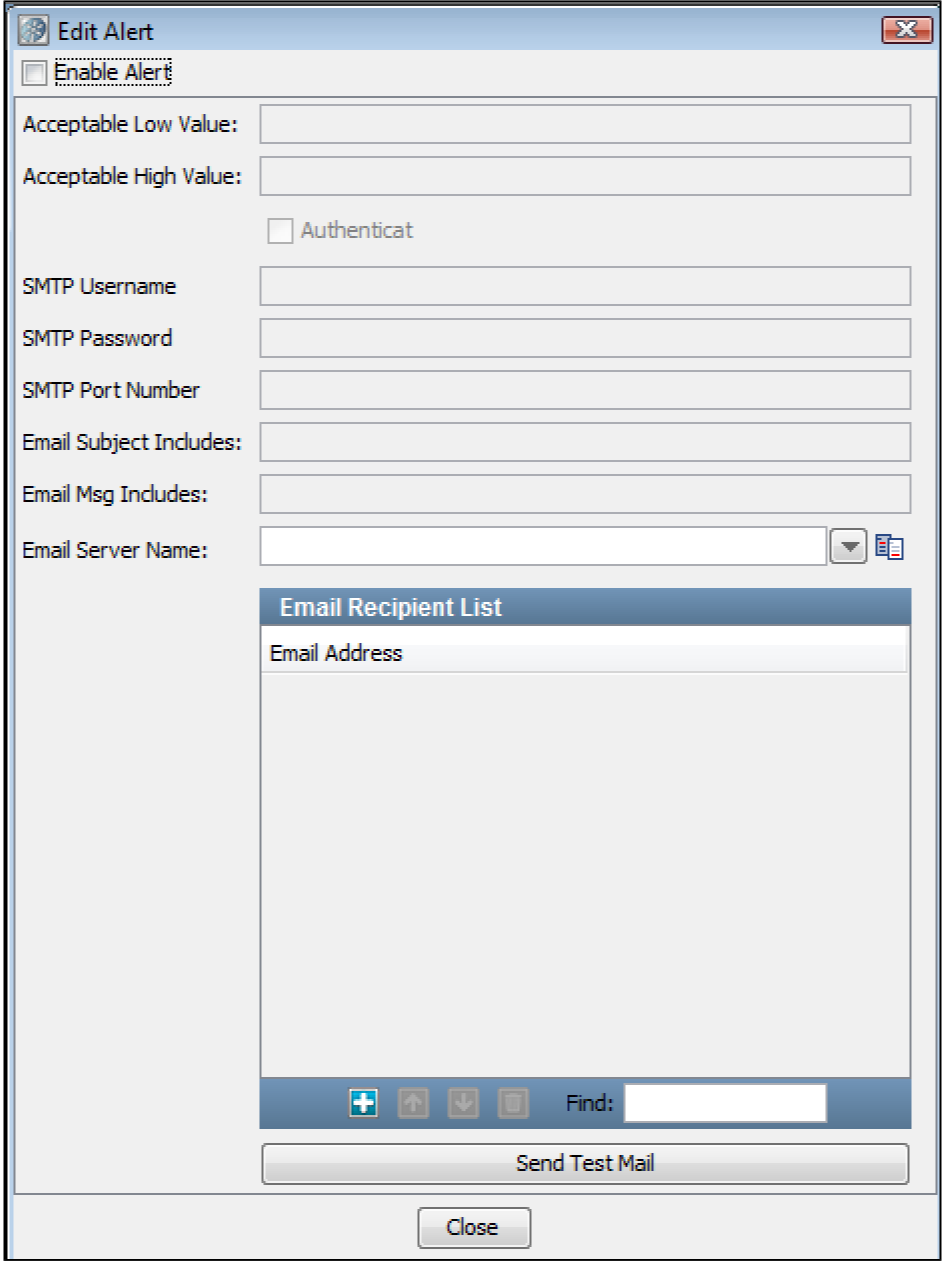
1. Click the Add button on the toolbar. The Add Metric dialog appears.
2. Select the target metric and click OK.
3. Repeat the procedure to configure metrics from the other categories.

For descriptions of all the metrics, in all categories, and details on how to configure them for inclusion in your reports, see Generating Metrics.

Setting Email Alerts

To set an email alert

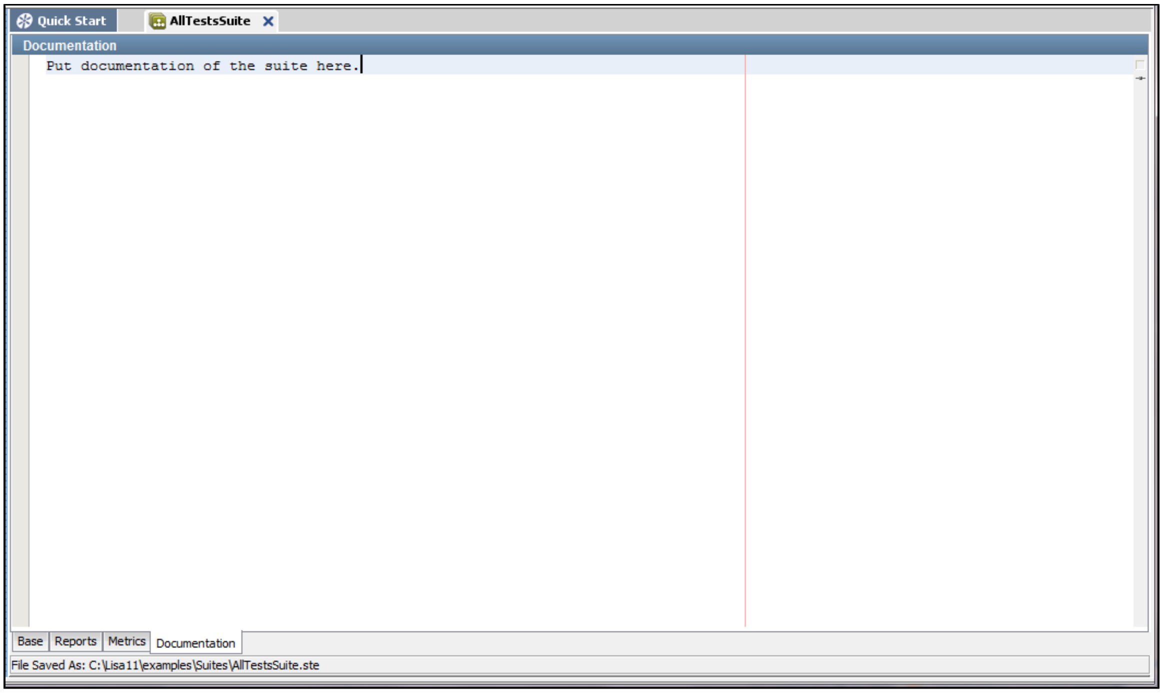
1. Click the Set Alert button corresponding to the metric. The Edit Alert dialog appears.



2. You can set the acceptable limits for the metric (low and high value), and the details to be sent in an email. You must provide the name of your email server, and a list of email addresses.
3. You can add and delete email addresses by using the Add and Delete icons at the bottom of the window.
4. When finished, click Close.

Test Suite Editor - Documentation Tab

The Documentation tab of the Test Suite Editor is where you enter the related documentation.



```
Put documentation of the suite here.
```

File Saved As: C:\Lisa11\examples\Suites\AllTestsSuite.ste

Working with Model Archives (MARs)

The main deployment artifact is a type of file named a Model Archive (MAR).

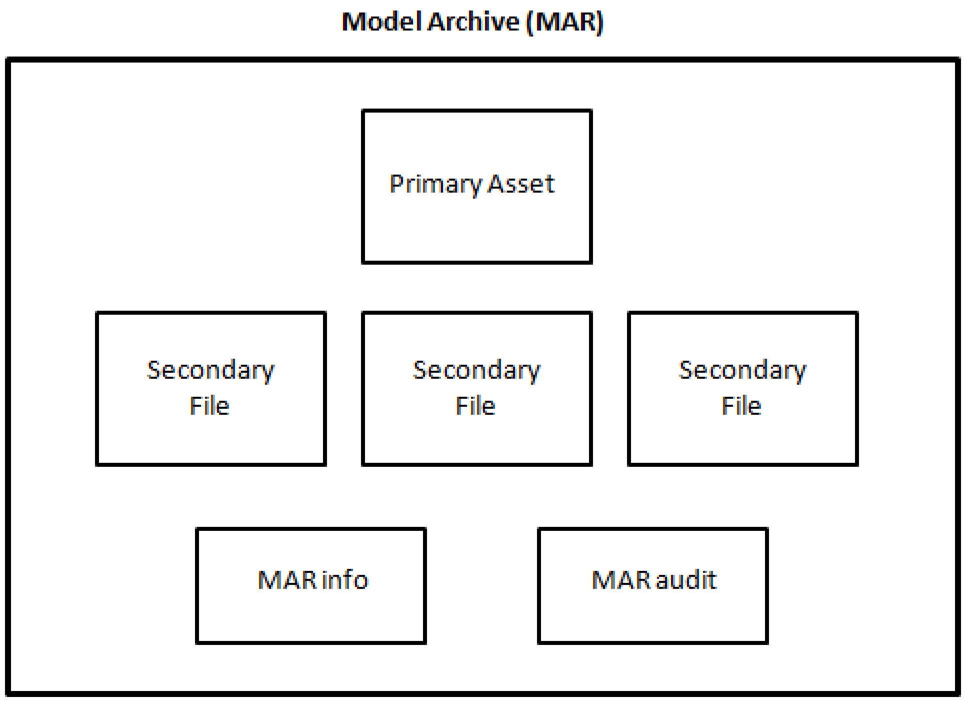
You can configure MARs from LISA Workstation or by using the Make Mar command-line utility.

In this section, the following topics are covered:

Model Archive (MAR) Overview

The main deployment artifact is a type of file referred to as a Model Archive (MAR). The file extension is **.mar**.

MAR files contain a primary asset, all secondary files that are needed to run the primary asset, an info file, and an audit file.



MAR files can also include the contents of the **.settings** directory, which is used in LISA Web 2.0.

MAR files are created from files in a project. After a MAR file is created, it is independent of the project.

Contents of a MAR

MAR files contain one of the following primary assets:

- Test case
- Suite
- Virtual service
- Test case monitor
- Suite monitor

MAR files also contain any secondary files that are required by the primary asset.

For example, if the primary asset is a virtual service model, then the MAR file also contains a service image.

In addition, MAR files contain the following files:

- MAR info file
- MAR audit file

You can specify that a MAR file be *optimized*. When the MAR file is built, only those project files that are required will be added. However, you can also configure an optimized MAR file to include one or more non-required project files.

If a MAR file is not optimized, all the project files will be added.

> ⚠ An archive is typically held in memory. Therefore, the use of optimized archives is highly encouraged.

MAR Info File

MAR info files contain information needed to create a MAR. The file extension is **.marinfo**.

The information specified in a MAR info file depends on the type of primary asset.

Primary Asset	Information Specified

Test case	Test case, configuration file, and staging document
Suite	Suite and configuration file
Virtual service	Virtual service model, configuration file, concurrent capacity, think time scale, and auto restart flag
Test case monitor	All the information specified for a test case, plus the service name, notification email, priority, and run schedule
Suite monitor	All the information specified for a suite, plus the service name, notification email, priority, and run schedule

MAR info files are located in the **MARInfos** folder of a project. However, when you use the stage/deploy related options, a MAR info file is created but not saved.

The **examples** project contains a variety of MAR info files that you can review.

MAR Audit File

MAR audit files contain metadata about the creation of a MAR. The file extension is **.maraudit**.

The following metadata is included in a MAR audit file:

- The date and time when the MAR was created
- The name of the computer on which the MAR was created
- The name of the user that created the MAR

Explicit and Implicit MAR Creation

There are two approaches to creating MARs: explicit and implicit.

To use the web-based dashboards or command-line utilities to deploy assets, you must explicitly create a MAR. An exception is the Test Runner command-line utility, which you can use to run test cases and suites that are not packaged in a MAR.

If you are staging a test case, staging a suite, deploying a virtual service, deploying a test case as a monitor, or deploying a suite as a monitor from within LISA Workstation, then you use the implicit approach.

Explicit MAR Creation

In the explicit approach, you perform steps to create a MAR info file, which you then use to build the MAR.

The general steps are as follows:

1. Identify the primary asset that you want to execute.
2. Create a MAR info file.
3. Build the MAR.

Implicit MAR Creation

In the implicit approach, both a MAR info file and a MAR are automatically created.

The general steps are as follows:

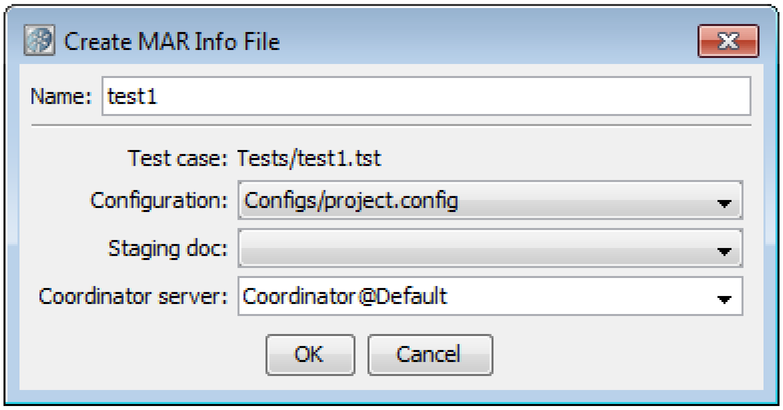
1. Identify the primary asset that you want to execute.
2. Stage a test case, stage a suite, deploy a virtual service, deploy a test case as a monitor, or deploy a suite as a monitor from within LISA Workstation or by using Test Runner.

Creating MAR Info Files

You can create a MAR info file from an existing test case, suite, or virtual service.

To create a MAR info file from an existing test case

1. In the Project panel of LISA Workstation, right-click the test case and select Create MAR Info File.
   The Create MAR Info File dialog appears.



Name: test1

Test case: Tests/test1.tst

Configuration: Configs/project.config

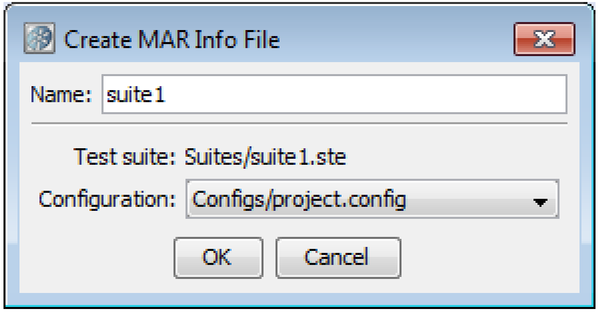
Staging doc:

Coordinator server: Coordinator@Default

2. In the Name field, enter a name for the MAR info file.
3. In the Configuration field, select the configuration file for this test case.
4. In the Staging doc field, select the staging document for this test case.
5. In the Coordinator server field, select the coordinator server for this test case (if applicable).
6. Click OK.
   The .mari file is created in the **MARInfos** folder of the Project panel.

To create a MAR info file from an existing suite

1. In the Project panel of LISA Workstation, right-click the suite and select Create MAR Info File.
   The Create MAR Info File dialog appears.



Name: suite1

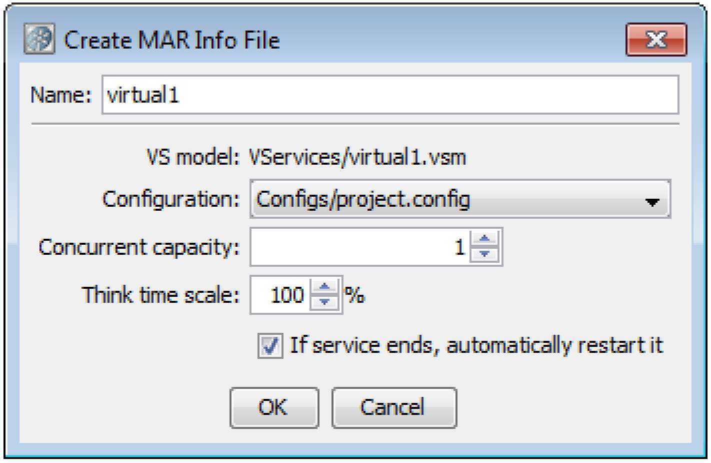
Test suite: Suites/suite1.ste

Configuration: Configs/project.config

2. In the Name field, enter a name for the MAR info file.
3. In the Configuration field, select the configuration file for this suite.
4. Click OK.
   The .mari file is created in the **MARInfos** folder of the Project panel.

To create a MAR info file from an existing virtual service

1. In the Project panel of LISA Workstation, right-click the virtual service and select Create MAR Info File.
   The Create MAR Info File dialog appears.



Name: virtual1

VS model: VServices/virtual1.vsm

Configuration: Configs/project.config

Concurrent capacity: 1

Think time scale: 100 %

☑ If service ends, automatically restart it

2. In the Name field, enter a name for the MAR info file.
3. In the Configuration field, select the configuration file for this virtual service.
4. In the Concurrent capacity field, enter a number to indicate the load capacity. The default value is 1, but it can be increased to provide additional capacity. **Capacity** is how many virtual users (instances) can execute with the virtual service model at one time. **Capacity** here indicates how many threads there are to service requests for this service model.
5. In the Think time scale field, enter the think time percentage with respect to the recorded think time. To double the think time, use 200. To halve the think time, use 50. The default value is 100.

6. The **If service ends, automatically restart it** check box keeps the service running even after an emulation session has reached its end point. Clear the check box, or leave it selected.
7. Click OK.
   The .mari file is created in the **MARInfos** folder of the Project panel.

Creating Monitor MAR Info Files

You can create a Monitor MAR info file from an existing test case or suite.

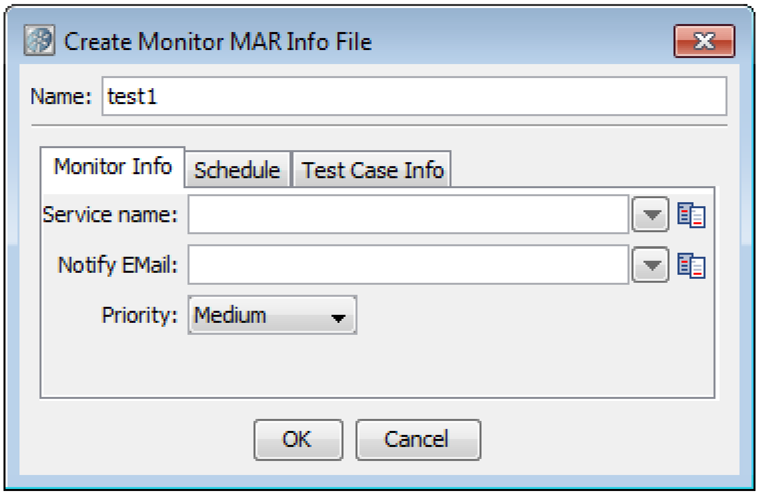
For more information on monitors, see Continuous Validation Service (CVS).

To create a Monitor MAR info file

1. In the Project panel of LISA Workstation, right-click the test case or suite and select Create Monitor MAR Info File.
   The Create Monitor MAR Info File dialog appears.
2. In the Name field, enter a name for the MAR info file.
3. In the Monitor Info tab, specify the monitor information.
4. In the Schedule tab, specify the time schedule information.
5. (Test Case) In the Test Case Info tab, specify the test case information.
6. (Suite) In the Suite Info tab, specify the suite information.
7. Click OK.
   The .mari file is created in the **MARInfos** folder of the Project panel.

Monitor Info Tab

The following image shows the Monitor Info tab of the Create Monitor MAR Info File dialog.



Name: test1

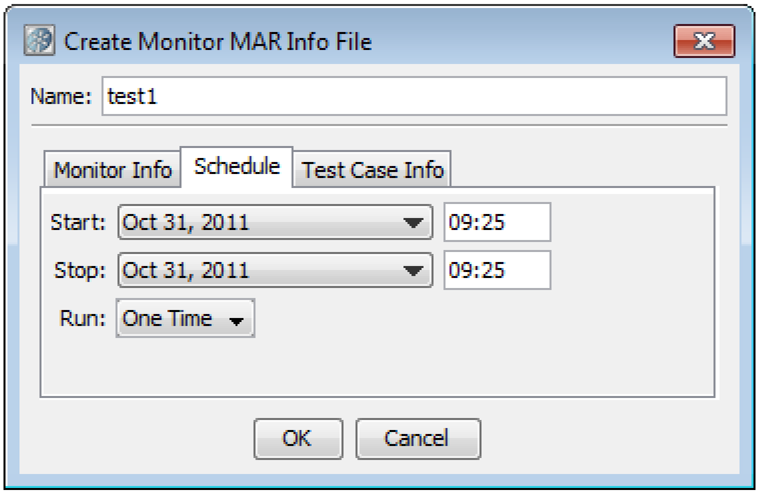
Priority: Medium

Specify the following information:

- **Service Name:** The name of the service. This name appears in the CVS Dashboard.
- **Notify Email:** The email address of the person for notification.
- **Priority:** Set the priority to High, Medium High, Medium, Medium Low, or Low.

Schedule Tab

The following image shows the Schedule tab of the Create Monitor MAR Info File dialog.



Name: test1

Start: Oct 31, 2011 09:25

Stop: Oct 31, 2011 09:25

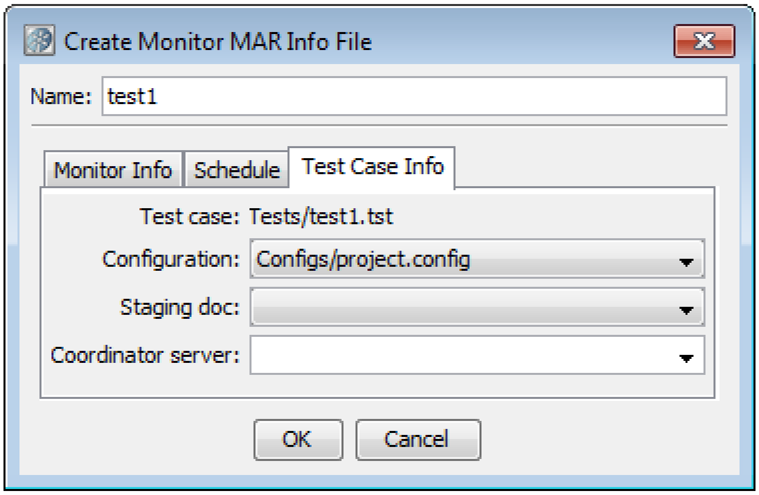
Run: One Time

Specify the following information:

- **Start:** The date and time when the monitor will start. For the time value, you must use a 24-hour clock.
- **Stop:** The date and time when the monitor will stop. For the time value, you must use a 24-hour clock. The stop date and time must be later than the start date and time.
- **Run:** How often the monitor will run. Depending on the selection, additional options may appear.
  - **One Time:** The monitor will run once, at the start date and time.
  - **Every:** Enter the frequency in minutes. The monitor will run every NN minutes. However, if the previous run has not completed, CVS will skip the run and try again at the next scheduled time. CVS will not terminate a monitor after it has started.
  - **Daily:** The monitor will run once a day, at the time specified in the start time.
  - **Weekly:** Select one or more days of the week. The monitor will run once on each selected day, at the time specified in the start time.
  - **Monthly:** Enter one or more days of the month. If you enter multiple days, they must be separated by spaces. For example, you could enter **1 15 30**. If you specify a day that does not occur in a particular month (such as 31), the day is ignored for that month. The monitor will run once on each day, at the time specified in the start time.
  - **CRON:** Enter a standard cron specification. Cron is a standard UNIX syntax for specifying time intervals. This approach allows the most flexibility when specifying a run schedule.

Test Case Info Tab

The following image shows the Test Case Info tab of the Create Monitor MAR Info File dialog.



Name: test1

Test case: Tests/test1.tst

Configuration: Configs/project.config

Staging doc:

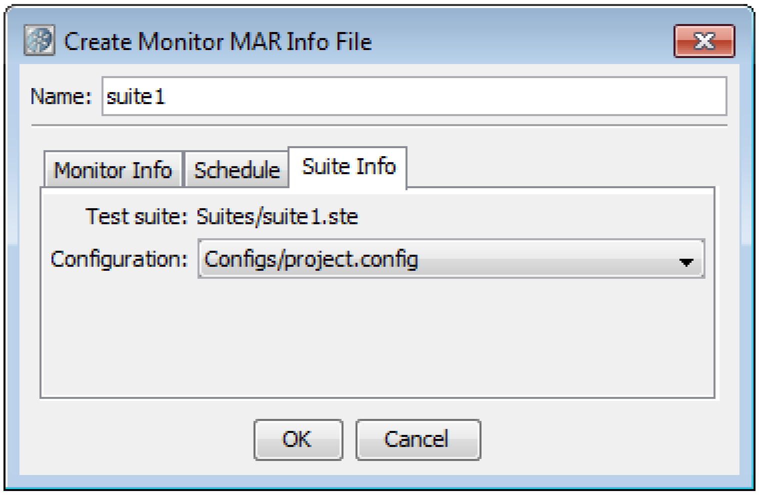
Coordinator server:

Specify the following information:

- **Configuration:** The name of the configuration. If this field is blank, the configuration active in the test case is used.
- **Staging doc:** The name of the staging document.
- **Coordinator Server:** The name of the coordinator server.

Suite Info Tab

The following image shows the Suite Info tab of the Create Monitor MAR Info File dialog.



Name: suite1

Test suite: Suites/suite1.ste

Configuration: Configs/project.config

Specify the following information:

- **Configuration:** The name of the configuration.

Editing MAR Info Files

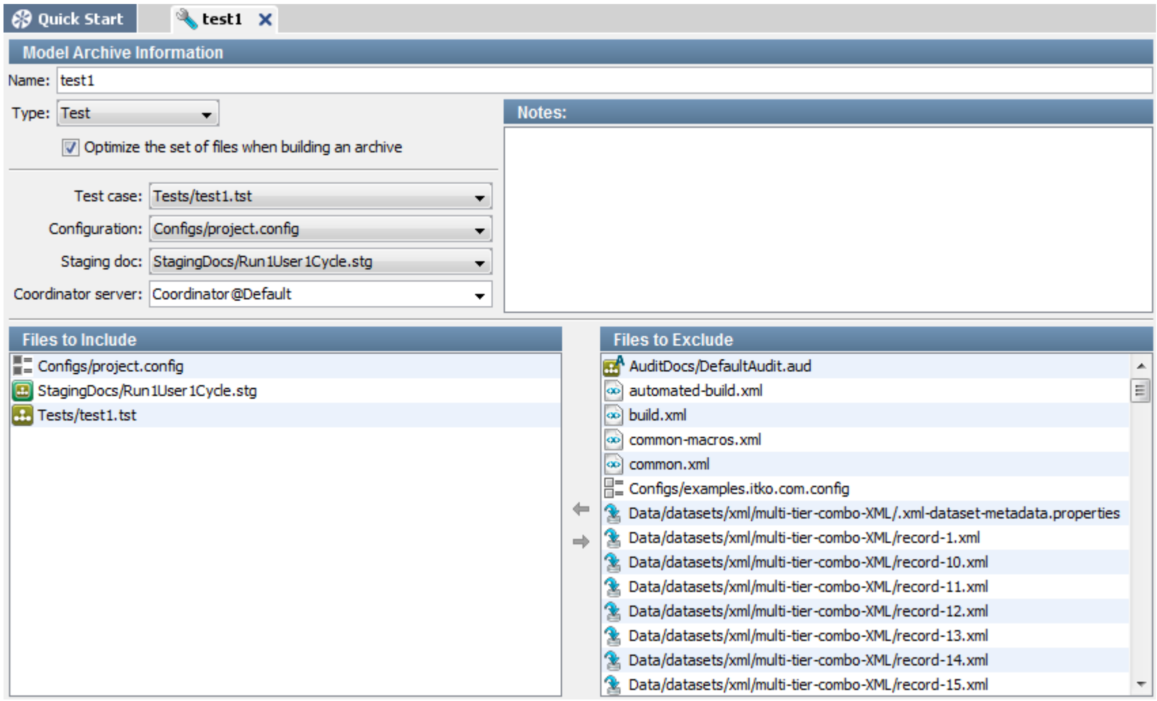
The MAR info editor lets you change the information that was specified during the creation of the MAR info file.

The editor includes an optimize check box. By default, the check box is selected. When the MAR file is built, only those project files that are required by the primary asset will be added. However, the editor lets you specify that an optimized MAR file include one or more non-required project files.

⚠ An archive is typically held in memory. Therefore, the use of optimized archives is highly encouraged.

If you clear the optimize check box, then all the project files will be added to the MAR file.

The following image shows the MAR info file editor.



Name: test1

Type: Test

☑ Optimize the set of files when building an archive

Test case: Tests/test1.tst

Configuration: Configs/project.config

Staging doc: StagingDocs/Run1User1Cycle.stg

Coordinator server: Coordinator@Default

Files to Include:

- Configs/project.config
- StagingDocs/Run1User1Cycle.stg
- Tests/test1.tst

Files to Exclude:

- AuditDocs/DefaultAudit.aud
- automated-build.xml
- build.xml
- common-macros.xml
- common.xml
- Configs/examples.itko.com.config
- Data/datasets/xml/multi-tier-combo-XML/.xml-dataset-metadata.properties
- Data/datasets/xml/multi-tier-combo-XML/record-1.xml
- Data/datasets/xml/multi-tier-combo-XML/record-10.xml
- Data/datasets/xml/multi-tier-combo-XML/record-11.xml
- Data/datasets/xml/multi-tier-combo-XML/record-12.xml
- Data/datasets/xml/multi-tier-combo-XML/record-13.xml
- Data/datasets/xml/multi-tier-combo-XML/record-14.xml
- Data/datasets/xml/multi-tier-combo-XML/record-15.xml

To edit MAR info files

1. In the Project panel of LISA Workstation, double-click the MAR info file.
   The editor appears.
2. The information that you can edit in the upper area depends on the type of MAR info file.
3. In the Notes area, add any notes about the MAR info file.
4. By default, the optimize check box is selected. The Files to Include area lists the required project files. The Files to Exclude area lists the non-required project files. If you want the MAR to include any non-required project files, then move the files to the Files to Include area.
5. If you want the MAR to include all of the project files, then clear the optimize check box. The Files to Include and Files to Exclude areas will be disabled.
6. From the main menu, select File > Save.

Building MARs

After you have created and edited a MAR info file, you can use the file to build a Model Archive (MAR).

You can save the MAR to a folder that is inside or outside the project directory.

To build a MAR

1. In the Project panel of LISA Workstation, right-click the MAR info file and select Build Model Archive.
   The Build Model Archive dialog appears.
2. Navigate to the folder where you want to save the MAR.
3. Click Save.

Deploying to CVS

You can deploy a Monitor MAR info file to CVS.

To deploy to CVS

1. In the Project panel of LISA Workstation, right-click the .mari file and select Deploy to CVS.
   A confirmation message appears.
2. You can go to the CVS Dashboard and see the newly added monitor.

Make Mar

You can use the Make Mar command-line utility to show the contents of MAR info files (standalone or in an archive) or to create model archive files from MAR info files.

This utility is located in the **LISA_HOME/bin** directory.

Options

Each option has a short version and a long version. The short version begins with a single dash. The long version begins with two dashes.

-h, --help

Displays help text.

-s file-name, --show=file-name

Shows the contents of a MAR info file.

If the file name refers to a MAR info file, then it is written out. If the file name refers to an archive, then both the audit and info entries are written out.

-c, --create

Creates one or more model archives from MAR info files.

Use the --marinfo argument to specify the MAR info file name to build the archive.

Use the --archive argument to specify the name of the archive to create.

If the --source-dir argument is used, then the --marinfo argument is taken as a name pattern to look for in the full directory tree. In this case, the --target-dir argument must be used to note where to place the created archives. This also implies auto-naming of the created archives.

-m mar-info-name, --marinfo=mar-info-name

Specifies the name of the MAR info file to read (if the --source-dir argument is not used) or the MAR info file name pattern to look for (if the --source-dir argument is used). In the latter case, if not specified, it will default to **.mari**.

-s directory, --source-dir=directory

Specifies the directory where the tool will search for MAR info files to make archives from. The full directory tree will be searched for files that match the pattern specified by the --marinfo argument.

-t output-directory, --target-dir=output-directory

Specifies the directory where archives are to be written.

When this argument is used, the created archives are automatically named based on the MAR info file name with a numeric suffix as appropriate to help ensure that no files are overwritten.

-a archive-file, --archive=archive-file

Specifies the name of the archive file to create.

When this argument is used, the --marinfo argument must specify a single existing MAR info file and neither --source-dir nor --target-dir are allowed.

--version

Print the version number.

Examples

The following example shows the contents of a MAR info file.

```
MakeMar --show=C:\Lisa\examples\MARInfos\AllTestsSuite.mari
```

The following example creates a model archive named **rawSoap.mar**.

```
MakeMar --create --marinfo=C:\Lisa\examples\MARInfos\rawSoap.mari --archive=rawSoap.mar
```

The following example creates a model archive for every MAR info file in the **examples\MARInfos** directory. The destination directory **examples\MARs** must exist before you run this command.

```
MakeMar --create --source-dir=examples\MARInfos --target-dir=examples\MARs
```

Running Test Cases and Suites

You have many options for running test cases:

- In LISA Workstation, you can use the Interactive Test Run (ITR) utility to walk through and verify the steps of a test case.
- In LISA Workstation, you can run a test case quickly with minimal setup.
- In LISA Workstation, you can stage a test case. This option requires you to specify a configuration, staging document, and coordinator server. Often, the same test cases are used with different staging documents to perform a variety of different tests. That is, a separate test case does not have to be written for functional, regression, and load testing. Instead, a different staging document is prepared using the same test case.
- Test Runner is a command-line utility that lets you run tests as a batch process.
- LISA Invoke is a REST-like web application that enables you to run test cases and suites with a URL.
- In the Server Console, you can deploy the Model Archive (MAR) for a test case to a lab.
- You can schedule tests to run at regular time intervals by using the Continuous Validation Service (CVS).
- You can run tests as part of an automatic build process by using Ant and JUnit.

Using the Interactive Test Run (ITR) Utility

The Interactive Test Run (ITR) utility lets you walk through and verify the steps of a test case. Therefore, you can verify that test cases are running correctly before staging them. The ITR runs the test that is currently in memory, rather than loading a test case document. You can make changes in real time to alter test properties or the workflow.

For example, you can choose to run a particular test step out of the natural workflow sequence. The real-time changes that you make as the test is running will be integrated into the test case, and it will continue running without requiring a restart, unless the change was global in nature, like changing the active configuration. If you execute a step and get an unexpected result, perhaps because you provided a bad parameter, you can leave the ITR open, edit the offending test step, then go back to the ITR and execute the same step again.

The ITR lets you save the current ITR state, and to load an ITR state that was previously saved. This feature lets you communicate an error, in context, to another member of your team. You can send the test case and the saved ITR state to provide the exact actions and the results that you observed.

The following topics are available.

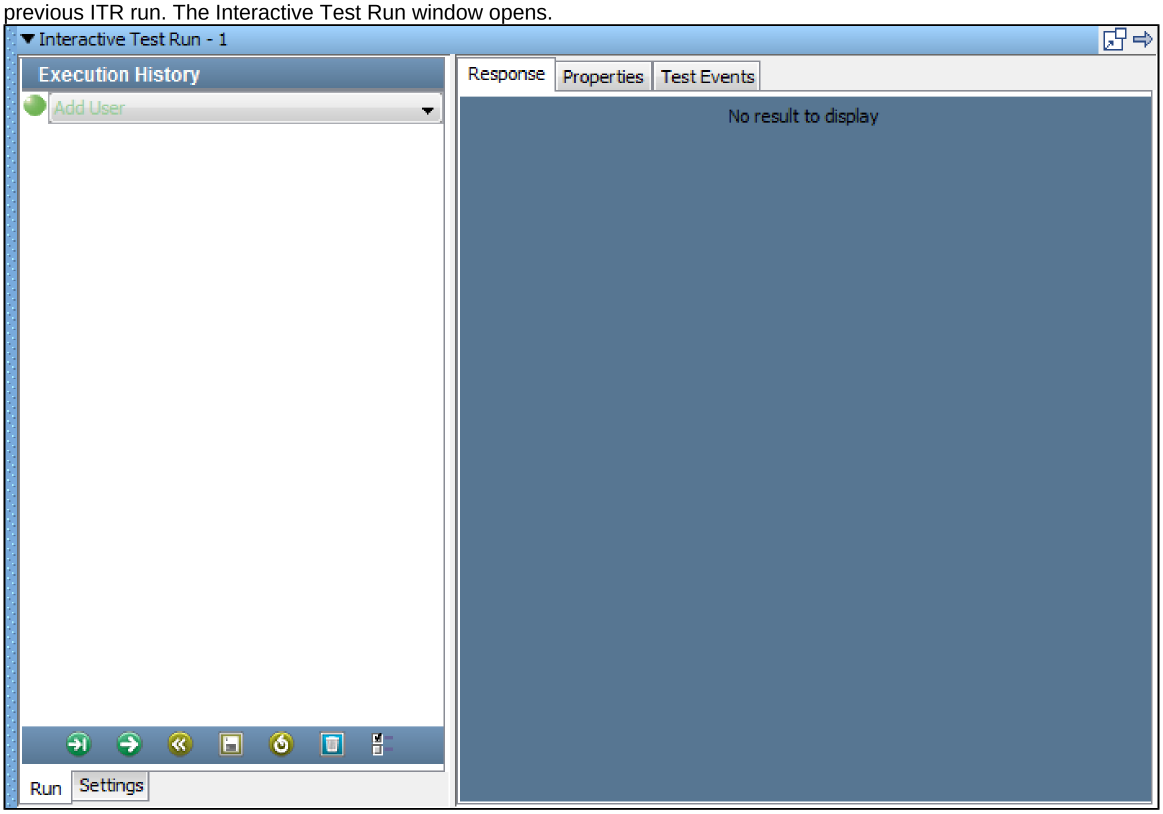
Starting an ITR Run

The Interactive Test Run (ITR) utility is run from an open test case in LISA Workstation.

To start an ITR run

1. From the main menu, select Actions > Start Interactive Test Run.

   Alternately, you can click the ITR icon on the toolbar. The icon gives you the option of starting a new ITR run or opening a previous ITR run. The Interactive Test Run window opens.

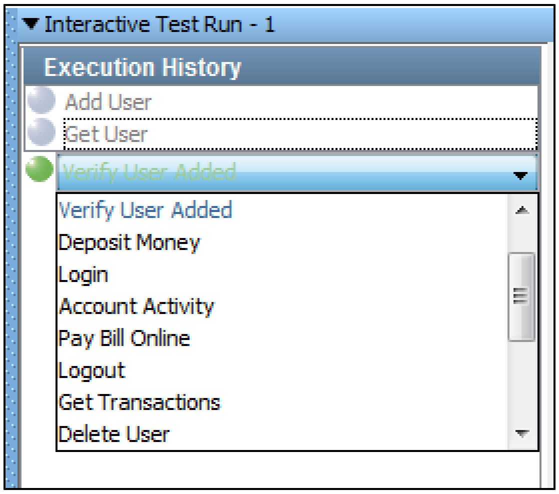


2. (Optional) Use the Settings tab to change one or more settings.
3. Use the Run tab to execute all or part of the test case.

Run Tab

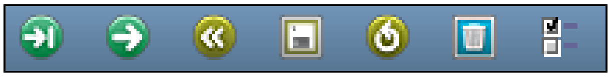
The Run tab shows the steps that have been executed (in gray) and the next step that will be executed (in green).

In the following image, the first two steps (Add User and Get User) have been executed. The third step (Verify User Added) is the next step that will be executed.



Each step has a pull-down menu that contains all the steps in the test case. You can use this menu to change the next step that will be executed. Select a step, and it will replace the current next step with the one of your choice. After you change the step, the workflow will continue from the new step.

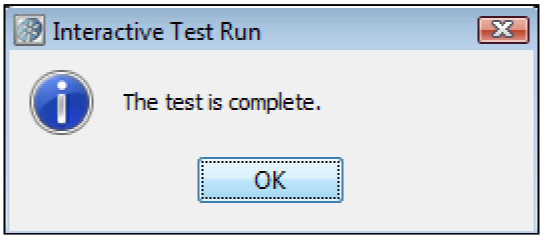
You manage the ITR by using the toolbar at the bottom of the Run tab.



To run the next step in the test case, click the Execute Next Step icon . After the step has run, you can examine the results in the right panel. Click the icon again to run the next step listed, or select a different step to run as described previously.

To run all of the steps in the test case, click the Automatically Execute Test icon . While the test is running, the icon will change into a Stop icon. You can stop the test by clicking the Stop icon.

When the last step has been executed, a dialog indicates that the test is complete.



The test is complete.

To start the test again, click the Restart Test icon. This feature is useful if you make a change to your test case and want to execute it from the beginning.

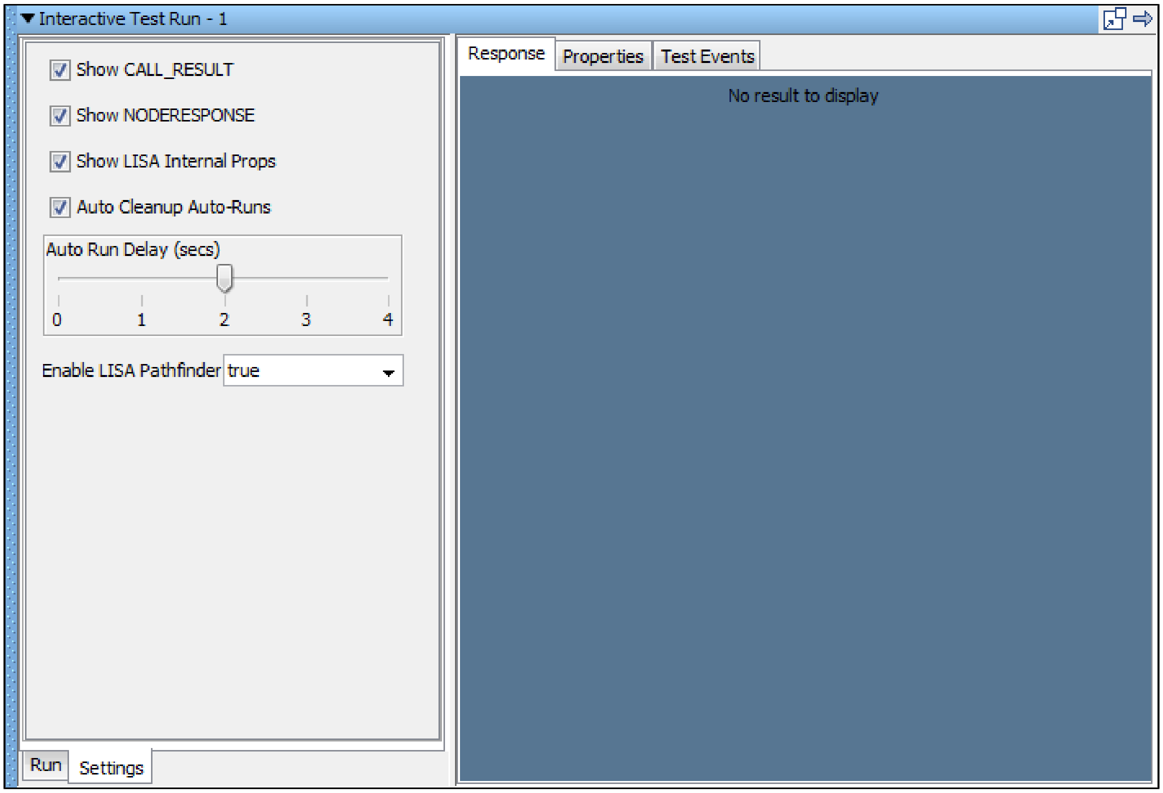
To save the current ITR state, click the Save ITR State icon . In the dialog, enter the name of the save file. The suffix .itr will be appended to the name. Subsequent saves will overwrite this file.

To load a saved ITR, click the Load Saved ITR State icon and browse to the .itr file that you want to load. The Saved ITR State will contain the information displayed in the tabs, thereby capturing all the testing performed. To make use of the Saved State, you must first open the test case that was used when the original tests were run.

To open the property watch window to add and/or watch properties, click the Add properties to watch icon.

Settings Tab

The Settings tab lets you control various aspects of the ITR.



☑ Show CALL_RESULT

☑ Show NODERESPONSE

☑ Show LISA Internal Props

☑ Auto Cleanup Auto-Runs

Auto Run Delay (secs): 2 (0 1 2 3 4)

Enable LISA Pathfinder: true

No result to display

You can filter out events and properties from the results tabs. There are some verbose events that you may not want to see. Likewise, you may not always want to clutter the property list with LISA internal properties.

- **Show CALL_RESULT**: Include EVENT_CALL_RESULT events in the Test Events list.
- **Show NODERESPONSE**: Include EVENT_NODERESPONSE events in the Test Events list.
- **Show LISA Internal Props**: Include all LISA internal events in the property lists in the Initial State and Post Exec State tabs.
- **Auto Cleanup Auto-Runs**: Clean up the Auto Runs automatically.

In the Automatically Execute Test mode, the ITR pauses between each step execution. The Auto Run Delay (secs) slider lets you change the pause interval. The ITR does not honor the think time, so this setting lets you add a constant delay between each step execution.

To control whether Pathfinder is enabled, select or clear the Enable LISA Pathfinder check box.

Examining the Results of an ITR Run

You can examine the results of the execution of a particular test step, during the run or between two steps or at the conclusion of the run.

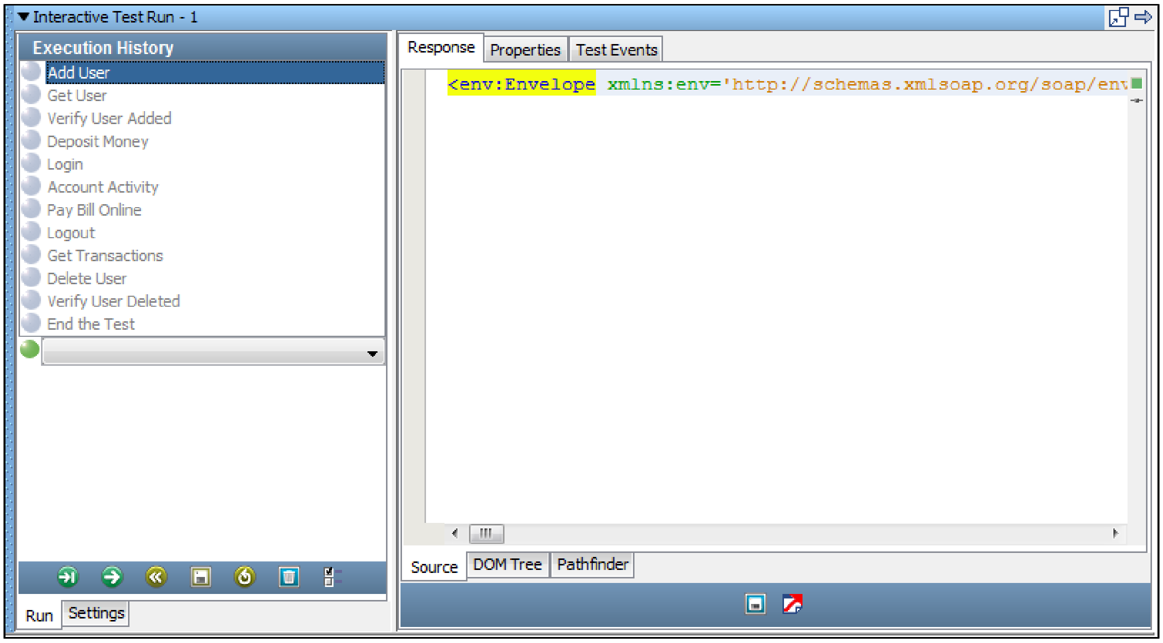
Select the step in the Execution History list. Examine the information in the right panel, which contains the following tabs:

- Response Tab
- Properties Tab
- Test Events Tab

Response Tab

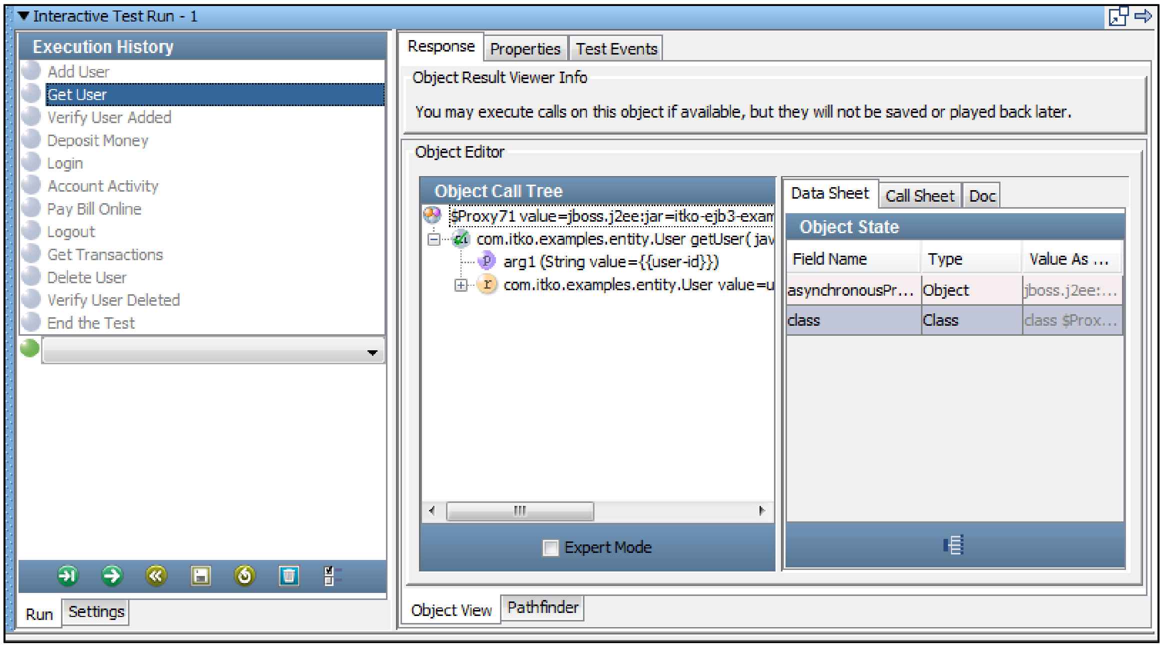
The Response tab displays the step response after executing a step.

The display will use the editor appropriate for the response type. For example, the Add User step in the multi-tier-combo test case invokes the HTML/XML response.



```
<env:Envelope xmlns:env='http://schemas.xmlsoap.org/soap/env
```

The Get User step in the multi-tier-combo test case invokes an EJB that returns an object that is displayed in the Complex Object Editor.

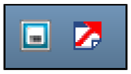


You may execute calls on this object if available, but they will not be saved or played back later.

```
$Proxy71 value=jboss.j2ee:jar=itko-ejb3-exam
  com.itko.examples.entity.User getUser( jav
    arg1 (String value={{user-id}})
    com.itko.examples.entity.User value=u
```

Field Name	Type	Value As ...
asynchronousPr...	Object	jboss.j2ee:...
class	Class	class $Prox...

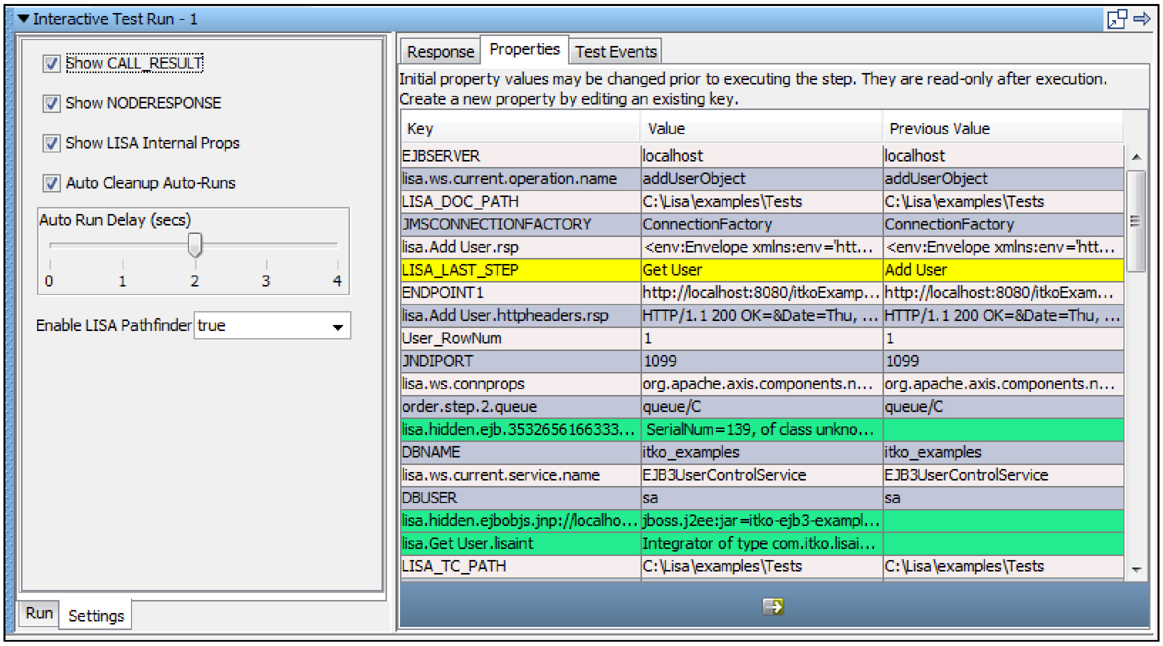
Some editors include two icons that you can use to save the ITR state and launch an external browser.



Properties Tab

The Properties tab displays the state of the step after execution (Value) and immediately before execution (Previous Value).

To show or hide LISA internal properties, select or clear the Show LISA Internal Props check box in the Settings tab.



☑ Show CALL_RESULT

☑ Show NODERESPONSE

☑ Show LISA Internal Props

☑ Auto Cleanup Auto-Runs

Auto Run Delay (secs): 2

Enable LISA Pathfinder: true

Initial property values may be changed prior to executing the step. They are read-only after execution.
Create a new property by editing an existing key.

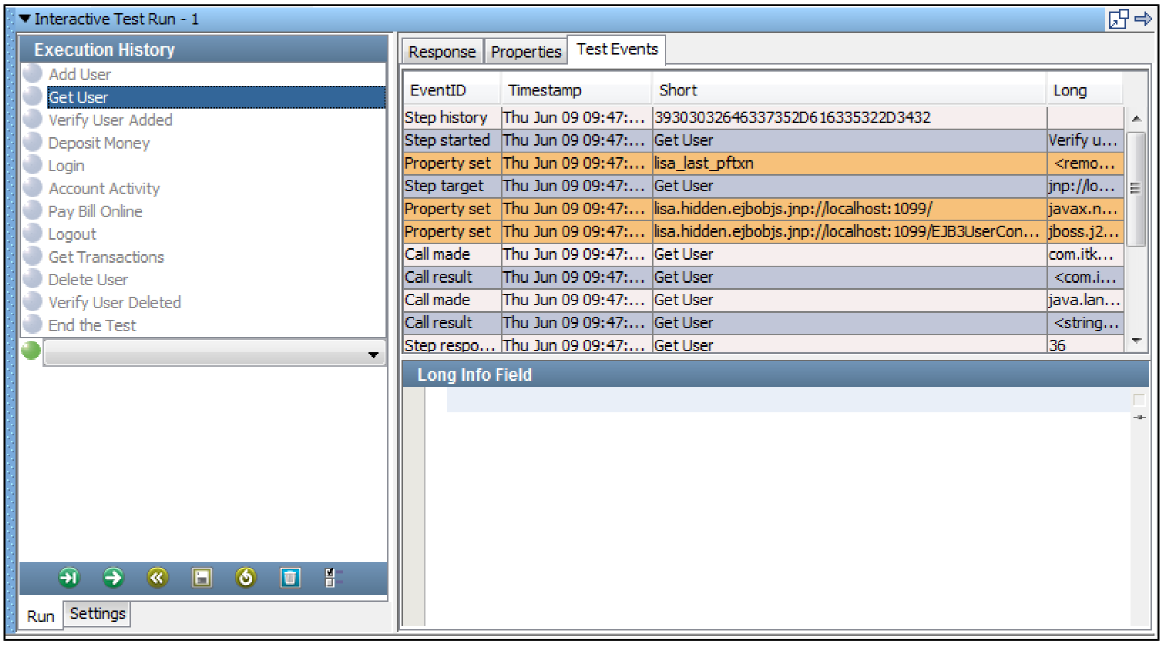
Key	Value	Previous Value
EJBSERVER	localhost	localhost
lisa.ws.current.operation.name	addUserObject	addUserObject
LISA_DOC_PATH	C:\Lisa\examples\Tests	C:\Lisa\examples\Tests
JMSCONNECTIONFACTORY	ConnectionFactory	ConnectionFactory
lisa.Add User.rsp	<env:Envelope xmlns:env='htt...	<env:Envelope xmlns:env='htt...
LISA_LAST_STEP	Get User	Add User
ENDPOINT1	http://localhost:8080/itkoExamp...	http://localhost:8080/itkoExam...
lisa.Add User.httpheaders.rsp	HTTP/1.1 200 OK=&Date=Thu, ...	HTTP/1.1 200 OK=&Date=Thu, ...
User_RowNum	1	1
JNDIPORT	1099	1099
lisa.ws.connprops	org.apache.axis.components.n...	org.apache.axis.components.n...
order.step.2.queue	queue/C	queue/C
lisa.hidden.ejb.3532656166333...	SerialNum=139, of class unkno...	
DBNAME	itko_examples	itko_examples
lisa.ws.current.service.name	EJB3UserControlService	EJB3UserControlService
DBUSER	sa	sa
lisa.hidden.ejbobjs.jnp://localho...	jboss.j2ee:jar=itko-ejb3-exampl...	
lisa.Get User.lisaint	Integrator of type com.itko.lisai...	
LISA_TC_PATH	C:\Lisa\examples\Tests	C:\Lisa\examples\Tests

Properties that were newly set by the step are highlighted in green. Properties that were modified in the step are highlighted in yellow.

Test Events Tab

The Test Events tab displays the events that were fired when the step was executed, in the order that the events were fired.

To show or hide the events, select or clear the event check boxes in the Settings tab (Show CALL_RESULT and Show NODE_RESPONSE).



Execution History

- Add User
- Get User
- Verify User Added
- Deposit Money
- Login
- Account Activity
- Pay Bill Online
- Logout
- Get Transactions
- Delete User
- Verify User Deleted
- End the Test

EventID	Timestamp	Short	Long
Step history	Thu Jun 09 09:47:...	39303032646337352D616335322D3432	
Step started	Thu Jun 09 09:47:...	Get User	Verify u...
Property set	Thu Jun 09 09:47:...	lisa_last_pftxn	<remo...
Step target	Thu Jun 09 09:47:...	Get User	jnp://lo...
Property set	Thu Jun 09 09:47:...	lisa.hidden.ejbobjs.jnp://localhost:1099/	javax.n...
Property set	Thu Jun 09 09:47:...	lisa.hidden.ejbobjs.jnp://localhost:1099/EJB3UserCon...	jboss.j2...
Call made	Thu Jun 09 09:47:...	Get User	com.itk...
Call result	Thu Jun 09 09:47:...	Get User	<com.i...
Call made	Thu Jun 09 09:47:...	Get User	java.lan...
Call result	Thu Jun 09 09:47:...	Get User	<string...
Step respo...	Thu Jun 09 09:47:...	Get User	36

Long Info Field

The Test Events tab contains the following columns:

- **EventID**: The ID of the event that fired.
- **Timestamp**: The time the event fired.
- **Short**: The short description of the event.
- **Long**: The long description of the event.

The Long description field can be truncated in the display. To view the full text, click the cell and its full contents will be displayed in the Long Info Field panel.

You can enter the complete or full description of the event in the Long Info Field.

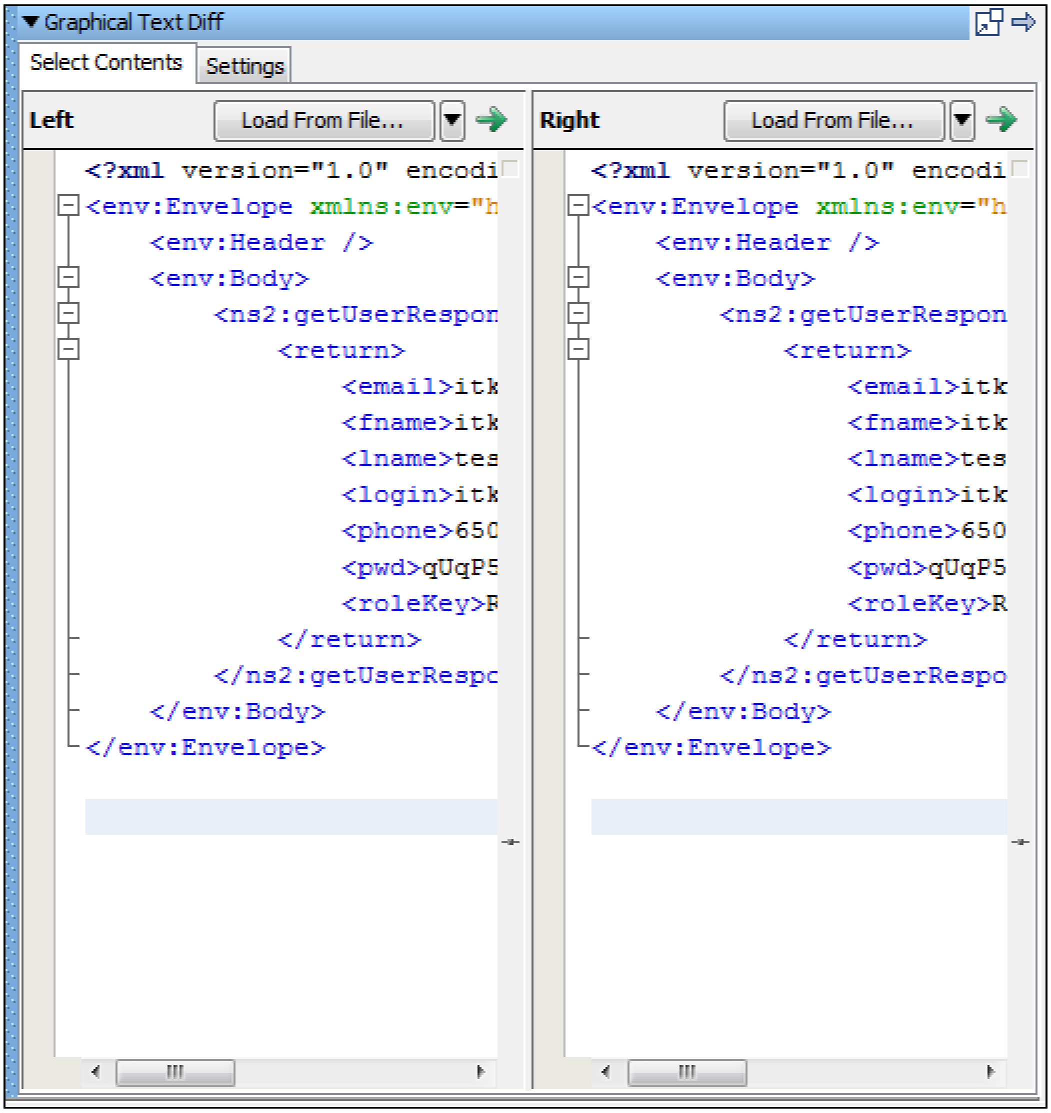
An event is generated on all assertions that are fired, and are evaluated.

Graphical Text Diff Utility

A graphical XML diff engine and visualizer are available to compare XML files and graphically display their differences.

Starting the Graphical Text Diff Utility

To start the Graphical Text Diff utility from the main menu, select Actions > Graphical Text Diff. From this panel, you can compare the contents of two XML files.



Left:

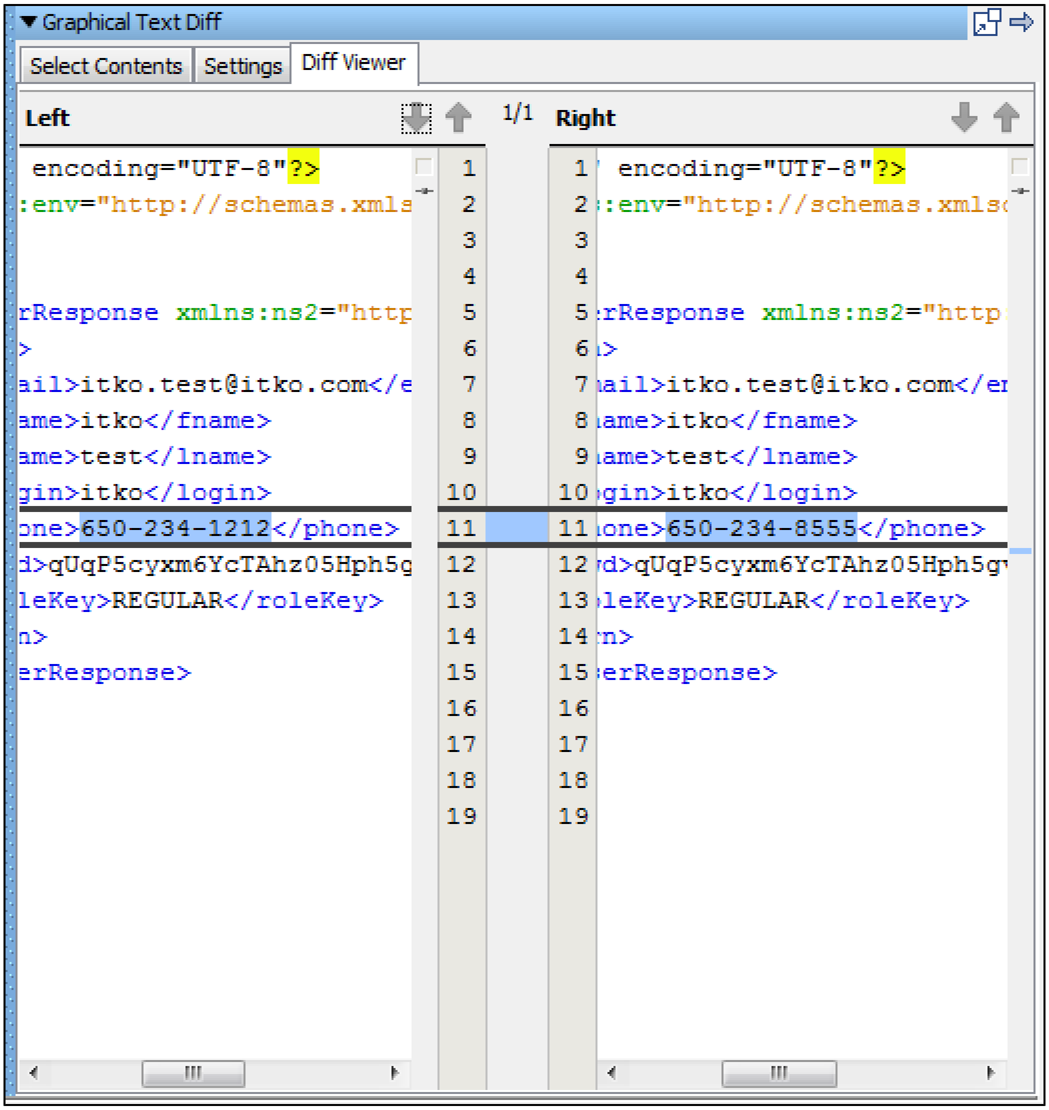
```
<?xml version="1.0" encodi
<env:Envelope xmlns:env="h
    <env:Header />
    <env:Body>
        <ns2:getUserRespon
            <return>
                <email>itk
                <fname>itk
                <lname>tes
                <login>itk
                <phone>650
                <pwd>qUqP5
                <roleKey>R
            </return>
        </ns2:getUserRespo
    </env:Body>
</env:Envelope>
```

Right:

```
<?xml version="1.0" encodi
<env:Envelope xmlns:env="h
    <env:Header />
    <env:Body>
        <ns2:getUserRespon
            <return>
                <email>itk
                <fname>itk
                <lname>tes
                <login>itk
                <phone>650
                <pwd>qUqP5
                <roleKey>R
            </return>
        </ns2:getUserRespo
    </env:Body>
</env:Envelope>
```

Click the Compare icon ➔ to start the compare.

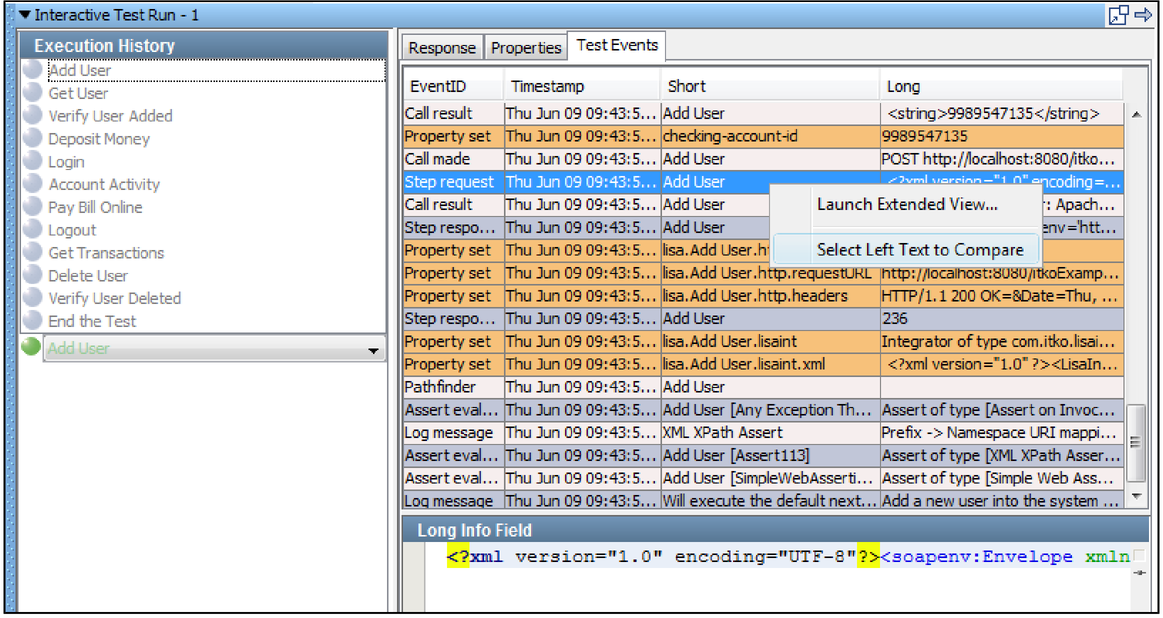
The following screen appears, which shows the actual comparison. The Diff Viewer tab shows the comparison.



```
 encoding="UTF-8"?>
:env="http://schemas.xmls


rResponse xmlns:ns2="http
>
ail>itko.test@itko.com</e
ame>itko</fname>
ame>test</lname>
gin>itko</login>
one>650-234-1212</phone>
d>qUqP5cyxm6YcTAhz05Hph5g
leKey>REGULAR</roleKey>
n>
erResponse>
```

```
' encoding="UTF-8"?>
:env="http://schemas.xmls


:rResponse xmlns:ns2="http
1>
nail>itko.test@itko.com</em
iame>itko</fname>
iame>test</lname>
gin>itko</login>
ione>650-234-8555</phone>
vd>qUqP5cyxm6YcTAhz05Hph5gv
leKey>REGULAR</roleKey>
n>
erResponse>
```

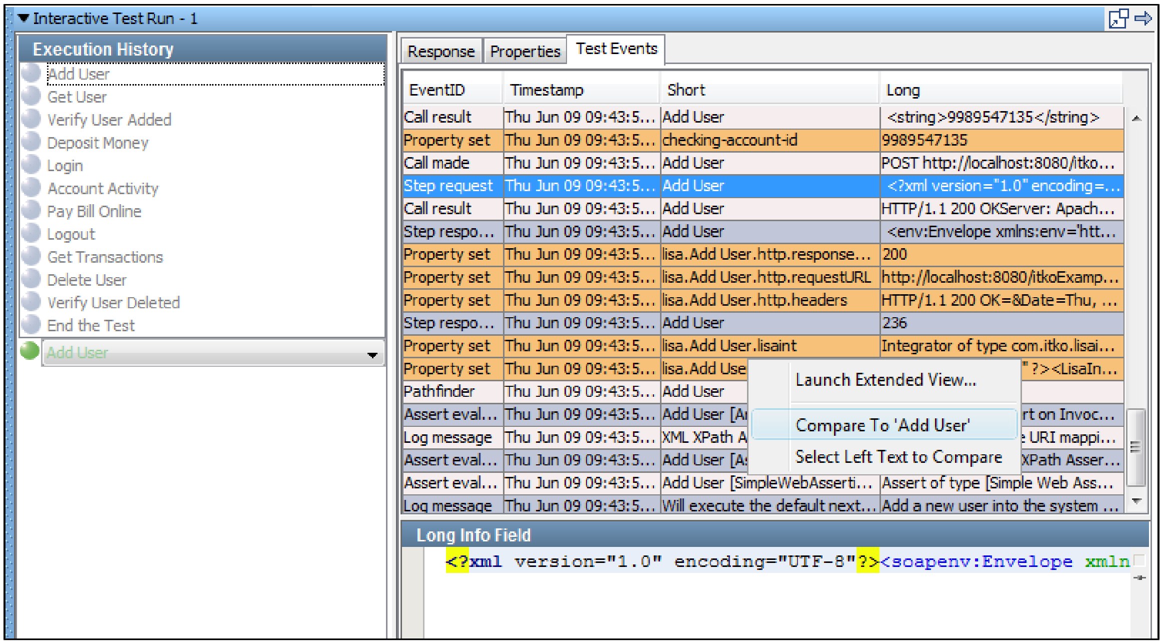
To start the Graphical Text Diff utility from the Interactive Test Run (ITR) utility, select the Test Events tab in the ITR. Right-click a table row and select Select Left Text to Compare.



EventID	Timestamp	Short	Long
Call result	Thu Jun 09 09:43:5...	Add User	<string>9989547135</string>
Property set	Thu Jun 09 09:43:5...	checking-account-id	9989547135
Call made	Thu Jun 09 09:43:5...	Add User	POST http://localhost:8080/itko...
Step request	Thu Jun 09 09:43:5...	Add User	<?xml version="1.0" encoding=...
Call result	Thu Jun 09 09:43:5...	Add User	: Apach...
Step respo...	Thu Jun 09 09:43:5...	Add User	env='htt...
Property set	Thu Jun 09 09:43:5...	lisa.Add User.h	
Property set	Thu Jun 09 09:43:5...	lisa.Add User.http.requestURL	http://localhost:8080/itkoExamp...
Property set	Thu Jun 09 09:43:5...	lisa.Add User.http.headers	HTTP/1.1 200 OK=&Date=Thu, ...
Step respo...	Thu Jun 09 09:43:5...	Add User	236
Property set	Thu Jun 09 09:43:5...	lisa.Add User.lisaint	Integrator of type com.itko.lisai...
Property set	Thu Jun 09 09:43:5...	lisa.Add User.lisaint.xml	<?xml version="1.0" ?><LisaIn...
Pathfinder	Thu Jun 09 09:43:5...	Add User	
Assert eval...	Thu Jun 09 09:43:5...	Add User [Any Exception Th...	Assert of type [Assert on Invoc...
Log message	Thu Jun 09 09:43:5...	XML XPath Assert	Prefix -> Namespace URI mappi...
Assert eval...	Thu Jun 09 09:43:5...	Add User [Assert113]	Assert of type [XML XPath Asser...
Assert eval...	Thu Jun 09 09:43:5...	Add User [SimpleWebAsserti...	Assert of type [Simple Web Ass...
Log message	Thu Jun 09 09:43:5...	Will execute the default next...	Add a new user into the system ...

```
<?xml version="1.0" encoding="UTF-8"?><soapenv:Envelope xmln
```

Right-click a table row to compare with and select Compare To.



Execution History

- Add User
- Get User
- Verify User Added
- Deposit Money
- Login
- Account Activity
- Pay Bill Online
- Logout
- Get Transactions
- Delete User
- Verify User Deleted
- End the Test

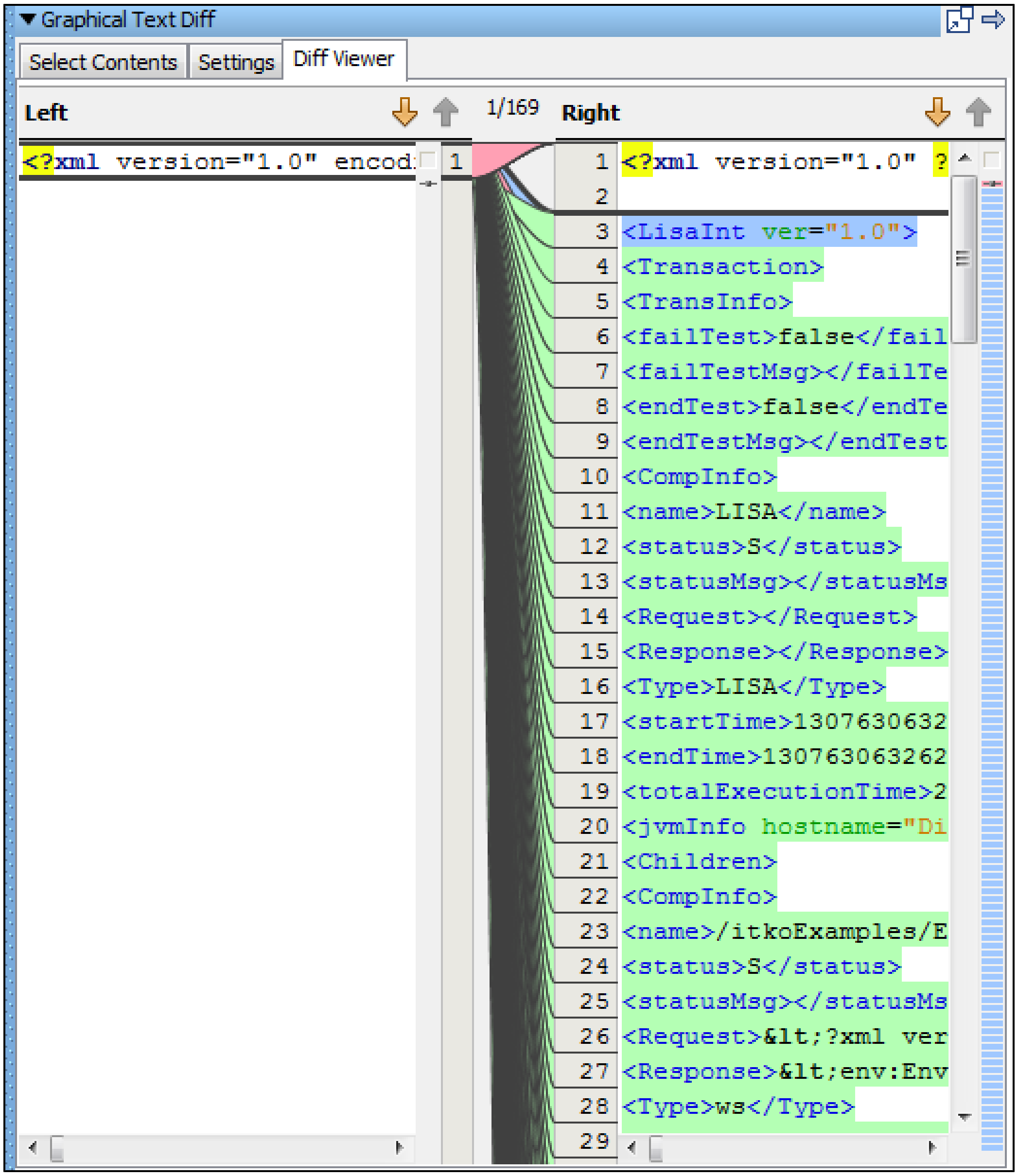
Add User

EventID	Timestamp	Short	Long
Call result	Thu Jun 09 09:43:5…	Add User	<string>9989547135</string>
Property set	Thu Jun 09 09:43:5…	checking-account-id	9989547135
Call made	Thu Jun 09 09:43:5…	Add User	POST http://localhost:8080/itko…
Step request	Thu Jun 09 09:43:5…	Add User	<?xml version="1.0" encoding=…
Call result	Thu Jun 09 09:43:5…	Add User	HTTP/1.1 200 OKServer: Apach…
Step respo…	Thu Jun 09 09:43:5…	Add User	<env:Envelope xmlns:env='htt…
Property set	Thu Jun 09 09:43:5…	lisa.Add User.http.response…	200
Property set	Thu Jun 09 09:43:5…	lisa.Add User.http.requestURL	http://localhost:8080/itkoExamp…
Property set	Thu Jun 09 09:43:5…	lisa.Add User.http.headers	HTTP/1.1 200 OK=&Date=Thu, …
Step respo…	Thu Jun 09 09:43:5…	Add User	236
Property set	Thu Jun 09 09:43:5…	lisa.Add User.lisaint	Integrator of type com.itko.lisai…
Property set	Thu Jun 09 09:43:5…	lisa.Add Use	" ?><LisaIn…
Pathfinder	Thu Jun 09 09:43:5…	Add User	
Assert eval…	Thu Jun 09 09:43:5…	Add User [A	rt on Invoc…
Log message	Thu Jun 09 09:43:5…	XML XPath A	URI mappi…
Assert eval…	Thu Jun 09 09:43:5…	Add User [A	XPath Asser…
Assert eval…	Thu Jun 09 09:43:5…	Add User [SimpleWebAsserti…	Assert of type [Simple Web Ass…
Log message	Thu Jun 09 09:43:5…	Will execute the default next…	Add a new user into the system …

Long Info Field

```
<?xml version="1.0" encoding="UTF-8"?><soapenv:Envelope xmln
```

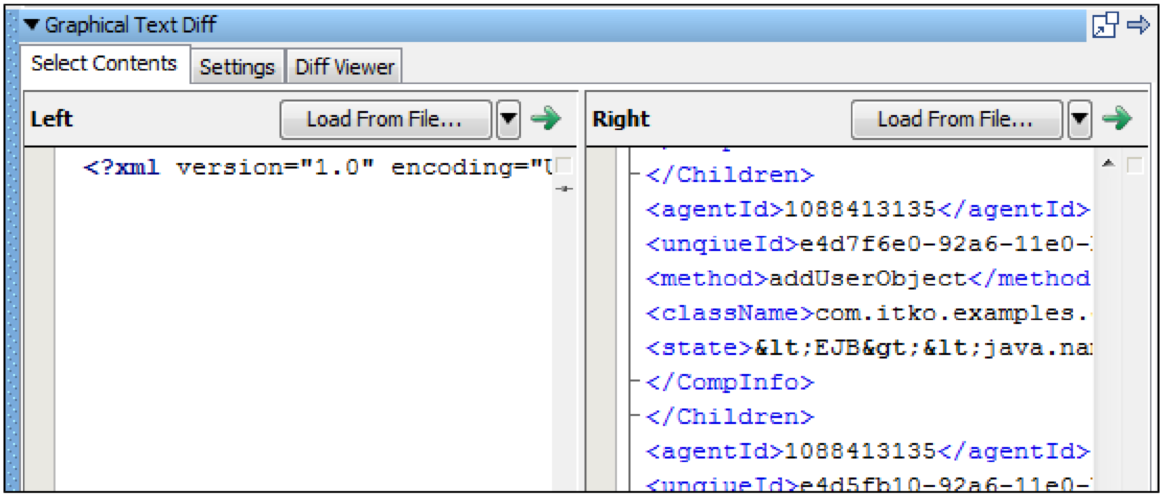
This will open the Graphical Text Diff for comparison.



```
<?xml version="1.0" encod
```

```
<?xml version="1.0" ?

<LisaInt ver="1.0">
<Transaction>
<TransInfo>
<failTest>false</fail
<failTestMsg></failTe
<endTest>false</endTe
<endTestMsg></endTest
<CompInfo>
<name>LISA</name>
<status>S</status>
<statusMsg></statusMs
<Request></Request>
<Response></Response>
<Type>LISA</Type>
<startTime>1307630632
<endTime>130763063262
<totalExecutionTime>2
<jvmInfo hostname="Di
<Children>
<CompInfo>
<name>/itkoExamples/E
<status>S</status>
<statusMsg></statusMs
<Request>&lt;?xml ver
<Response>&lt;env:Env
<Type>ws</Type>
```

While in the Graphical Text Diff utility, you can also load the contents by selecting a file. Click the Select Contents tab and click Load from File to load the file.

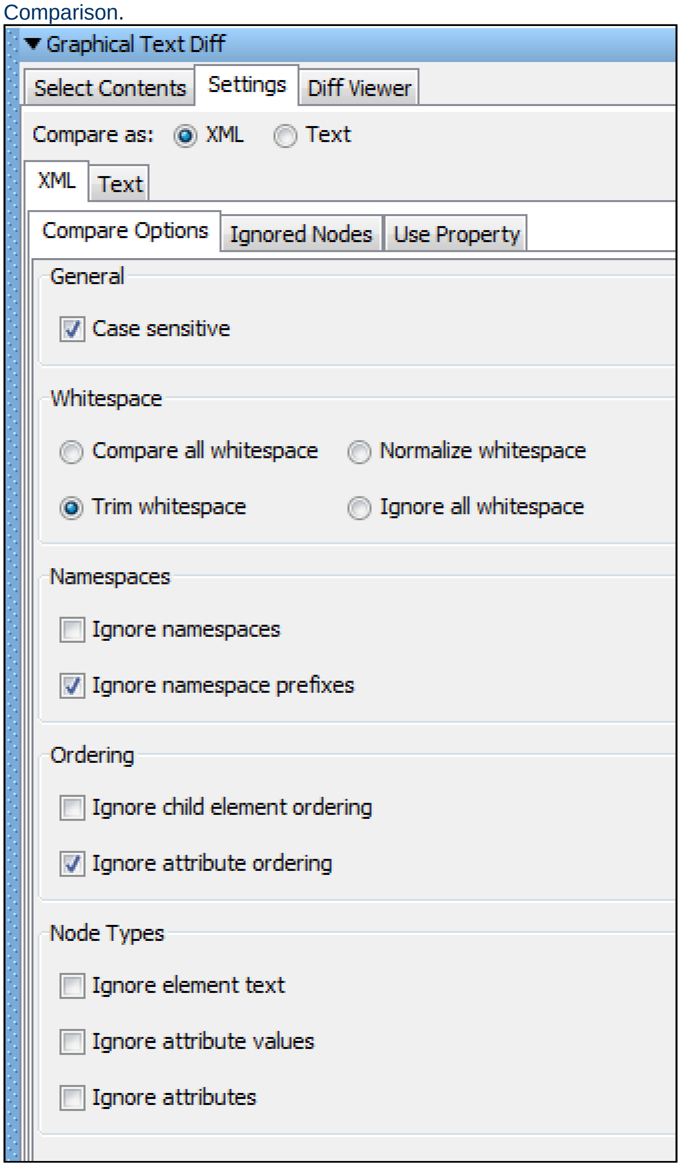


```
<?xml version="1.0" encoding="U
```

```
</Children>
<agentId>1088413135</agentId>
<unqiueId>e4d7f6e0-92a6-11e0-
<method>addUserObject</method
<className>com.itko.examples.
<state>&lt;EJB&gt;&lt;java.na
</CompInfo>
</Children>
<agentId>1088413135</agentId>
<unqiueId>e4d5fb10-92a6-11e0-
```

The results of the diff are then displayed in the global tray panel component.

You can also set the compare options in the Settings tab. For more information about the compare options, see Graphical XML Side-by-Side Comparison.



Compare as: ● XML ○ Text

General

☑ Case sensitive

Whitespace

○ Compare all whitespace ○ Normalize whitespace

● Trim whitespace ○ Ignore all whitespace

Namespaces

□ Ignore namespaces

☑ Ignore namespace prefixes

Ordering

□ Ignore child element ordering

☑ Ignore attribute ordering

Node Types

□ Ignore element text

□ Ignore attribute values

□ Ignore attributes

LISA Properties for the Graphical Text Diff Utility

The following LISA properties are used by the Graphical XML diff feature. These properties can be overridden in `local.properties` or at runtime.

- **lisa.graphical.xml.diff.engine.max.differences**: This property configures the maximum number of differences that are detected before the XML diff algorithm stops. The default value is `100`. Set to `-1` to compute all differences. When there are many differences, this causes the XML diff algorithm to finish faster.
- **lisa.graphical.xml.diff.report.max.linewidth**: This property configures the maximum width of a line of text in the XML diff results report. The default value is `80`. If the input XML has very long lines of text, this property causes the XML diff results report to consume less memory.
- **lisa.graphical.xml.diff.report.max.numberoflines**: This property configures the maximum number of context lines for a difference in the XML diff results report. The default value is `5`. If the XML being compared has different elements that are very large, this property causes the XML diff results report to consume less memory.

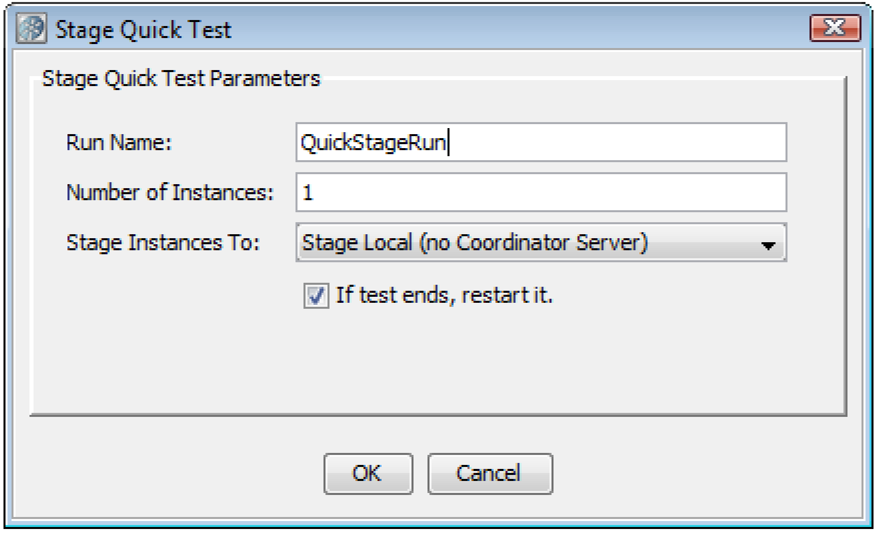
⚠ In most cases these values do not need to be changed. Changing the default values for these properties may result in the Graphical XML diff engine using more CPU and/or running out of memory.

Staging Quick Tests

LISA Workstation lets you run a test case quickly with minimal setup. A quick test runs the test that is currently in memory, rather than loading a test case document, and uses a simple pre-built staging specification with few options. The test has minimal instances so it is easy to stage/run, but lacks much of the functionality of a test staged with a staging document as in the case of a proper test case execution. A quick test lets you select and monitor events and metrics, and view a standard performance report.

To stage a quick test

1. Open a test case in LISA Workstation.
2. From the main menu, select Actions > Stage a Quick Test.
   The Stage Quick Test dialog appears.



Stage Quick Test

Stage Quick Test Parameters

Run Name: QuickStageRun

Number of Instances: 1

Stage Instances To: Stage Local (no Coordinator Server)

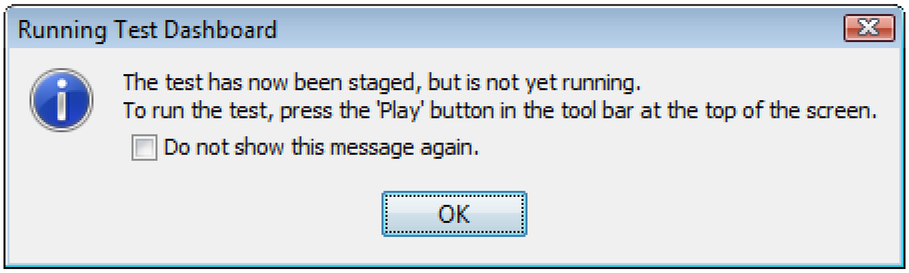
☑ If test ends, restart it.

   If you have coordinators and simulators attached to a Registry, it will show you the coordinator or simulator servers attached.
3. Specify the following parameters:
   - **Run Name:** The name of the quick test.
   - **Number of Instances:** The number of concurrent users (instances) to be used. Your license determines the maximum number of users that you can specify.
   - **Stage Instances To:** The name of the coordinator server where the test will be staged.
   - **If test ends, restart it:** Select this check box if you want to run the test continuously until you stop it manually.
4. Click OK.
   The Test Monitor window opens.

Test Monitor Window - Quick Test

When you stage a quick test, the Test Monitor window opens in LISA Workstation. The Test Monitor window for a quick test is similar to the Test Monitor window for a test case.

At first, the test is staged, but the test has not started running yet. The following image shows the message that is displayed:



Running Test Dashboard

The test has now been staged, but is not yet running.
To run the test, press the 'Play' button in the tool bar at the top of the screen.

□ Do not show this message again.

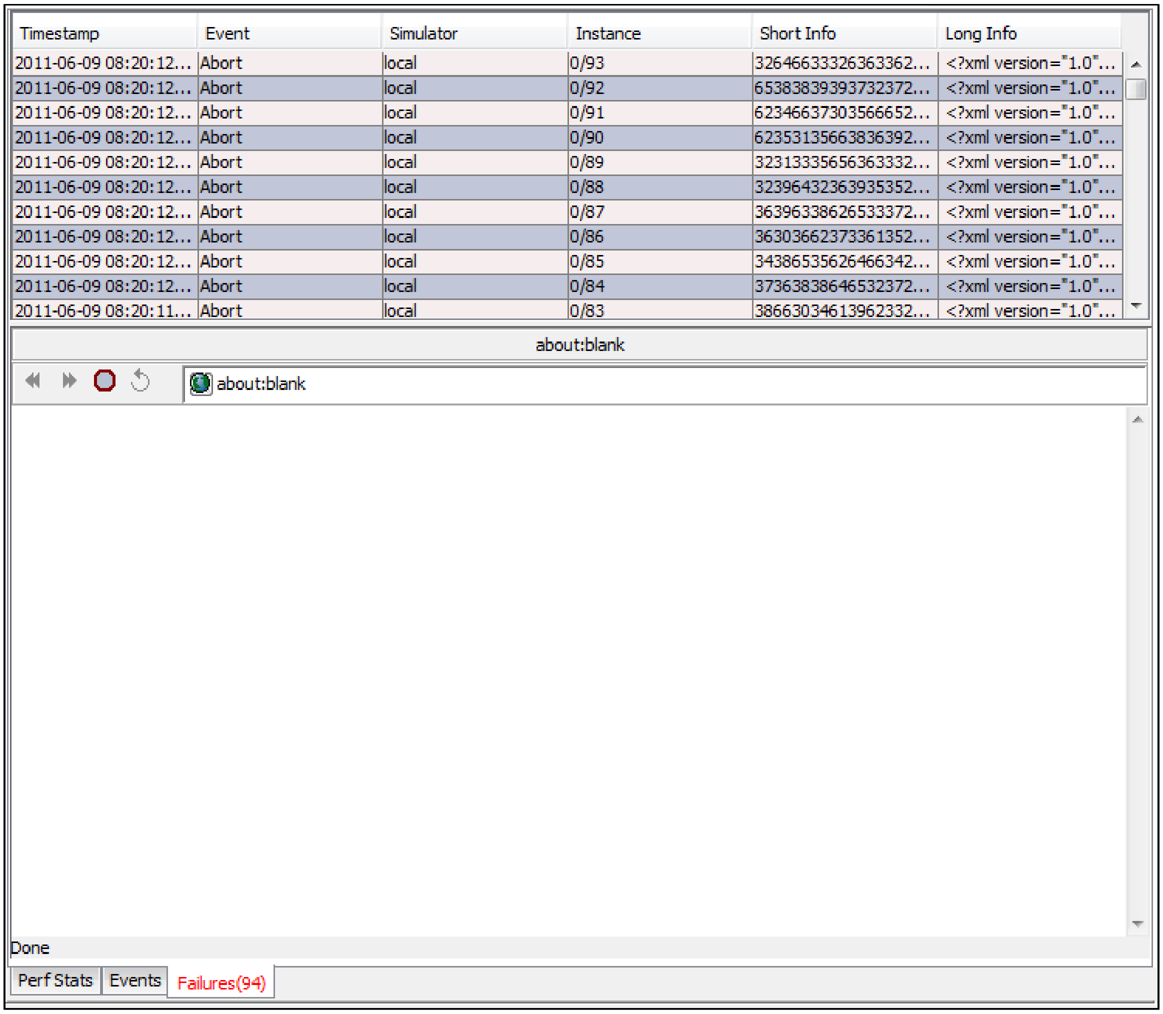
At this point you can select the metrics and events that you want to monitor during the test run.

The Test Monitor consists of two tabs:

- The Perf Stats Tab displays collected metrics as a function of time. After the test starts, this cannot be changed.
- The Events Tab displays events as they are recorded during the test run.

If you run the test case and there are failures in the test, then a third tab labeled **Failures** is displayed.

You can limit the size of the failure events by adding the **lisa.coord.failure.list.size** property to the **local.properties** file.



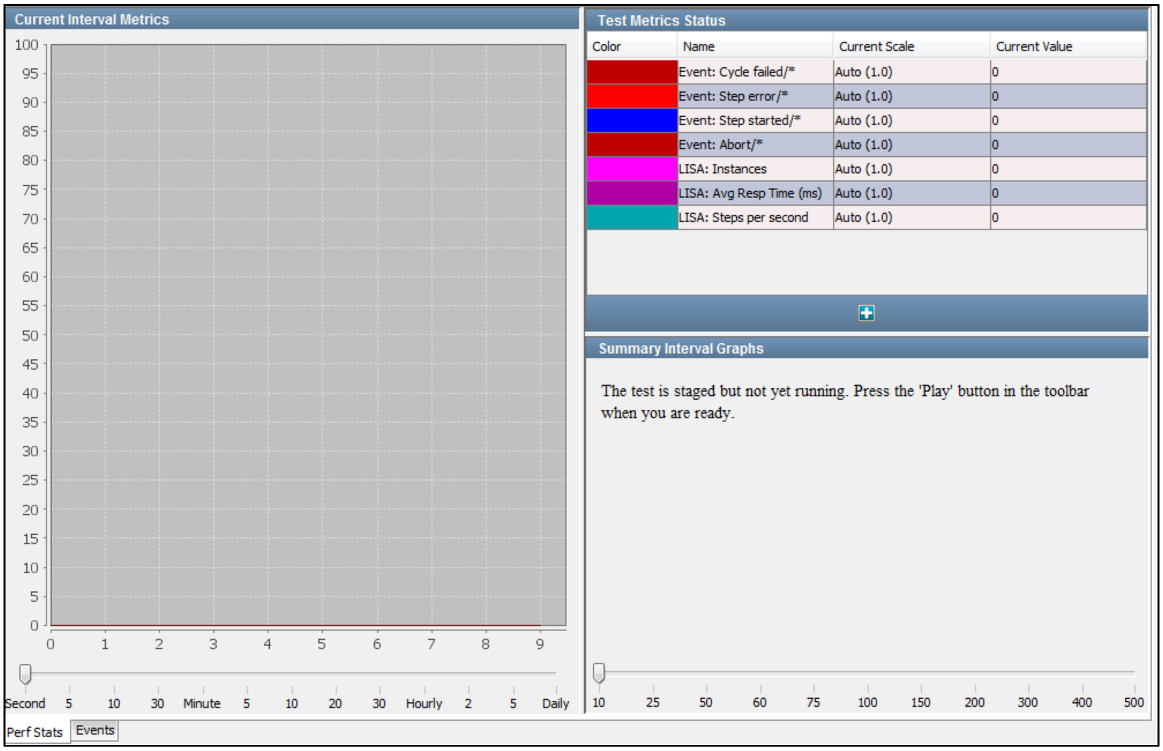
Timestamp	Event	Simulator	Instance	Short Info	Long Info
2011-06-09 08:20:12...	Abort	local	0/93	32646633326363362...	<?xml version="1.0"...
2011-06-09 08:20:12...	Abort	local	0/92	65383839393732372...	<?xml version="1.0"...
2011-06-09 08:20:12...	Abort	local	0/91	62346637303566652...	<?xml version="1.0"...
2011-06-09 08:20:12...	Abort	local	0/90	62353135663836392...	<?xml version="1.0"...
2011-06-09 08:20:12...	Abort	local	0/89	32313335656363332...	<?xml version="1.0"...
2011-06-09 08:20:12...	Abort	local	0/88	32396432363935352...	<?xml version="1.0"...
2011-06-09 08:20:12...	Abort	local	0/87	36396338626533372...	<?xml version="1.0"...
2011-06-09 08:20:12...	Abort	local	0/86	36303662373361352...	<?xml version="1.0"...
2011-06-09 08:20:12...	Abort	local	0/85	34386535626466342...	<?xml version="1.0"...
2011-06-09 08:20:12...	Abort	local	0/84	37363838646532372...	<?xml version="1.0"...
2011-06-09 08:20:11...	Abort	local	0/83	38663034613962332...	<?xml version="1.0"...

about:blank

Done

Perf Stats Tab

The **Perf Stats** tab enables you to add the metrics and events that you want to monitor during the test run.



Color	Name	Current Scale	Current Value
🟥	Event: Cycle failed/*	Auto (1.0)	0
🟥	Event: Step error/*	Auto (1.0)	0
🟦	Event: Step started/*	Auto (1.0)	0
🟥	Event: Abort/*	Auto (1.0)	0
🟪	LISA: Instances	Auto (1.0)	0
🟪	LISA: Avg Resp Time (ms)	Auto (1.0)	0
🟦	LISA: Steps per second	Auto (1.0)	0

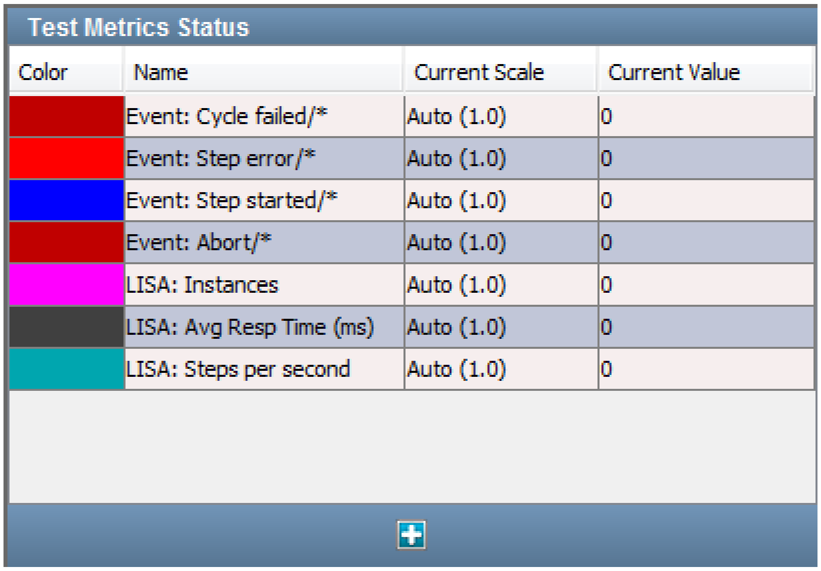
The test is staged but not yet running. Press the 'Play' button in the toolbar when you are ready.

The **Perf Stats** tab consists of the following panels:

- Test Metrics Status
- Current Interval Metrics
- Summary Interval Graphs

Test Metrics Status

The **Test Metrics Status** panel lists the current metrics being monitored.



Color	Name	Current Scale	Current Value
🟥	Event: Cycle failed/*	Auto (1.0)	0
🟥	Event: Step error/*	Auto (1.0)	0
🟦	Event: Step started/*	Auto (1.0)	0
🟥	Event: Abort/*	Auto (1.0)	0
🟪	LISA: Instances	Auto (1.0)	0
⬛	LISA: Avg Resp Time (ms)	Auto (1.0)	0
🟦	LISA: Steps per second	Auto (1.0)	0

Certain metrics are shown by default. The metrics are color coded to help you distinguish between them.

You can add additional metrics by clicking the Add icon. A pull-down menu displays all the metrics categories that can be added.

The **Test Metrics Status** panel contains the following columns:

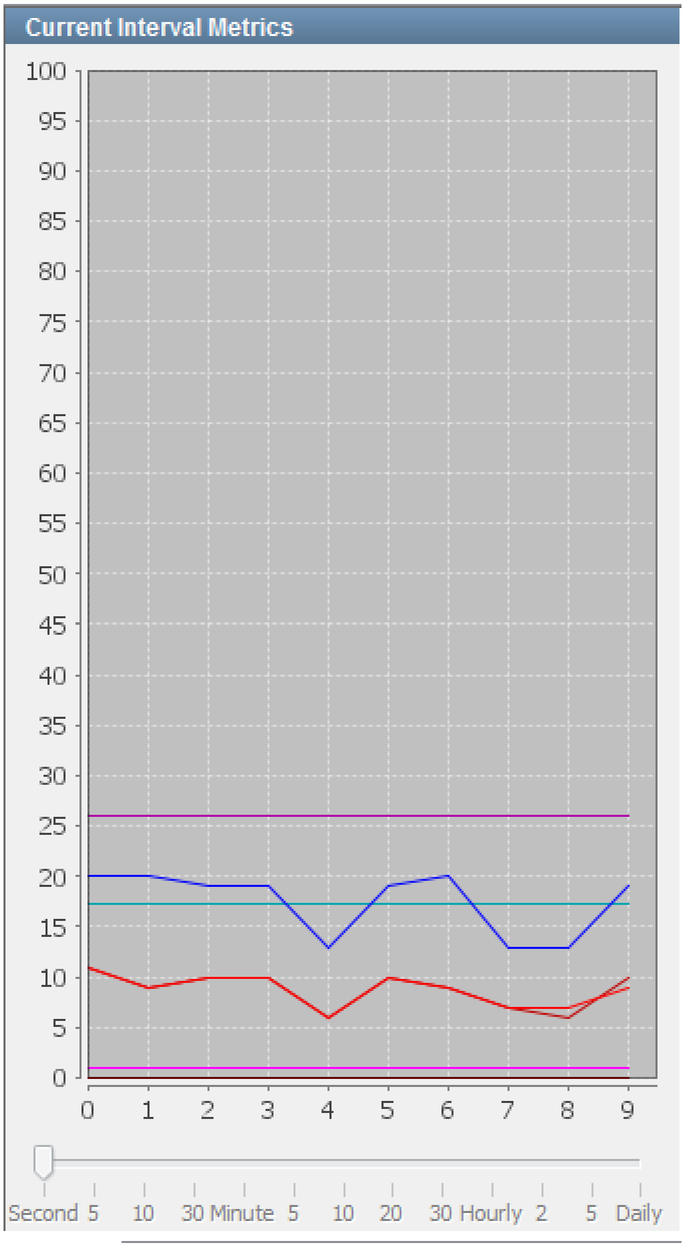
- **Color**: The color coding used on the graphs.
- **Name**: The name of the metric. If the metric has been filtered using its short name, then the short name appears after a slash in the name field. An asterisk means use just the event name for the metric.

- **Current Scale**: The vertical scale used on the graphs. You can adjust the graph scale for any metric by clicking the scale and selecting a new value from the drop-down menu.
- **Current Value**: The instantaneous value of the metric.

For detailed information about metrics, see Generating Metrics.

Current Interval Metrics

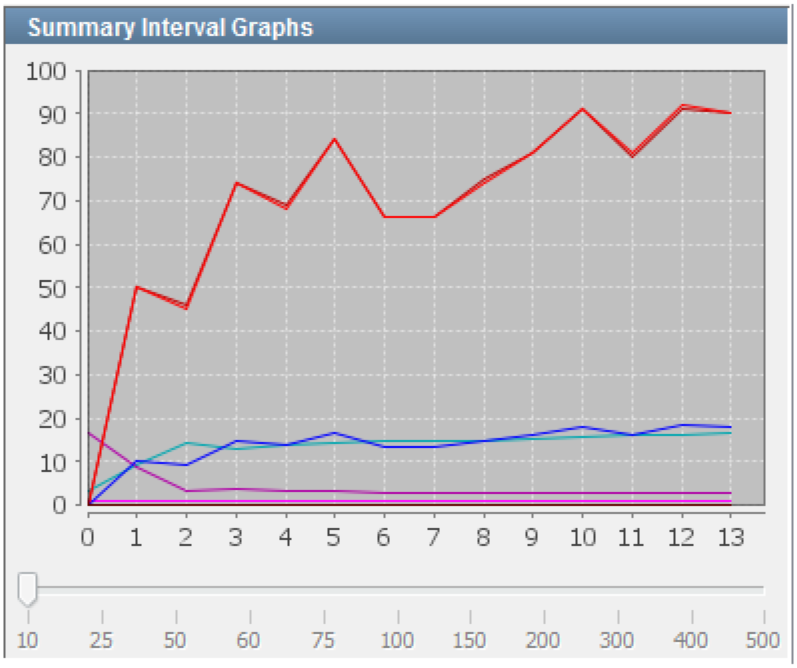
For each metric in the **Test Metrics Status** panel, the **Current Interval Metrics** panel displays the value at a specified time interval during the run.



To specify the time interval, use the slider at the bottom of the panel. You cannot adjust the value after the run has started.

Summary Interval Graphs

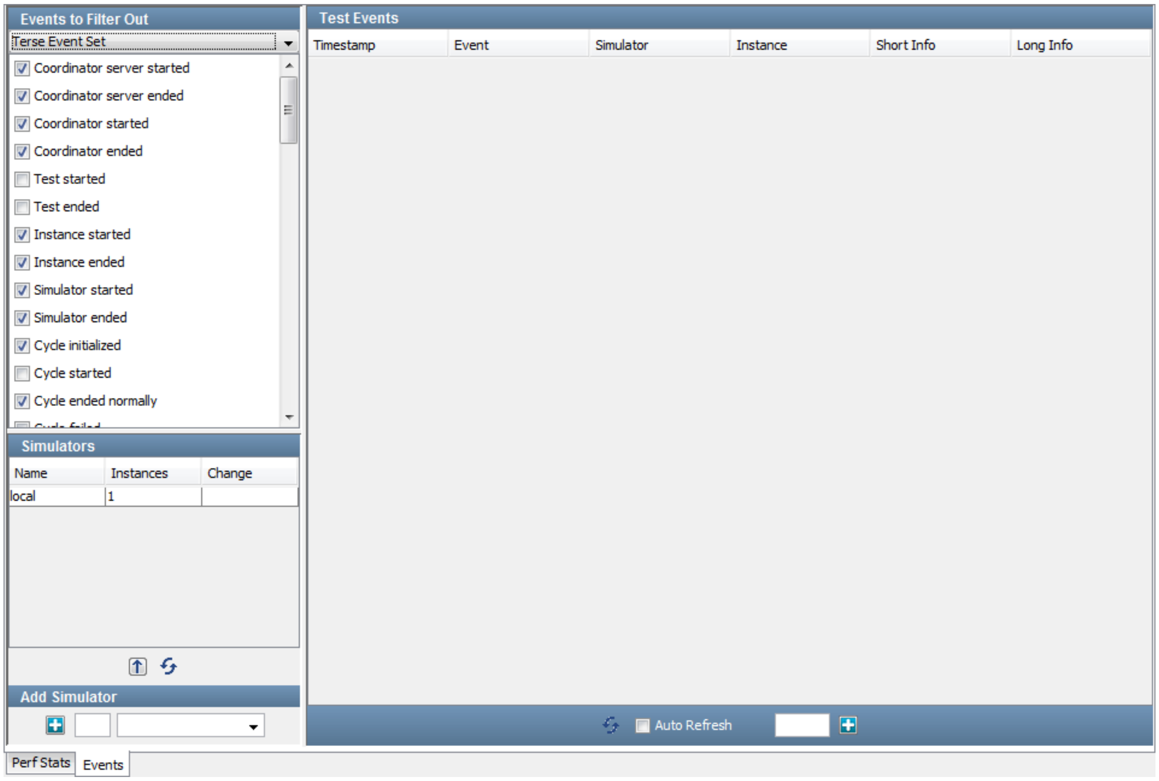
For each metric in the **Test Metrics Status** panel, the **Summary Interval Graphs** panel displays the value of each metric averaged over a specified time interval.



To specify the number of samples per interval, use the slider at the bottom of the panel. You cannot adjust the value after the run has started.

Events Tab

The **Events** tab displays the events that are generated during the run.



Events to Filter Out

Terse Event Set

- ☑ Coordinator server started
- ☑ Coordinator server ended
- ☑ Coordinator started
- ☑ Coordinator ended
- □ Test started
- □ Test ended
- ☑ Instance started
- ☑ Instance ended
- ☑ Simulator started
- ☑ Simulator ended
- ☑ Cycle initialized
- □ Cycle started
- ☑ Cycle ended normally
- □ Cycle failed

Simulators

Name	Instances	Change
local	1	

Add Simulator

Test Events

Timestamp	Event	Simulator	Instance	Short Info	Long Info

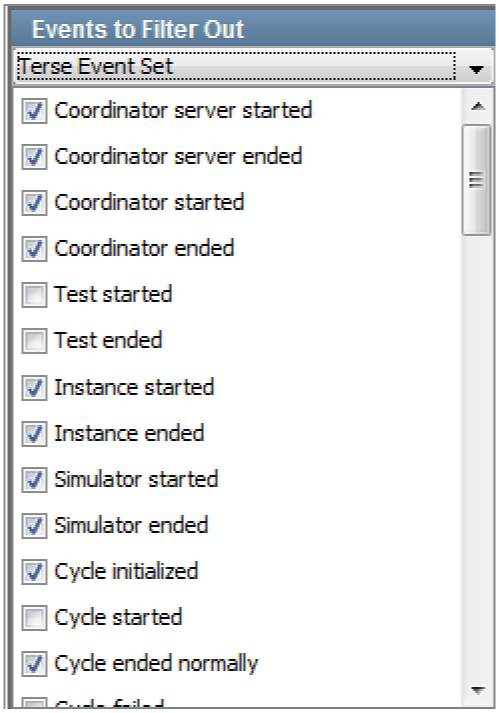
□ Auto Refresh

The **Events** tab consists of the following panels:

- Events to Filter Out
- Simulators
- Test Events

Events to Filter Out

The **Events to Filter Out** panel enables you to restrict the events that appear during the run. The panel contains a list of events. If the check box for an event is selected, then the event will *not* appear during the run.



Terse Event Set

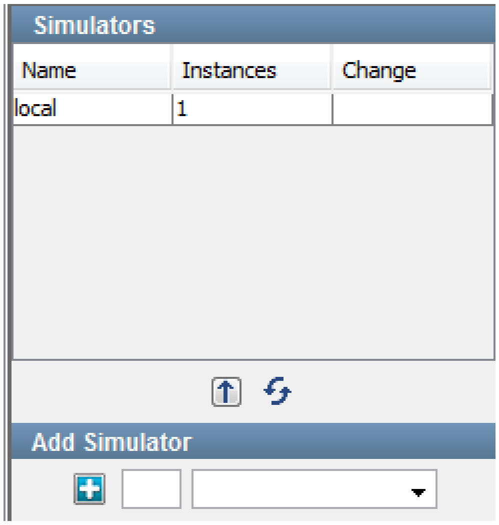
☑ Coordinator server started
☑ Coordinator server ended
☑ Coordinator started
☑ Coordinator ended
□ Test started
□ Test ended
☑ Instance started
☑ Instance ended
☑ Simulator started
☑ Simulator ended
☑ Cycle initialized
□ Cycle started
☑ Cycle ended normally
□ Cycle failed

The panel includes a drop-down menu. The options enable you to filter none of the events, include a predefined set of events, or filter all of the events. The options are:

- No Filter
- Terse Event Set
- Common Event Set
- Verbose Event Set
- Load Test Set (as of release 6.0.6)
- Custom Event Set
- Filter All Events

Simulators

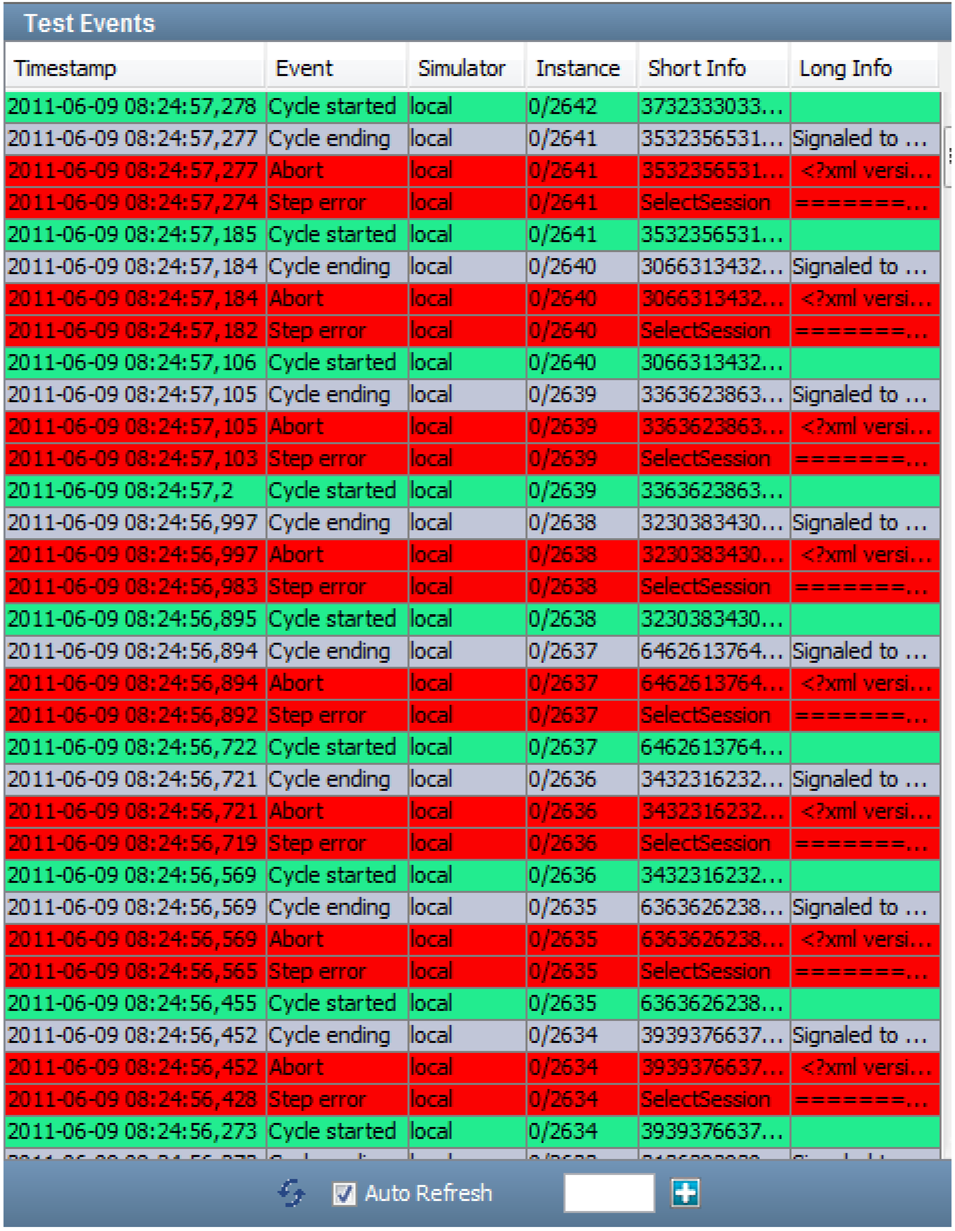
The **Simulators** panel shows the status of the Simulators in use.



Name	Instances	Change
local	1	

Test Events

The **Test Events** panel displays the events. The most recent event appears at the top. The oldest event appears at the bottom.



Test Events

Timestamp	Event	Simulator	Instance	Short Info	Long Info
2011-06-09 08:24:57,278	Cycle started	local	0/2642	3732333033…	
2011-06-09 08:24:57,277	Cycle ending	local	0/2641	3532356531…	Signaled to …
2011-06-09 08:24:57,277	Abort	local	0/2641	3532356531…	<?xml versi…
2011-06-09 08:24:57,274	Step error	local	0/2641	SelectSession	=======…
2011-06-09 08:24:57,185	Cycle started	local	0/2641	3532356531…	
2011-06-09 08:24:57,184	Cycle ending	local	0/2640	3066313432…	Signaled to …
2011-06-09 08:24:57,184	Abort	local	0/2640	3066313432…	<?xml versi…
2011-06-09 08:24:57,182	Step error	local	0/2640	SelectSession	=======…
2011-06-09 08:24:57,106	Cycle started	local	0/2640	3066313432…	
2011-06-09 08:24:57,105	Cycle ending	local	0/2639	3363623863…	Signaled to …
2011-06-09 08:24:57,105	Abort	local	0/2639	3363623863…	<?xml versi…
2011-06-09 08:24:57,103	Step error	local	0/2639	SelectSession	=======…
2011-06-09 08:24:57,2	Cycle started	local	0/2639	3363623863…	
2011-06-09 08:24:56,997	Cycle ending	local	0/2638	3230383430…	Signaled to …
2011-06-09 08:24:56,997	Abort	local	0/2638	3230383430…	<?xml versi…
2011-06-09 08:24:56,983	Step error	local	0/2638	SelectSession	=======…
2011-06-09 08:24:56,895	Cycle started	local	0/2638	3230383430…	
2011-06-09 08:24:56,894	Cycle ending	local	0/2637	6462613764…	Signaled to …
2011-06-09 08:24:56,894	Abort	local	0/2637	6462613764…	<?xml versi…
2011-06-09 08:24:56,892	Step error	local	0/2637	SelectSession	=======…
2011-06-09 08:24:56,722	Cycle started	local	0/2637	6462613764…	
2011-06-09 08:24:56,721	Cycle ending	local	0/2636	3432316232…	Signaled to …
2011-06-09 08:24:56,721	Abort	local	0/2636	3432316232…	<?xml versi…
2011-06-09 08:24:56,719	Step error	local	0/2636	SelectSession	=======…
2011-06-09 08:24:56,569	Cycle started	local	0/2636	3432316232…	
2011-06-09 08:24:56,569	Cycle ending	local	0/2635	6363626238…	Signaled to …
2011-06-09 08:24:56,569	Abort	local	0/2635	6363626238…	<?xml versi…
2011-06-09 08:24:56,565	Step error	local	0/2635	SelectSession	=======…
2011-06-09 08:24:56,455	Cycle started	local	0/2635	6363626238…	
2011-06-09 08:24:56,452	Cycle ending	local	0/2634	3939376637…	Signaled to …
2011-06-09 08:24:56,452	Abort	local	0/2634	3939376637…	<?xml versi…
2011-06-09 08:24:56,428	Step error	local	0/2634	SelectSession	=======…
2011-06-09 08:24:56,273	Cycle started	local	0/2634	3939376637…	

You can list events in real time by checking the **Auto Refresh** box at the bottom of the panel, or you can refresh the list manually by clicking the Refresh icon, with the Auto Refresh box cleared. Only those events that have not been filtered out are displayed.

For each event, the following information is displayed:

- **Timestamp**: The time of the event
- **Event**: The name of the event
- **Simulator**: The name of the Simulator in which the event was generated
- **Instance**: The instance ID and run number (separated by a forward slash)
- **Short Info**: A short piece of information about the event
- **Long Info**: A long piece of information about the event (if available)

Starting and Stopping Quick Tests

With the Test Monitor window open, you can start and stop a quick test by clicking icons on the main toolbar.

To start a quick test, click the **Play** icon on the main toolbar.

When the quick test starts, the **Play** icon changes to a **Stop** icon.

To stop a quick test, click the **Stop** icon on the main toolbar.

If you click the **Stop** icon again, you will be asked if you want to kill the test run.

When you stop a test, you are saying "do not start any new instances". When you kill a test, you are saying "stop all running instance as soon as possible".

To replay a quick test, click the **Replay** icon on the main toolbar.

To close a quick test, click the **Close** icon on the main toolbar.

Actions Menu

When you run a test and click the **Play** button to actually run it, the following options are added to the Actions menu.

- Pause Metrics Collection
- Un-pause Metrics Collection
- Save Recent Metrics to XML Document
- Save Recent Metrics to CSV Document
- Save Interval Data to CSV Document
- Save Interval Data to XML Document
- Save Recent Events to CSV Document
- Stage This Test
- Stop Test
- Restart Test (Reloads Documents)

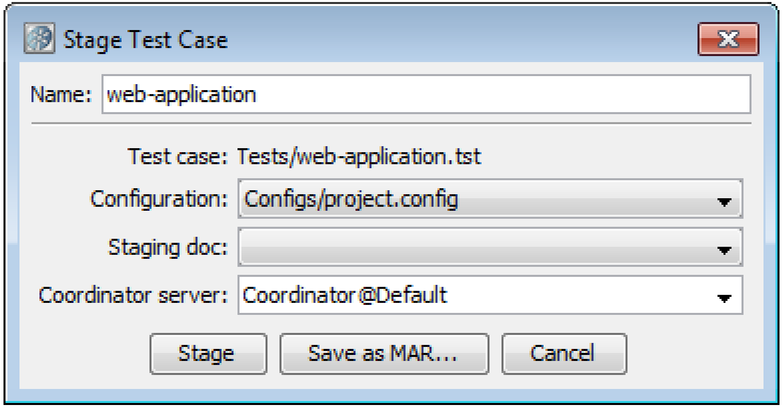
Staging Test Cases

You can run a test case from LISA Workstation by specifying the configuration, staging document, and coordinator server.

The main toolbar in LISA Workstation includes a Lab Status icon. If you stage a test case to a cloud-based lab, the icon indicates when the lab is being provisioned and then ready.

To stage a test case

1. Open a test case in LISA Workstation.
2. From the main menu, choose Actions > Stage Test.
   The Stage Test Case dialog appears.



Stage Test Case

Name: web-application

Test case: Tests/web-application.tst

Configuration: Configs/project.config

Staging doc:

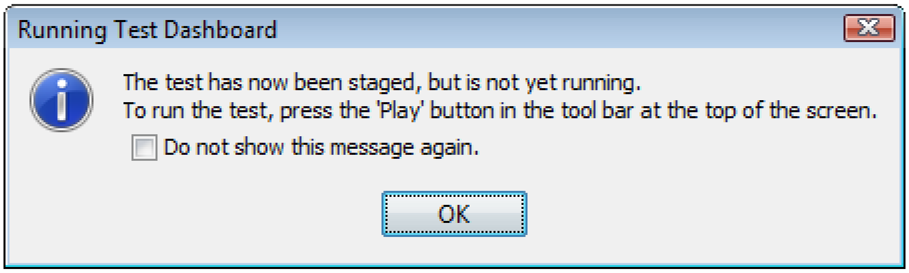
Coordinator server: Coordinator@Default

3. In the Configuration drop-down list, select the configuration. If you leave this field blank, then the default configuration is used.
4. In the Staging doc drop-down list, select the staging document.
5. In the Coordinator server drop-down list, select the coordinator server or (if available) a cloud-based lab. Coordinator server names include the associated lab. For example, **Coordinator@Default** indicates that the Default lab will be used. For cloud-based labs, the lab will be started and the test case will be staged to the coordinator in the lab.
6. Click Stage.
7. If you selected a coordinator server, the Test Monitor window opens. You can now choose the metrics and events that you want to monitor, and then start the test case.
8. If you selected a cloud-based lab, a message indicates that it will take some time to provision the lab and you will be notified when the process has completed. Click OK. When the Provisioning Complete message appears, click OK. The Test Monitor window opens. You can now choose the metrics and events that you want to monitor, and then start the test case.

Test Monitor Window - Test Case

When you stage a test case, the Test Monitor window opens in LISA Workstation. The Test Monitor window for a quick test is similar to the Test Monitor window for a test case.

At first, the test is staged, but the test has not started running yet. The following image shows the message that is displayed:



Running Test Dashboard

The test has now been staged, but is not yet running.
To run the test, press the 'Play' button in the tool bar at the top of the screen.

□ Do not show this message again.

At this point you can select the metrics and events that you want to monitor during the test run.

The Test Monitor consists of two tabs:

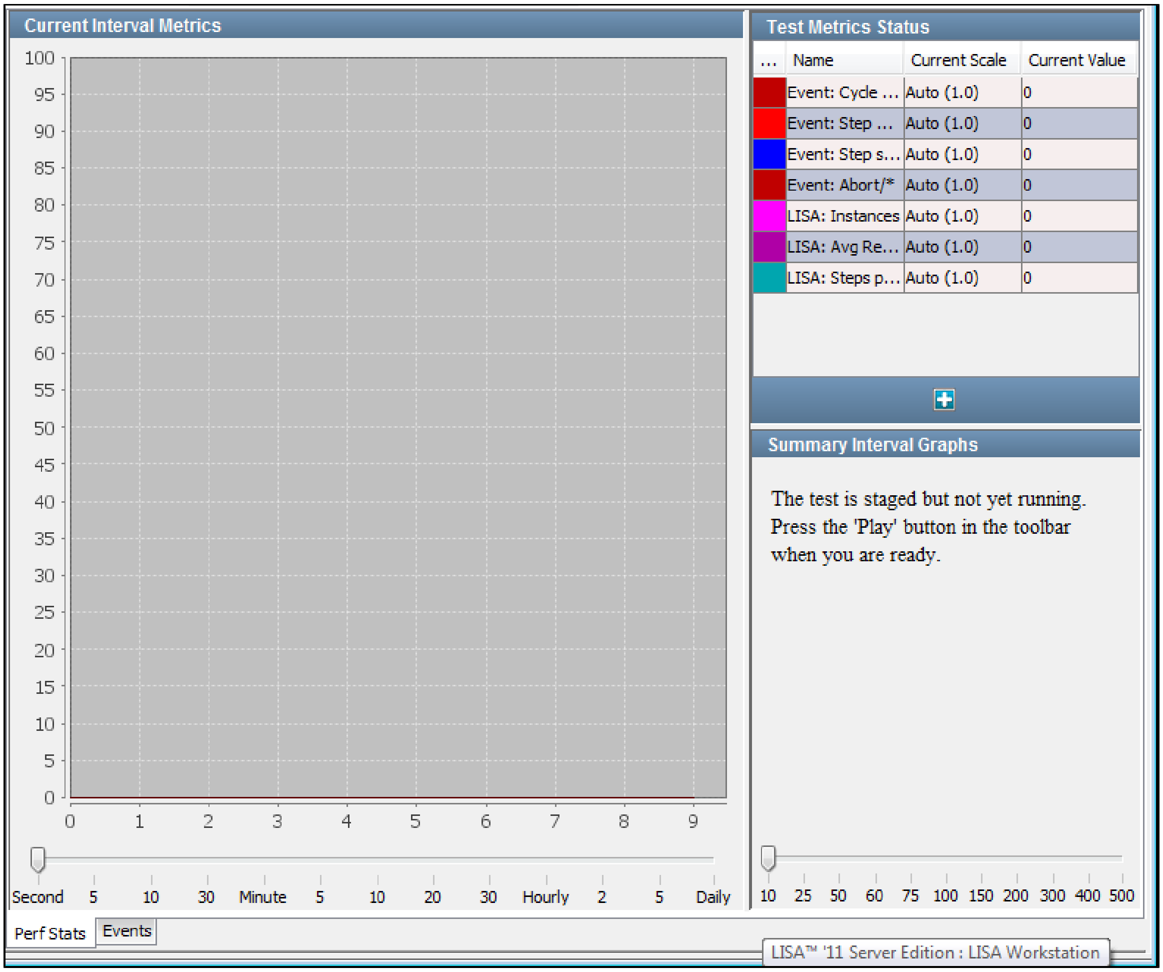
- The Perf Stats Tab lets you select metrics, and display them as a function of time.
- The Events Tab lets you select and display events as they occur during the test run.

If you run the test case and there are failures in the test, then a third tab labeled **Failures** is displayed.

You can limit the size of the failure events by adding the **lisa.coord.failure.list.size** property to the **local.properties** file.

Perf Stats Tab

The **Perf Stats** tab enables you to add the metrics and events that you want to monitor during the test run.



...	Name	Current Scale	Current Value
🟥	Event: Cycle ...	Auto (1.0)	0
🟥	Event: Step ...	Auto (1.0)	0
🟦	Event: Step s...	Auto (1.0)	0
🟥	Event: Abort/*	Auto (1.0)	0
🟪	LISA: Instances	Auto (1.0)	0
🟪	LISA: Avg Re...	Auto (1.0)	0
🟦	LISA: Steps p...	Auto (1.0)	0

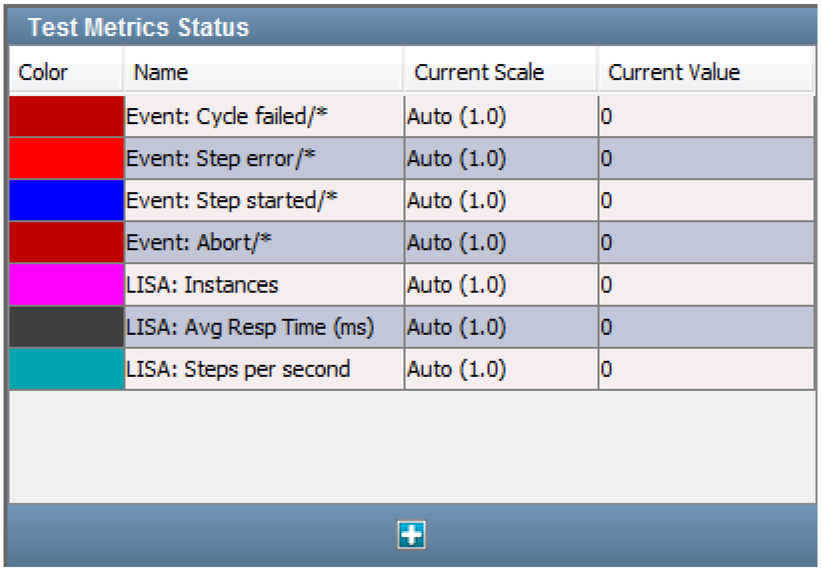
The test is staged but not yet running.
Press the 'Play' button in the toolbar
when you are ready.

The **Perf Stats** tab consists of the following panels:

- Test Metrics Status
- Current Interval Metrics
- Summary Interval Graphs

Test Metrics Status

The **Test Metrics Status** panel lists the current metrics being monitored.



Test Metrics Status

Color	Name	Current Scale	Current Value
🟥	Event: Cycle failed/*	Auto (1.0)	0
🟥	Event: Step error/*	Auto (1.0)	0
🟦	Event: Step started/*	Auto (1.0)	0
🟥	Event: Abort/*	Auto (1.0)	0
🟪	LISA: Instances	Auto (1.0)	0
⬛	LISA: Avg Resp Time (ms)	Auto (1.0)	0
🟦	LISA: Steps per second	Auto (1.0)	0

Certain metrics are shown by default. The metrics are color coded to help you distinguish between them.

You can add additional metrics by clicking the **Add** icon. A pull-down menu displays all the metrics categories that can be added.

The **Test Metrics Status** panel contains the following columns:

- **Color**: The color coding used on the graphs.
- **Name**: The name of the metric. If the metric has been filtered using its short name, then the short name appears after a slash in the name field. An asterisk means use just the event name for the metric.
- **Current Scale**: The vertical scale used on the graphs. You can adjust the graph scale for any metric by clicking the scale and selecting a new value from the drop-down menu.
- **Current Value**: The instantaneous value of the metric.

For detailed information about metrics, see Generating Metrics.

Current Interval Metrics

For each metric in the **Test Metrics Status** panel, the **Current Interval Metrics** panel displays the value at a specified time interval during the run.

To specify the time interval, use the slider at the bottom of the panel. You cannot adjust the value after the run has started.

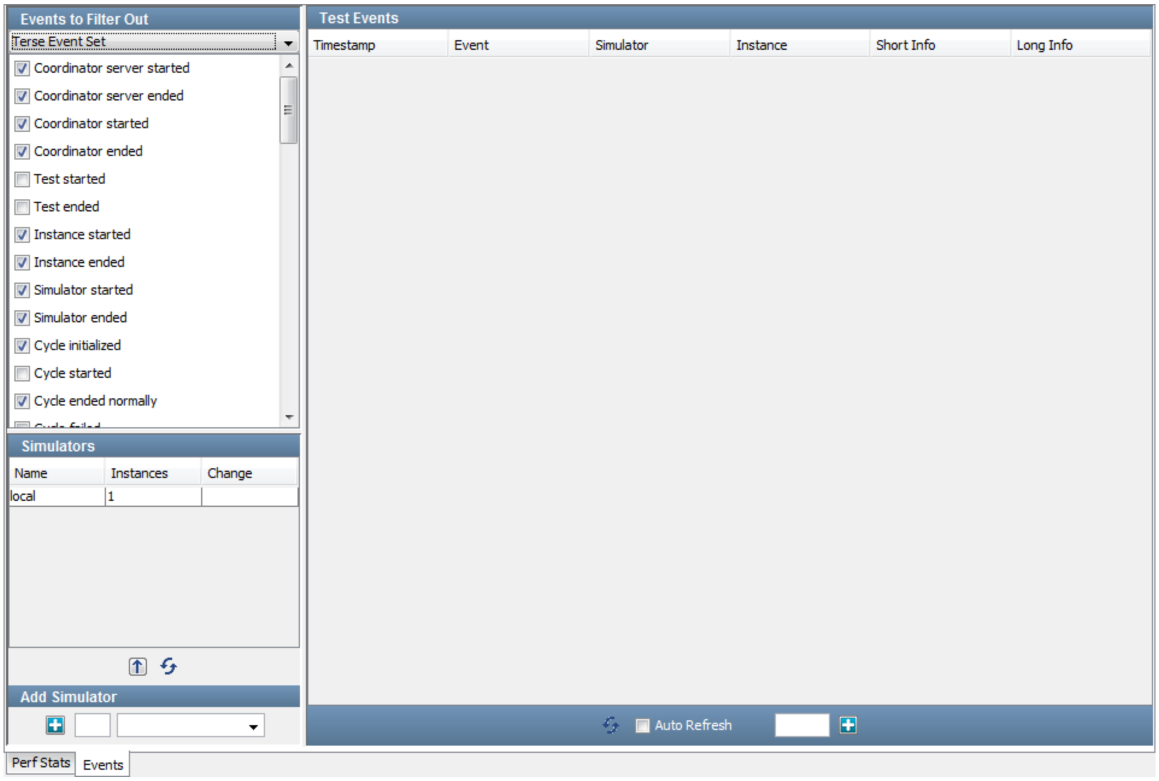
Summary Interval Graphs

For each metric in the **Test Metrics Status** panel, the **Summary Interval Graphs** panel displays the value of each metric averaged over a specified time interval.

To specify the number of samples per interval, use the slider at the bottom of the panel. You cannot adjust the value after the run has started.

Events Tab

The **Events** tab displays the events that are generated during the run.



Terse Event Set

- ☑ Coordinator server started
- ☑ Coordinator server ended
- ☑ Coordinator started
- ☑ Coordinator ended
- □ Test started
- □ Test ended
- ☑ Instance started
- ☑ Instance ended
- ☑ Simulator started
- ☑ Simulator ended
- ☑ Cycle initialized
- □ Cycle started
- ☑ Cycle ended normally
- □ Cycle failed

Name	Instances	Change
local	1	

Timestamp	Event	Simulator	Instance	Short Info	Long Info

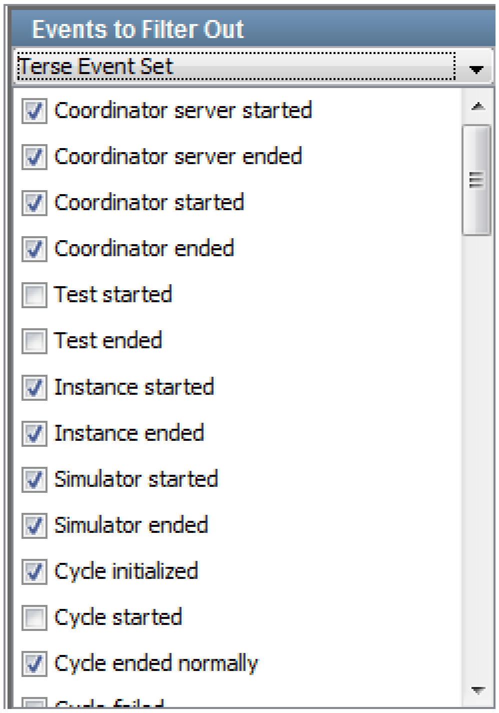
□ Auto Refresh

The **Events** tab consists of the following panels:

- Events to Filter Out
- Simulators
- Test Events

Events to Filter Out

The **Events to Filter Out** panel enables you to restrict the events that appear during the run. The panel contains a list of events. If the check box for an event is selected, then the event will *not* appear during the run.



Terse Event Set

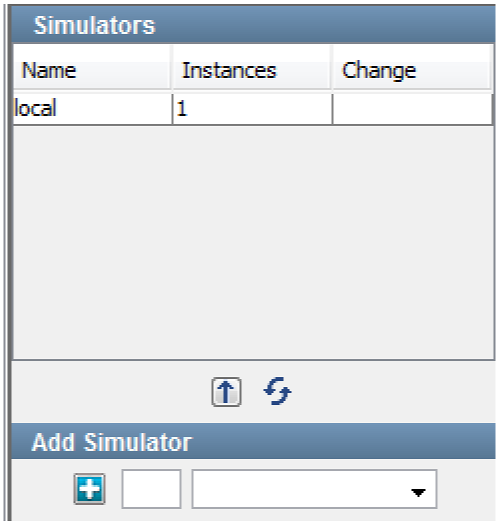
- ☑ Coordinator server started
- ☑ Coordinator server ended
- ☑ Coordinator started
- ☑ Coordinator ended
- □ Test started
- □ Test ended
- ☑ Instance started
- ☑ Instance ended
- ☑ Simulator started
- ☑ Simulator ended
- ☑ Cycle initialized
- □ Cycle started
- ☑ Cycle ended normally
- □ Cycle failed

The panel includes a drop-down menu. The options enable you to filter none of the events, include a predefined set of events, or filter all of the events. The options are:

- No Filter
- Terse Event Set
- Common Event Set
- Verbose Event Set
- Load Test Set (as of release 6.0.6)
- Custom Event Set
- Filter All Events

Simulators

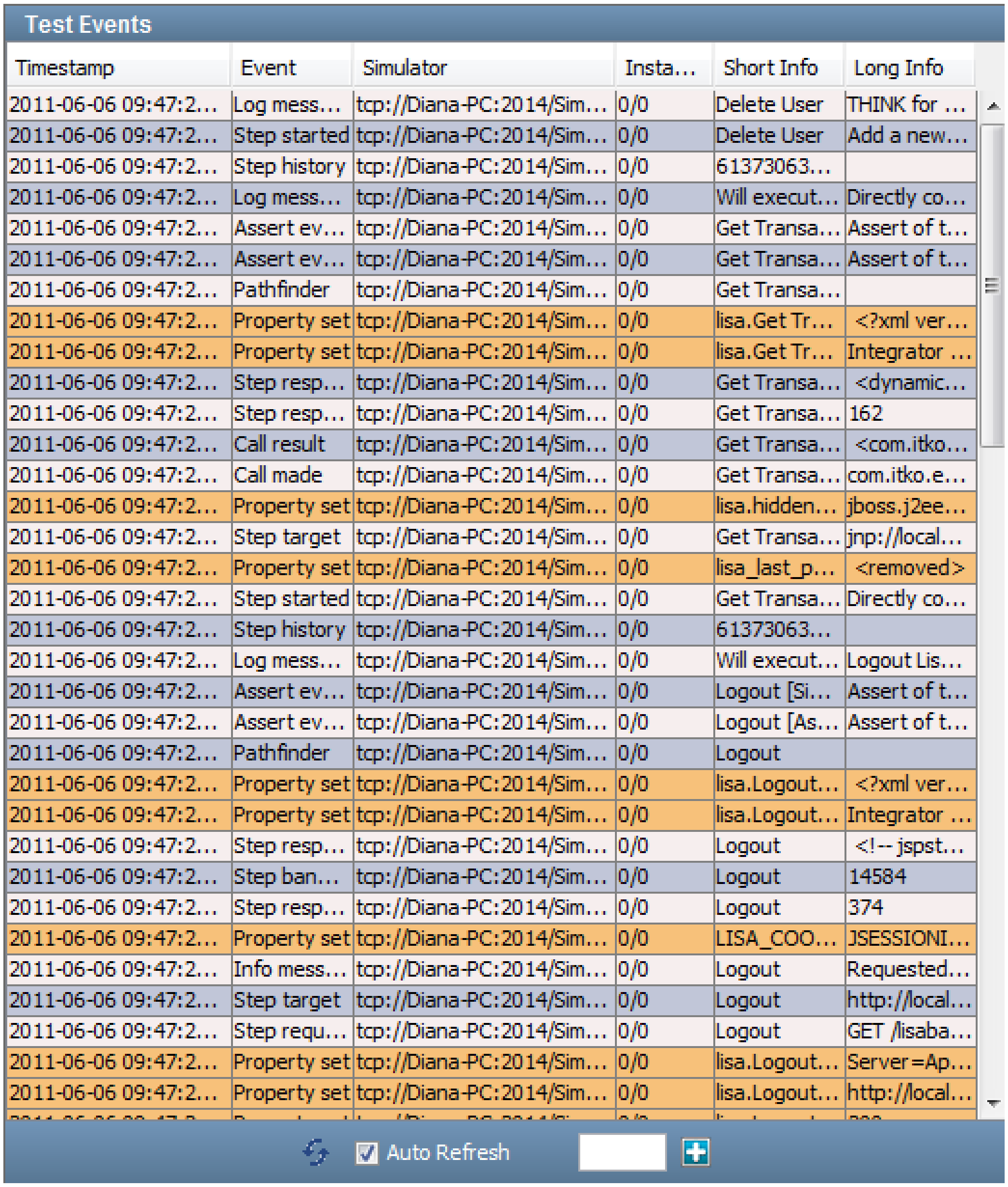
The **Simulators** panel shows the status of the simulators in use.



Name	Instances	Change
local	1	

Test Events

The **Test Events** panel displays the events. The most recent event appears at the top. The oldest event appears at the bottom.



Test Events

Timestamp	Event	Simulator	Insta…	Short Info	Long Info
2011-06-06 09:47:2…	Log mess…	tcp://Diana-PC:2014/Sim…	0/0	Delete User	THINK for …
2011-06-06 09:47:2…	Step started	tcp://Diana-PC:2014/Sim…	0/0	Delete User	Add a new…
2011-06-06 09:47:2…	Step history	tcp://Diana-PC:2014/Sim…	0/0	61373063…	
2011-06-06 09:47:2…	Log mess…	tcp://Diana-PC:2014/Sim…	0/0	Will execut…	Directly co…
2011-06-06 09:47:2…	Assert ev…	tcp://Diana-PC:2014/Sim…	0/0	Get Transa…	Assert of t…
2011-06-06 09:47:2…	Assert ev…	tcp://Diana-PC:2014/Sim…	0/0	Get Transa…	Assert of t…
2011-06-06 09:47:2…	Pathfinder	tcp://Diana-PC:2014/Sim…	0/0	Get Transa…	
2011-06-06 09:47:2…	Property set	tcp://Diana-PC:2014/Sim…	0/0	lisa.Get Tr…	<?xml ver…
2011-06-06 09:47:2…	Property set	tcp://Diana-PC:2014/Sim…	0/0	lisa.Get Tr…	Integrator …
2011-06-06 09:47:2…	Step resp…	tcp://Diana-PC:2014/Sim…	0/0	Get Transa…	<dynamic…
2011-06-06 09:47:2…	Step resp…	tcp://Diana-PC:2014/Sim…	0/0	Get Transa…	162
2011-06-06 09:47:2…	Call result	tcp://Diana-PC:2014/Sim…	0/0	Get Transa…	<com.itko…
2011-06-06 09:47:2…	Call made	tcp://Diana-PC:2014/Sim…	0/0	Get Transa…	com.itko.e…
2011-06-06 09:47:2…	Property set	tcp://Diana-PC:2014/Sim…	0/0	lisa.hidden…	jboss.j2ee…
2011-06-06 09:47:2…	Step target	tcp://Diana-PC:2014/Sim…	0/0	Get Transa…	jnp://local…
2011-06-06 09:47:2…	Property set	tcp://Diana-PC:2014/Sim…	0/0	lisa_last_p…	<removed>
2011-06-06 09:47:2…	Step started	tcp://Diana-PC:2014/Sim…	0/0	Get Transa…	Directly co…
2011-06-06 09:47:2…	Step history	tcp://Diana-PC:2014/Sim…	0/0	61373063…	
2011-06-06 09:47:2…	Log mess…	tcp://Diana-PC:2014/Sim…	0/0	Will execut…	Logout Lis…
2011-06-06 09:47:2…	Assert ev…	tcp://Diana-PC:2014/Sim…	0/0	Logout [Si…	Assert of t…
2011-06-06 09:47:2…	Assert ev…	tcp://Diana-PC:2014/Sim…	0/0	Logout [As…	Assert of t…
2011-06-06 09:47:2…	Pathfinder	tcp://Diana-PC:2014/Sim…	0/0	Logout	
2011-06-06 09:47:2…	Property set	tcp://Diana-PC:2014/Sim…	0/0	lisa.Logout…	<?xml ver…
2011-06-06 09:47:2…	Property set	tcp://Diana-PC:2014/Sim…	0/0	lisa.Logout…	Integrator …
2011-06-06 09:47:2…	Step resp…	tcp://Diana-PC:2014/Sim…	0/0	Logout	<!-- jspst…
2011-06-06 09:47:2…	Step ban…	tcp://Diana-PC:2014/Sim…	0/0	Logout	14584
2011-06-06 09:47:2…	Step resp…	tcp://Diana-PC:2014/Sim…	0/0	Logout	374
2011-06-06 09:47:2…	Property set	tcp://Diana-PC:2014/Sim…	0/0	LISA_COO…	JSESSIONI…
2011-06-06 09:47:2…	Info mess…	tcp://Diana-PC:2014/Sim…	0/0	Logout	Requested…
2011-06-06 09:47:2…	Step target	tcp://Diana-PC:2014/Sim…	0/0	Logout	http://local…
2011-06-06 09:47:2…	Step requ…	tcp://Diana-PC:2014/Sim…	0/0	Logout	GET /lisaba…
2011-06-06 09:47:2…	Property set	tcp://Diana-PC:2014/Sim…	0/0	lisa.Logout…	Server=Ap…
2011-06-06 09:47:2…	Property set	tcp://Diana-PC:2014/Sim…	0/0	lisa.Logout…	http://local…

You can list events in real time by checking the **Auto Refresh** box at the bottom of the panel, or you can refresh the list manually by clicking the Refresh icon, with the Auto Refresh box cleared. Only those events that have not been filtered out are displayed.

For each event, the following information is displayed:

- **Timestamp**: The time of the event
- **Event**: The name of the event
- **Simulator**: The name of the simulator in which the event was generated
- **Instance**: The instance ID and run number (separated by a forward slash)
- **Short Info**: A short piece of information about the event
- **Long Info**: A long piece of information about the event (if available)

Starting and Stopping Test Cases

With the Test Monitor window open, you can start and stop a test case by clicking icons on the main toolbar.

To start a test case, click the Play icon on the main toolbar:

When the test case starts, the Play icon changes to a Stop icon.

To stop a test case, click the Stop icon on the main toolbar:

If you click the Stop icon again, you will be asked if you want to kill the test run.

When you stop a test, you are saying "do not start any new instances". When you kill a test, you are saying "stop all running instances as soon as possible."

To replay a test case, click the Replay icon on the main toolbar:

To close a test case, click the Close icon on the main toolbar:

Running Test Suites

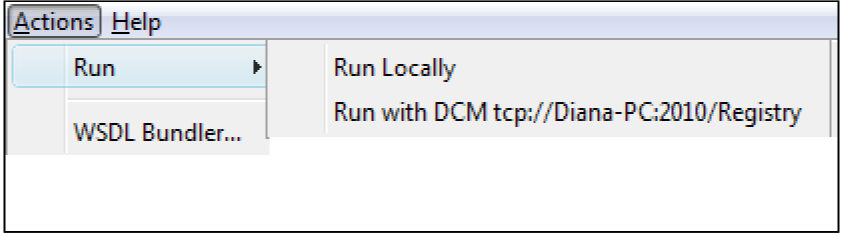
A test suite lets you group related test cases and test suites and run them as a single test. A test suite document specifies the contents of the suite, the reports to generate, and the metrics to collect. These reports and metrics relate to the suite as a whole. Each test within the suite will still produce its own reports and metrics. Each test still retains the ability to use its own staging document, configuration, and audit document. Therefore, tests in a suite can be run in a distributed environment.

When a suite is included within a suite, the individual tests in the included suite are extracted from the suite and run as individual tests in the current suite. The defaults from the included suite and the startup and teardown settings are ignored.

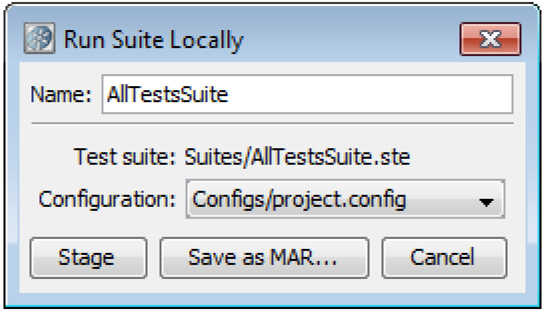
Within LISA Workstation, you can configure and stage the test suite, monitor the tests while they are running, and view the reports at the conclusion of the test.

To run a test suite

1. Open the test suite in LISA Workstation.
2. From the main menu, select Actions > Run. This action opens a menu where you select whether to run locally or to run with a specified registry.



The Run Suite Locally dialog appears.



Name: AllTestsSuite

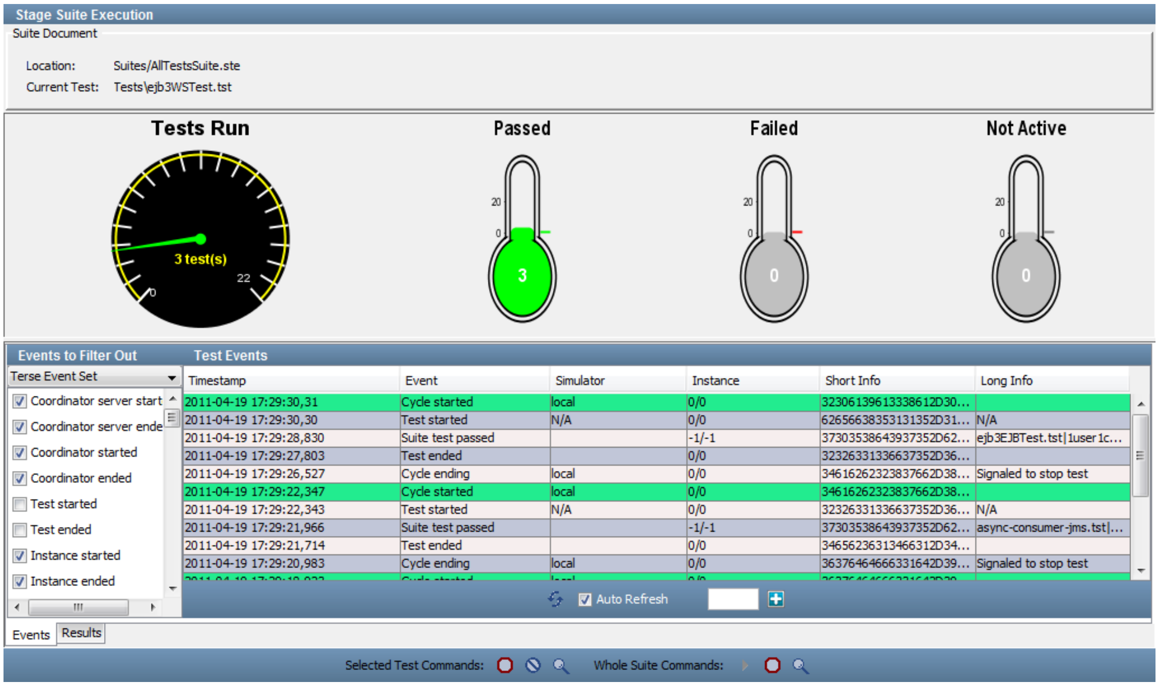
Test suite: Suites/AllTestsSuite.ste

Configuration: Configs/project.config

3. Enter the name of the test suite.
4. Select the configuration from the drop-down list.
5. Click Stage.
   The Stage Suite Execution screen is displayed and the tests start.

Stage Suite Execution

When you click Stage during the process of running a test suite in LISA Workstation, the Stage Suite Execution tab is displayed and the tests start.



Location: Suites/AllTestsSuite.ste

Current Test: Tests\ejb3WSTest.tst

Terse Event Set

- ☑ Coordinator server start
- ☑ Coordinator server ende
- ☑ Coordinator started
- ☑ Coordinator ended
- □ Test started
- □ Test ended
- ☑ Instance started
- ☑ Instance ended

Timestamp	Event	Simulator	Instance	Short Info	Long Info
2011-04-19 17:29:30,31	Cycle started	local	0/0	323061396133386I2D30...	
2011-04-19 17:29:30,30	Test started	N/A	0/0	626566383531313352D31...	N/A
2011-04-19 17:29:28,830	Suite test passed		-1/-1	373035386439373520D62...	ejb3EJBTest.tst|1user1c...
2011-04-19 17:29:27,803	Test ended		0/0	3232633133663735 2D36...	
2011-04-19 17:29:26,527	Cycle ending	local	0/0	346162623238376620D38...	Signaled to stop test
2011-04-19 17:29:22,347	Cycle started	local	0/0	346162623238376620D38...	
2011-04-19 17:29:22,343	Test started	N/A	0/0	3232633133663735 2D36...	N/A
2011-04-19 17:29:21,966	Suite test passed		-1/-1	373035386439373520D62...	async-consumer-jms.tst|...
2011-04-19 17:29:21,714	Test ended		0/0	346562363134663120D34...	
2011-04-19 17:29:20,983	Cycle ending	local	0/0	363764646663316420D39...	Signaled to stop test

☑ Auto Refresh

The top panel displays the following information:

- The location and name of the test suite document
- The name of the current test

The middle panel displays the **Tests Run** meter and three thermometers. The **Tests Run** meter shows the number of tests completed and the total number of tests. The thermometers have the following labels: **Passed**, **Failed**, and **Not Active**. The colors of the thermometers indicate that you have passed our preselected number of cases. The known behavior when you run more than 50 to 100 test cases in a suite is:

- If number of tests passed is greater than 50, then the thermometer is orange.
- If number of tests passed is greater than 75, then the thermometer is red.
- If number of tests passed is greater than 100, then the thermometer is gray.

The lower panel has the following tabs:

- **Events Tab**: Lists requested events as they occur
- **Results Tab**: Shows the status of each individual test

Stage Suite Execution - Events Tab

The Events tab lists the events that you have selected from the panel on the left of the tab.

Events to Filter Out

The **Events to Filter Out** area contains a list of the available events. Each event has a check box. Use this list to select events that you do *not* want to monitor.

One approach is to manually select the events to filter out. Another approach is to select one of the following event sets from the drop-down menu and then customize the selections as necessary:

- Terse Event Set
- Common Event Set
- Verbose Event Set
- Load Test Set (as of release 6.0.6)

You can clear all the check boxes by selecting No Filter. You can select all the check boxes by selecting Filter All Events.

Test Events

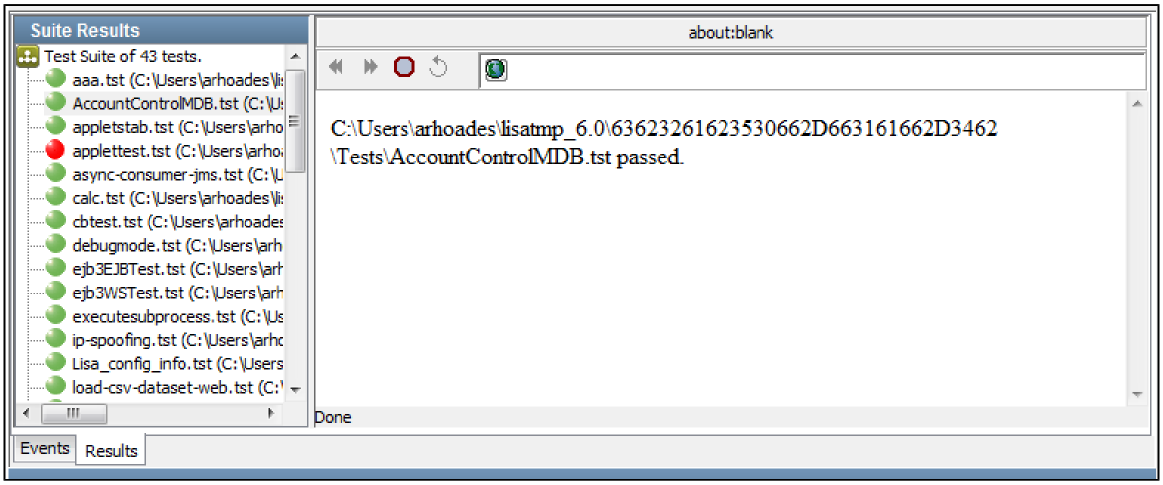
The **Test Events** area lists the events as they occur, with the most recent on the top of the list. You can list events in real time by selecting the **Auto Refresh** check box at the bottom of the panel, or you can refresh the list manually by clicking the Refresh icon, with the **Auto Refresh** check box cleared.

For each event, the following information is displayed:

- **Timestamp**: When the event was generated
- **Event**: The name of the event
- **Simulator**: The name of the simulator where the instance is running
- **Instance**: The instance ID and run number (separated by a forward slash)
- **Short Info**: The short information for the event
- **Long Info**: The long information for the event

Stage Suite Execution - Results Tab

The Results tab shows the status of the individual tests in the test suite.


Suite Results
Test Suite of 43 tests.
aaa.tst (C:\Users\arhoades\li:
AccountControlMDB.tst (C:\U:
appletstab.tst (C:\Users\arho
applettest.tst (C:\Users\arho
async-consumer-jms.tst (C:\L
calc.tst (C:\Users\arhoades\li:
cbtest.tst (C:\Users\arhoades
debugmode.tst (C:\Users\arh
ejb3EJBTest.tst (C:\Users\arh
ejb3WSTest.tst (C:\Users\arh
executesubprocess.tst (C:\Us
ip-spoofing.tst (C:\Users\arhc
Lisa_config_info.tst (C:\Users
load-csv-dataset-web.tst (C:\
about:blank
Done
Events
Results

C:\Users\arhoades\lisatmp_6.0\63623261623530662D663161662D3462 \Tests\AccountControlMDB.tst passed.

Suite Results

The **Suite Results** area displays a list of all the tests in the suite.

The tests are shown by the following icons.

- The green icon indicates that the test has completed and passed.
- The red icon indicates that the test has completed and failed.
- The blue icon indicates that the test is still running.

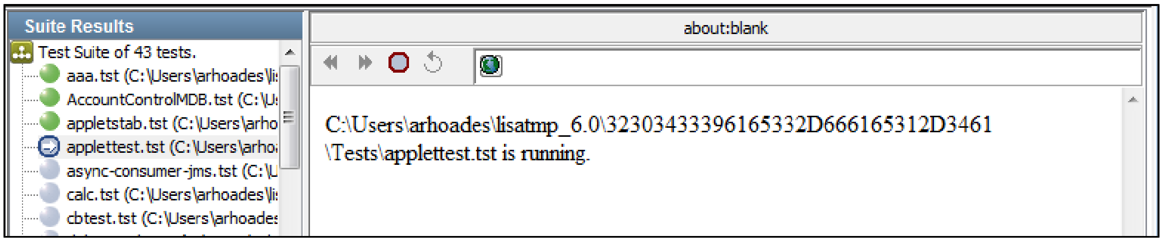
For tests that are still running, you can click the View Test icon at the bottom of the panel to display the test monitor for this test.

For completed tests, you can select the test name to see a status in the text area to the right. This is most useful to see why a given test failed.

Status Window

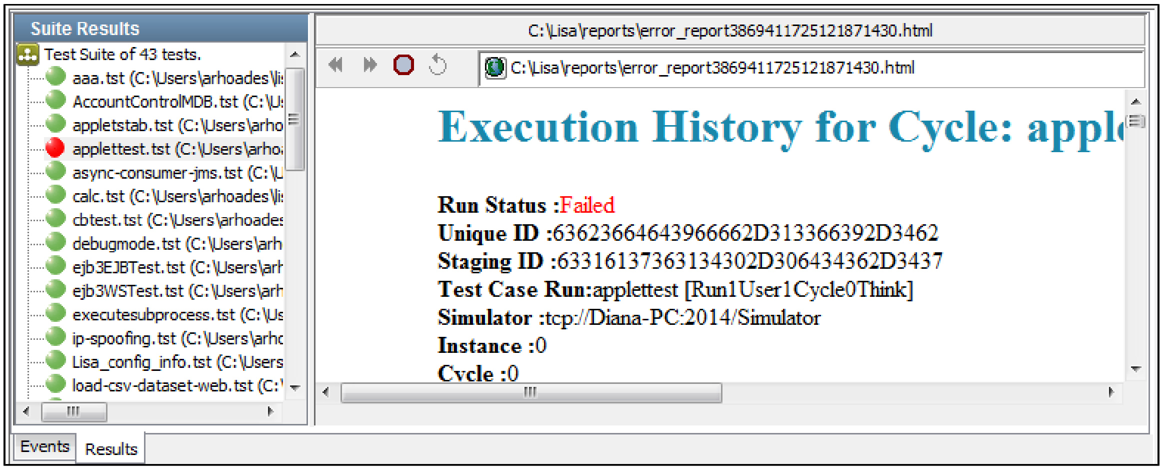
The status window displays the status of the test selected on the left.

The following test is running:


Suite Results
Test Suite of 43 tests.
aaa.tst (C:\Users\arhoades\li:
AccountControlMDB.tst (C:\U:
appletstab.tst (C:\Users\arho
applettest.tst (C:\Users\arho
async-consumer-jms.tst (C:\L
calc.tst (C:\Users\arhoades\li:
cbtest.tst (C:\Users\arhoades
about:blank

C:\Users\arhoades\lisatmp_6.0\32303433396165332D666165312D3461 \Tests\applettest.tst is running.

The following test has failed:



Suite Results

Test Suite of 43 tests.
aaa.tst (C:\Users\arhoades\li
AccountControlMDB.tst (C:\U
appletstab.tst (C:\Users\arho
applettest.tst (C:\Users\arho
async-consumer-jms.tst (C:\L
calc.tst (C:\Users\arhoades\li
cbtest.tst (C:\Users\arhoades
debugmode.tst (C:\Users\arh
ejb3EJBTest.tst (C:\Users\arh
ejb3WSTest.tst (C:\Users\arh
executesubprocess.tst (C:\Us
ip-spoofing.tst (C:\Users\arhc
Lisa_config_info.tst (C:\Users
load-csv-dataset-web.tst (C:\

C:\Lisa\reports\error_report3869411725121871430.html

C:\Lisa\reports\error_report3869411725121871430.html

Execution History for Cycle: appl

Run Status :Failed
Unique ID :63623664643966662D313366392D3462
Staging ID :63316137363134302D306434362D3437
Test Case Run:applettest [Run1User1Cycle0Think]
Simulator :tcp://Diana-PC:2014/Simulator
Instance :0
Cycle :0

A failed test status shows as much information as is available as to why the test failed.

At the bottom of the Stage Suite Execution panel is a toolbar.

The first set of icons manages individual tests in the suite. To use these icons, first select a test in the Test List section of the Results tab. The following functions are available for a particular selected test:

Stop Test: Stops the test after it reaches the next logical end/fail. It does not start a new cycle.

Kill Test: Stops the test immediately after the current step completes.

View Test: This tool will work only for currently running tests. This launches the test monitor for the selected test.

The second set of icons manage the test suite itself, or when suite is running:

Run Suite: Starts/restarts the suite test.

Close Suite: Close the Stage Suite Execution window.

View Test: Launches the test monitor for the selected test.

For more information about Test Monitor, see Using the LISA Registry Monitor.

Using the Load Test Optimizer

The Load Test Optimizer appears within the LISA Registry Monitor. It lets you run a simple load test on the system under test. A load test determines how many users the system under test supports.

In the Load Test Optimizer, the system under test is run continuously while more and more simulated users access it, until a specific target is reached, usually a predefined average response time.

The Load Test Optimizer helps determine how many users the system under test supports. An optimizer can be set on any LISA metric like average response time or a metric pulled from an external source (for example, SNMP, JMX, Windows Perfmon). It increases the number of users at a preset frequency and informs you when a preset metric threshold is achieved. For example, you can configure the optimizer to increase the number of test instances spawned by the simulator by five every ten seconds until the average response time hits two seconds.

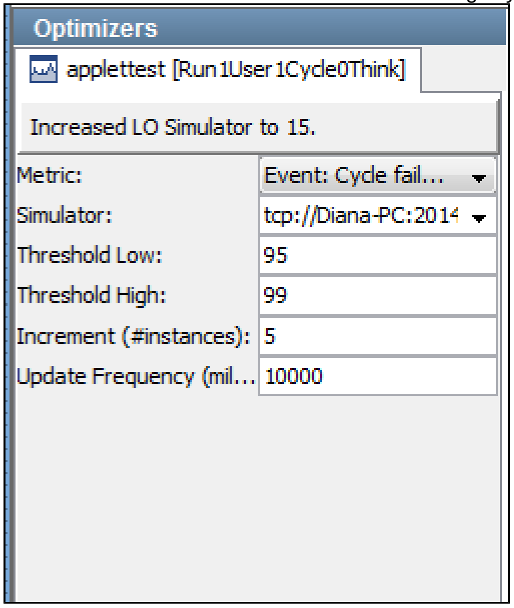
To optimize the test, it must be running.

To configure and start an optimizer

1. In the LISA Registry Monitor, select a test from the running tests list and click the Optimize Test icon .
2. The Optimizer panel opens with the name of the staging document listed at the top.
3. Configure the following parameters:

- **Metric**: The metric to use for the optimization. Select from the pull-down list.
- **Simulator**: The simulator used to spawn test instances. Select from the pull-down list.
- **Threshold Low**: The metric value at which the optimizer reports that the system will require more virtual users. Enter a numeric value.
- **Threshold High**: The metric value at which the optimizer reports that the system cannot support any more virtual users. Enter a numeric value.
- **Increment (#instances)**: The number of additional virtual users to add to the system at the update frequency. Enter a numeric value.
- **Update Frequency (millis)**: the number of milliseconds between increments

4. Click the Start Optimizer icon .
   As the optimizer increases the number of virtual users, it displays the number of virtual users on both the Optimizers section and in the Instances column of the Tests tab in the LISA Registry Monitor.



Optimizers

applettest [Run1User1Cycle0Think]

Increased LO Simulator to 15.

Metric: Event: Cycle fail...

Simulator: tcp://Diana-PC:2014

Threshold Low: 95

Threshold High: 99

Increment (#instances): 5

Update Frequency (mil... 10000

5. As the optimizer executes, you can change the parameters. To update the optimizer with the changed information, click the Update icon .
6. To close the optimizer, click the Close icon .

Test Runner

The Test Runner command-line utility is a "headless" version of LISA Workstation with the same functionality but no user interface. That is, it can be run as a stand-alone application.

Test Runner lets you run tests as batch applications. You give up the opportunity to monitor tests in real time, but you still have the capability to request reports for later viewing.

On Windows, Test Runner is available in the **LISA_HOME/bin** directory as a Windows executable, **TestRunner.exe**.

On UNIX, Test Runner is available as a UNIX executable, **TestRunner**, and a UNIX script, **TestRunner.sh**.

This lets you incorporate LISA tests into a continuous build workflow, or use it with JUnit to run standard JUnit tests in Ant or some other build tool.

Test Runner provides the following options:

```
TestRunner [-h] [[-r StagingDocument] [-t TestCaseDocument] [-cs CoordinatorServerName]] | [-s
TestSuiteDocument] [-m TestRegistryName] [-a]
```

As a Java class, you can run it as follows:

```
java com.itko.lisa.coordinator.TestRunner [-h] [[-r StagingDocument] [-t TestCaseDocument] [-cs
CoordinatorServerName]] | [-s TestSuiteDocument] [-m TestRegistryName] [-a]
```

To display help information for Test Runner, use the **-h** or **--help** option.

```
TestRunner -h
```

To display the version number, use the **--version** option.

As Part of an Automated Build

LISA test cases can be incorporated into an automatic build and test process. LISA provides the additional software required to use Java Ant, and Java JUnit, and an example Ant build script.

LISA test cases will be run and reported as native JUnit tests.

Test Runner can be used in standard JUnit tests using a custom LISA Java class. For more information, see Running LISA with Ant and JUnit.

Running a Model Archive (MAR) with Test Runner

To run a Model Archive (MAR) with Test Runner, specify the following option:

- **-mar** or **--mar** name of the MAR file

Example

The following example runs a MAR file called **test1.mar**.

```
TestRunner -mar C:\test1.mar
```

Running a Test Case with Test Runner

To run a single test case with Test Runner, specify the following options:

- **-t** or **--testCase** name of the test case document
- **-r** or **--stagingDoc** name of the staging document

If you want to stage remotely, then also specify the following options:

- **-cs** or **--coordinatorService** name of the Coordinator Server
- **-m** or **--testRegistry** name of the LISA Registry

For additional options that you can use, see Other Test Runner Options.

Example

The following example runs the multi-tier-combo test case located in the **examples** project.

```
TestRunner -t ../examples/Tests/multi-tier-combo.tst -r ../examples/StagingDocs/Run1User1Cycle.stg
-a
```

Running a Suite with Test Runner

To run a suite with Test Runner, specify the following option:

- **-s** or **--testSuite** name of the suite document

No auditing is performed.

If you want to stage remotely, also specify the following option:

- **-m** or **--testRegistry** name of the LISA Registry

To stage a suite that is within a project and reference project test cases with a property name `LISA_PROJ_ROOT`, you must have a copy of the project available somewhere under the folder defined by **lisa.projects.home** property on the server side (primarily coordinator and all simulators). This property defaults to `LISA_HOME`, and can be set to any value in local.properties file. A template is available in **_local.properties**.

The projects do not have to be siblings under this root. Another important restriction for remote staging project documents is to have a unique name for all projects. This is the case not only with project suites, but also remotely staging project test cases that refer to other assets (like data sets) within the project.

For additional options that you can use, see Other Test Runner Options.

Example

The following example runs the AllTestsSuite suite located in the examples project.

```
TestRunner -s ../examples/Suites/AllTestsSuite.ste
```

The following example assumes that the registry is running on another computer.

```
TestRunner -s ../examples/Suites/AllTestsSuite.ste -m somemachine/Registry
```

Other Test Runner Options

This topic describes additional options that you can use while running a Model Archive (MAR), test case, or suite with Test Runner.

You can use LISA properties when specifying file names. However, in this context only system properties and properties defined in your property files (**lisa.properties**, **local.properties** and **site.properties**) will work.

When the test(s) have been completed, you can view your reports in the standard way using LISA Workstation.

Automatically Starting the Test

The **-a** or **--autoStart** option automatically starts the test, so you do not need to press Enter after the test has been staged.

Specifying the Configuration

The **-config** or **--configFile** option lets you specify the configuration to use for a test run.

Test Runner does not understand LISA projects, so you must provide the fully qualified path to the configuration file. For example:

```
-config /path/to/lisa/home/examples/Configs/examples.itko.com.config
```

Generating an HTML Report

The **-html** or **--htmlReport** option enables you to have Test Runner produce an HTML summary report for a test case. This option is not supported for suites.

The parameter after this option must be a fully qualified filename. For example:

```
-html /some/directory/MyReport.html
```

Changing the Update Interval

The **-u** or **--update** option enables you to change the update interval from the default value of 5 seconds. This interval refers to how often the Test Runner writes a status message to the log file.

```
-u 10
```

Multiple Test Runner Instances

You can stage multiple instances of Test Runner from a single workstation to a LISA Server.

The following things are required:

- You need 512 MB for each instance.
- Your license must support launching multiple instances of Test Runner.

Test Runner Log File

Logging output is written to the **trunner.log** file. For information about the location of this file, see Log Files.

The logging level used is the same as that set in the **LISA_HOME/logging.properties** file.

To change the logging level, edit the **log4j.rootCategory** property in the **logging.properties** file from:

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

to

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

LISA Invoke

LISA Invoke is a REST-like web application that lets you run test cases and suites with a URL. The response consists of an XML document.

You can invoke synchronously or asynchronously.

You can display the online help by running http://*hostname*:1505/lisa-invoke/, where *hostname* is the computer where the registry is running.

Running Test Cases with LISA Invoke

The syntax for running test cases with LISA Invoke is:

```
/lisa-invoke/runTest?testCasePath=testCasePath&stagingDocPath=stagingDocPath&[configPath=configPath]&[a
```

The following table describes the parameters.

Parameter	Description
testCasePath	The path to the test case that you want to invoke.

stagingDocPath	The path to the staging document. If not provided, a default staging document is created.
configPath	The path to the configuration. If not provided, the project's project.config file is used.
async	If true, then the response includes a callback key. If not provided, the parameter is set to false.
coordName	The path to the coordinator. If not provided, the default coordinator name is used.

Running Suites with LISA Invoke

The syntax for running suites with LISA Invoke is:

```
/lisa-invoke/runSuite?suitePath=[suitePath]&configPath=[configPath]&[async=true|false]
```

The following table describes the parameters.

Parameter	Description
suitePath	The path to the suite that you want to invoke.
configPath	The path to the configuration.
async	If true, then the response includes a callback key. If not provided, the parameter is set to false.

Running Model Archives (MARs) with LISA Invoke

The syntax for running Model Archives (MARs) with LISA Invoke is:

```
/lisa-invoke/runMar?marOrMariPath=[marOrMariPath]&[async=true|false]
```

The following table describes the parameters.

Parameter	Description
marOrMariPath	The path to the MAR or MAR info file that you want to invoke.
async	If true, then the response includes a callback key. If not provided, the parameter is set to false.

Invoking the Callback Service

To use the callback service, you must have performed an asynchronous invocation of a test case or suite. The XML response contains the callback key.

The syntax for invoking the callback service is:

```
/lisa-invoke/callback?testOrSuitePath=[testOrSuitePath]&callbackKey=[callbackKey]&command=[status|kill|
```

The following table describes the parameters.

Parameter	Description
testOrSuitePath	The path to the test case or suite.
callbackKey	The callback key that was included in the response to the asynchronous invocation.
command	The action that you want to perform: status, kill, or stop.

LISA Invoke Responses

This section describes the elements that can appear in the XML document returned by LISA Invoke.

The method element indicates what type of run was performed.

The status element contains the status of the run: OK or ERROR.

The result element contains one or more of the following child elements:

- The status element contains the status of a test case: RUNNING or ENDED.
- The reportURL element contains the URL to the report in the Reporting Console.
- The runId element contains the unique identifier of the run.
- The pass element contains the number of tests that passed.
- The fail element contains the number of tests that failed.
- The warning element contains the number of tests that had warnings.
- The error element contains the number of tests that had errors.
- The message element contains information that is specific to the type of run.
- The tc element contains the name of a test case included in a suite.
- The callbackKey element contains a string that you can use to perform additional actions on the test case or suite.

Example: Synchronous Invocation

The following URL performs a synchronous invocation of the **AccountControlMDB** test case in the examples project.

```
http://localhost:1505/lisa-invoke/runTest/?testCasePath=examples/Tests/AccountControlMDB.tst&stagingDoc
```

The following XML response indicates that the test case passed.

```
<?xml version="1.0" encoding="UTF-8"?>
<invokeResult>
  <method name="RunTest">
    <params>
      <param name="stagingDocPath" value="examples/StagingDocs/1user1cycle0think.stg" />
      <param name="coordName" value="Coordinator" />
      <param name="configPath" value="" />
      <param name="testCasePath" value="examples/Tests/AccountControlMDB.tst" />
      <param name="callbackKey" value="64343533653737312D343765312D3439" />
    </params>
  </method>
  <status>OK</status>
  <result>
    <status>ENDED</status>

<reportUrl><![CDATA[http://localhost:1505/index.html?lisaPortal=reporting/printPreview_functional.html#
<runId>61653261643936342D613636392D3435</runId>
    <pass count="1" />
    <fail count="0" />
    <warning count="0" />
    <error count="0" />
    <message>AccountControlMDB,Run1User1Cycle0Think</message>
  </result>
</invokeResult>
```

Example: Asynchronous Invocation

The following URL performs an asynchronous invocation of the **AccountControlMDB** test case in the examples project.

```
http://localhost:1505/lisa-invoke/runTest/?testCasePath=examples/Tests/AccountControlMDB.tst&stagingDoc
```

The following XML response shows the callback key in the result element.

```
<?xml version="1.0" encoding="UTF-8"?>
<invokeResult>
  <method name="RunTest">
    <params>
      <param name="stagingDocPath" value="examples/StagingDocs/1user1cycle0think.stg" />
      <param name="coordName" value="" />
      <param name="configPath" value="" />
      <param name="testCasePath" value="examples/Tests/AccountControlMDB.tst" />
      <param name="callbackKey" value="61663038653562382D663566372D3432" />
      <param name="async" value="true" />
    </params>
  </method>
  <status>OK</status>
  <result>
    <callbackKey>61663038653562382D663566372D3432</callbackKey>
    <message>The LISA test 'examples/Tests/AccountControlMDB.tst' was launched asynchronously at
Mon Mar 26 16:05:39 PDT 2012.</message>
  </result>
</invokeResult>
```

Cloud DevTest Labs

You can use cloud-based infrastructure to provision development and test environments.

In this section, the following topics are covered:

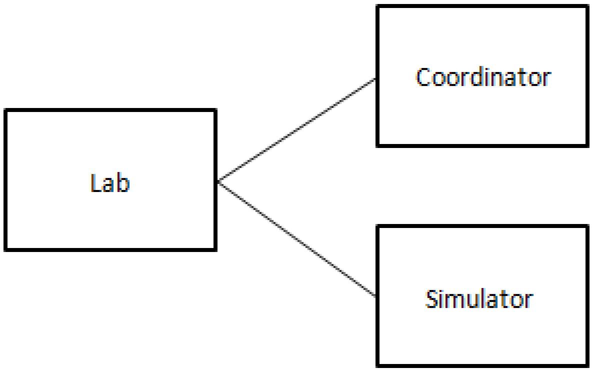
Labs and Lab Members

A *lab* is a logical container for one or more lab members.

A *lab member* can be a LISA server or a non-LISA server.

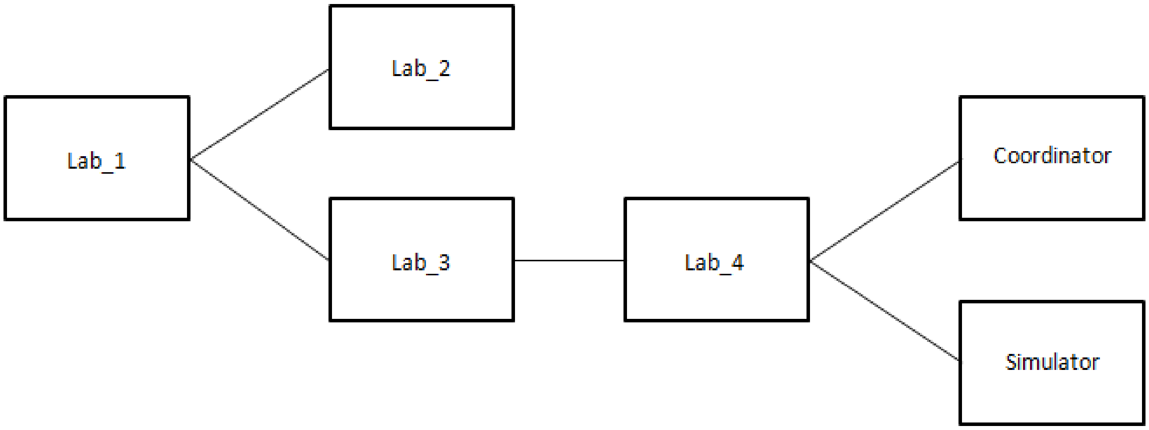
- The valid types of LISA servers are coordinator, simulator, and Virtual Service Environment.
- Examples of non-LISA servers include a database and a web server.

The following diagram shows a lab with two members: a coordinator and a simulator.



A lab can have one or more child labs.

The following diagram shows the hierarchical nature of labs. Lab_1 is the parent of Lab_2 and Lab_3. Lab_3 is the parent of Lab_4.



The fully qualified name of a child lab uses a forward slash as the separator. For example, the fully qualified name of Lab_4 in the previous diagram is Lab_1/Lab_3/Lab_4.

A lab named Default is included with LISA. If you start a coordinator, simulator, or Virtual Service Environment without specifying a lab, the Default lab is used.

A lab member has one of the following statuses:

- Uninitialized
- Starting
- Running
- Unknown

If a lab member is a LISA server, then the status proceeds from Uninitialized to Starting to Running.

If a lab member is a non-LISA server, then the status goes directly from Uninitialized to Running.

Virtual Lab Manager (VLM)

The cloud-based environment where the labs run is known as a Virtual Lab Manager (VLM).

The following VLM providers are supported:

- ServiceMesh Agility Platform 7.1
- VMware vCloud Director 1.0

You can configure LISA to use more than one VLM provider at the same time.

Each VLM provider has a unique prefix. The following table describes the prefixes.

VLM Provider	Prefix
ServiceMesh Agility Platform	AGL

VMware vCloud Director	VCD

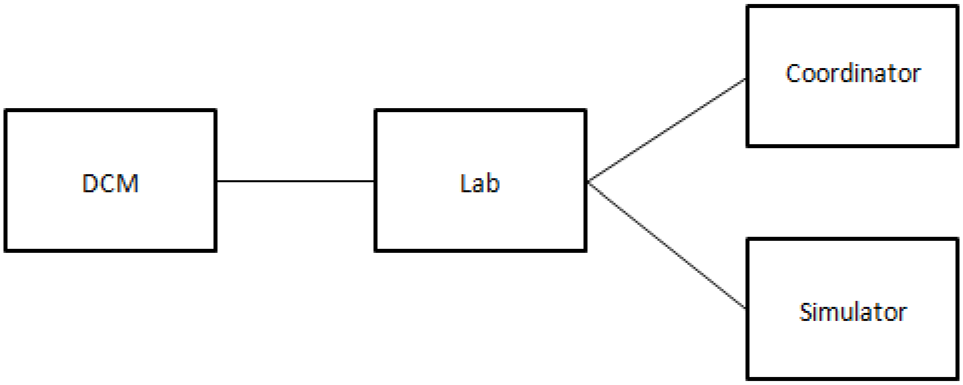
In LISA Workstation, the prefix and a colon appear at the beginning of a fully qualified lab name to indicate which VLM provider is being used. For example:

- AGL:Root/MyLab
- VCD:MyOrganization/MyCatalog/MyLab

The prefixes make it possible for you to use the same lab name in different VLM providers.

DevTest Cloud Manager (DCM)

The DevTest Cloud Manager (DCM) is the LISA component that interacts with the labs.

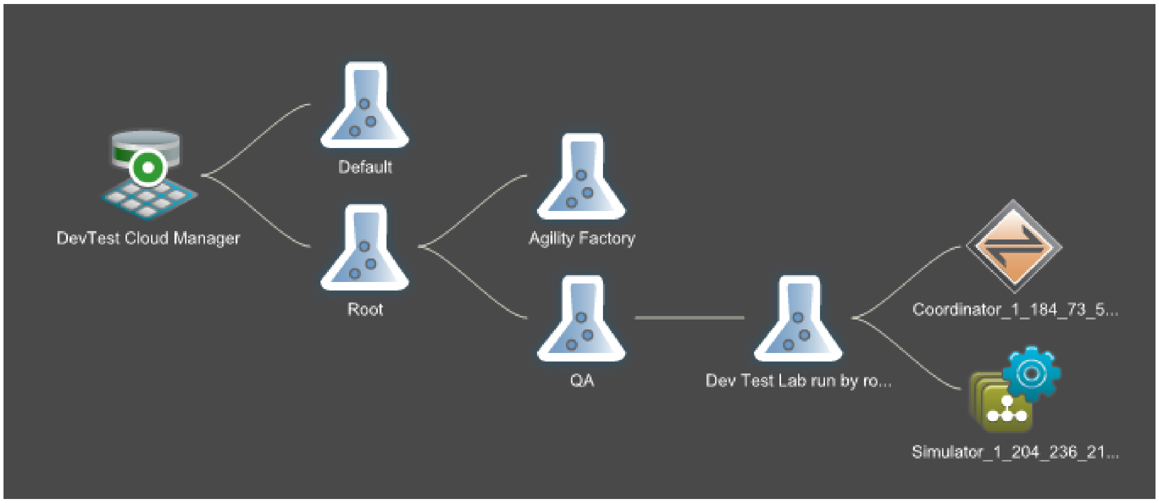


The responsibilities of the DCM include:

- Sending cloud-related properties to a lab
- Notifying the Virtual Lab Manager (VLM) provider that a lab should be started
- Sending the Model Archive (MAR) to a lab

The DCM lets you view, launch, monitor, and shut down labs from the Server Console.

The Server Console displays the DCM as part of the network graph. In the following image, the DCM is interacting with the Default lab and a lab called Root. The Root lab has two child labs: Agility Factory and QA. The QA lab has a child lab called Dev Test Lab. The fully qualified name of Dev Test Lab is Root/QA/Dev Test Lab.


DevTest Cloud Manager
Default
Root
Agility Factory
QA
Dev Test Lab run by ro...
Coordinator_1_184_73_5...
Simulator_1_204_236_21...

Configuring LISA DCM Properties

To enable the provisioning of development and test labs, you must configure properties on the computer where the registry is located.

Some properties are applicable to all Virtual Lab Manager (VLM) providers. Other properties are specific to a VLM provider.

- General Properties
- ServiceMesh Properties

General Properties

This section describes the properties that are applicable to all VLM providers.

You must configure the following property in the **local.properties** file:

- lisa.net.bindToAddress

You must configure the following properties in the **site.properties** file:

- lisa.dcm.labstartup.min
- lisa.dcm.lisastartup.min
- lisa.dcm.lisashutdown.min
- lisa.dcm.lab.factories
- lisa.net.timeout.ms

If you want to enable access control (ACL), then you must configure the following property in either the **local.properties** file or the **site.properties** file. It does not matter which file you choose.

- lisa.acl.auth.enabled

lisa.net.bindToAddress

The fully qualified domain name or IP address of the computer where the registry is located. The domain name or IP address must be addressable by any computer that is launched to a network outside of the network in which the registry resides.

Syntax

```
lisa.net.bindToAddress=<fully-qualified-domain-name-or-IP-address>
```

lisa.dcm.labstartup.min

The number of minutes that LISA will wait for the VLM provider to start a lab.

Syntax

```
lisa.dcm.labstartup.min=<integer>
```

lisa.dcm.lisastartup.min

The number of minutes that LISA will wait after the VLM provider has started a lab for LISA to be properly initialized.

Syntax

```
lisa.dcm.lisastartup.min=<integer>
```

lisa.dcm.lisashutdown.min

The number of minutes that LISA will wait after the test execution has completed to shut down a lab.

Syntax

```
lisa.dcm.lisashutdown.min=<integer>
```

lisa.dcm.lab.factories

The name of the object that contains cloud support logic for a specific VLM provider. The valid values are **com.itko.lisa.cloud.serviceMesh.ServiceMeshCloudSupport** and **com.itko.lisa.cloud.vCloud.vCloudDirectorCloudSupport**.

You can specify more than one VLM provider. If you do so, then you must separate the values with a semicolon.

Syntax

```
lisa.dcm.lab.factories=<object-name>[;<object-name>]
```

lisa.net.timeout.ms

The timeout value (in milliseconds) used by the underlying messaging system. This property is not cloud specific. Modify the value for cloud integration because certain operations can take longer than the default 30 seconds.

Syntax

```
lisa.net.timeout.ms=<integer>
```

lisa.acl.auth.enabled

If you want to enable access control (ACL), then set the value of this property to **true**.

Syntax

```
lisa.acl.auth.enabled=<true|false>
```

ServiceMesh Properties

This section describes the properties that are specific to ServiceMesh Agility Platform.

You must configure the following property in the **site.properties** file:

- lisa.dcm.SERVICEMESH.baseUri

You must also configure the following properties. The location depends on whether ACL is enabled.

- lisa.dcm.SERVICEMESH.userId
- lisa.dcm.SERVICEMESH.password

If ACL is not enabled, then you must configure the properties in the **site.properties** file.

If ACL is enabled, then LISA creates the **lisa.dcm.SERVICEMESH.userId** and **lisa.dcm.SERVICEMESH.password** custom permissions and assigns them to each role. You can specify the initial values for these custom permissions by configuring the **lisa.dcm.SERVICEMESH.userId** and **lisa.dcm.SERVICEMESH.password** properties in the **site.properties** file. If you do not specify the initial values, then you must set the values from the Server Console. See Adding, Updating, and Deleting Roles.

You can optionally configure the following property in the **site.properties** file:

- lisa.dcm.SERVICEMESH.lab.cache.sec

lisa.dcm.SERVICEMESH.baseUri

The base URL of ServiceMesh Agility Platform's REST interface.

Syntax

```
lisa.dcm.SERVICEMESH.baseUri=https://<fully-qualified-domain-name>:<ssl-port-number>/agility/api/v1.0
```

lisa.dcm.SERVICEMESH.userId

A user name that can log in to the ServiceMesh Agility Platform web interface.

Syntax

```
lisa.dcm.SERVICEMESH.userId=<user-id>
```

lisa.dcm.SERVICEMESH.password

The password for the user name defined in the **lisa.dcm.SERVICEMESH.userId** property.

Syntax

```
lisa.dcm.SERVICEMESH.password=<password>
```

lisa.dcm.SERVICEMESH.lab.cache.sec

LISA maintains a cache of the ServiceMesh project configuration. This property specifies how often LISA will access ServiceMesh to see whether the cache needs to be updated (for example, an environment may have been added). The value is in seconds. The default value is 300.

Syntax

```
lisa.dcm.SERVICEMESH.lab.cache.sec=<integer>
```

vCloud Director Properties

This section describes the properties that are specific to VMware vCloud Director.

You must configure the following property in the **site.properties** file:

- lisa.dcm.vCLOUD.baseUri

You must also configure the following properties. The location depends on whether ACL is enabled.

- lisa.dcm.vCLOUD.userId
- lisa.dcm.vCLOUD.password

If ACL is not enabled, then you must configure the properties in the **site.properties** file.

If ACL is enabled, then LISA creates the **lisa.dcm.vCLOUD.userId** and **lisa.dcm.vCLOUD.password** custom permissions and assigns them to each role. You can specify the initial values for these custom permissions by configuring the **lisa.dcm.vCLOUD.userId** and **lisa.dcm.vCLOUD.password** properties in the **site.properties** file. If you do not specify the initial values, then you must set the values from the Server Console. See Adding, Updating, and Deleting Roles.

lisa.dcm.vCLOUD.baseUri

The login URL of the vCloud API.

Syntax

```
lisa.dcm.vCLOUD.baseUri=<url>
```

lisa.dcm.vCLOUD.userId

A user name that can log in to vCloud Director. Typically, the format consists of the user name, followed by an ampersand (@), followed by the organization name.

Syntax

```
lisa.dcm.vCLOUD.userId=<user-name>@<organization-name>
```

lisa.dcm.vCLOUD.password

The password for the user name defined in the **lisa.dcm.vCLOUD.userId** property.

Syntax

```
lisa.dcm.vCLOUD.password=<password>
```

Example: Properties for ServiceMesh-Based Configuration

The following example shows a ServiceMesh-based configuration. This example assumes that ACL is not enabled.

This property is located in the **local.properties** file:

```
lisa.net.bindToAddress=myserver.example.com
```

These properties are located in the **site.properties** file:

```
lisa.dcm.labstartup.min=6
lisa.dcm.lisastartup.min=4
lisa.dcm.lisashutdown.min=2
lisa.dcm.lab.factories=com.itko.lisa.cloud.serviceMesh.ServiceMeshCloudSupport;

lisa.dcm.SERVICEMESH.baseUri=https://www.example.com:8443/agility/api/v1.0
lisa.dcm.SERVICEMESH.userId=admin
lisa.dcm.SERVICEMESH.password=mypassword
lisa.dcm.SERVICEMESH.lab.cache.sec=360

lisa.net.timeout.ms=60000
```

Example: Properties for vCloud Director-Based Configuration

The following example shows a vCloud Director-based configuration. This example assumes that ACL is not enabled.

This property is located in the **local.properties** file:

```
lisa.net.bindToAddress=myserver.example.com
```

These properties are located in the **site.properties** file:

```
lisa.dcm.labstartup.min=6
lisa.dcm.lisastartup.min=4
lisa.dcm.lisashutdown.min=2
lisa.dcm.lab.factories=com.itko.lisa.cloud.vCloud.vCloudDirectorCloudSupport;

lisa.dcm.vCLOUD.baseUri=https://www.example.com/api/versions
lisa.dcm.vCLOUD.userId=administrator@System
lisa.dcm.vCLOUD.password=mypassword

lisa.net.timeout.ms=60000
```

Configuring ServiceMesh

To enable the provisioning of development and test labs with ServiceMesh Agility Platform as the Virtual Lab Manager (VLM) provider, you must perform configuration steps in ServiceMesh.

This topic uses the following ServiceMesh concepts: environment, image, instance, package, script, stack, and template. Some of these terms can be mapped to LISA concepts:

- An environment in ServiceMesh corresponds to a lab in LISA.
- An instance in ServiceMesh corresponds to a lab member in LISA.

Startup scripts must be written for the following LISA server components:

- Coordinator
- Simulator
- Virtual Service Environment

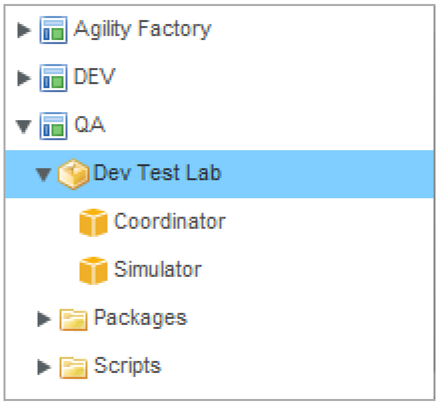
The startup script must include a command that starts the server component in **remoteInit** mode. This mode causes the server component to start and then wait until the DevTest Cloud Manager (DCM) sends the required initialization settings. For example:

```
su -c "nohup lisa/bin/CoordinatorServer -remoteInit" - itko &
```

If you want the labs to include only a Coordinator and Simulator, then you do not need to write a startup script for the Virtual Service Environment.

If you want the labs to include only a Virtual Service Environment, then you do not need to write startup scripts for the Coordinator and Simulator.

The following image shows an example configuration in ServiceMesh. The Dev Test Lab environment contains a Coordinator template and a Simulator template.



When a lab is started, LISA makes a copy of the corresponding ServiceMesh environment. At any given time, there can be multiple independent copies of the same environment.

To configure ServiceMesh

1. Configure and run a base image supported by ServiceMesh.
2. Download the installer for LISA Server.
3. Install LISA Server onto the base image. When the installation is finished, do not configure the license properties or make any changes to the install.
4. Create a new stack. Use the base image with LISA installed as the base for the stack. This stack is the "bare metal" image of LISA. Be sure to note where LISA is installed so that the startup scripts can properly reference the location.
5. To add a Coordinator, do the following:
    a. Select an environment and create a new virtual machine template. Set the template name to **Coordinator**, or ensure that **Coordinator** appears as part of the template name. Use the stack that you created as the base for the template. The startup order of the template does not matter to LISA.
    b. In the same project, create a new script that starts the Coordinator in **remoteInit** mode. Add the script to a package as a startup script. Add the package to the template that you created in the previous step.
6. To add a Simulator, do the following:
    a. Select an environment and create a new virtual machine template. Set the template name to **Simulator**, or ensure that **Simulator** appears as part of the template name. Use the stack that you created as the base for the template. The startup order of the template does not matter to LISA.
    b. In the same project, create a new script that starts the Simulator in **remoteInit** mode. Add the script to a package as a startup script. Add the package to the template that you created in the previous step.
7. To add a Virtual Service Environment, do the following:
    a. Select an environment and create a new virtual machine template. Set the template name to **VSE**, or ensure that **VSE** appears

as part of the template name. Use the stack that you created as the base for the template. The startup order of the template does not matter to LISA.

b. In the same project, create a new script that starts the Virtual Service Environment in **remoteInit** mode. Add the script to a package as a startup script. Add the package to the template that you created in the previous step.

Configuring vCloud Director

To enable the provisioning of development and test labs with VMware vCloud Director as the Virtual Lab Manager (VLM) provider, you must perform configuration steps in vCloud Director.

This topic uses the following vCloud Director concepts: catalog, vApp, and vApp template. vApps and vApp templates in vCloud Director correspond to labs in LISA.

During the configuration procedure, you create virtual machine images for the following LISA server components:

- Coordinator
- Simulator
- Virtual Service Environment

If you want the labs to include only a Coordinator and Simulator, then you do not need to create an image for the Virtual Service Environment.

If you want the labs to include only a Virtual Service Environment, then you do not need to create images for the Coordinator and Simulator.

Each server component must be configured to start in **remoteInit** mode. This mode causes the server component to start and then wait until the DevTest Cloud Manager (DCM) sends the required initialization settings.

To configure vCloud Director

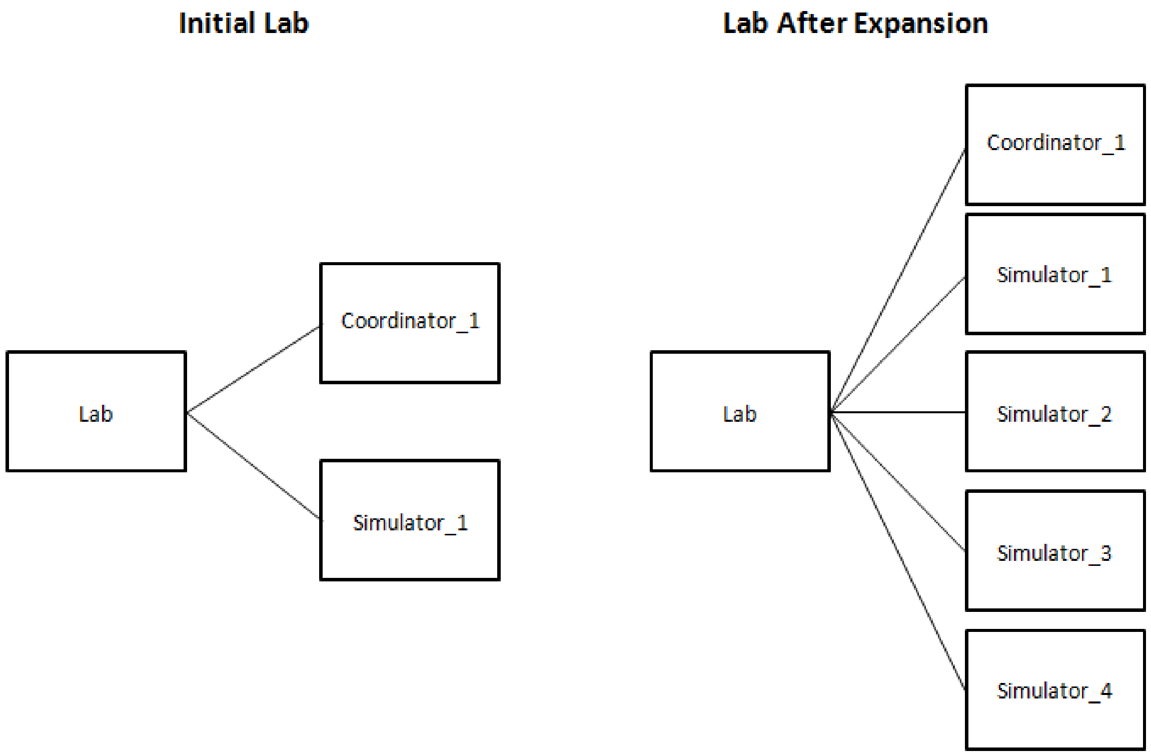
1. Using an application such as VMware Workstation, create a virtual machine image for each server component that you want to include in a lab. Configure each image to start the server component in **remoteInit** mode. The name of a Coordinator must include the term **Coordinator**. The name of a Simulator must include the term **Simulator**. The name of a Virtual Service Environment must include the term **VSE**.
2. Log in to the vCloud Director web console as an administrator.
3. Import each virtual machine image that you created as a vApp template.
4. Create a vApp from the vApp templates.
5. Add the vApp to a catalog.

Dynamic Expansion of Test Labs

LISA can determine that more capacity is required to meet the needs of a running test and then automatically expand the lab.

You must use a staging document that has the Dynamic Simulator Scaling with DCM distribution pattern. For more information, see Distribution Selection.

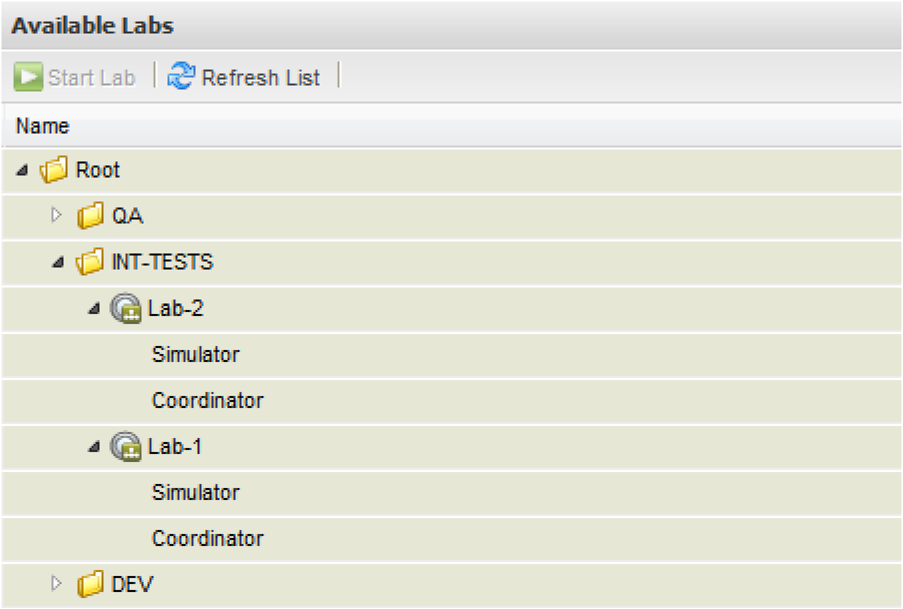
The following diagram shows an example. In the left portion, the lab initially has one Coordinator and one Simulator. In the right portion, the lab has one Coordinator and four Simulators after the expansion takes place.



Listing the Available Labs

The Server Console enables you to list the development and test labs that are available in the cloud environment.

The following image shows a list of available labs in the Server Console. The tree structure is expanded to show two child labs.



- Root
  - QA
  - INT-TESTS
    - Lab-2
      - Simulator
      - Coordinator
    - Lab-1
      - Simulator
      - Coordinator
  - DEV

For instructions on accessing the Server Console, see Opening the LISA Console.

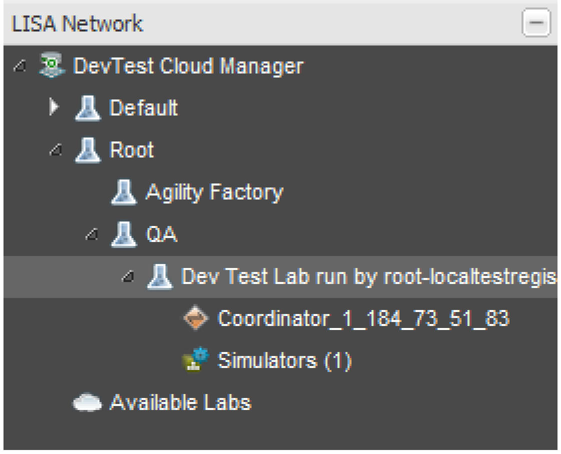
To list the available labs

1. In the Server Console, display the LISA Network panel.
2. Expand the DevTest Cloud Manager node.
3. Click the Available Labs node.
   The available labs appear in the right panel. The tree structure is initially collapsed.

Starting a Lab

When you start a lab, you are actually starting an instance of the lab. A lab can have multiple instances, all of which are independent of each other.

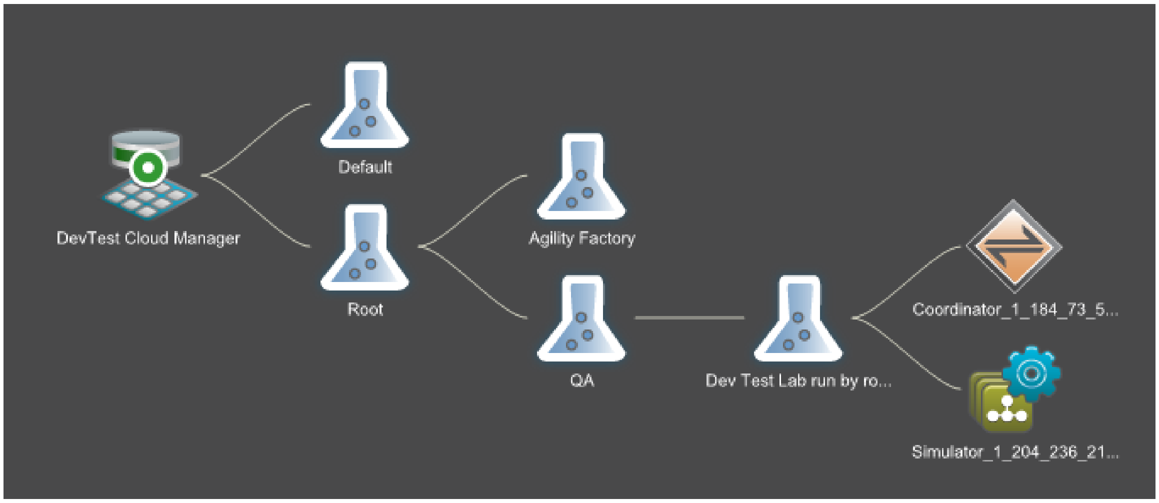
In the Server Console, the LISA Network panel displays the started lab in a tree structure. The following image shows the LISA Network panel.



LISA Network

- DevTest Cloud Manager
  - Default
  - Root
    - Agility Factory
    - QA
      - Dev Test Lab run by root-localtestregis
        - Coordinator_1_184_73_51_83
        - Simulators (1)
  - Available Labs

The right panel displays the started lab in the *network graph*. The following image shows the network graph.


DevTest Cloud Manager
Default
Root
Agility Factory
QA
Dev Test Lab run by ro...
Coordinator_1_184_73_5...
Simulator_1_204_236_21...

The **lisa.dcm.labstartup.min** property controls the number of minutes that LISA will wait for the Virtual Lab Manager (VLM) provider to start the lab. For more information, see Configuring LISA DCM Properties.

> ⚠ You do not need to explicitly start the Default lab.

Starting a Lab from the Command Line

You can start a lab by invoking one of the LISA server executables, and specifying a server name and a lab name.

For example, the following command starts a lab named MyLab, which is a child of MyParentLab. The MyLab lab has one lab member: a coordinator named Dev-Coord.

```
CoordinatorServer -n Dev-Coord -l MyParentLab/MyLab
```

Starting a Lab from the Server Console

This procedure assumes that you have listed the available labs.

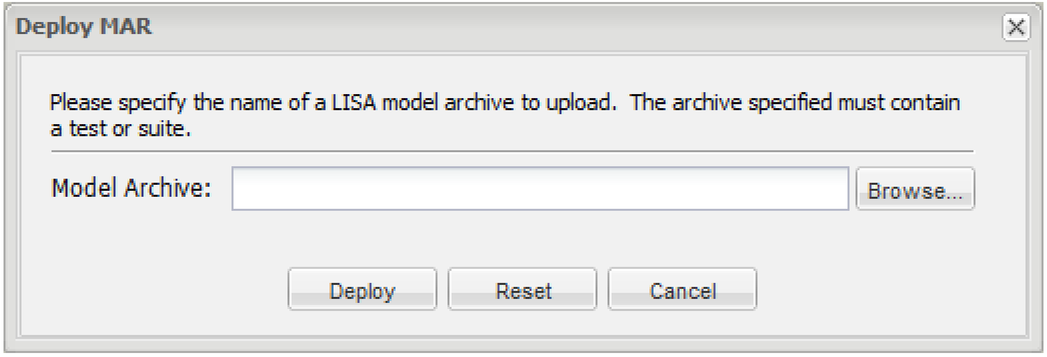
To start a lab from the Server Console

1. In the list of available labs, select the lab.
2. Click Start Lab.
3. Wait for the lab to start.
   When the startup sequence is finished, a message indicates that the lab was started.
   The LISA Network panel displays the started lab in a tree structure. The right panel displays the started lab in the network graph.
4. Wait for the lab members to start.
   When the startup sequence for a lab member is finished, the member's status changes to Running. In addition, statistics about the lab member begin appearing in the Component Health Summary tab.

Deploying a Model Archive (MAR) to a Lab

You can deploy the Model Archive (MAR) for a test case or suite to a lab that has been started. The lab must include a coordinator and (in most cases) a simulator.

To deploy a Model Archive (MAR) to a lab

1. In the LISA Network panel of the Server Console, click the coordinator.
   The details window for the coordinator appears in the right panel.
2. In the right panel, click Deploy MAR.
   The Deploy MAR dialog appears.
   
   Deploy MAR
   Please specify the name of a LISA model archive to upload. The archive specified must contain a test or suite.
   Model Archive:

3. Click Browse and select the .mar file.
4. Click Deploy.
   A message indicates that the model archive has been successfully deployed. The test case or suite is now running.
5. Click OK.

Stopping a Lab

You can stop a currently running lab from the Server Console. This action is permanent and cannot be undone.

If the lab is a child lab, the parent labs are not stopped.

⚠ You cannot stop the Default lab.

To stop a lab

- In the network graph, right-click the lab and select Stop.
  When the action is finished, a message indicates that the lab was stopped.

Cloud DevTest Lab Videos

The following videos explore different aspects of using cloud-based infrastructure to provision development and test environments.

Get Runnable Labs

This video shows how to get a list of runnable labs available to be launched. The length of the video is 30 seconds.

Get Runnable Labs

Start a Runnable Lab

This video shows how to start a lab that you can deploy tests to. The length of the video is 1 minute.

Start a Runnable Lab

Stage a Test to Running Lab

This video shows how to take a started lab and initialize a test on the server that was provisioned in the cloud. The length of the video is 1 minute.

Stage a Test to Running Lab

Deploy a Virtual Service to Launched Lab

This video shows how to stage a virtual service to a deployed virtual server. The length of the video is 2 minutes and 30 seconds.

Deploy a Virtual Service to Launched Lab

Stop a Running Lab

This video shows how to shut down a lab. The length of the video is 1 minute.

Stop a Running Lab

Continuous Validation Service (CVS)

The Continuous Validation Service (CVS) lets you schedule tests and test suites to run on a regular basis over an extended time period.

CVS has a dashboard that maintains a list of all scheduled tests (services) and the status of each one. From the CVS Dashboard, you can select a test and monitor each test run.

A monitor contains a single test or an entire test suite. A service contains one or many monitors within itself.

The tests are managed by a coordinator and run on a simulator. State is maintained in a database on the LISA registry.

You can either choose to run a service or individual monitors within it from the CVS Dashboard.

Prerequisite

CVS runs in a LISA Server environment.

There must be a coordinator server and a simulator server running, registered with a LISA registry.

For details on setting up the LISA Server environment, see the *Installation and Configuration Guide*.

The following sections are available:

Opening the CVS Dashboard

You can open the CVS Dashboard from LISA Workstation or from a web browser.

To open the CVS Dashboard from LISA Workstation

- From the main menu, select View > CVS Dashboard.

To open the CVS Dashboard from a web browser

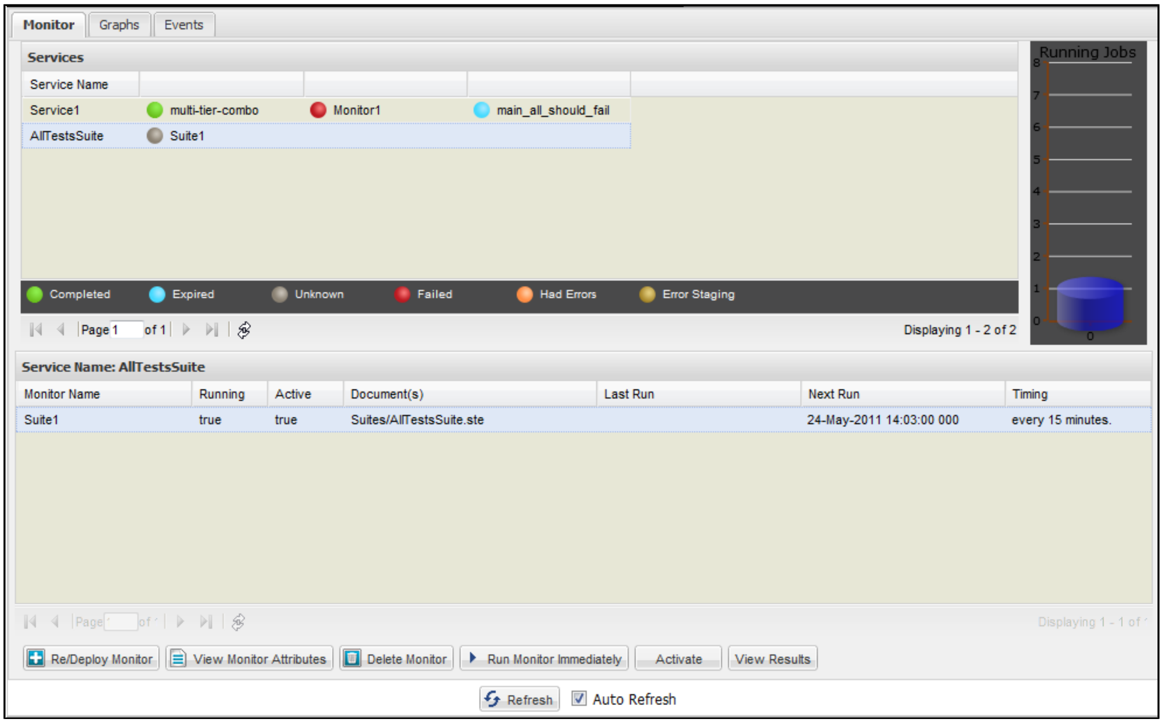
1. Ensure that the registry is running.
2. Enter **http://localhost:1505/** in a web browser. If the registry is located on a remote computer, replace **localhost** with the name or IP address of the computer.
   The LISA Console appears.
3. Click Continuous Validation Service.

Closing the CVS dashboard or closing LISA Workstation will not interfere with the CVS scheduled tasks. When you reconnect to the registry, you will see the dashboard with the current data and status information.

CVS Dashboard Overview

The CVS Dashboard has the following tabs:

- **Monitors**: Displays a list of all the monitors (tests/test suites) that are added to the CVS Dashboard. For more information, see Monitor Tab. Your CVS dashboard will list all the monitors running on an attached LISA Registry, not just the ones initiated by you.
- **Graphs:** Displays the dashboard status graphically. It will also show the percentage of the tests passed or failed. For more information, see Graphs Tab.
- **Events:** Displays the status events recorded by the monitors. For more information, see Events Tab.



Service Name			
Service1	● multi-tier-combo	● Monitor1	● main_all_should_fail
AllTestsSuite	● Suite1		

Service Name: AllTestsSuite

Monitor Name	Running	Active	Document(s)	Last Run	Next Run	Timing
Suite1	true	true	Suites/AllTestsSuite.ste		24-May-2011 14:03:00 000	every 15 minutes.

A refresh button, which lets you refresh the list on the dashboard, appears at the bottom of the dashboard.

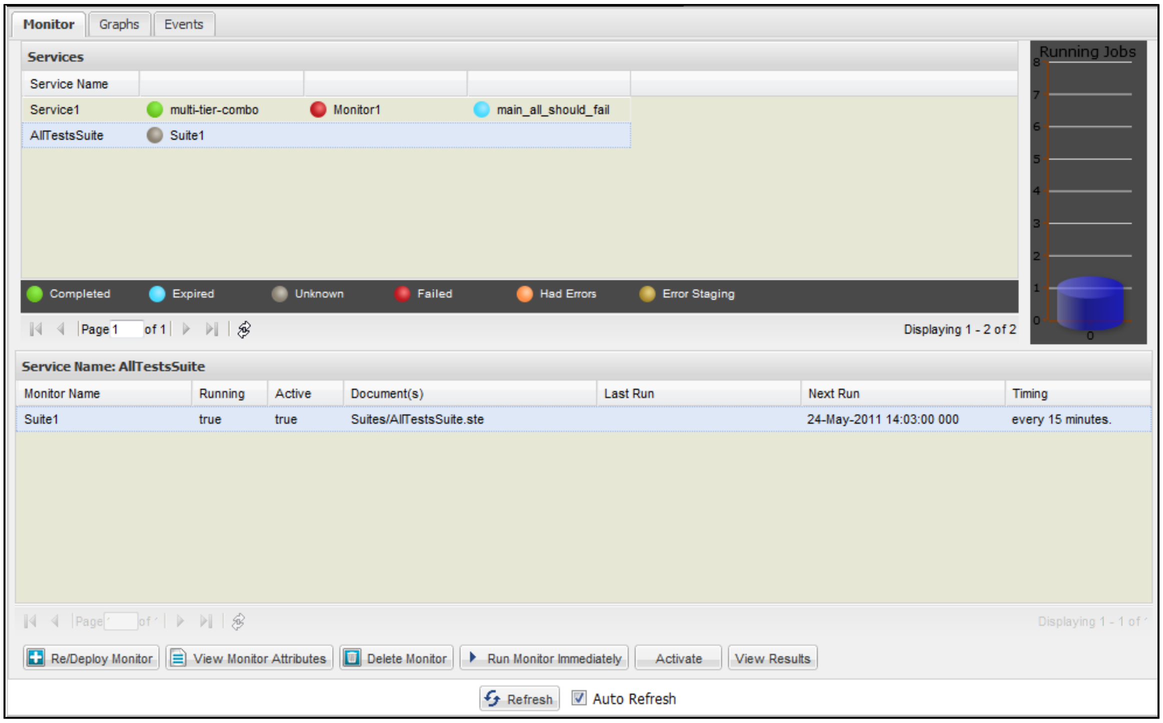
Monitor Tab

The CVS Dashboard opens in the Monitor tab.

You can add many monitors to run in one or many services. A service contains one or more monitors.

You can schedule to run a service. All monitors in that service are run at the scheduled time. The services are run in the background at scheduled intervals.

You can either add the monitors in one service (in the following example, **Service1**) or add the monitors in different services (in the following example, **Service1** and **AllTestsSuite**). All the monitors added in one service (for example, **Service1**) are shown added after that service name.



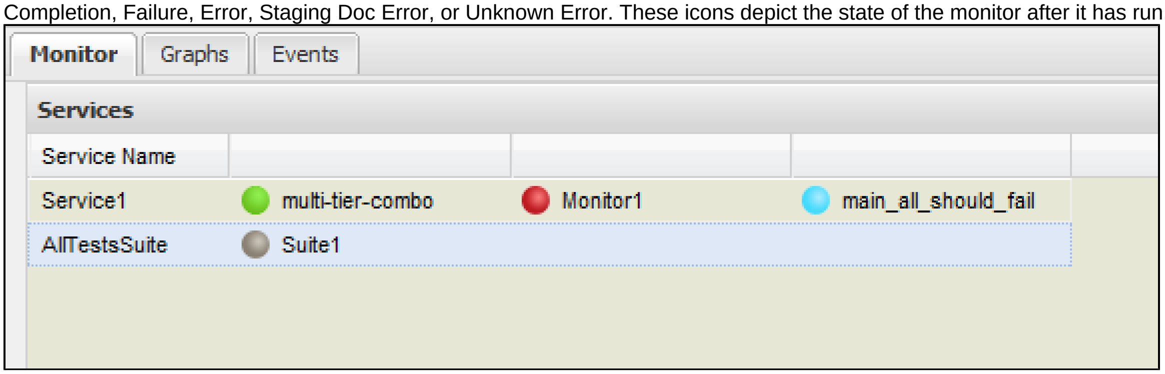
Service Name			
Service1	● multi-tier-combo	● Monitor1	● main_all_should_fail
AllTestsSuite	● Suite1		

Service Name: AllTestsSuite

Monitor Name	Running	Active	Document(s)	Last Run	Next Run	Timing
Suite1	true	true	Suites/AllTestsSuite.ste		24-May-2011 14:03:00 000	every 15 minutes.

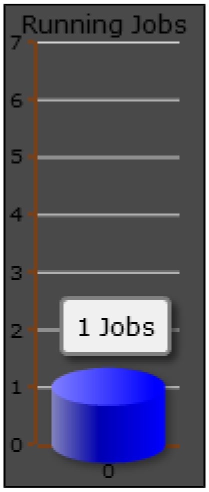
Top Panel

All the tests in a particular service that have run are depicted with a colored ball. The completed tests are shown by a green ball, failed tests are shown by a red ball, tests that had an error in their staging documents are shown with a yellow ball, and so on. The icons depict the Test Run Completion, Failure, Error, Staging Doc Error, or Unknown Error. These icons depict the state of the monitor after it has run.



Service Name			
Service1	● multi-tier-combo	● Monitor1	● main_all_should_fail
AllTestsSuite	● Suite1		

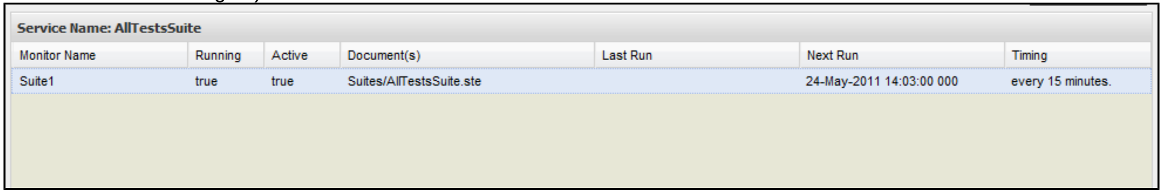
At the right side, the top panel shows a graphical representation of the jobs that are currently running. For example, the following graph shows only one job currently running.



Bottom Panel

The bottom panel displays the status of each monitor run in a service.

The bottom panel shows all the monitors that are running or have finished their last run. The monitors that have finished running completely (are not scheduled to run again) will not be seen in this list.



Service Name: AllTestsSuite

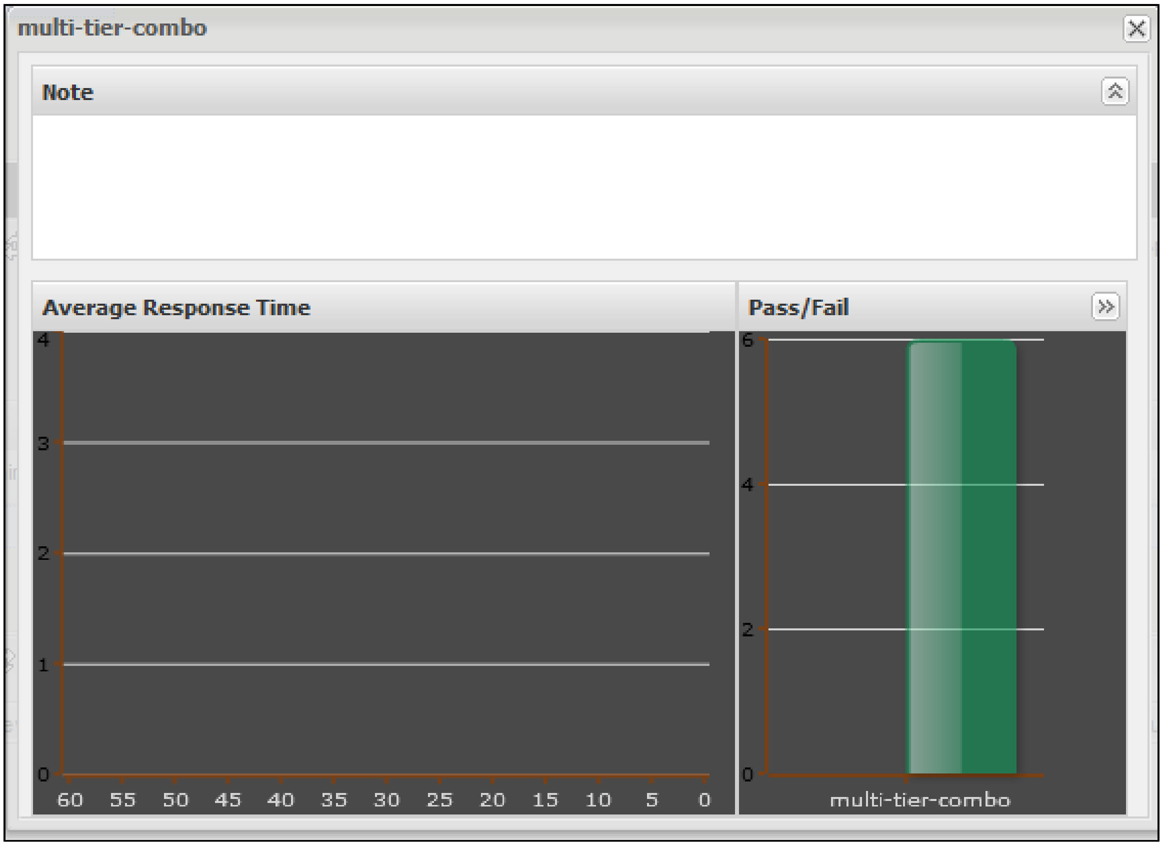
Monitor Name	Running	Active	Document(s)	Last Run	Next Run	Timing
Suite1	true	true	Suites/AllTestsSuite.ste		24-May-2011 14:03:00 000	every 15 minutes.

The table shows the following information:

- **Monitor Name**: The name given to the monitor
- **Running**: Shows whether this monitor is currently running (True/False)
- **Active**: Shows whether this monitor is currently Active (True/False)
- **Documents**: The names of the documents, such as test case, staging, and suite documents associated with this monitor
- **Last Run**: The last run date and time of this monitor
- **Next Run**: The next scheduled start time of this monitor
- **Timing**: The schedule details for this monitor

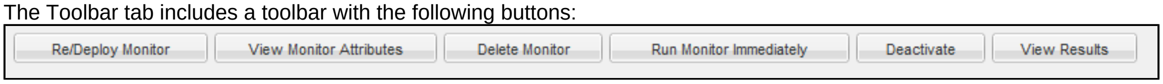
You can customize the columns by arranging them in sorted order or showing/hiding of the columns.

Double-click a monitor to see the monitor-related information.

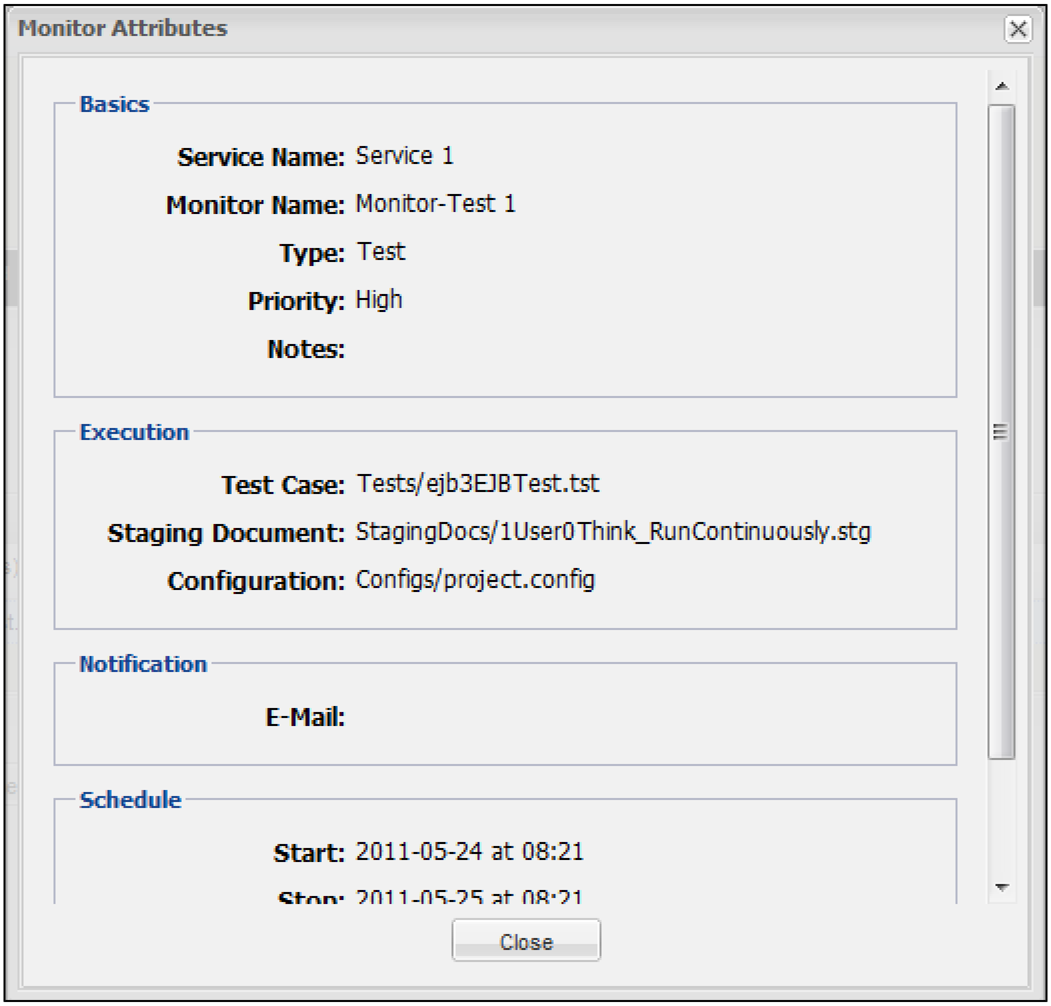


Toolbar

The Toolbar tab includes a toolbar with the following buttons:



- **Re/Deploy Monitor**: Deploy or redeploy a monitor to the Dashboard. See Deploying a Monitor to CVS.
- **View Monitor Attributes**: View monitor-related information



Monitor Attributes

Basics

Service Name: Service 1

Monitor Name: Monitor-Test 1

Type: Test

Priority: High

Notes:

Execution

Test Case: Tests/ejb3EJBTest.tst

Staging Document: StagingDocs/1User0Think_RunContinuously.stg

Configuration: Configs/project.config

Notification

E-Mail:

Schedule

Start: 2011-05-24 at 08:21

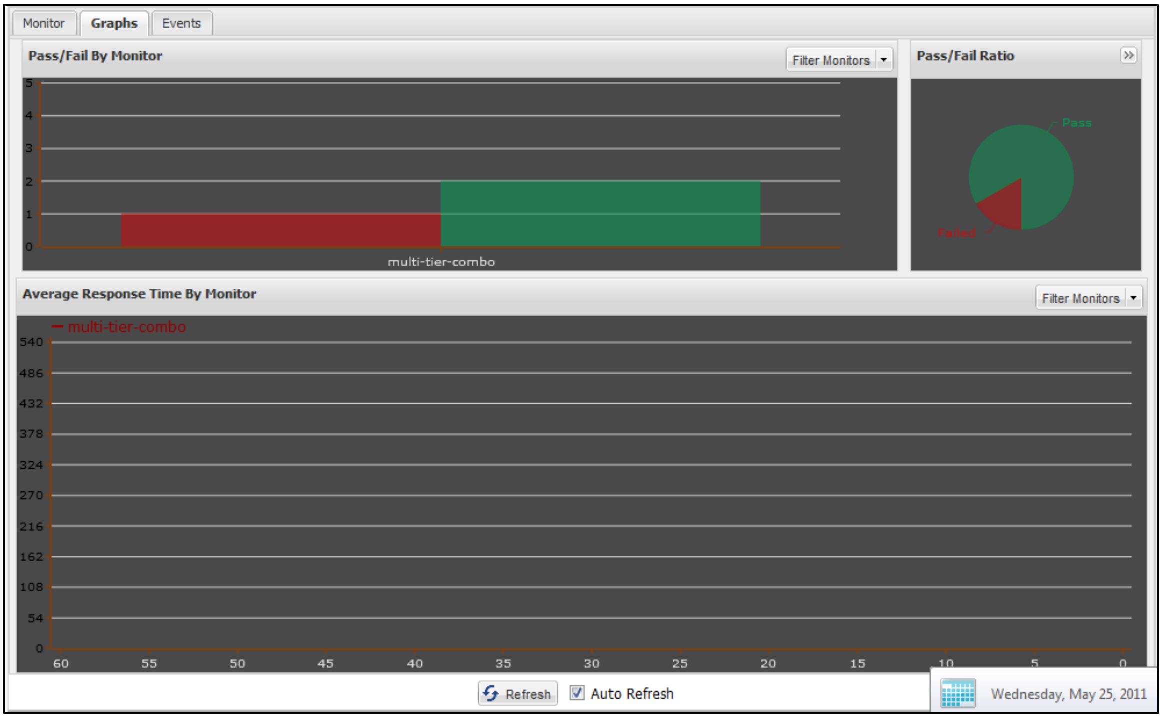
Stop: 2011-05-25 at 08:21

- **Delete Monitor**: Delete a monitor from the Dashboard.
- **Run Monitor Immediately**: Run the monitor immediately.
- **Deactivate**: Click to deactivate the monitor. Deactivate means it remains in the list, but will not run. You can also click this button to reactivate a previously-deactivated monitor.
- **View Results**: Click to view the results of the run in the Report Viewer.

When you are running the CVS Dashboard directly from LISA Workstation (by selecting to view the Dashboard from LISA), you will see buttons to **Refresh** the list on the Dashboard and AutoRefresh ☑ to auto-refresh the list on the Dashboard after a certain period. When you run the CVS Dashboard directly from your browser, these buttons do not appear.

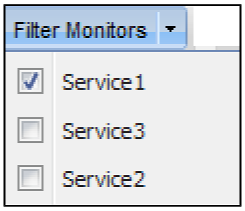
Graphs Tab

The Graphs tab displays the tests in the CVS Dashboard in a graphical format. It gives you a graphical summary of the tests that have passed and failed.



The Graphs tab consists of three graphical displays, which show and track the last 60 minutes of activity.

Filter Monitor: You can filter the monitors for viewing from the Filter Monitor button.



- Click the Filter Monitor button to get a list of available monitors.
- Select one to view the details regarding that monitor.
  - **Pass/Fail by Monitor**: The **Pass Fail by Monitor** graph shows stacked histograms of Pass/Fail results for each monitor. Moving the cursor over the histogram shows the result in text form.

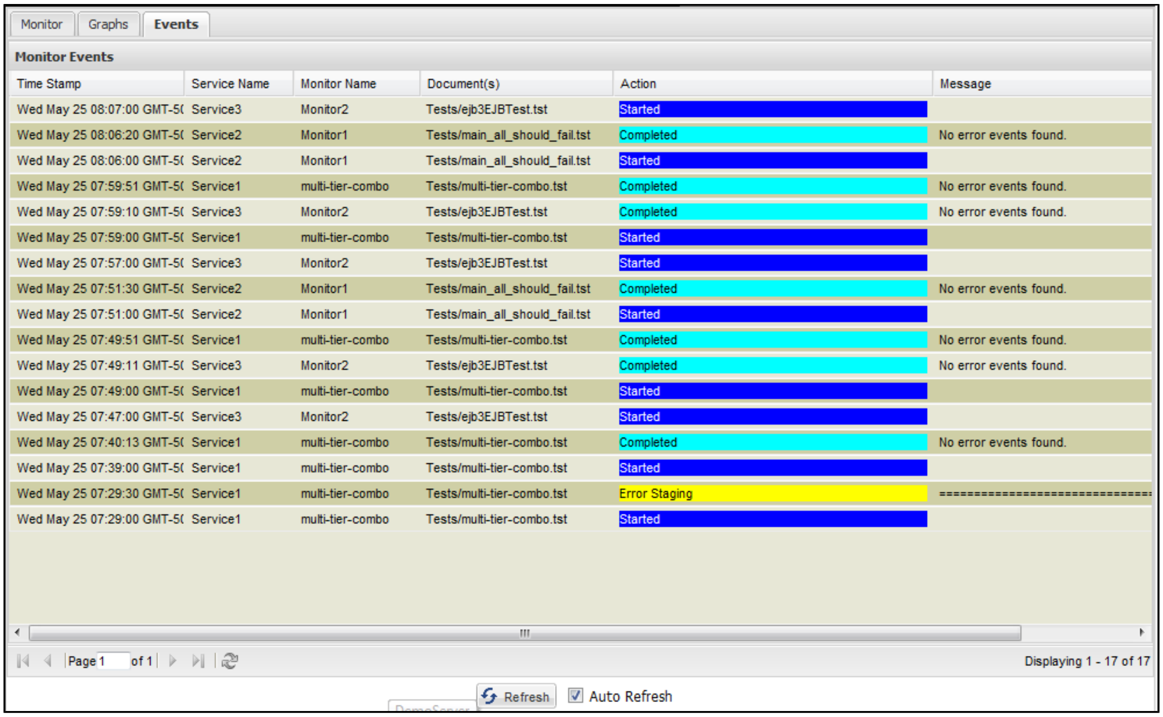
In the top **Pass/Fail By Monitor** panel, you can choose to see either all the tests or selective tests/suites by clicking the Filter Monitor button, which lets you select the tests/suites.

  - **Pass/Fail Ratio**: The Pass/Fail Ratio graph displays the percentage of total number of passing tests (green), and the total number of failing tests (red) as a pie chart.
  - **Average Response Time by Monitor**: **Average Response Time by Monitor** graphs the average response time for each monitor over the last 60 minutes of activity. Each monitor is color-coded. Hover over a monitor to show the average response time in a tooltip.

You can choose to display the test/suite to be seen in the graphical format in the **Pass/Fail By Monitor**.

Events Tab

The Events tab displays various events corresponding to the stages of the monitor run, like starting of a test run, ending, failing, and so on.



Monitor Events

Time Stamp	Service Name	Monitor Name	Document(s)	Action	Message
Wed May 25 08:07:00 GMT-5(	Service3	Monitor2	Tests/ejb3EJBTest.tst	Started	
Wed May 25 08:06:20 GMT-5(	Service2	Monitor1	Tests/main_all_should_fail.tst	Completed	No error events found.
Wed May 25 08:06:00 GMT-5(	Service2	Monitor1	Tests/main_all_should_fail.tst	Started	
Wed May 25 07:59:51 GMT-5(	Service1	multi-tier-combo	Tests/multi-tier-combo.tst	Completed	No error events found.
Wed May 25 07:59:10 GMT-5(	Service3	Monitor2	Tests/ejb3EJBTest.tst	Completed	No error events found.
Wed May 25 07:59:00 GMT-5(	Service1	multi-tier-combo	Tests/multi-tier-combo.tst	Started	
Wed May 25 07:57:00 GMT-5(	Service3	Monitor2	Tests/ejb3EJBTest.tst	Started	
Wed May 25 07:51:30 GMT-5(	Service2	Monitor1	Tests/main_all_should_fail.tst	Completed	No error events found.
Wed May 25 07:51:00 GMT-5(	Service2	Monitor1	Tests/main_all_should_fail.tst	Started	
Wed May 25 07:49:51 GMT-5(	Service1	multi-tier-combo	Tests/multi-tier-combo.tst	Completed	No error events found.
Wed May 25 07:49:11 GMT-5(	Service3	Monitor2	Tests/ejb3EJBTest.tst	Completed	No error events found.
Wed May 25 07:49:00 GMT-5(	Service1	multi-tier-combo	Tests/multi-tier-combo.tst	Started	
Wed May 25 07:47:00 GMT-5(	Service3	Monitor2	Tests/ejb3EJBTest.tst	Started	
Wed May 25 07:40:13 GMT-5(	Service1	multi-tier-combo	Tests/multi-tier-combo.tst	Completed	No error events found.
Wed May 25 07:39:00 GMT-5(	Service1	multi-tier-combo	Tests/multi-tier-combo.tst	Started	
Wed May 25 07:29:30 GMT-5(	Service1	multi-tier-combo	Tests/multi-tier-combo.tst	Error Staging	================================
Wed May 25 07:29:00 GMT-5(	Service1	multi-tier-combo	Tests/multi-tier-combo.tst	Started	

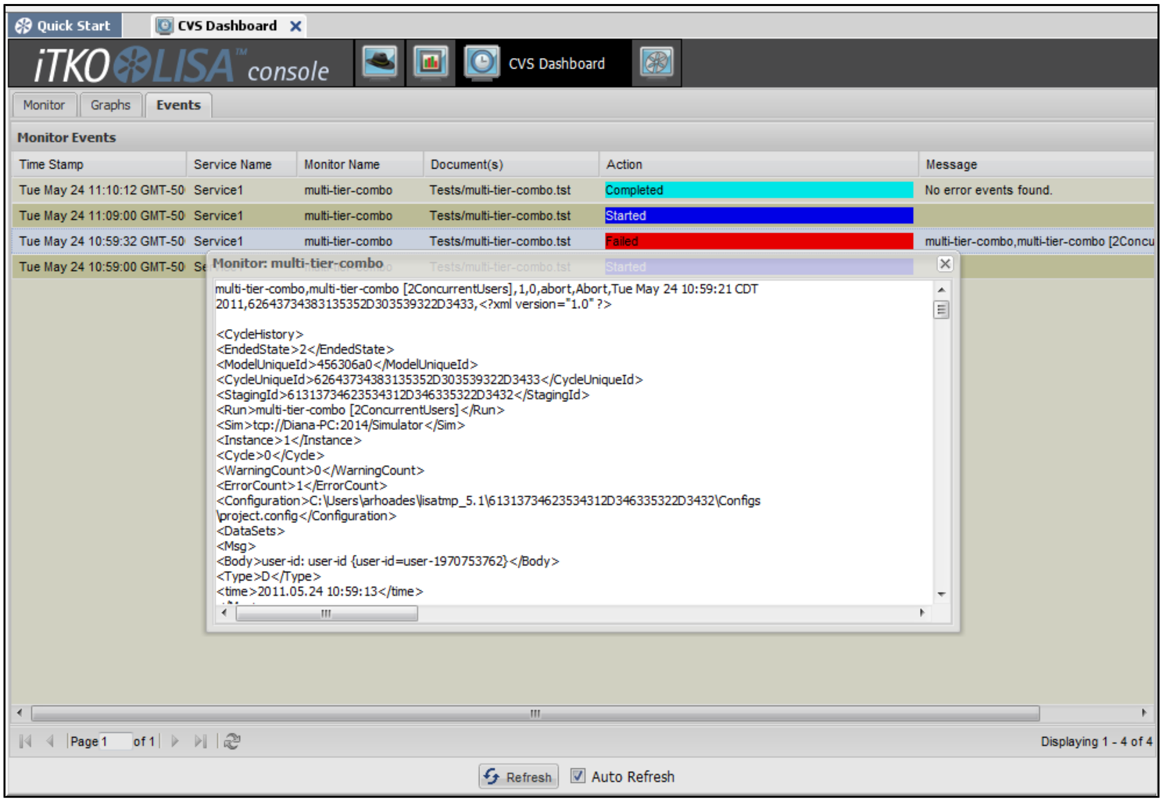
☑ Auto Refresh

The following information is displayed:

- **Time Stamp**: The time the event occurred.
- **Service Name**: The name of the Service in which the monitor resides.
- **Monitor Name**: The name of the monitor in which the event occurred.
- **Document(s)**: A list of the documents associated with the monitor for which the event occurred.
- **Action**: The action reported by the event. Start events are blue, staging error events yellow, completion events green, and failed events are red.
- **Message**: The message reported by the event. If the text is larger than the message cell, you can right-click the cell to invoke the extended view window.

You can display the events in real time by checking the Auto Refresh box at the bottom of the panel, or you can refresh the list manually by clicking the Refresh icon, with the Auto Refresh box cleared.

You can adjust the column sizes so that all columns can display on the screen. If you double-click on a Message field, you can see the full text of the message.



Monitor Events

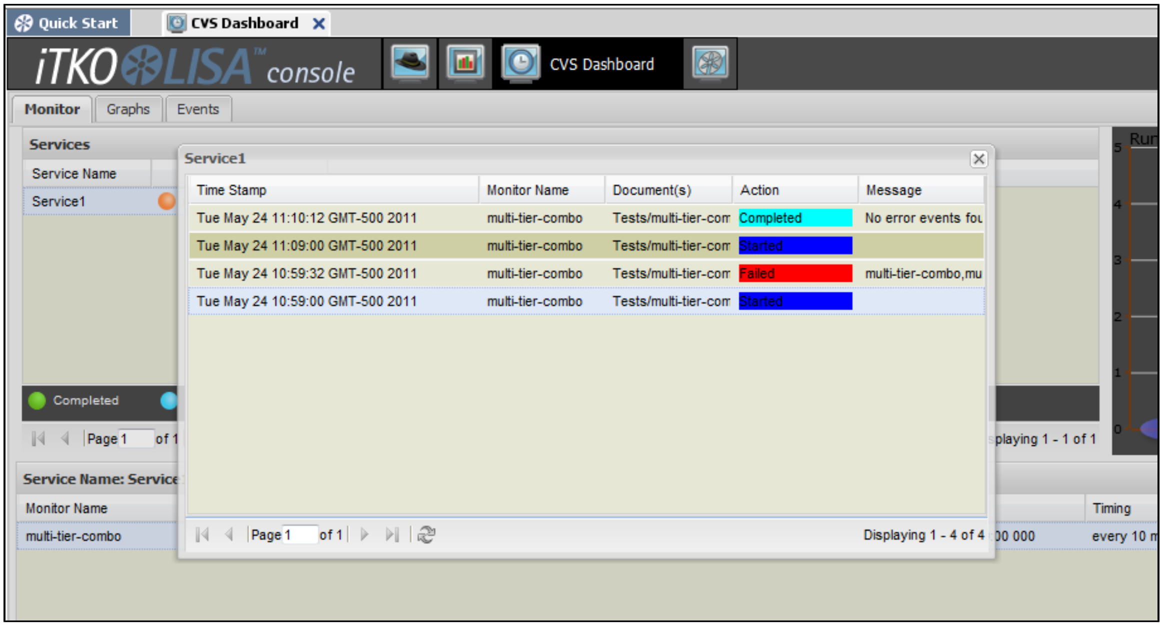
Time Stamp	Service Name	Monitor Name	Document(s)	Action	Message
Tue May 24 11:10:12 GMT-50	Service1	multi-tier-combo	Tests/multi-tier-combo.tst	Completed	No error events found.
Tue May 24 11:09:00 GMT-50	Service1	multi-tier-combo	Tests/multi-tier-combo.tst	Started	
Tue May 24 10:59:32 GMT-50	Service1	multi-tier-combo	Tests/multi-tier-combo.tst	Failed	multi-tier-combo,multi-tier-combo [2Concu
Tue May 24 10:59:00 GMT-50	Se	multi-tier-combo	Tests/multi-tier-combo.tst	Started	

Monitor: multi-tier-combo

```
multi-tier-combo,multi-tier-combo [2ConcurrentUsers],1,0,abort,Abort,Tue May 24 10:59:21 CDT
2011,62643734383135352D303539322D3433,<?xml version="1.0" ?>

<CycleHistory>
<EndedState>2</EndedState>
<ModelUniqueId>456306a0</ModelUniqueId>
<CycleUniqueId>62643734383135352D303539322D3433</CycleUniqueId>
<StagingId>61313734623534312D346335322D3432</StagingId>
<Run>multi-tier-combo [2ConcurrentUsers]</Run>
<Sim>tcp://Diana-PC:2014/Simulator</Sim>
<Instance>1</Instance>
<Cycle>0</Cycle>
<WarningCount>0</WarningCount>
<ErrorCount>1</ErrorCount>
<Configuration>C:\Users\arhoades\lisatmp_5.1\61313734623534312D346335322D3432\Configs
\project.config</Configuration>
<DataSets>
<Msg>
<Body>user-id: user-id {user-id=user-1970753762}</Body>
<Type>D</Type>
<time>2011.05.24 10:59:13</time>
```

Test Cycle with Errors

At times, the test cycle may fail. The cycle that has errors can be seen in the Actions list in different color. You can double-click any event, to see the associated messages in an expanded window.



Service1

Time Stamp	Monitor Name	Document(s)	Action	Message
Tue May 24 11:10:12 GMT-500 2011	multi-tier-combo	Tests/multi-tier-com	Completed	No error events fou
Tue May 24 11:09:00 GMT-500 2011	multi-tier-combo	Tests/multi-tier-com	Started	
Tue May 24 10:59:32 GMT-500 2011	multi-tier-combo	Tests/multi-tier-com	Failed	multi-tier-combo,mu
Tue May 24 10:59:00 GMT-500 2011	multi-tier-combo	Tests/multi-tier-com	Started	

Page 1 of 1
Displaying 1 - 4 of 4

Deploying a Monitor to CVS

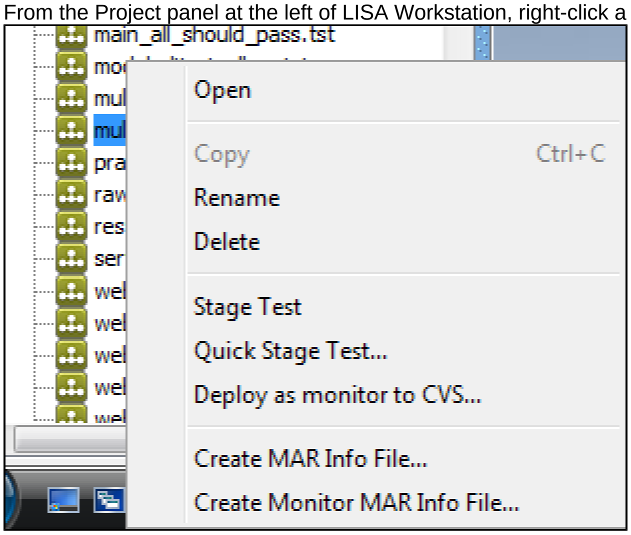
You can deploy a monitor to the Continuous Validation Service (CVS) by using any of the following approaches:

- Deploying a Monitor through LISA Workstation
- Deploying a Monitor through the CVS Dashboard
- Deploying a Monitor through the cvsMonitors Directory
- Deploying a Monitor through CVS Manager

Deploying a Monitor through LISA Workstation

The only prerequisite for this approach is the existence of a test case or suite.

1. From the Project panel at the left of LISA Workstation, right-click a test case or suite and select Deploy as monitor to CVS.

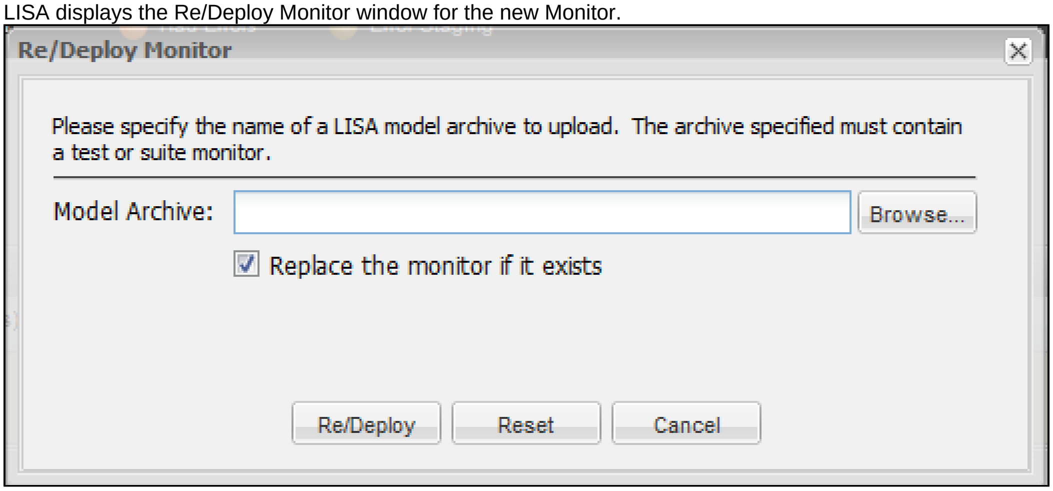


2. You will see a collection of tabs to provide monitor information, the same as needed for Creating Monitor MAR Info Files. See that section of the documentation for information, then click Deploy to deploy the monitor.

Deploying a Monitor through the CVS Dashboard

Before you can perform this procedure, a Model Archive for the monitor must have been created.

1. In the Monitor tab of the CVS Dashboard, click the Re/Deploy Monitor button.
LISA displays the Re/Deploy Monitor window for the new Monitor.

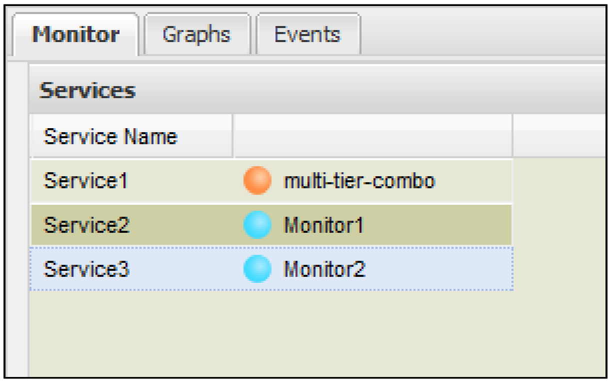


Please specify the name of a LISA model archive to upload. The archive specified must contain a test or suite monitor.

Model Archive:

☑ Replace the monitor if it exists

2. In the **Model Archive** area, enter the name of the MAR file.
3. If the MAR has previously been deployed, select or clear the Replace the monitor if it exists check box.
4. Click Re/Deploy.
5. The Monitor is added to the CVS Dashboard.



Service Name	
Service1	● multi-tier-combo
Service2	● Monitor1
Service3	● Monitor2

Deploying a Monitor through the cvsMonitors Directory

Before you can perform this procedure, a Model Archive for the monitor must have been created.

1. Go to the LISA_HOME/cvsMonitors directory.
2. Add the MAR file to the directory. At a later time, you can update the monitor by placing a new version of the MAR file in the same directory.

Deploying a Monitor through CVS Manager

For details on the command-line option for deploying a CVS Monitor, see CVS Manager.

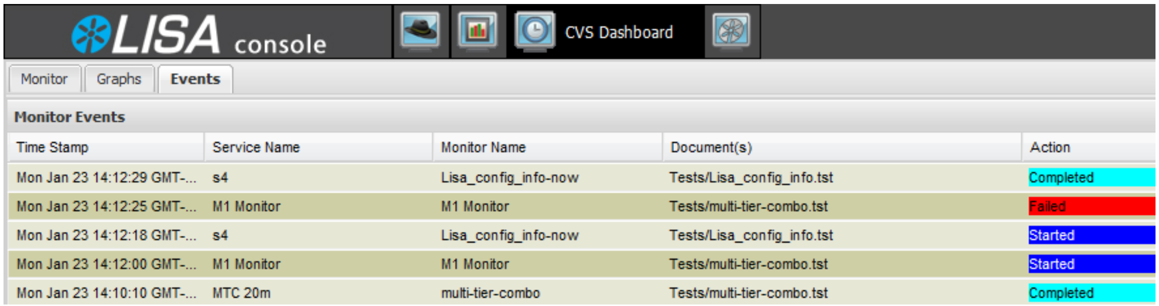
Running a Monitor Immediately

When you add the services in the top panel, they will be run automatically in the background at the scheduled time intervals.

If you want to run a particular monitor immediately, you can use the toolbar.

To run a monitor immediately

1. Deploy the monitor.
2. When it gets listed in the bottom panel, select it.
3. Click Run Monitor Immediately to run the monitor immediately. You will not see the monitor run in the bottom panel.
4. Go to the Events tab of the CVS Dashboard to see the monitor that has run, with a suffix **-now** (for example, **Test 3-now**).



Monitor Events

Time Stamp	Service Name	Monitor Name	Document(s)	Action
Mon Jan 23 14:12:29 GMT-...	s4	Lisa_config_info-now	Tests/Lisa_config_info.tst	Completed
Mon Jan 23 14:12:25 GMT-...	M1 Monitor	M1 Monitor	Tests/multi-tier-combo.tst	Failed
Mon Jan 23 14:12:18 GMT-...	s4	Lisa_config_info-now	Tests/Lisa_config_info.tst	Started
Mon Jan 23 14:12:00 GMT-...	M1 Monitor	M1 Monitor	Tests/multi-tier-combo.tst	Started
Mon Jan 23 14:10:10 GMT-...	MTC 20m	multi-tier-combo	Tests/multi-tier-combo.tst	Completed

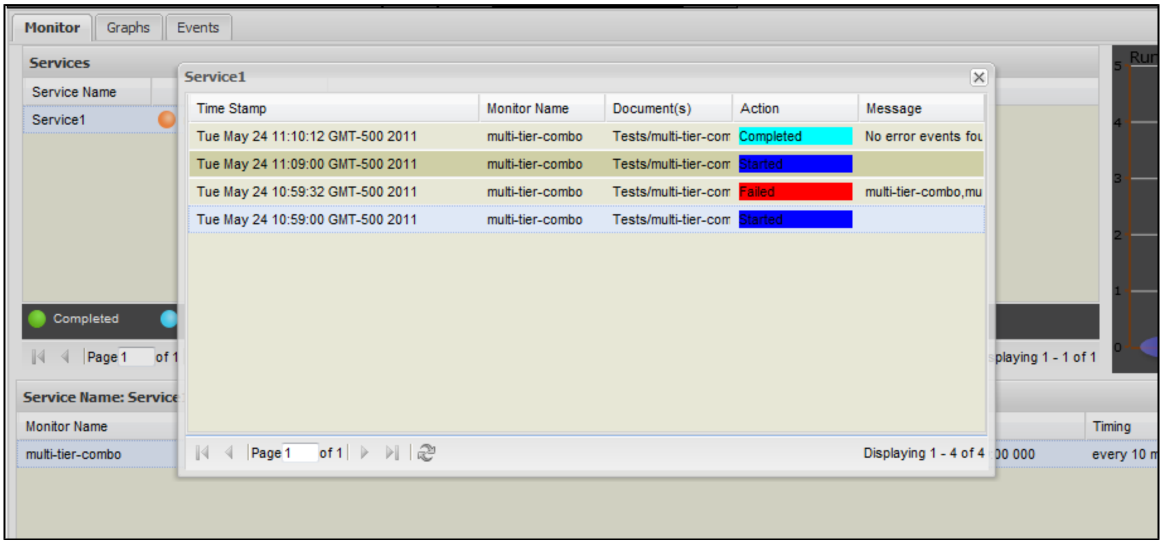
Viewing Test Details

You can view the details of a test run within the CVS Dashboard.

To view test details

- Double-click the service to see the service-related details.
- Double-click the monitor whose details you want to view in the top panel.

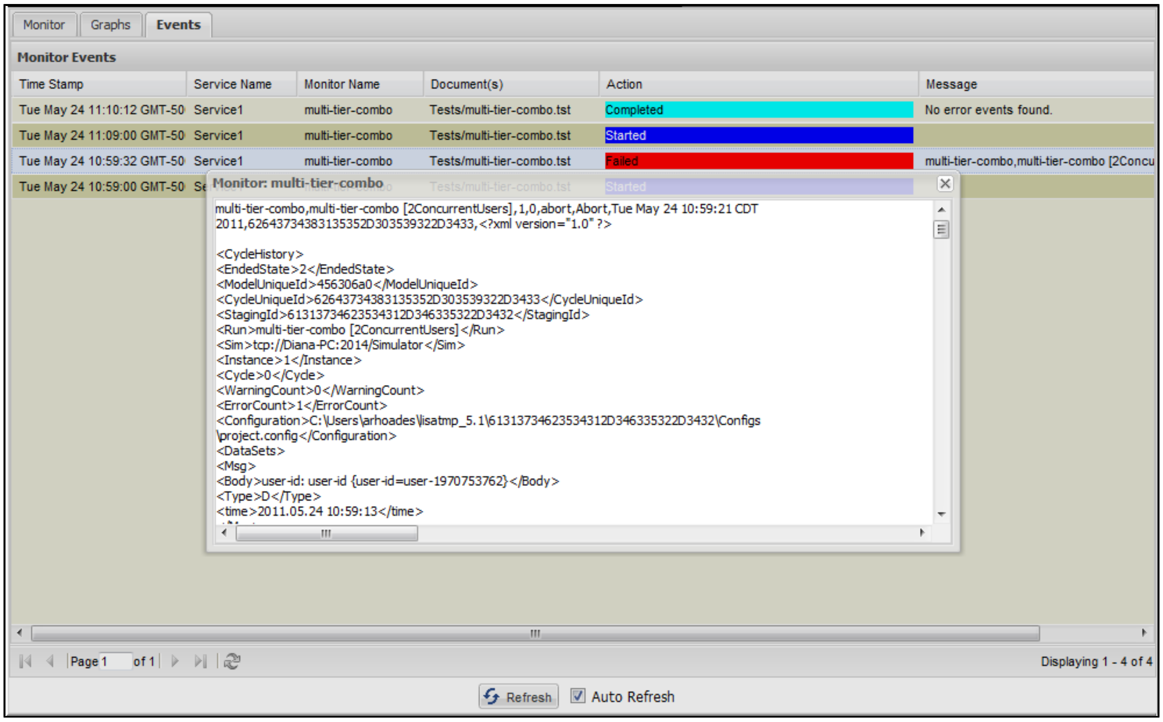
This will open the test-related detail window.



Service1

Time Stamp	Monitor Name	Document(s)	Action	Message
Tue May 24 11:10:12 GMT-500 2011	multi-tier-combo	Tests/multi-tier-com	Completed	No error events fou
Tue May 24 11:09:00 GMT-500 2011	multi-tier-combo	Tests/multi-tier-com	Started	
Tue May 24 10:59:32 GMT-500 2011	multi-tier-combo	Tests/multi-tier-com	Failed	multi-tier-combo,mu
Tue May 24 10:59:00 GMT-500 2011	multi-tier-combo	Tests/multi-tier-com	Started	

Also, you can display the Events tab in the CVS Dashboard. You can adjust the column sizes so that all columns can display on the screen. If you double-click on a Message field, you can see the full text of the message.



Time Stamp	Service Name	Monitor Name	Document(s)	Action	Message
Tue May 24 11:10:12 GMT-50	Service1	multi-tier-combo	Tests/multi-tier-combo.tst	Completed	No error events found.
Tue May 24 11:09:00 GMT-50	Service1	multi-tier-combo	Tests/multi-tier-combo.tst	Started	
Tue May 24 10:59:32 GMT-50	Service1	multi-tier-combo	Tests/multi-tier-combo.tst	Failed	multi-tier-combo,multi-tier-combo [2Concu
Tue May 24 10:59:00 GMT-50	Se	multi-tier-combo	Tests/multi-tier-combo.tst	Started	

Monitor: multi-tier-combo

```
multi-tier-combo,multi-tier-combo [2ConcurrentUsers],1,0,abort,Abort,Tue May 24 10:59:21 CDT
2011,62643734383135352D303539322D3433,<?xml version="1.0" ?>

<CycleHistory>
<EndedState>2</EndedState>
<ModelUniqueId>456306a0</ModelUniqueId>
<CycleUniqueId>62643734383135352D303539322D3433</CycleUniqueId>
<StagingId>61313734623534312D346335322D3432</StagingId>
<Run>multi-tier-combo [2ConcurrentUsers]</Run>
<Sim>tcp://Diana-PC:2014/Simulator</Sim>
<Instance>1</Instance>
<Cycle>0</Cycle>
<WarningCount>0</WarningCount>
<ErrorCount>1</ErrorCount>
<Configuration>C:\Users\arhoades\lisatmp_5.1\61313734623534312D346335322D3432\Configs
\project.config</Configuration>
<DataSets>
<Msg>
<Body>user-id: user-id {user-id=user-1970753762}</Body>
<Type>D</Type>
<time>2011.05.24 10:59:13</time>
```

Any reports generated during scheduled test runs are available and can be seen in the Report Viewer.

Email Notification Settings

Every time you schedule a new test in the CVS utility, you must deploy a monitor in the CVS Dashboard.

Within the monitor window, you can also set an email address, so that you receive an email notification every time the monitor is run within the specified time period.

To set the email notification, you must update the **lisa.properties** file.

You also must change the mail server configuration and enter the mail host as shown in the following example.

```
# If you use performance monitoring alerts, this is the "from" email address of those alerts
# lisa.alert.email.emailAddr=lisa@itko.com

# And this is the email server we will attempt to route emails with (smtp server)
# lisa.alert.email.defHosts=localhost
```

Remove the # from the first column of the file to uncomment this information in the **lisa.properties** file.

Restart LISA to set the mail server configuration.

- **Notification email:** Enter the email address for email notification of the test run result.

Every time the test is run, you will receive an email.

CVS Manager

The CVS Manager command-line utility lets you manage the set of monitors deployed to the Continuous Validation Service. The utility is located in the **LISA_HOME/bin** directory.

This utility has the following format:

```
CVSManager [-h] [-m registry-spec] [-d archive-file] [-r archive-file] [-l] [-D] [-A] [-e] [x] [-X]
[-s name] [-n name] [-u username] [-p password] [--version]
```

-h, --help

Displays help text.

-m *registry-spec*, --registry=*registry-spec*

Defines the registry to which to connect.

-d *archive-file*, --deploy=*archive-file*

Deploys the specified model archive to CVS as a monitor. The monitor defined in the archive must refer to a monitor and service name combination that does not already exist.

-r *archive-file*, --redeploy=*archive-file*

Redeploys the specified model archive to CVS as a monitor. The monitor defined in the archive must refer to a monitor and service name combination that exists.

-l, --list

Lists the currently deployed monitors with interesting information about each.

-D, --pause

Pauses the scheduled execution of the indicated monitor.

-A, --resume

Resumes the scheduled execution of the indicated monitor.

-e, --execute-now

Causes the indicated monitor to be executed immediately, regardless of its schedule. This action does not affect any scheduled executions of the monitor.

-x, --remove

Removes a monitor from CVS. Use the service name and monitor name arguments to indicate which monitor to remove.

-X, --remove-all

Removes all monitors from CVS. If a service name is specified, then only monitors defined with that service name are removed.

-s *name*, --service-name=*name*

Defines the service name for the monitors to affect.

-n *name*, --monitor-name=*name*

Defines the monitor name to affect.

-u *username*, --username=*username*

Defines the LISA security user name. If access control (ACL) is enabled, then this option is required.

-p *password*, --password=*password*

Defines the LISA security password. If access control (ACL) is enabled, then this option is required.

--version

Print the version number.

Example: Deploy Monitor

This example deploys a monitor to CVS.

```
CVSManager -d monitor.mar
```

Example: Delete Monitor

This example deletes that same monitor (assuming the service and monitor names).

```
CVSManager -x -s OrderManager -n CheckOrders
```

This example deletes all monitors in the OrderManager service.

```
CVSManager -X -s OrderManager
```

This example deletes all monitors.

```
CVSManager -X
```

Reports

There are a wealth of features related to the generation and capture of data for the purpose of reporting results. You determine what data LISA collects by specifying reporting parameters in one of three areas:

- Quick Tests
- Staging Documents
- Test Suites

You can specify the specific events or metrics you want collected for each test case or test suite you run, resulting in maximum flexibility and control over the reporting data you collect. You choose between two report generators that store reporting data in either a database or an XML file. You also determine how long the reporting data should be kept.

After the reports have been generated, they can be viewed and managed at a later date, shared with colleagues, or exported to other locations.

In this section, the following topics are covered:

Report Generator Types
Opening the Reporting Portal
Reporting Portal Layout
Filtering Reports
Viewing Reports
Exporting Reports
Auto Expiring Reports
Changing Reporting Databases

Report Generator Types

You can select the following types of report generators in the Reports tab of the Staging Document Editor or the Test Suite Editor.

- Default Report Generator
- Load Test Report Generator
- XML Report Generator

Default Report Generator

The default report generator captures functional, performance, and metric information and publishes that data to the reporting database referenced by the registry. The reporting portal uses the reporting database.

For load testing, check the Record Performance Metrics in the Parameters section:

- **Record All Events**: If selected, will record all events.
- **Record Properties Set/Referenced**: If selected, will record the set or referenced properties.
- **Record Performance Metrics**: If selected, will record the performance metrics.
- **Record Request/Response**: If selected, will record the request and response.

Load Test Report Generator

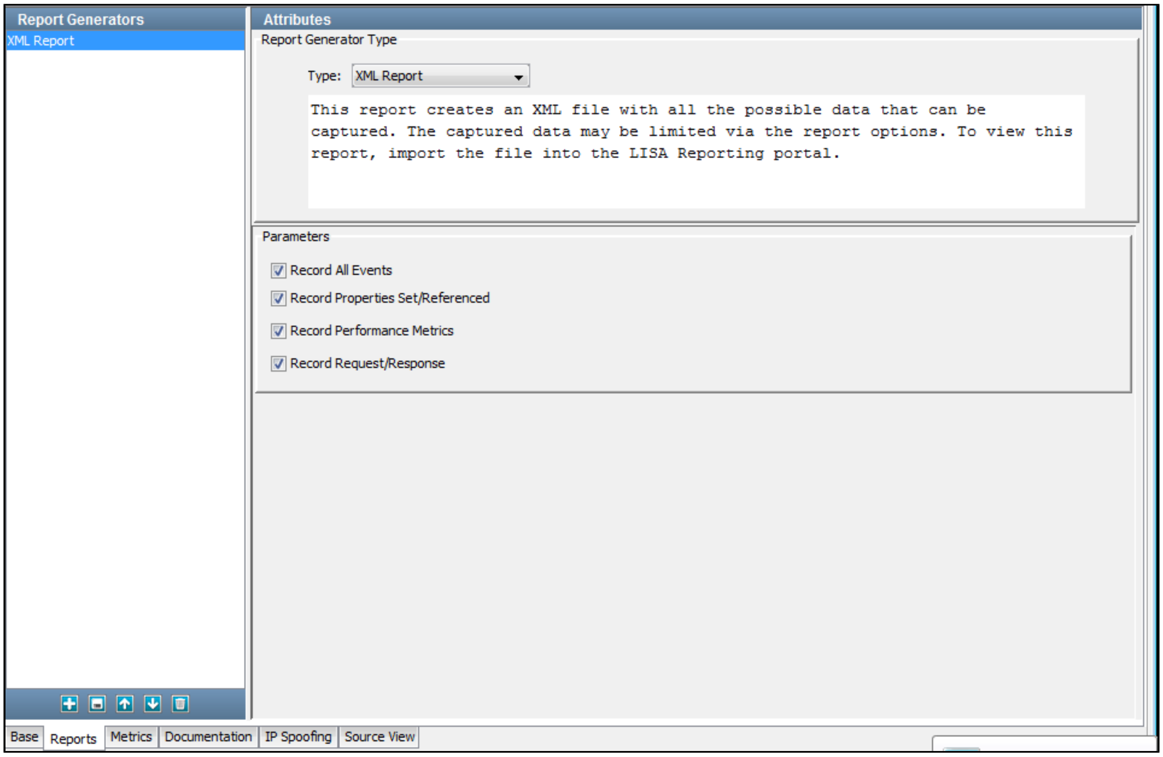
The Load Test report generator is designed for load tests with thousands of virtual users.

This report captures load metrics but not step-level metrics (there would be too much data and the reporting database would slow down the test).

XML Report Generator

This report generator creates an XML file with all the possible data that can be captured. The captured data can be limited by using the report options in the Test Suite Editor or Staging Document Editor. To view this report, import the file into the LISA Reporting portal.

The data in this table can be used for any custom reporting needs.



This report creates an XML file with all the possible data that can be captured. The captured data may be limited via the report options. To view this report, import the file into the LISA Reporting portal.

☑ Record All Events
☑ Record Properties Set/Referenced
☑ Record Performance Metrics
☑ Record Request/Response

After your tests and/or suites are complete, you can view the report data in the reporting console. To export the XML data to a file, see Exporting Reports.

Opening the Reporting Portal

You can open the Reporting Portal from LISA Workstation or from a web browser.

To open the Reporting Portal from LISA Workstation

- From the main menu, select View > Reporting Console.

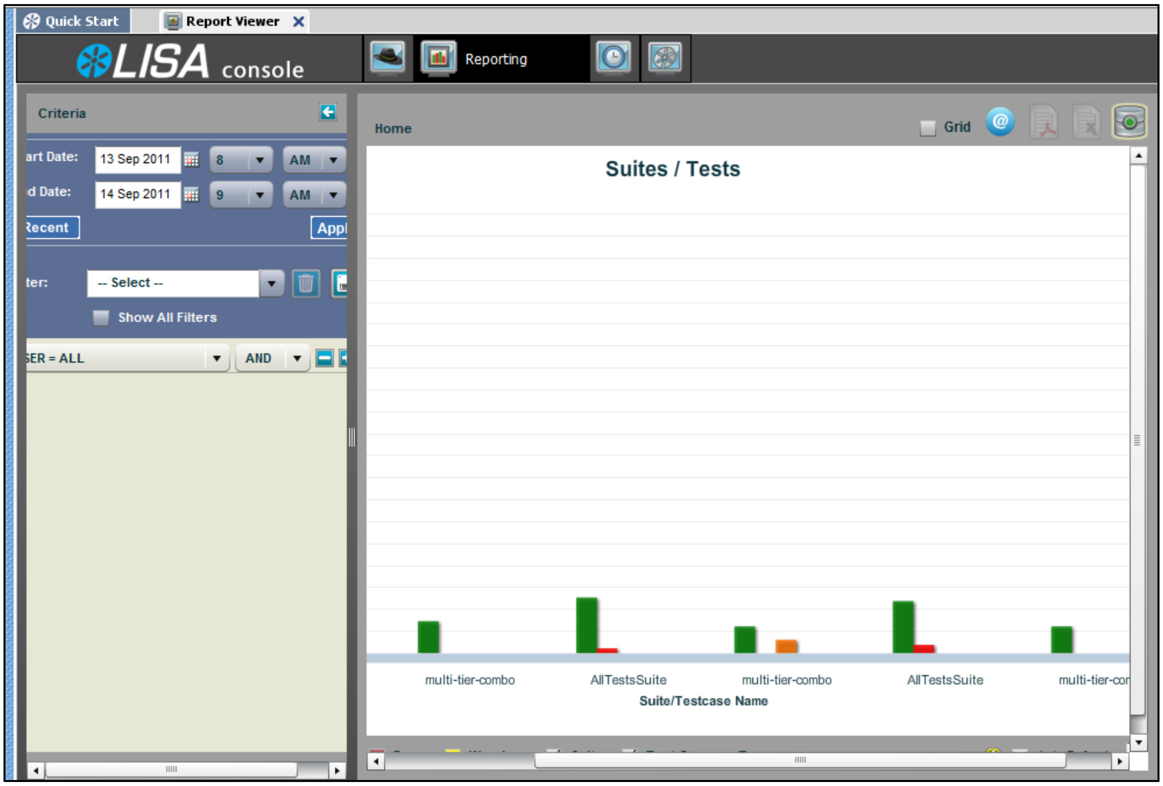
To open the Reporting Portal from a web browser

1. Ensure that the registry is running.
2. Enter **http://localhost:1505/** in a web browser. If the registry is located on a remote computer, replace **localhost** with the name or IP address of the computer.
   The LISA Console appears.
3. Click Reporting Dashboard.

Reporting Portal Layout

The Reporting Portal lets you view all the reports that have run earlier in LISA Workstation. You can configure the reports either through the staging documents, quick tests, running test cases or test suites.

This section looks at the reports generated by running the multi-tier-combo test case, which is located in the examples directory.



art Date: 13 Sep 2011 8 AM

d Date: 14 Sep 2011 9 AM

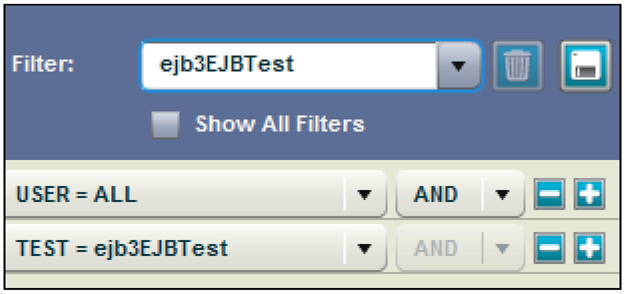
ter: -- Select --

SER = ALL AND

Criteria

In the left panel, you can select the date and time criteria and select the filters for use.

- **Start date/End date**: Select the start/end date by clicking the Calendar ▦ icon. By default, the start and end dates will be in the range of the last hour. Select the start and end time by clicking the 1-12 and AM/PM drop-downs. After entering start and end dates, select the Apply button to apply your changes.
- The Recent button will automatically reset the date and time criteria to find the latest report information available (the last test/suite run) and the hour previous from it.
- **Filters**: You can create filters of your choice here. Enter the name of the filter and click Save. You can also delete a filter by clicking Delete. Select Show All Filters to show all filters created by all users, not just ones you have created.



Filter: ejb3EJBTest

□ Show All Filters

USER = ALL AND

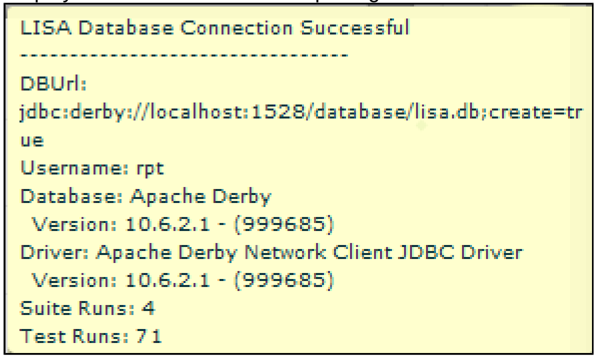
TEST = ejb3EJBTest AND

Select from the AND/OR operators for the criteria and click Add ⊞ or Delete ⊟.

Right Panel

The right panel consists of the graphs charted depending on the selected criteria. For more information, see Reports - Graphical View

The Reporting Toolbar is also shown at the top of the right panel.

Icon	Function
☐ Grid ☑ Grid	Changes the report view from the default chart view to the grid view, and back. For more information, see Reports - Grid View.
@	Copies the URL of the report you are viewing to the operating system clipboard so you can share the report with other users.
	Exports the report data to a PDF file or Excel file to view report details in those formats. For more information, see Exporting Reports.
	Displays information about the reporting database.  LISA Database Connection Successful -------------------------------- DBUrl: jdbc:derby://localhost:1528/database/lisa.db;create=true Username: rpt Database: Apache Derby Version: 10.6.2.1 - (999685) Driver: Apache Derby Network Client JDBC Driver Version: 10.6.2.1 - (999685) Suite Runs: 4 Test Runs: 71

You can filter the reports on the results of test cases that have passed or failed. You can also choose to show or hide errors and warnings, and show or hide test suites or test cases. For more information, see Filtering Reports.

The Zoom slider lets you customize the size of the report display.

The Refresh button will refresh the display. To set an auto-refresh, select the Autorefresh check box and set an interval for auto-refresh.

Filtering Reports

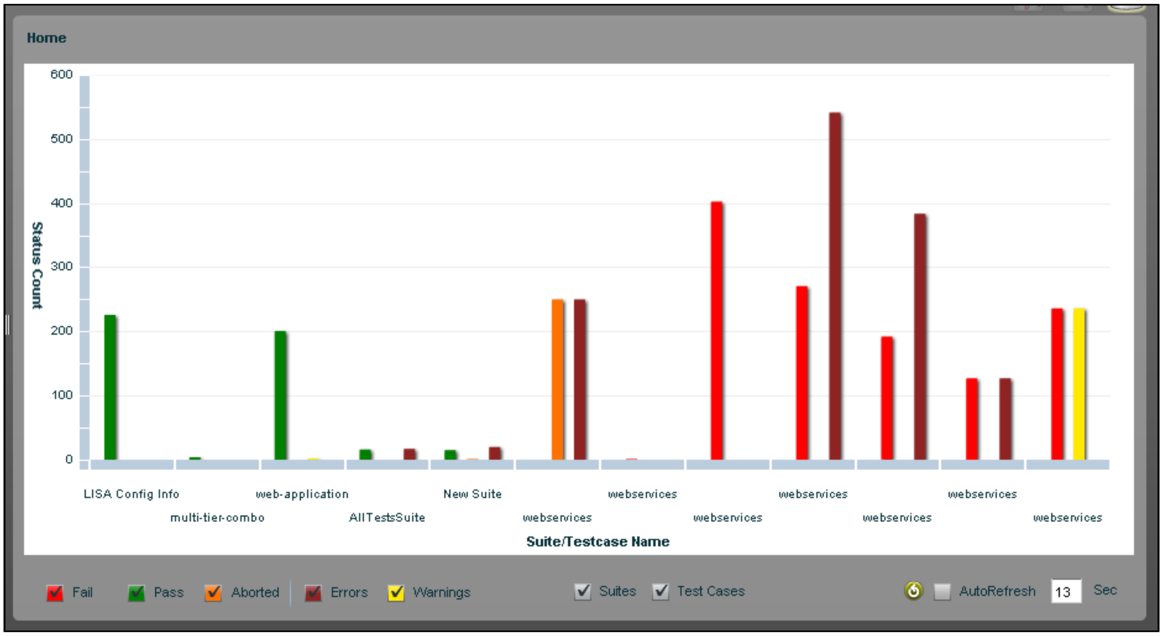
The right panel of the Reporting Portal displays the reports.

You can filter reports by using certain criteria. After you select the criteria, the report viewer will show graphs only for selected criteria.

- ☑ **Pass**: Show the test cases/suites that have passed
- ☑ **Fail**: Show the test cases/suites that have failed
- ☑ **Aborted**: Show the test cases/suites that have aborted
- ☑ **Errors**: Show the test cases/suites that have generated errors
- ☑ **Warnings**: Show the test cases/suites that have generated warnings
- ☑ **Suites**: Show the test suite results

- ☑ **Test Cases**: Show the test case results

The Reporting Portal will show the reports for all the test cases and test suites that have run and that are currently in its database.



In the previous report, there is no search criteria specified as all are clicked. Therefore, you will see the report for all the test cases and test suites that have passed, failed, aborted, had errors, and had warnings.

> ⚠ You can combine the **result** criteria with the **test run** criteria for more specificity. For example, you could select **Fail** and **Error** and **Test Case** to see only failed test cases or those that completed with some errors.

To refresh the reports

- Click the **Refresh** ⟳ icon.

To automatically refresh the reports

1. Select the **AutoRefresh** Auto Refresh 15 icon.
2. Enter the number of seconds after which you want the reports to be refreshed.

Viewing Reports

In the Reports viewer, the reports can be viewed in two formats:

- Graphical View: This is the default view of the reporting portal. You can see all the reports in the graphical format.
- Grid View: You can select the grid view to have the data arranged in a grid format.

Reports - Graphical View

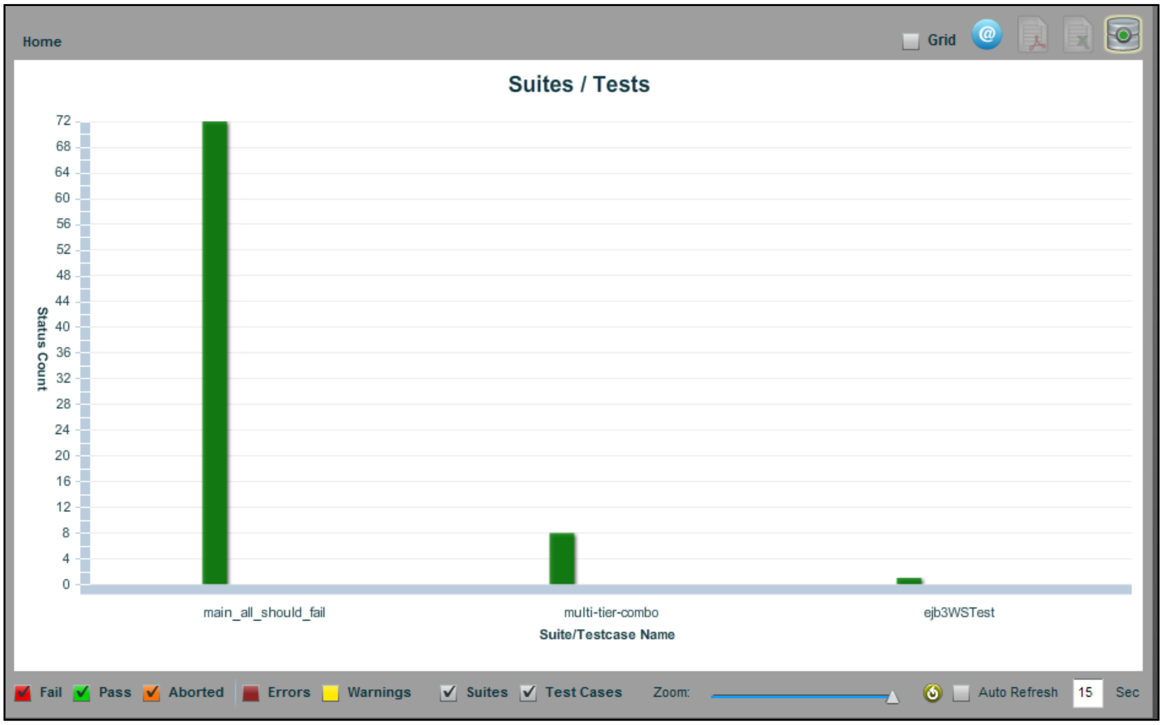
By default, all the reports can be seen as graphs.

Example: Graphical View

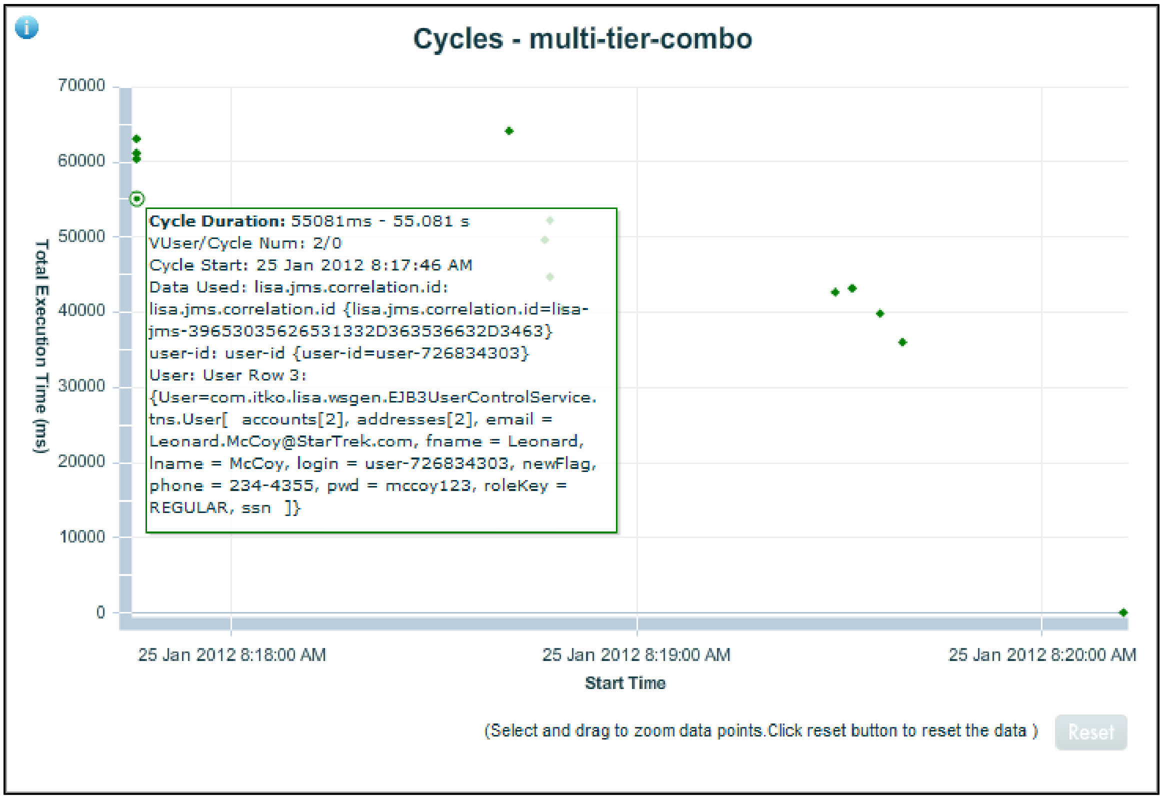
The following reports graph contains these test cases: multi-tier-combo.tst, main_all_should_fail.tst, and ejb3WSTest.tst.

Because not all of the filter criteria boxes are checked, we will see the report for only the test cases that have passed, failed, or aborted.

Mousing over a test case will display an information box that shows the test case name, test execution details like the number of pass/fail tests, and the date of execution.



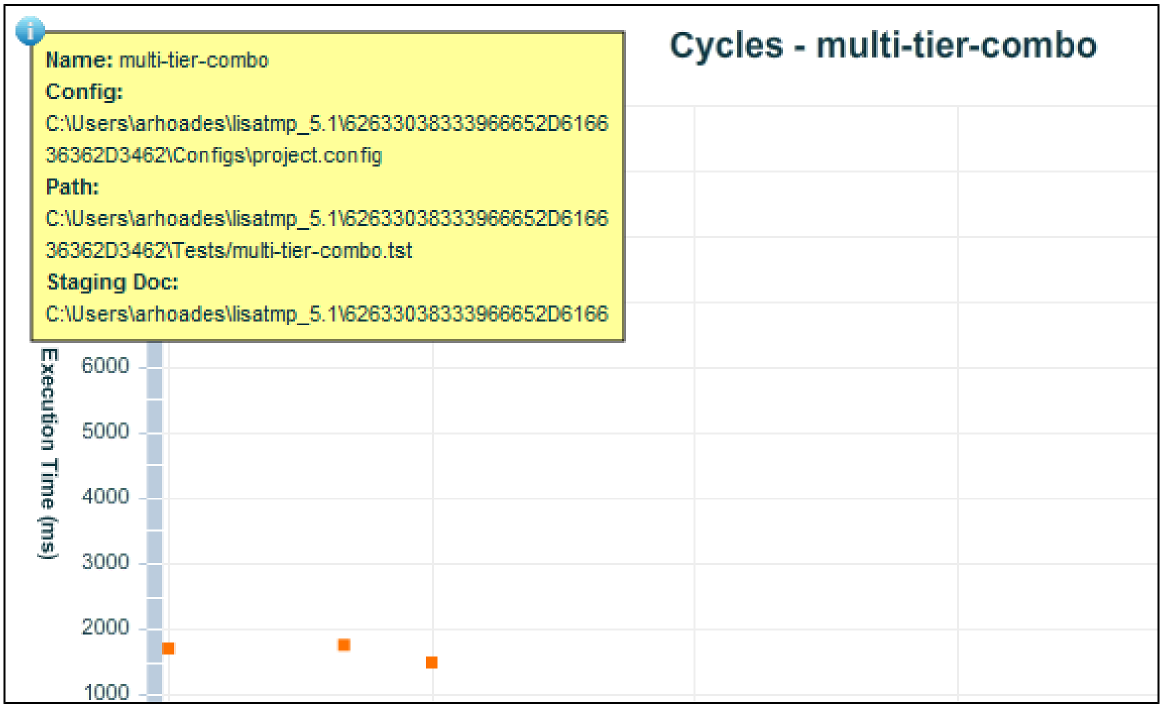
Double-clicking a test case in the graph that passed produces a Cycles diagram for the test case.


Cycles - multi-tier-combo
70000
60000
50000
40000
30000
20000
10000
0
Total Execution Time (ms)
25 Jan 2012 8:18:00 AM
25 Jan 2012 8:19:00 AM
25 Jan 2012 8:20:00 AM
Start Time
(Select and drag to zoom data points.Click reset button to reset the data)
Reset

```
Cycle Duration: 55081ms - 55.081 s
VUser/Cycle Num: 2/0
Cycle Start: 25 Jan 2012 8:17:46 AM
Data Used: lisa.jms.correlation.id:
lisa.jms.correlation.id {lisa.jms.correlation.id=lisa-
jms-39653035626531332D363536632D3463}
user-id: user-id {user-id=user-726834303}
User: User Row 3:
{User=com.itko.lisa.wsgen.EJB3UserControlService.
tns.User[  accounts[2], addresses[2], email =
Leonard.McCoy@StarTrek.com, fname = Leonard,
lname = McCoy, login = user-726834303, newFlag,
phone = 234-4355, pwd = mccoy123, roleKey =
REGULAR, ssn  ]}
```

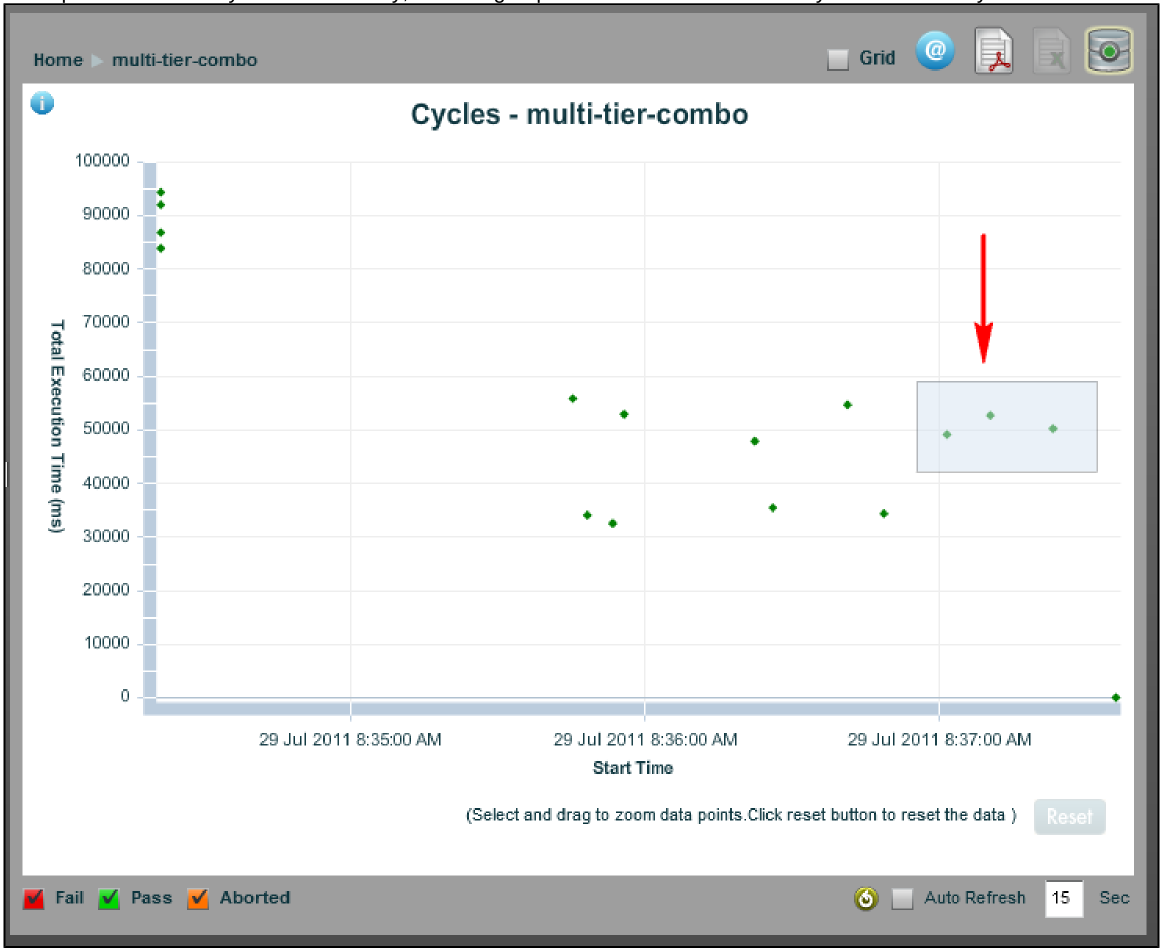
When you move your mouse over these dots, you see that each dot represents a cycle of the test case, and the details for each cycle are shown: response time in milliseconds, the cycle/instance number, the date of the cycle, and the data used.

Clicking the Information button at the upper-left of the screen gives you more information about the test case environment.

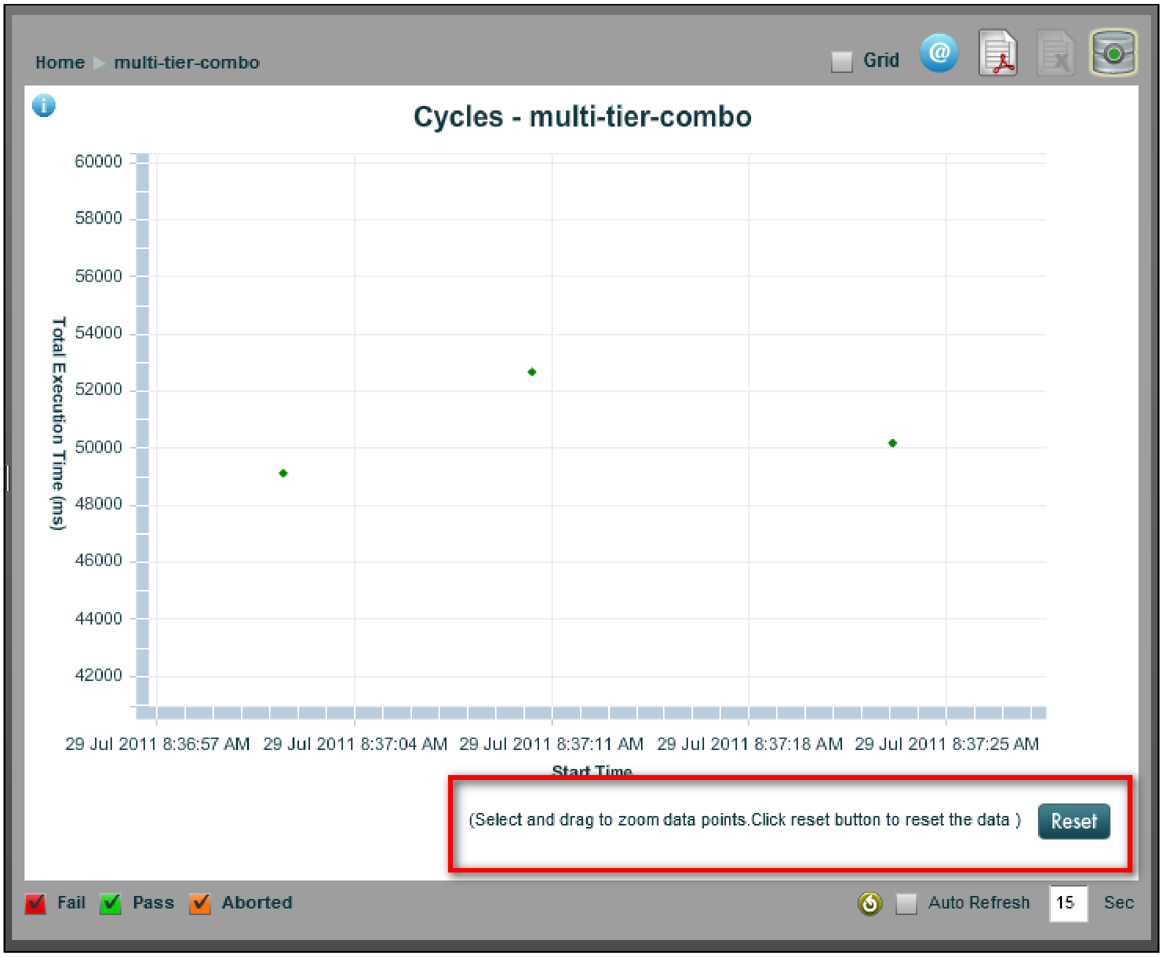


Name: multi-tier-combo
Config:
C:\Users\arhoades\lisatmp_5.1\62633038333966652D616636362D3462\Configs\project.config
Path:
C:\Users\arhoades\lisatmp_5.1\62633038333966652D616636362D3462\Tests/multi-tier-combo.tst
Staging Doc:
C:\Users\arhoades\lisatmp_5.1\62633038333966652D6166

To inspect a subset of cycles more closely, select a group of dots and zoom to see only that subset of cycles.

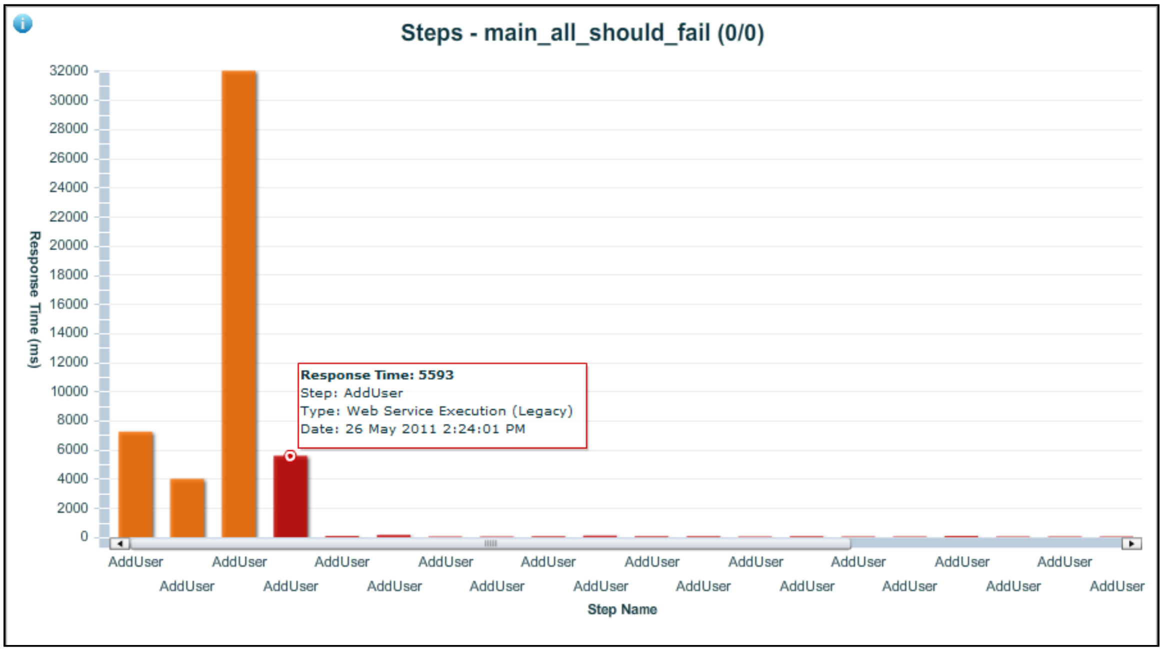


To return to the complete cycle view, click the Reset button.



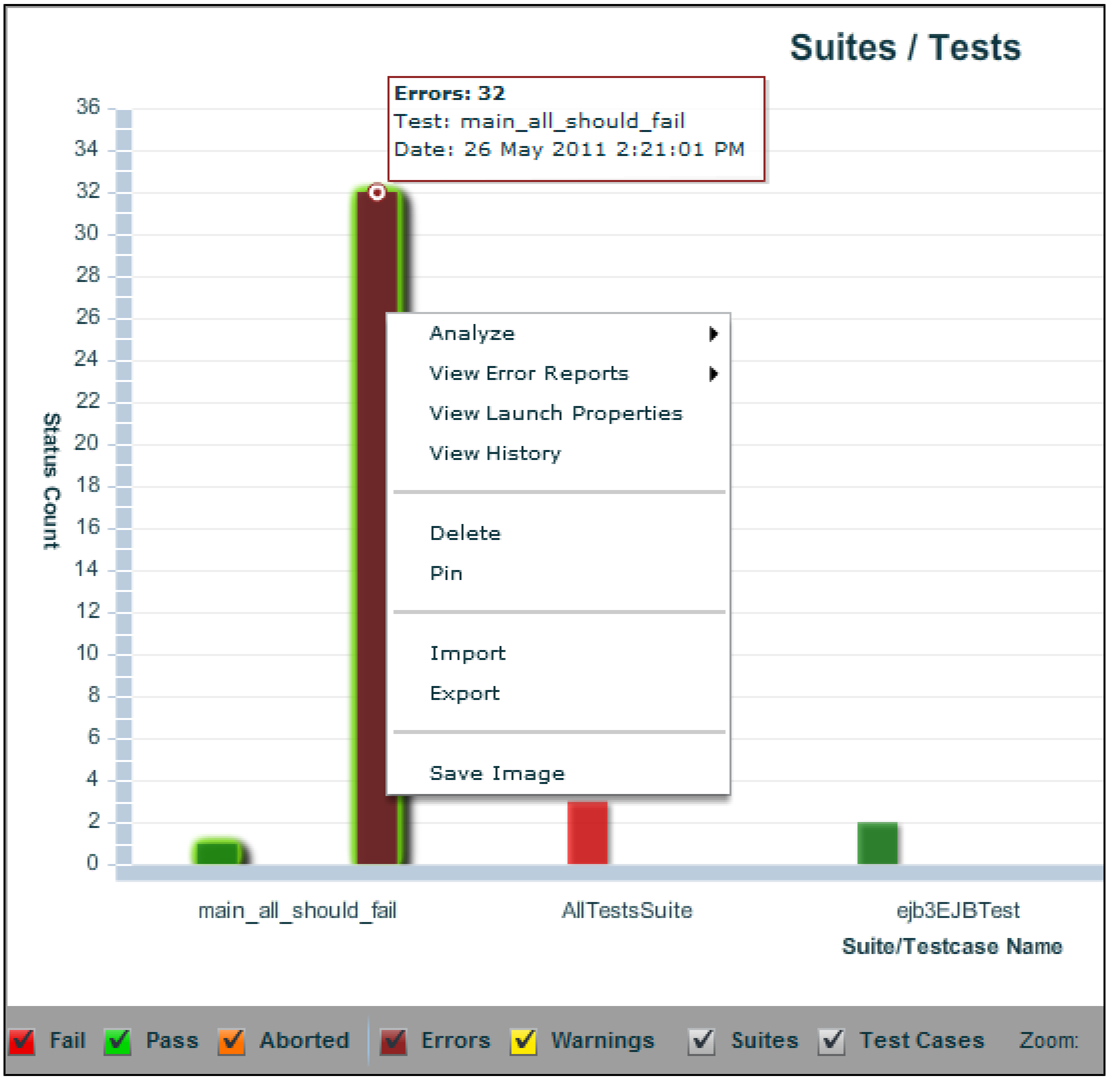
(Select and drag to zoom data points.Click reset button to reset the data)

Double-clicking a test case with errors shows a view of each step of the case, with information about response times.



Response Time: 5593
Step: AddUser
Type: Web Service Execution (Legacy)
Date: 26 May 2011 2:24:01 PM

Right-clicking a test case or suite produces a reporting menu.



Following are options for the reporting menu. To see examples of these reports, see Reports - Graphical View - Examples.

Analyze

- **Top Ten Longest Transactions**: The ten longest-running transactions in a pie chart
- **Average Transaction Response**: Transaction response time in a line chart
- **Metrics**: A variety of LISA event metrics in a line chart by time
- **Requests/Second**: Number of requests per second in a line chart by time
- **Performance Summary**: Average response time and standard deviation by step in a bar chart
- **Cycle Performance Summary**: Cycle time and cycle execution time in milliseconds in a bar chart
- **HTTP Details**: HTTP traffic details in a grid format by name
- **HTTP Summary**: Cumulative HTTP traffic summary in a line chart

Error reports report errors and warnings from steps that did not complete error-free. To see examples of these reports, see Reports - Graphical View - Examples. Reports available are:

View Error Reports

- **Detailed Failures**
- **Detailed Errors**
- **Detailed Warnings**
- **Detailed Aborts**

View Launch Properties

View all launch properties with a timestamp, property name and property value in grid form.

View History

Shrinks the current graph on to the top half of the screen and displays two additional reports: a grid listing of test cases run and their results and a line chart showing historic execution times. The history report shows information on previous runs of the same test case. For an example of a history report, see Reports - Graphical View - Examples.

Delete

Deletes that test case or suite from the report

Pin

Keeps the test case or suite from being auto-deleted after 30 days. Any unpinned test older than 30 days will be auto deleted. For more information, see Auto Expiring Reports.

Import

Imports test case or suite information from an XML file

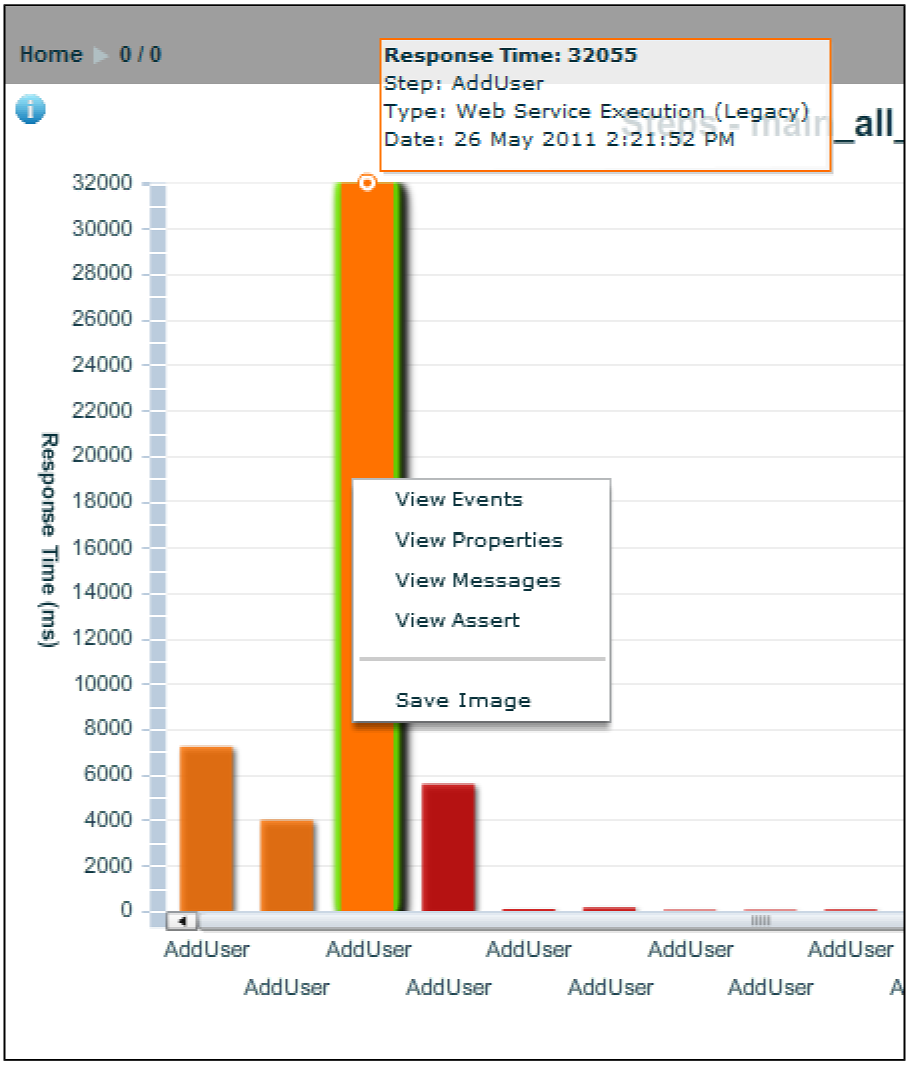
Export

Exports the test case or suite information to an XML file

Save Image

Saves the graphical image as a .png file

Right-click the step bar for a step menu.



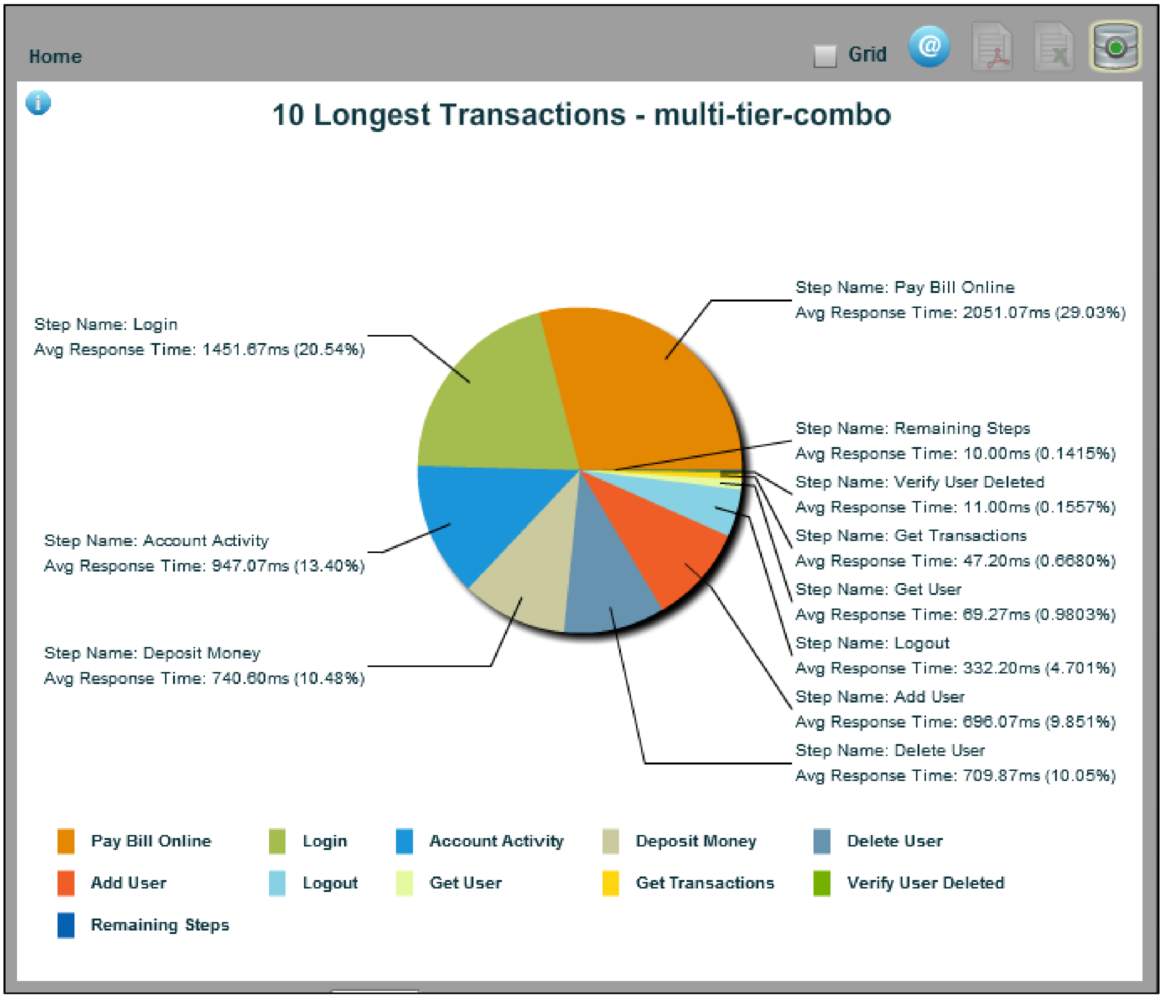
Response Time: 32055
Step: AddUser
Type: Web Service Execution (Legacy)
Date: 26 May 2011 2:21:52 PM

Reports - Graphical View - Examples

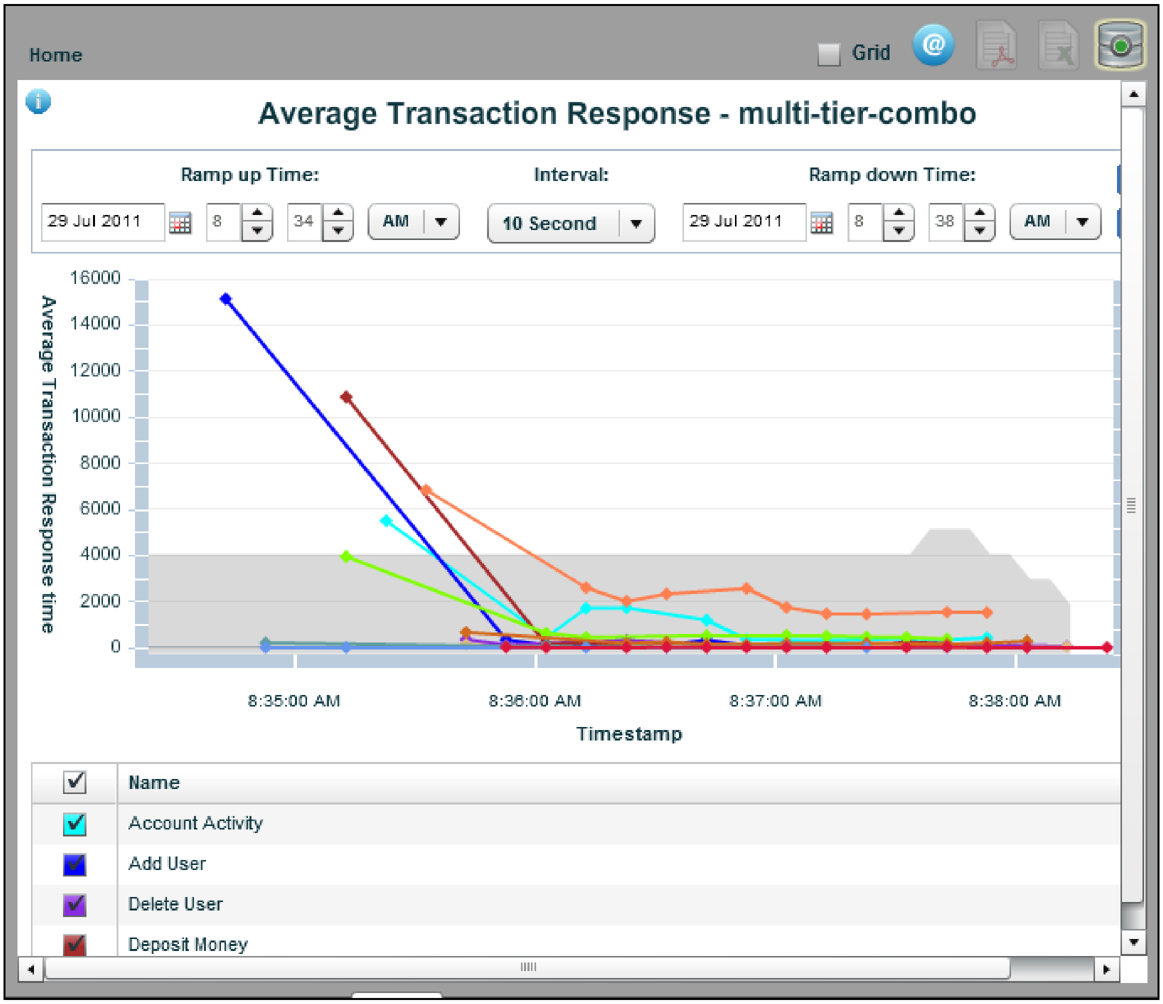
Following are examples of report output available when you right-click from a test/suite report.

Analyze Menu

Top Ten Longest Transactions


Home
Grid
10 Longest Transactions - multi-tier-combo
Step Name: Pay Bill Online
Avg Response Time: 2051.07ms (29.03%)
Step Name: Login
Avg Response Time: 1451.67ms (20.54%)
Step Name: Remaining Steps
Avg Response Time: 10.00ms (0.1415%)
Step Name: Verify User Deleted
Avg Response Time: 11.00ms (0.1557%)
Step Name: Get Transactions
Avg Response Time: 47.20ms (0.6680%)
Step Name: Account Activity
Avg Response Time: 947.07ms (13.40%)
Step Name: Get User
Avg Response Time: 69.27ms (0.9803%)
Step Name: Logout
Avg Response Time: 332.20ms (4.701%)
Step Name: Deposit Money
Avg Response Time: 740.60ms (10.48%)
Step Name: Add User
Avg Response Time: 696.07ms (9.851%)
Step Name: Delete User
Avg Response Time: 709.87ms (10.05%)
Pay Bill Online
Login
Account Activity
Deposit Money
Delete User
Add User
Logout
Get User
Get Transactions
Verify User Deleted
Remaining Steps

Average Transaction Response



Average Transaction Response - multi-tier-combo

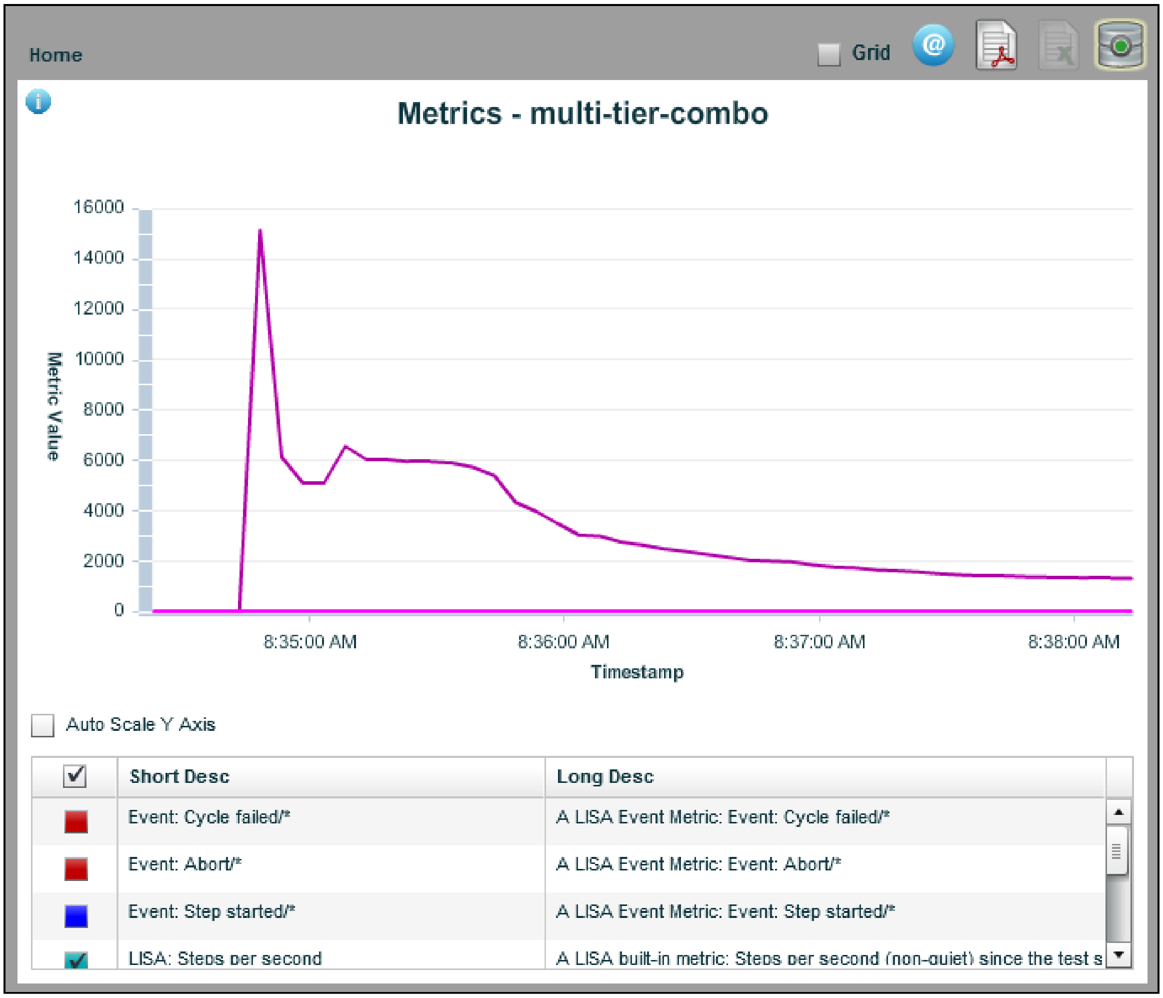
Ramp up Time: 29 Jul 2011 8 34 AM

Interval: 10 Second

Ramp down Time: 29 Jul 2011 8 38 AM

☑	Name
☑	Account Activity
☑	Add User
☑	Delete User
☑	Deposit Money

Metrics

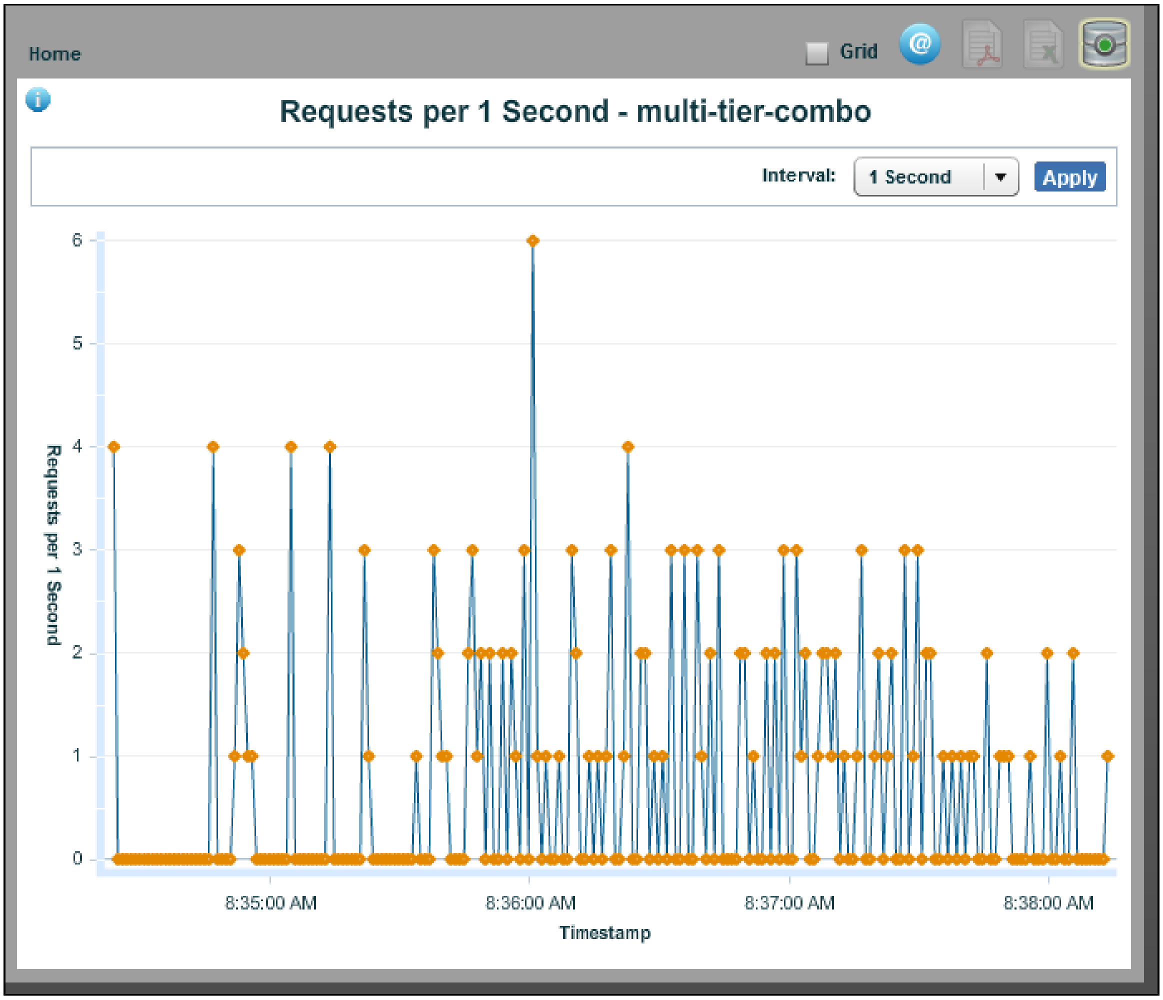


Metrics - multi-tier-combo

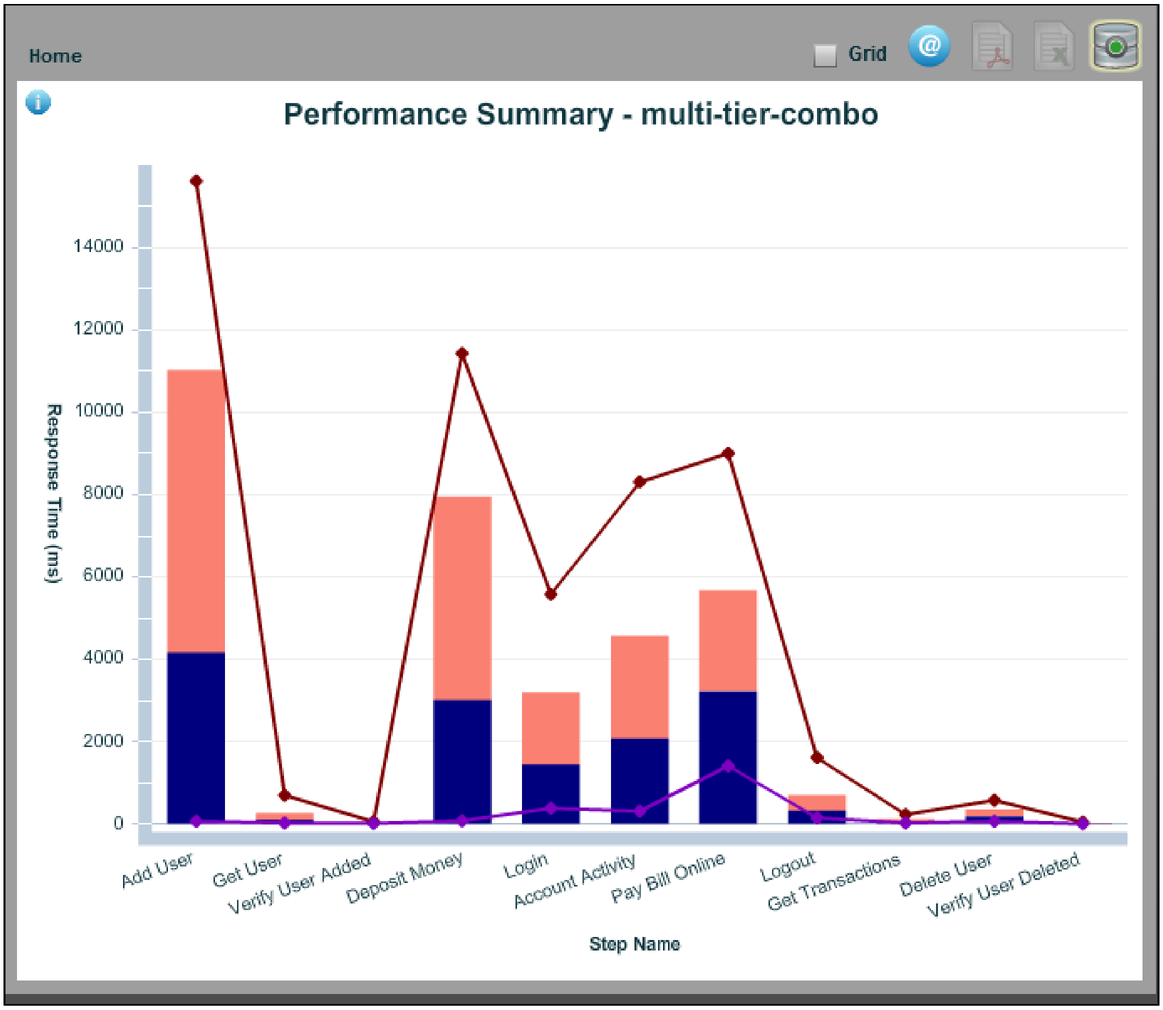
□ Auto Scale Y Axis

☑	Short Desc	Long Desc
🟥	Event: Cycle failed/*	A LISA Event Metric: Event: Cycle failed/*
🟥	Event: Abort/*	A LISA Event Metric: Event: Abort/*
🟦	Event: Step started/*	A LISA Event Metric: Event: Step started/*
☑	LISA: Steps per second	A LISA built-in metric: Steps per second (non-quiet) since the test s

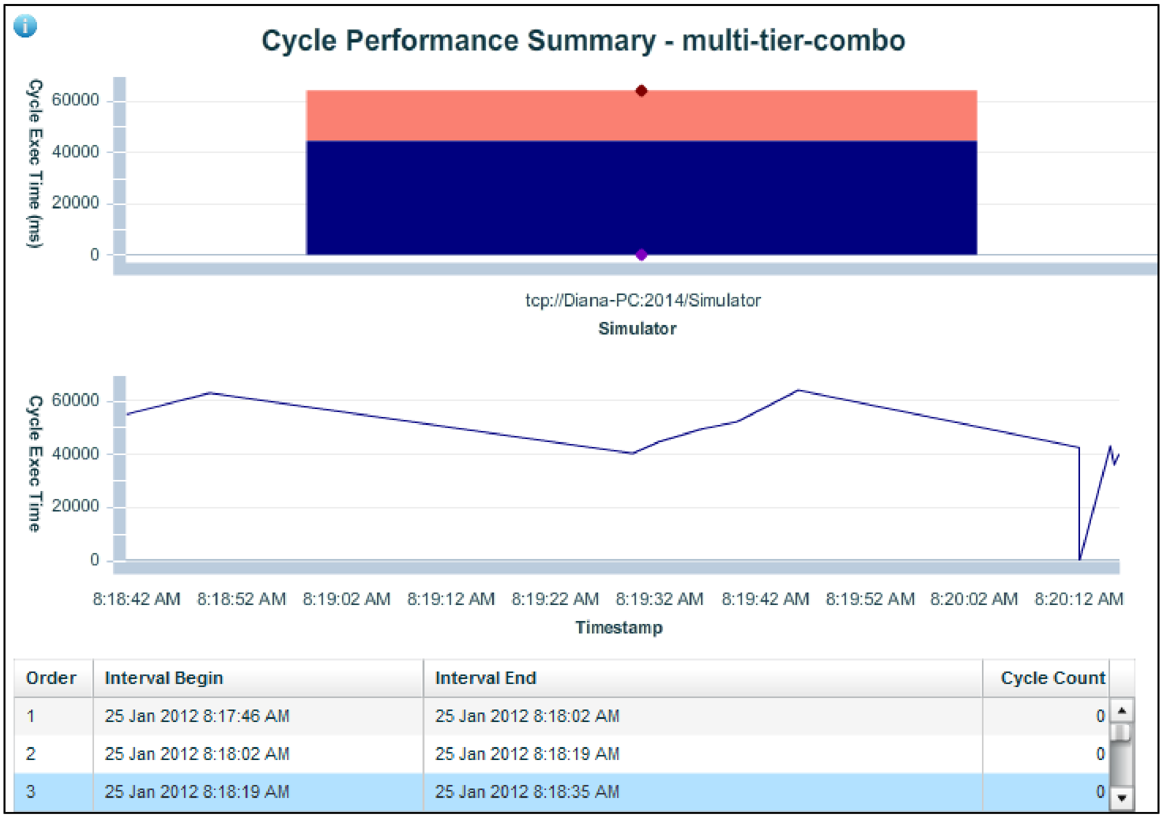
Requests/Second



Performance Summary



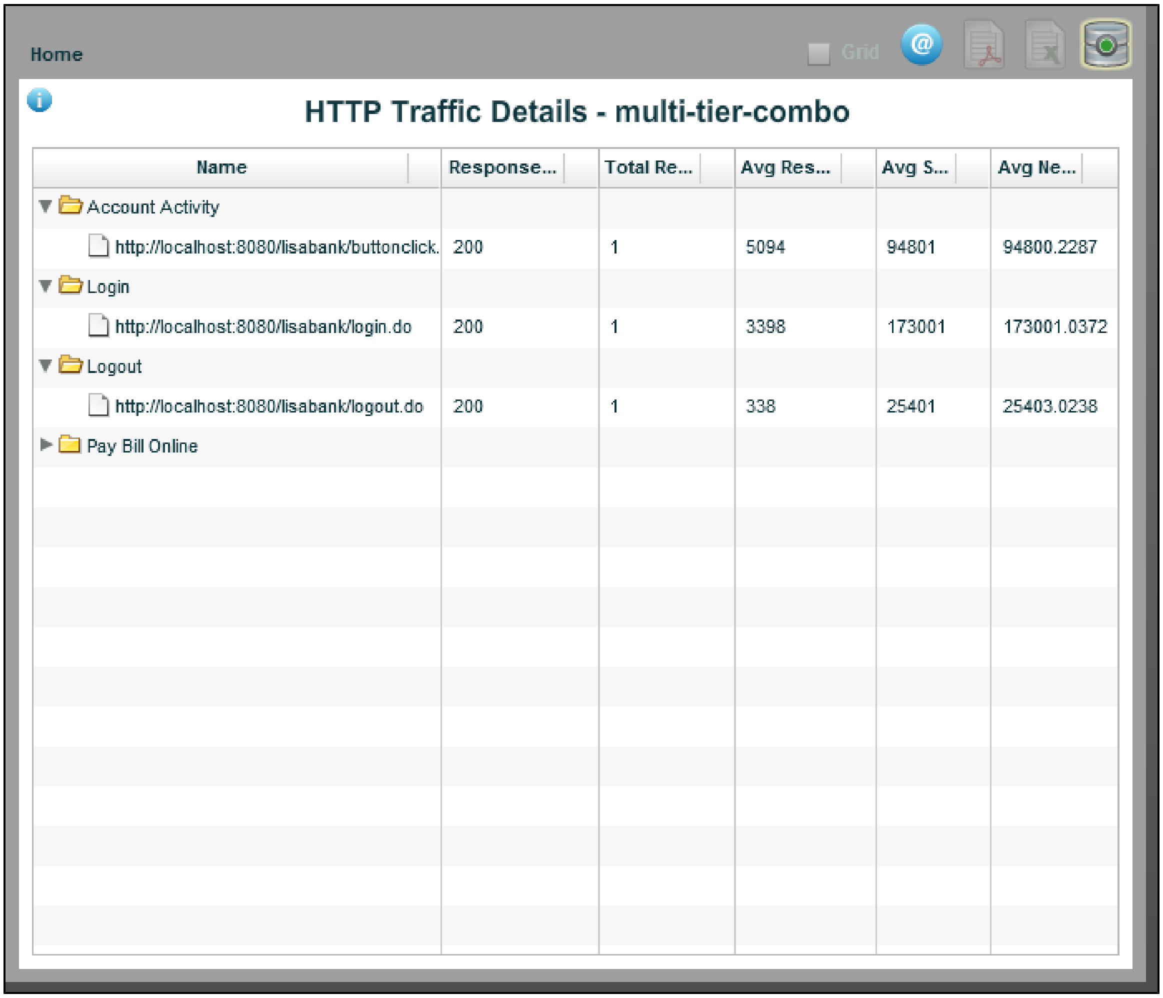
Cycle Performance Summary



Cycle Performance Summary - multi-tier-combo

Order	Interval Begin	Interval End	Cycle Count
1	25 Jan 2012 8:17:46 AM	25 Jan 2012 8:18:02 AM	0
2	25 Jan 2012 8:18:02 AM	25 Jan 2012 8:18:19 AM	0
3	25 Jan 2012 8:18:19 AM	25 Jan 2012 8:18:35 AM	0

HTTP Details

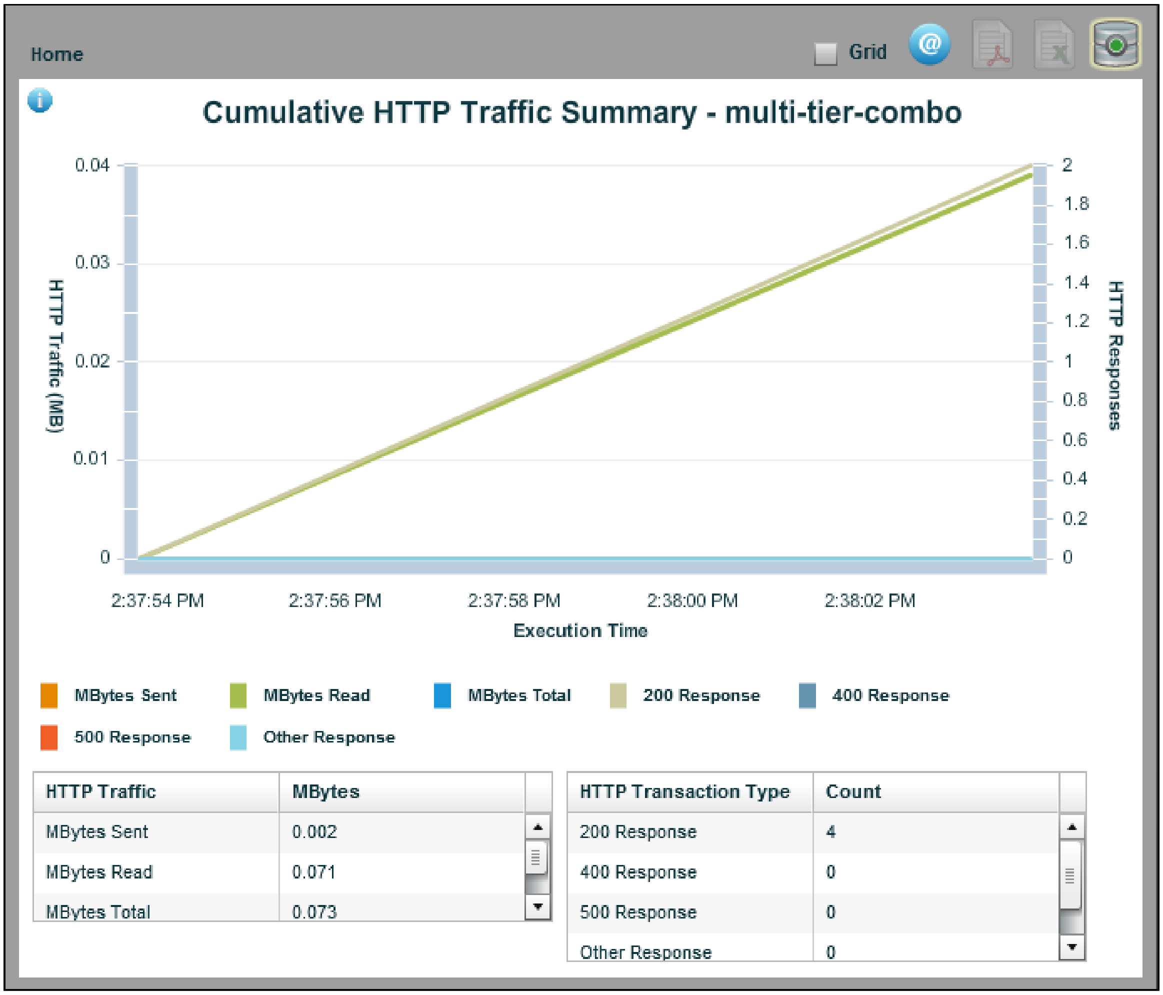


HTTP Traffic Details - multi-tier-combo

Name	Response...	Total Re...	Avg Res...	Avg S...	Avg Ne...
▼ Account Activity					
http://localhost:8080/lisabank/buttonclick.	200	1	5094	94801	94800.2287
▼ Login					
http://localhost:8080/lisabank/login.do	200	1	3398	173001	173001.0372
▼ Logout					
http://localhost:8080/lisabank/logout.do	200	1	338	25401	25403.0238
▶ Pay Bill Online					

HTTP Summary

To see the Cumulative HTTP Traffic Summary report, the following property must be set in either **local.properties** or **site.properties**: **lisa.commtrans.ctstats=true**.



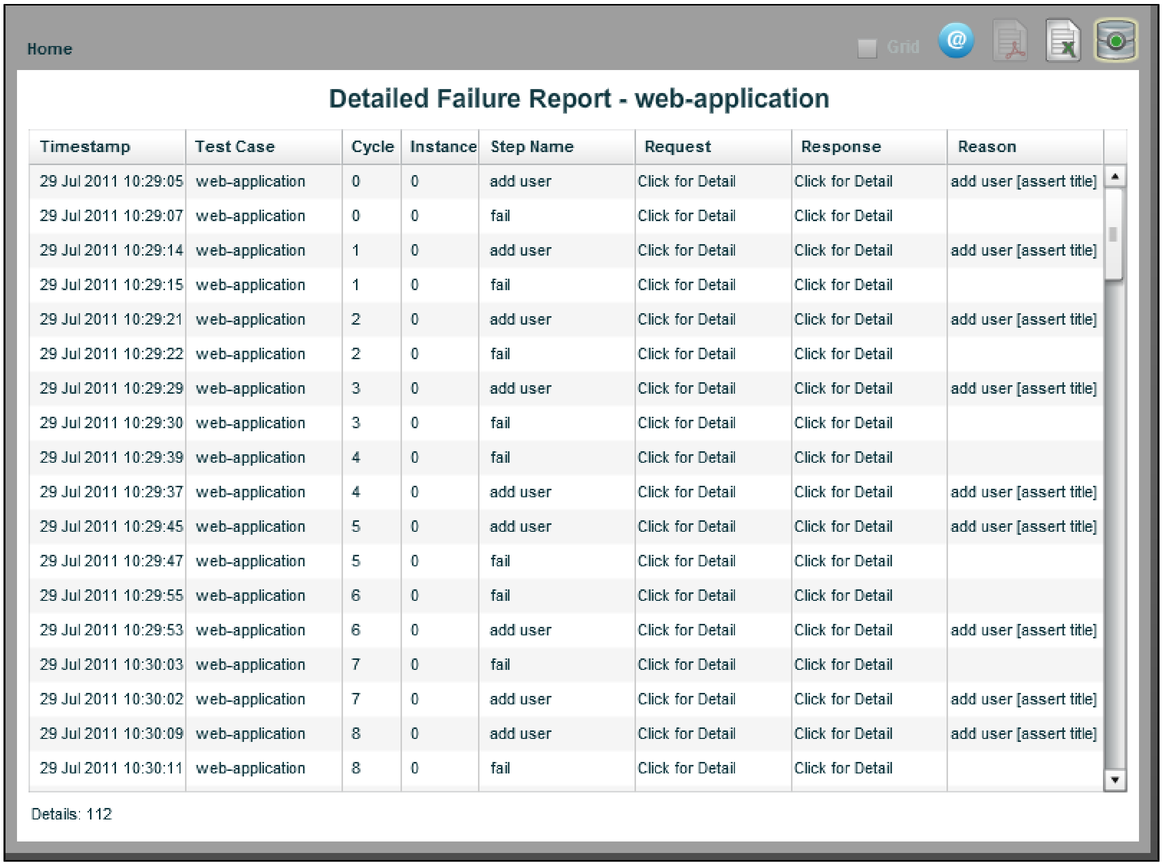
HTTP Traffic	MBytes
MBytes Sent	0.002
MBytes Read	0.071
MBytes Total	0.073

HTTP Transaction Type	Count
200 Response	4
400 Response	0
500 Response	0
Other Response	0

View Error Reports Menu

Error reports report errors and warnings from steps that did not complete error-free. Reports available are:

Detailed Failures

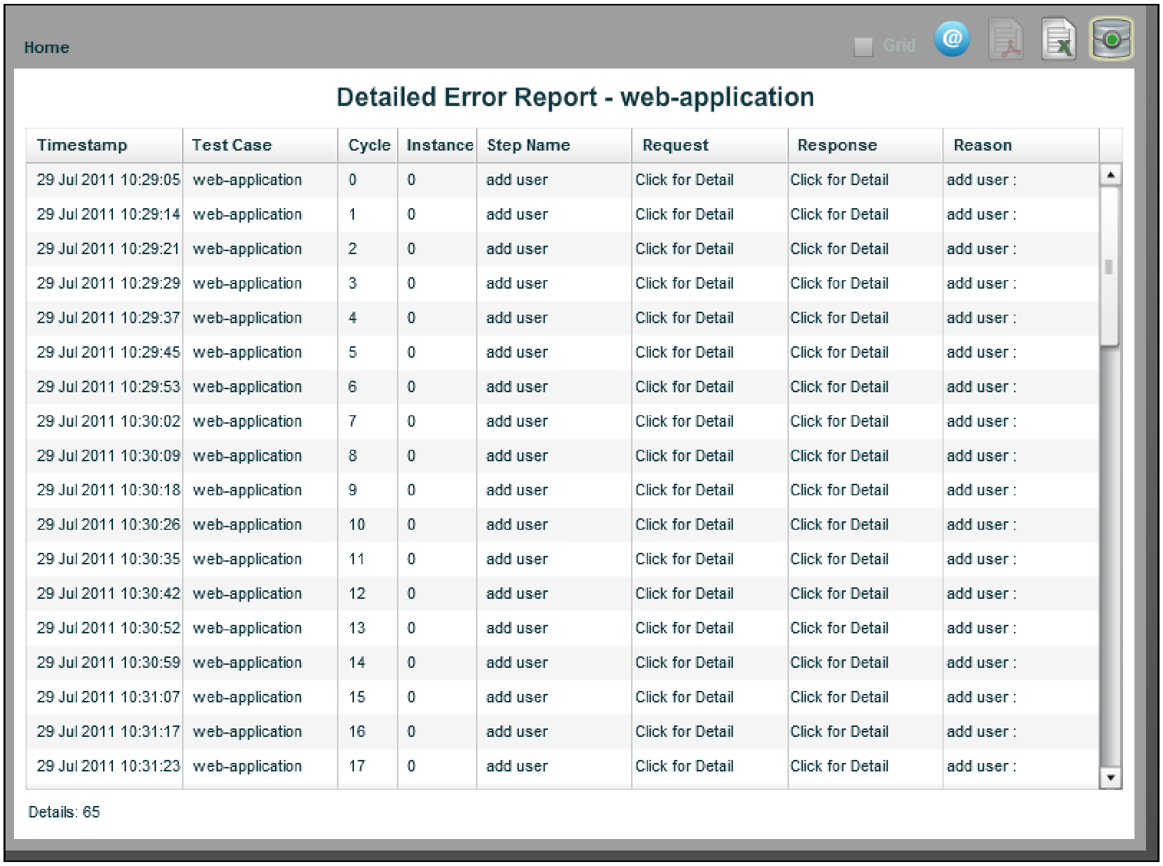


Detailed Failure Report - web-application

Timestamp	Test Case	Cycle	Instance	Step Name	Request	Response	Reason
29 Jul 2011 10:29:05	web-application	0	0	add user	Click for Detail	Click for Detail	add user [assert title]
29 Jul 2011 10:29:07	web-application	0	0	fail	Click for Detail	Click for Detail	
29 Jul 2011 10:29:14	web-application	1	0	add user	Click for Detail	Click for Detail	add user [assert title]
29 Jul 2011 10:29:15	web-application	1	0	fail	Click for Detail	Click for Detail	
29 Jul 2011 10:29:21	web-application	2	0	add user	Click for Detail	Click for Detail	add user [assert title]
29 Jul 2011 10:29:22	web-application	2	0	fail	Click for Detail	Click for Detail	
29 Jul 2011 10:29:29	web-application	3	0	add user	Click for Detail	Click for Detail	add user [assert title]
29 Jul 2011 10:29:30	web-application	3	0	fail	Click for Detail	Click for Detail	
29 Jul 2011 10:29:39	web-application	4	0	fail	Click for Detail	Click for Detail	
29 Jul 2011 10:29:37	web-application	4	0	add user	Click for Detail	Click for Detail	add user [assert title]
29 Jul 2011 10:29:45	web-application	5	0	add user	Click for Detail	Click for Detail	add user [assert title]
29 Jul 2011 10:29:47	web-application	5	0	fail	Click for Detail	Click for Detail	
29 Jul 2011 10:29:55	web-application	6	0	fail	Click for Detail	Click for Detail	
29 Jul 2011 10:29:53	web-application	6	0	add user	Click for Detail	Click for Detail	add user [assert title]
29 Jul 2011 10:30:03	web-application	7	0	fail	Click for Detail	Click for Detail	
29 Jul 2011 10:30:02	web-application	7	0	add user	Click for Detail	Click for Detail	add user [assert title]
29 Jul 2011 10:30:09	web-application	8	0	add user	Click for Detail	Click for Detail	add user [assert title]
29 Jul 2011 10:30:11	web-application	8	0	fail	Click for Detail	Click for Detail	

Details: 112

Detailed Errors

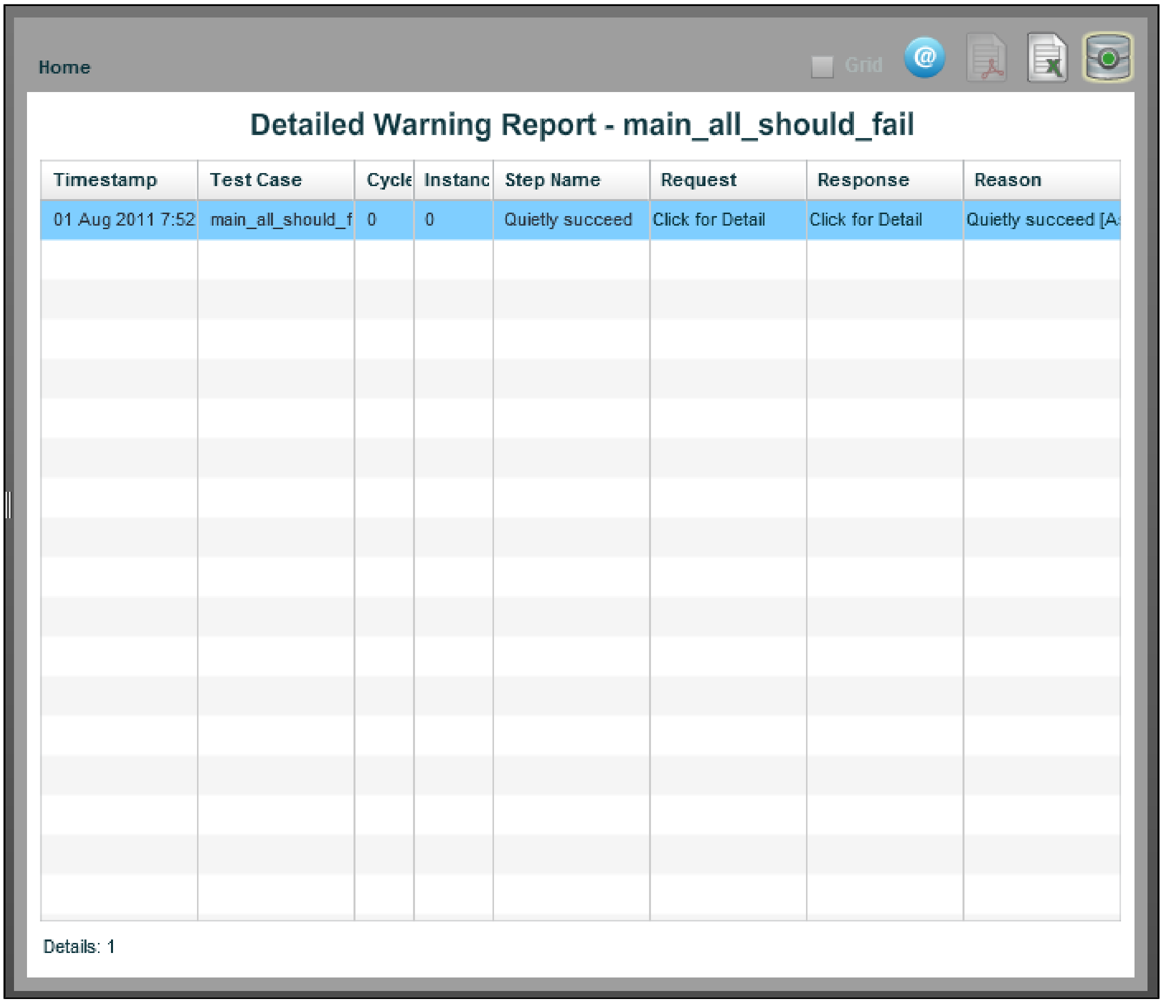


Detailed Error Report - web-application

Timestamp	Test Case	Cycle	Instance	Step Name	Request	Response	Reason
29 Jul 2011 10:29:05	web-application	0	0	add user	Click for Detail	Click for Detail	add user :
29 Jul 2011 10:29:14	web-application	1	0	add user	Click for Detail	Click for Detail	add user :
29 Jul 2011 10:29:21	web-application	2	0	add user	Click for Detail	Click for Detail	add user :
29 Jul 2011 10:29:29	web-application	3	0	add user	Click for Detail	Click for Detail	add user :
29 Jul 2011 10:29:37	web-application	4	0	add user	Click for Detail	Click for Detail	add user :
29 Jul 2011 10:29:45	web-application	5	0	add user	Click for Detail	Click for Detail	add user :
29 Jul 2011 10:29:53	web-application	6	0	add user	Click for Detail	Click for Detail	add user :
29 Jul 2011 10:30:02	web-application	7	0	add user	Click for Detail	Click for Detail	add user :
29 Jul 2011 10:30:09	web-application	8	0	add user	Click for Detail	Click for Detail	add user :
29 Jul 2011 10:30:18	web-application	9	0	add user	Click for Detail	Click for Detail	add user :
29 Jul 2011 10:30:26	web-application	10	0	add user	Click for Detail	Click for Detail	add user :
29 Jul 2011 10:30:35	web-application	11	0	add user	Click for Detail	Click for Detail	add user :
29 Jul 2011 10:30:42	web-application	12	0	add user	Click for Detail	Click for Detail	add user :
29 Jul 2011 10:30:52	web-application	13	0	add user	Click for Detail	Click for Detail	add user :
29 Jul 2011 10:30:59	web-application	14	0	add user	Click for Detail	Click for Detail	add user :
29 Jul 2011 10:31:07	web-application	15	0	add user	Click for Detail	Click for Detail	add user :
29 Jul 2011 10:31:17	web-application	16	0	add user	Click for Detail	Click for Detail	add user :
29 Jul 2011 10:31:23	web-application	17	0	add user	Click for Detail	Click for Detail	add user :

Details: 65

Detailed Warnings

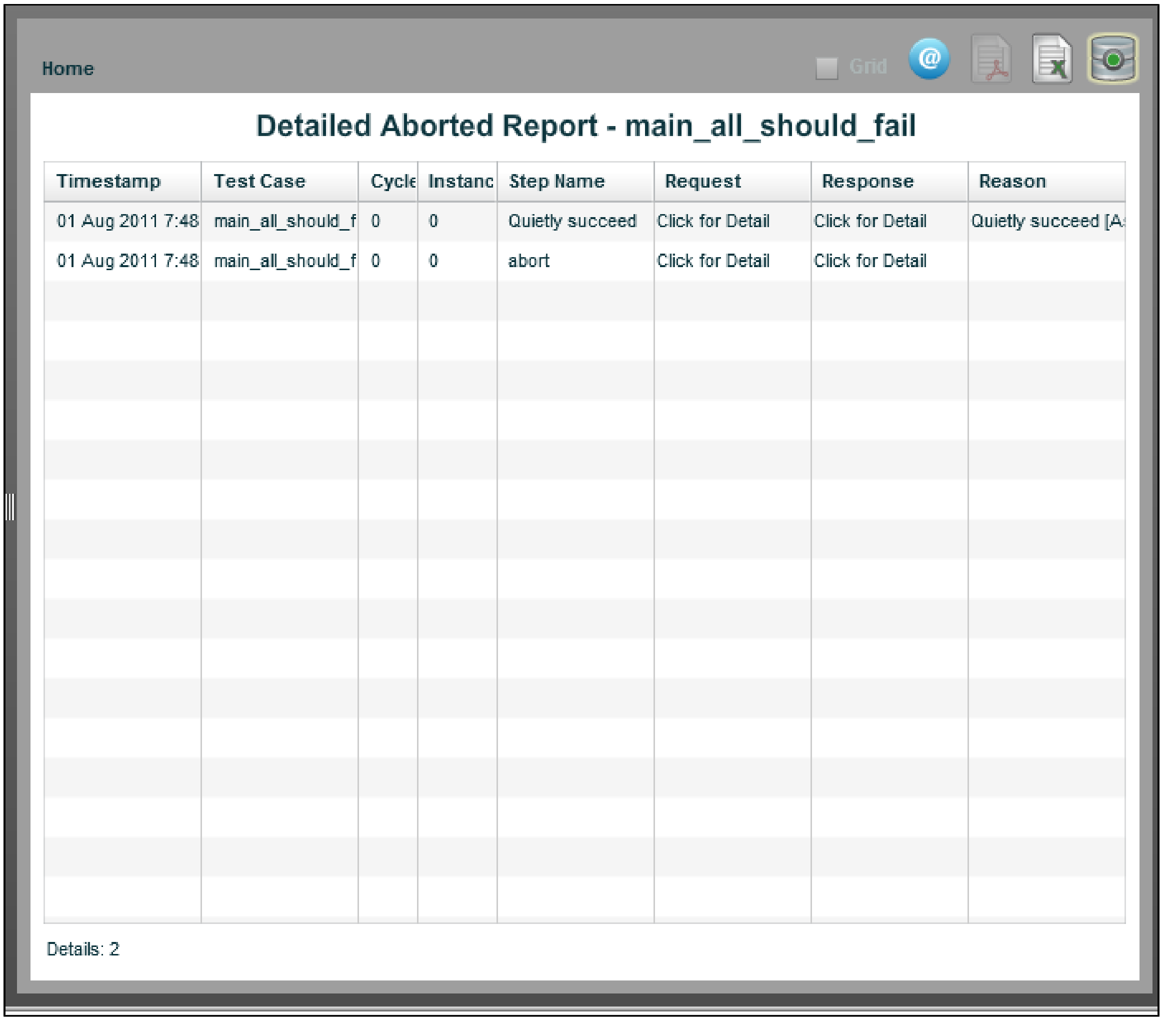


Detailed Warning Report - main_all_should_fail

Timestamp	Test Case	Cycle	Instanc	Step Name	Request	Response	Reason
01 Aug 2011 7:52	main_all_should_f	0	0	Quietly succeed	Click for Detail	Click for Detail	Quietly succeed [A

Details: 1

Detailed Aborts

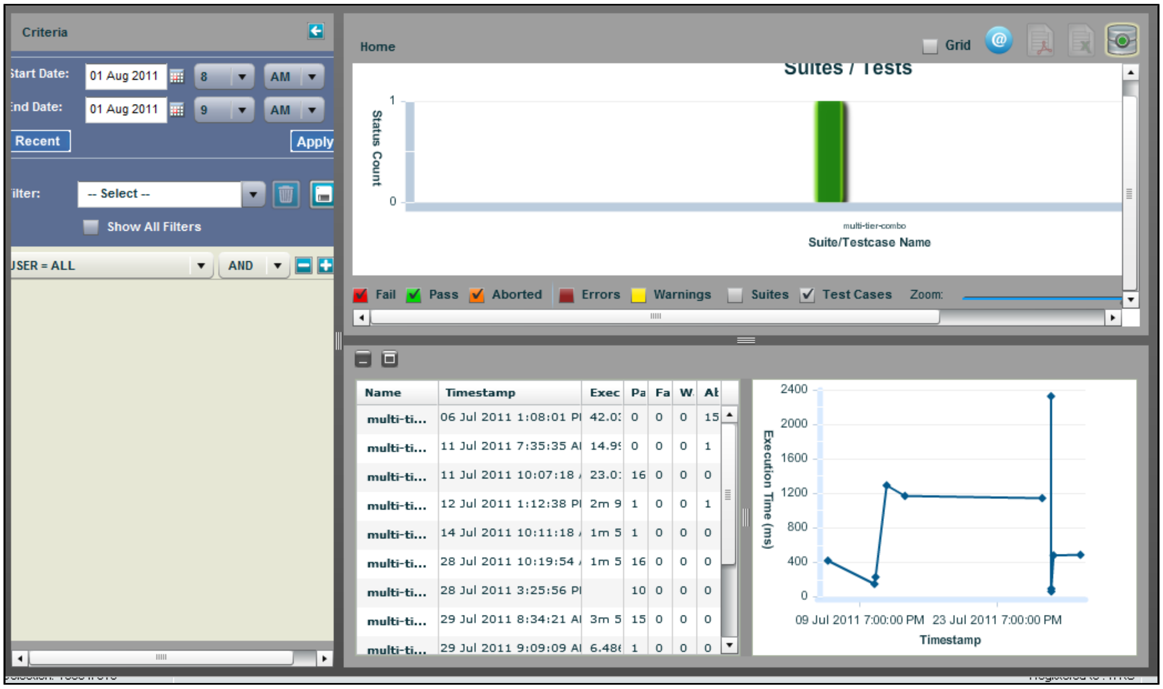


Detailed Aborted Report - main_all_should_fail

Timestamp	Test Case	Cycle	Instanc	Step Name	Request	Response	Reason
01 Aug 2011 7:48	main_all_should_f	0	0	Quietly succeed	Click for Detail	Click for Detail	Quietly succeed [A:
01 Aug 2011 7:48	main_all_should_f	0	0	abort	Click for Detail	Click for Detail	

Details: 2

View History



Start Date: 01 Aug 2011 8 AM

End Date: 01 Aug 2011 9 AM

Recent Apply

Filter: -- Select --

□ Show All Filters

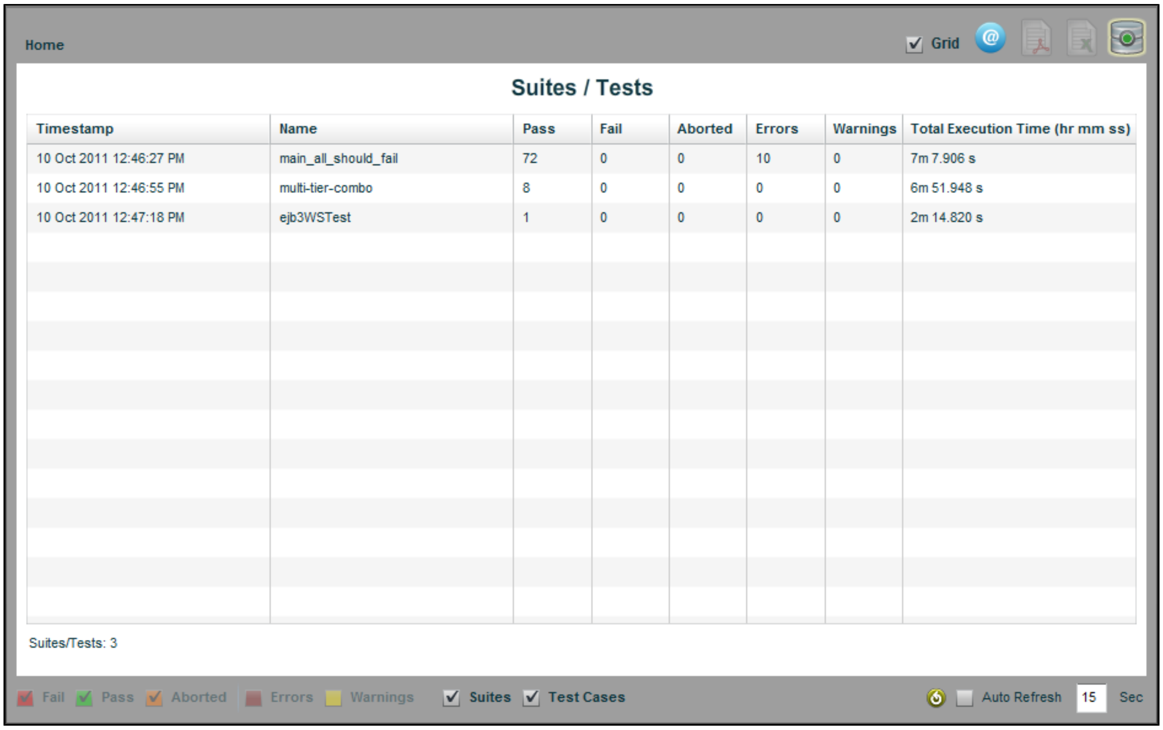
USER = ALL AND

🟥 Fail 🟩 Pass ☑ Aborted ■ Errors ■ Warnings □ Suites ☑ Test Cases

Name	Timestamp	Exec	Pa	Fa	W	Ak
multi-ti...	06 Jul 2011 1:08:01 PI	42.0:	0	0	0	15
multi-ti...	11 Jul 2011 7:35:35 AI	14.9!	0	0	0	1
multi-ti...	11 Jul 2011 10:07:18 /	23.0:	16	0	0	0
multi-ti...	12 Jul 2011 1:12:38 PI	2m 9	1	0	0	1
multi-ti...	14 Jul 2011 10:11:18 /	1m 5	1	0	0	0
multi-ti...	28 Jul 2011 10:19:54 /	1m 5	16	0	0	0
multi-ti...	28 Jul 2011 3:25:56 PI		10	0	0	0
multi-ti...	29 Jul 2011 8:34:21 AI	3m 5	15	0	0	0
multi-ti...	29 Jul 2011 9:09:09 AI	6.48(	1	0	0	0

Reports - Grid View

Reports can also display in a grid view. To view the results in a grid, select the Grid icon ☑ Grid on the top right corner. The information available in the grid view is the same as that in the graphs view; the only difference is the display. All report information filters work the same. You must be in grid view to export reports to Excel.



Home

☑ Grid

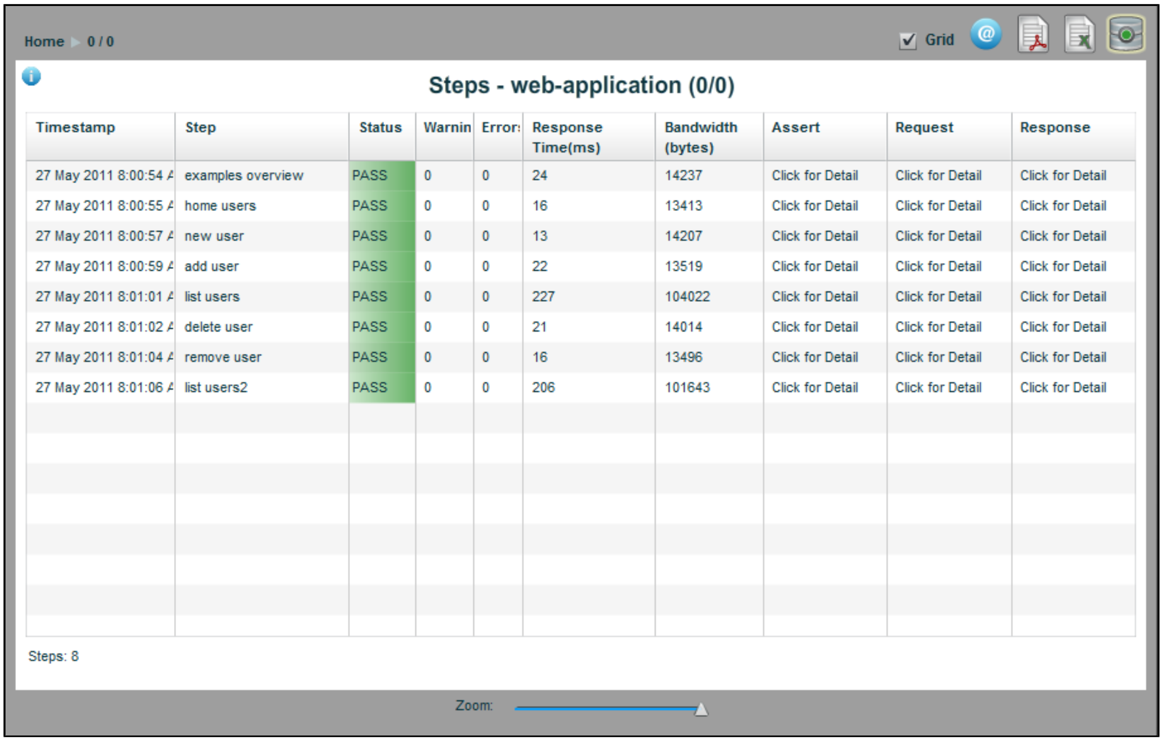
Suites / Tests

Timestamp	Name	Pass	Fail	Aborted	Errors	Warnings	Total Execution Time (hr mm ss)
10 Oct 2011 12:46:27 PM	main_all_should_fail	72	0	0	10	0	7m 7.906 s
10 Oct 2011 12:46:55 PM	multi-tier-combo	8	0	0	0	0	6m 51.948 s
10 Oct 2011 12:47:18 PM	ejb3WSTest	1	0	0	0	0	2m 14.820 s

Suites/Tests: 3

✓ Fail ✓ Pass ✓ Aborted Errors Warnings ☑ Suites ☑ Test Cases
□ Auto Refresh 15 Sec

You can select each of the test cases within the report to find out more details regarding it.



Home ▸ 0 / 0

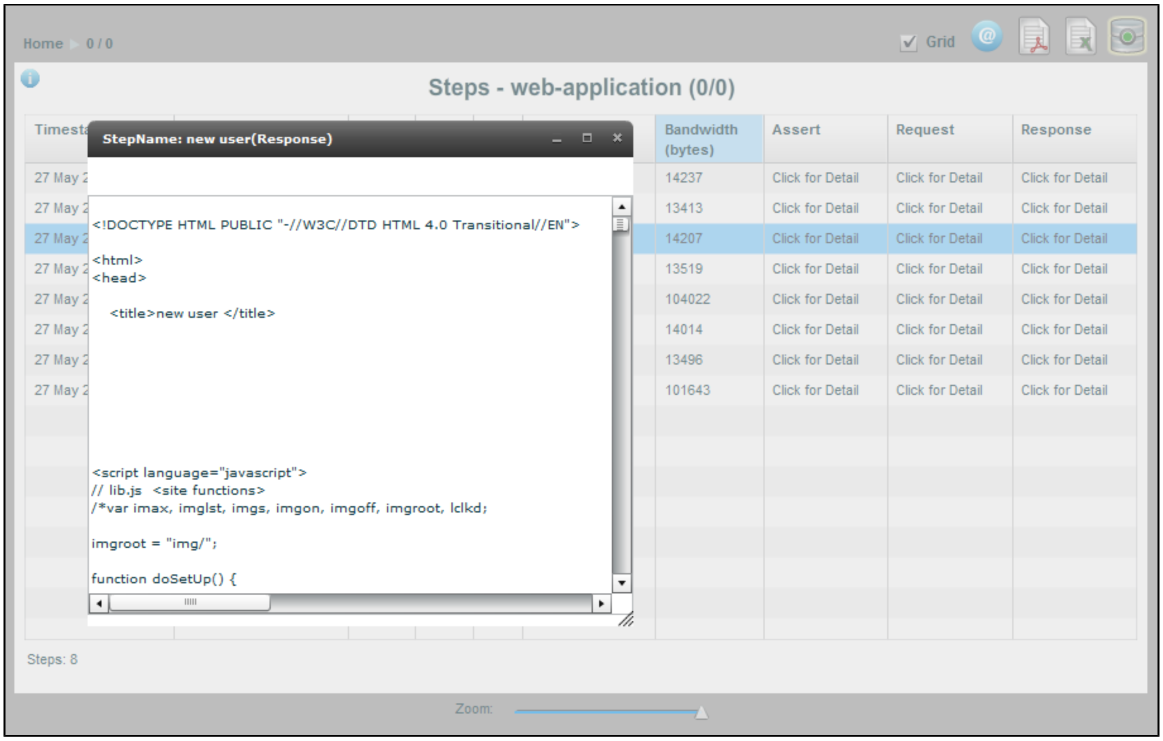
☑ Grid

Steps - web-application (0/0)

Timestamp	Step	Status	Warnin	Error:	Response Time(ms)	Bandwidth (bytes)	Assert	Request	Response
27 May 2011 8:00:54 A	examples overview	PASS	0	0	24	14237	Click for Detail	Click for Detail	Click for Detail
27 May 2011 8:00:55 A	home users	PASS	0	0	16	13413	Click for Detail	Click for Detail	Click for Detail
27 May 2011 8:00:57 A	new user	PASS	0	0	13	14207	Click for Detail	Click for Detail	Click for Detail
27 May 2011 8:00:59 A	add user	PASS	0	0	22	13519	Click for Detail	Click for Detail	Click for Detail
27 May 2011 8:01:01 A	list users	PASS	0	0	227	104022	Click for Detail	Click for Detail	Click for Detail
27 May 2011 8:01:02 A	delete user	PASS	0	0	21	14014	Click for Detail	Click for Detail	Click for Detail
27 May 2011 8:01:04 A	remove user	PASS	0	0	16	13496	Click for Detail	Click for Detail	Click for Detail
27 May 2011 8:01:06 A	list users2	PASS	0	0	206	101643	Click for Detail	Click for Detail	Click for Detail

Steps: 8

If you click **Click for detail** in the **Assert**, **Request**, or **Response** columns, you see more detailed information about each of those components of the step. In our example, we display the information about the Response on the **new user** step.



Steps - web-application (0/0)

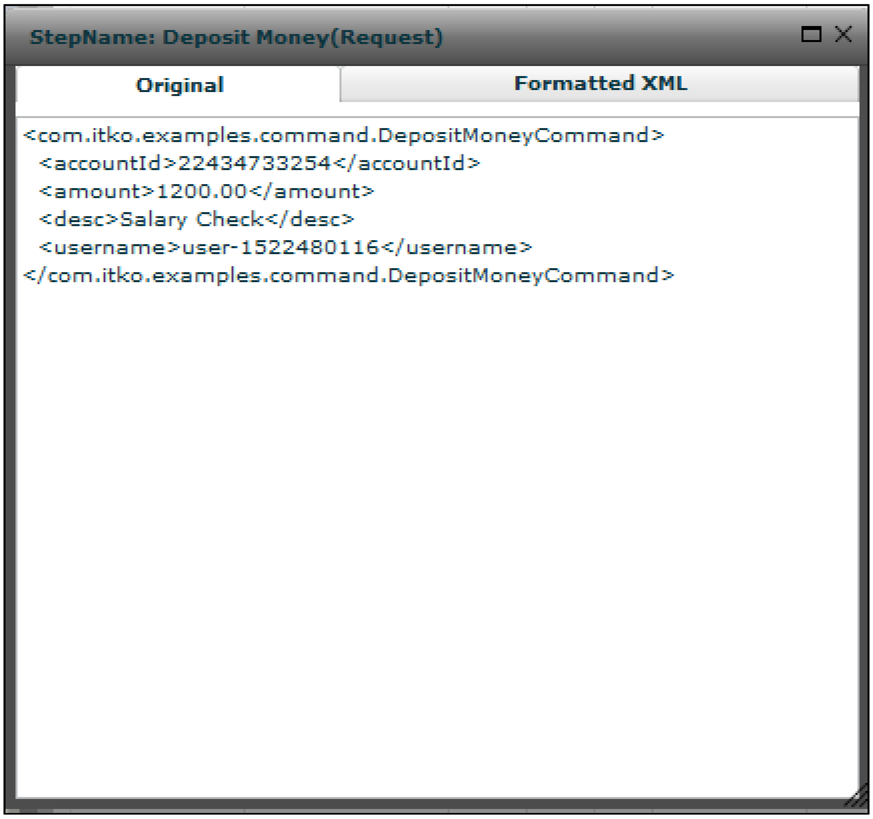
Timesta	Bandwidth (bytes)	Assert	Request	Response
27 May 2	14237	Click for Detail	Click for Detail	Click for Detail
27 May 2	13413	Click for Detail	Click for Detail	Click for Detail
27 May 2	14207	Click for Detail	Click for Detail	Click for Detail
27 May 2	13519	Click for Detail	Click for Detail	Click for Detail
27 May 2	104022	Click for Detail	Click for Detail	Click for Detail
27 May 2	14014	Click for Detail	Click for Detail	Click for Detail
27 May 2	13496	Click for Detail	Click for Detail	Click for Detail
27 May 2	101643	Click for Detail	Click for Detail	Click for Detail

StepName: new user(Response)

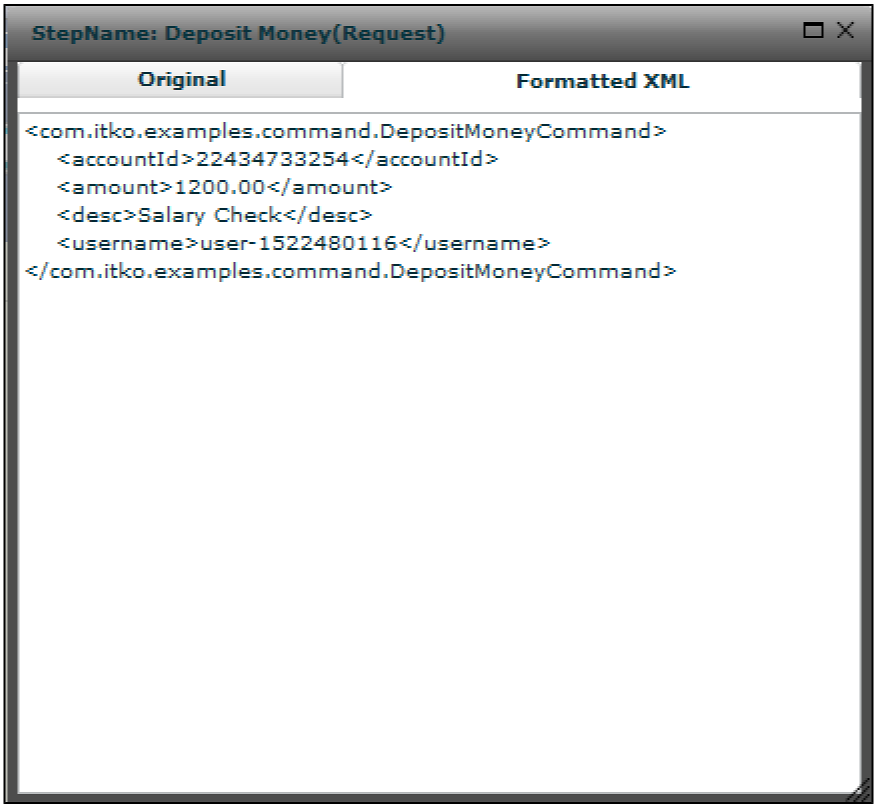
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<html>
<head>

  <title>new user </title>







<script language="javascript">
// lib.js  <site functions>
/*var imax, imglst, imgs, imgon, imgoff, imgroot, lclkd;

imgroot = "img/";

function doSetUp() {
```

Steps: 8

In this view, when you click to view request and response data, for those that are XML, you can select the **Formatted XML** tab to see formatted text. For those that are not XML, the Formatted tab shows "Text is not valid XML."


StepName: Deposit Money(Request)

```
<com.itko.examples.command.DepositMoneyCommand>
  <accountId>22434733254</accountId>
  <amount>1200.00</amount>
  <desc>Salary Check</desc>
  <username>user-1522480116</username>
</com.itko.examples.command.DepositMoneyCommand>
```



```
<com.itko.examples.command.DepositMoneyCommand>
   <accountId>22434733254</accountId>
   <amount>1200.00</amount>
   <desc>Salary Check</desc>
   <username>user-1522480116</username>
</com.itko.examples.command.DepositMoneyCommand>
```

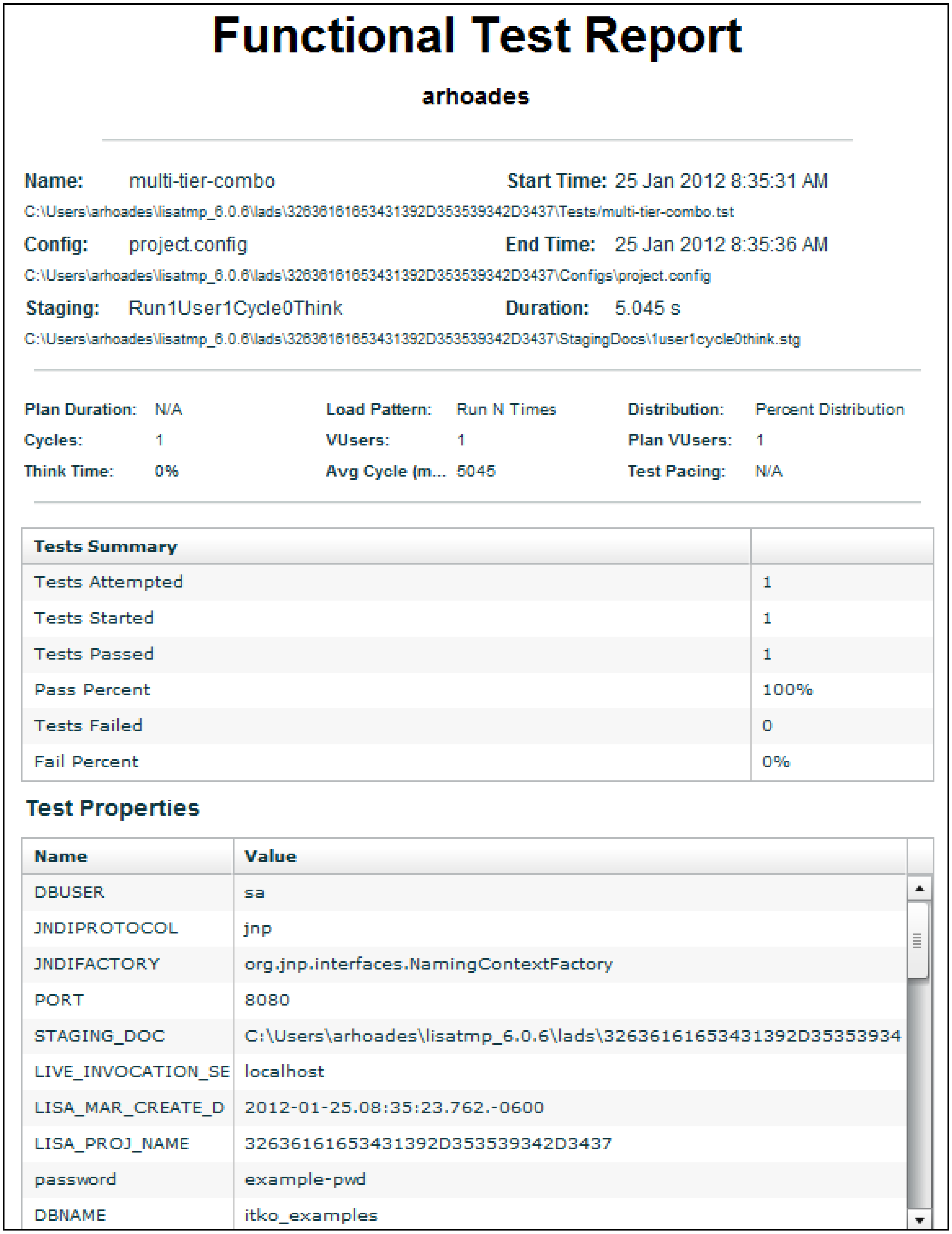
Standard LISA Reports

There are four pre-formatted LISA reports that can be generated and exported to PDF.

Functional Test Report

To produce a functional test report, double-click on a test to see the Cycles report. Select a cycle, then select the Export to PDF icon to open the report in a PDF format.

> ⚠ The Functional Test Report is produced when the test is run with fewer than three virtual users or fewer than 15 cycles. If the test is run with more virtual users and/or cycles, the Performance Test Report is produced.



Functional Test Report

arhoades

Name: multi-tier-combo **Start Time:** 25 Jan 2012 8:35:31 AM

C:\Users\arhoades\lisatmp_6.0.6\lads\32636161653431392D353539342D3437\Tests/multi-tier-combo.tst

Config: project.config **End Time:** 25 Jan 2012 8:35:36 AM

C:\Users\arhoades\lisatmp_6.0.6\lads\32636161653431392D353539342D3437\Configs\project.config

Staging: Run1User1Cycle0Think **Duration:** 5.045 s

C:\Users\arhoades\lisatmp_6.0.6\lads\32636161653431392D353539342D3437\StagingDocs\1user1cycle0think.stg

Plan Duration:	N/A	**Load Pattern:**	Run N Times	**Distribution:**	Percent Distribution
Cycles:	1	**VUsers:**	1	**Plan VUsers:**	1
Think Time:	0%	**Avg Cycle (m...**	5045	**Test Pacing:**	N/A

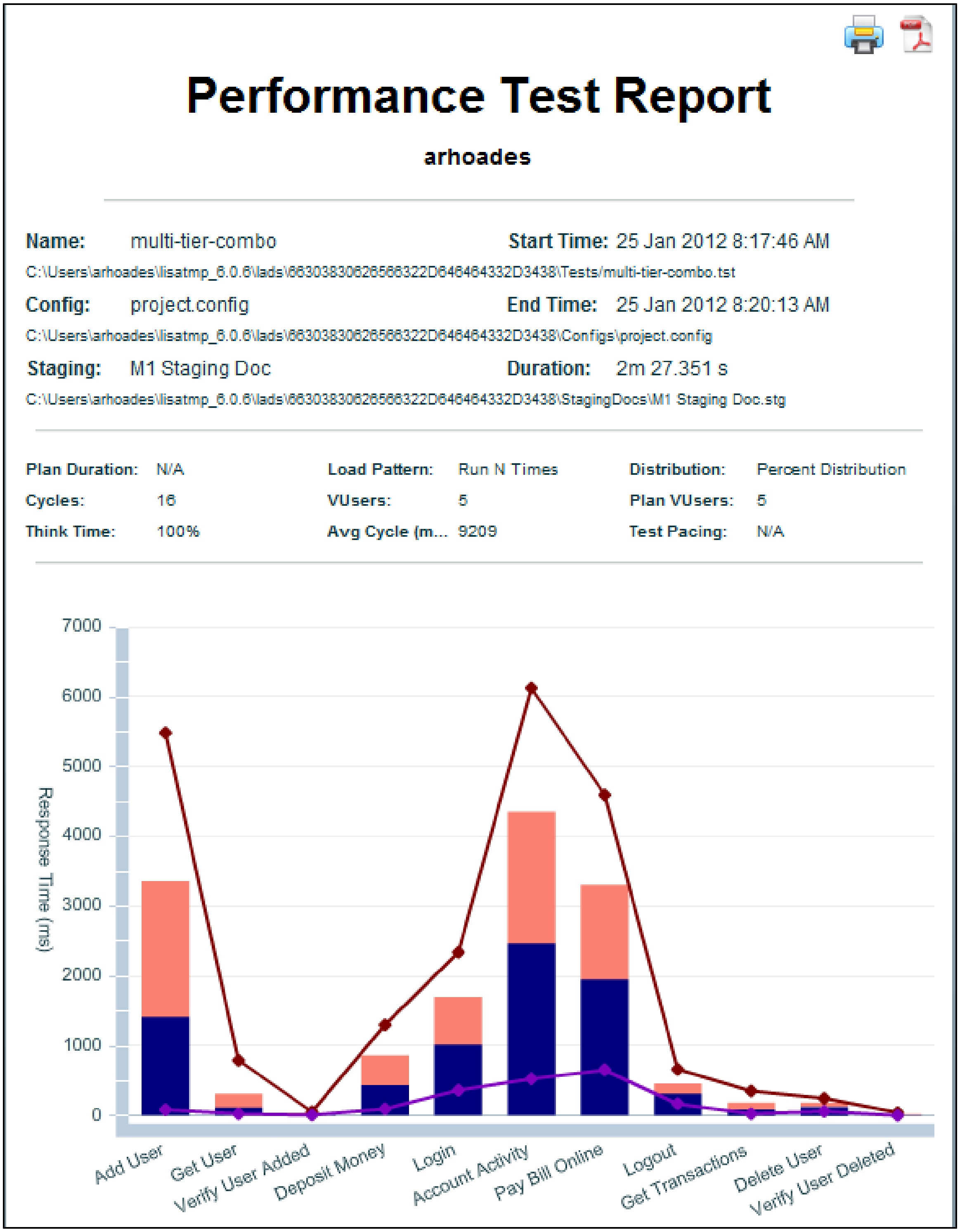
Tests Summary	
Tests Attempted	1
Tests Started	1
Tests Passed	1
Pass Percent	100%
Tests Failed	0
Fail Percent	0%

Test Properties

Name	Value
DBUSER	sa
JNDIPROTOCOL	jnp
JNDIFACTORY	org.jnp.interfaces.NamingContextFactory
PORT	8080
STAGING_DOC	C:\Users\arhoades\lisatmp_6.0.6\lads\32636161653431392D35353934
LIVE_INVOCATION_SE	localhost
LISA_MAR_CREATE_D	2012-01-25.08:35:23.762.-0600
LISA_PROJ_NAME	32636161653431392D353539342D3437
password	example-pwd
DBNAME	itko_examples

Performance Test Report

To produce a performance test report, double-click on a test to see the Cycles report. Select a cycle, then click the Export to PDF icon to open the report in a PDF format.



Performance Test Report

arhoades

Name: multi-tier-combo **Start Time:** 25 Jan 2012 8:17:46 AM

C:\Users\arhoades\lisatmp_6.0.6\lads\66303830626566322D646464332D3438\Tests/multi-tier-combo.tst

Config: project.config **End Time:** 25 Jan 2012 8:20:13 AM

C:\Users\arhoades\lisatmp_6.0.6\lads\66303830626566322D646464332D3438\Configs\project.config

Staging: M1 Staging Doc **Duration:** 2m 27.351 s

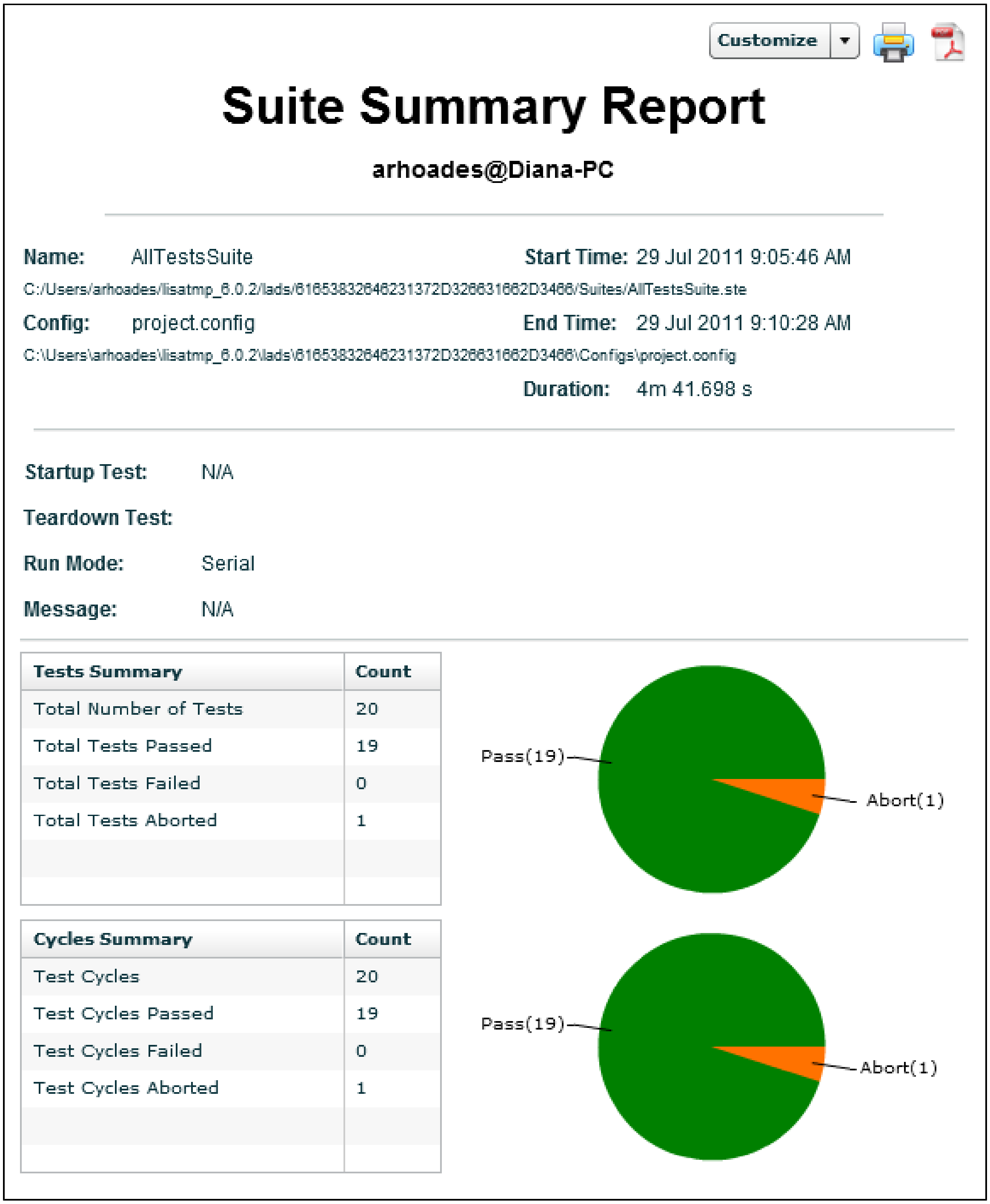
C:\Users\arhoades\lisatmp_6.0.6\lads\66303830626566322D646464332D3438\StagingDocs\M1 Staging Doc.stg

Plan Duration:	N/A	**Load Pattern:**	Run N Times	**Distribution:**	Percent Distribution
Cycles:	16	**VUsers:**	5	**Plan VUsers:**	5
Think Time:	100%	**Avg Cycle (m...**	9209	**Test Pacing:**	N/A

> ⚠ The Performance Test Report is produced when the test is run with more than 3 virtual users or more than 15 cycles. If the test is run with fewer virtual users and/or cycles, the Functional Test Report is produced.

Suite Summary Report

To produce a suite summary report, open a test suite report. Click the Export to PDF icon to open the suite summary report in a PDF format.



Suite Summary Report

arhoades@Diana-PC

Name: AllTestsSuite **Start Time:** 29 Jul 2011 9:05:46 AM

C:/Users/arhoades/lisatmp_6.0.2/lads/61653832646231372D326631662D3466/Suites/AllTestsSuite.ste

Config: project.config **End Time:** 29 Jul 2011 9:10:28 AM

C:\Users\arhoades\lisatmp_6.0.2\lads\61653832646231372D326631662D3466\Configs\project.config

Duration: 4m 41.698 s

Startup Test: N/A

Teardown Test:

Run Mode: Serial

Message: N/A

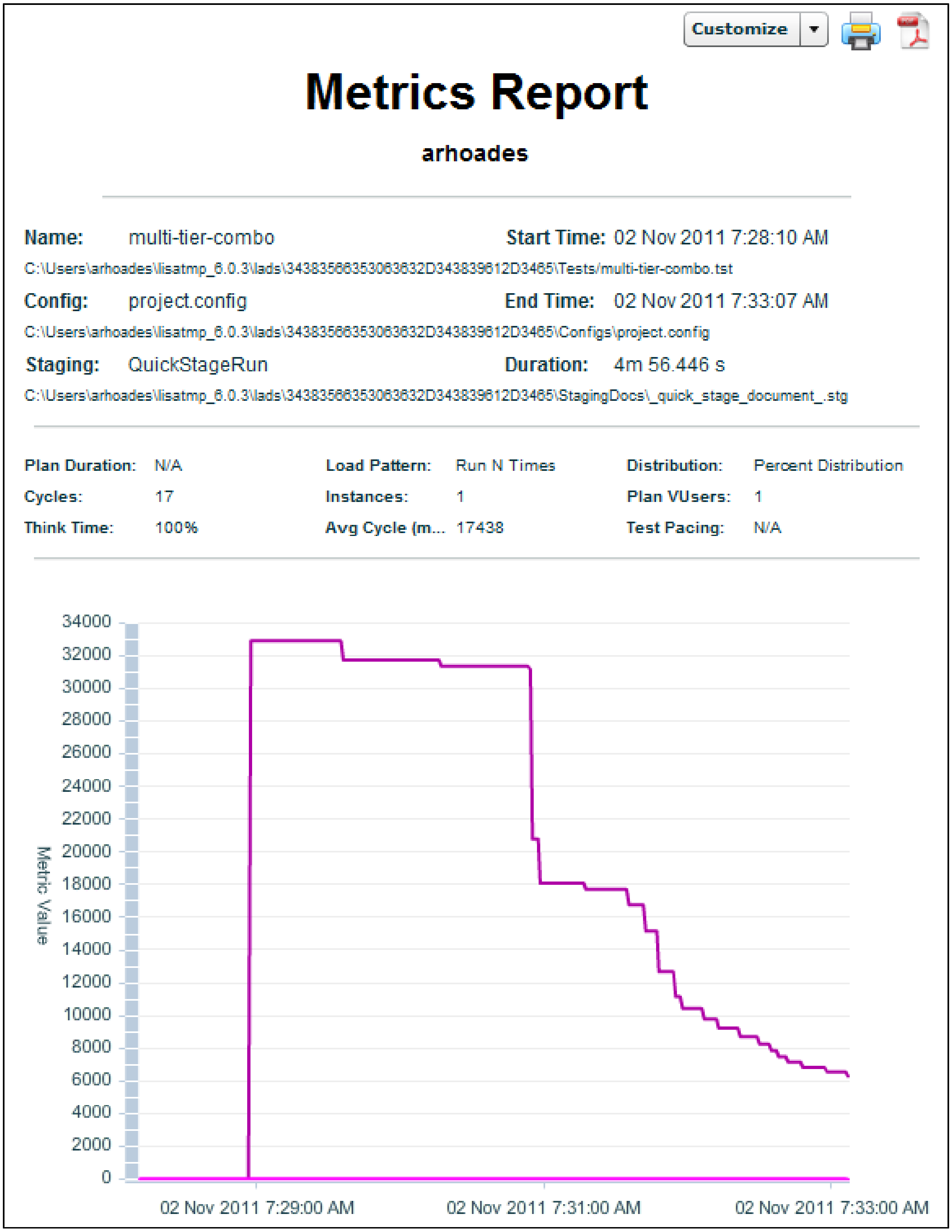
Tests Summary	Count
Total Number of Tests	20
Total Tests Passed	19
Total Tests Failed	0
Total Tests Aborted	1

Cycles Summary	Count
Test Cycles	20
Test Cycles Passed	19
Test Cycles Failed	0
Test Cycles Aborted	1

Metrics Report

To produce a Metrics Report, from a test/suite report, right-click on a test or suite and select Analyze > Metrics to display a metrics chart or grid.

From that report, click the Export to PDF icon to open the report in a PDF format.



Metrics Report

arhoades

Name: multi-tier-combo

Start Time: 02 Nov 2011 7:28:10 AM

C:\Users\arhoades\lisatmp_6.0.3\lads\34383566353063632D343839612D3465\Tests/multi-tier-combo.tst

Config: project.config

End Time: 02 Nov 2011 7:33:07 AM

C:\Users\arhoades\lisatmp_6.0.3\lads\34383566353063632D343839612D3465\Configs\project.config

Staging: QuickStageRun

Duration: 4m 56.446 s

C:\Users\arhoades\lisatmp_6.0.3\lads\34383566353063632D343839612D3465\StagingDocs_quick_stage_document_.stg

Plan Duration:	N/A	Load Pattern:	Run N Times	Distribution:	Percent Distribution
Cycles:	17	Instances:	1	Plan VUsers:	1
Think Time:	100%	Avg Cycle (m...	17438	Test Pacing:	N/A

Interpreting Reports

There are some situations where the results of reports are not what you expect, and it may appear that the reporting data is incorrect.

Looping Tests

Loops in test cases can cause unexpected results in the reporting engine.

The LISA workflow engine is designed to flow from the beginning step to the end step, with no loops. One execution of this chain of steps is considered to be one pass or fail or abort event. Looping is designed to be initiated externally to the test case using a staging document. By using a staging document to specify 10 virtual users executing a test case once, you will have 10 pass/fail/abort events. When the looping is done within the test case, a pass/fail/abort will not occur until the end step is reached, therefore creating only a single pass/fail/abort event.

Think Time and Reports

Think time is how long LISA waits to begin the execution of a test step. The purpose of think time is to simulate a real user interacting with any system. You can set think time in the step editor, in a staging document, or in the Web 2.0 parameters.

In the main reporting panel, we display total execution time, which is the length of time from start to finish of the test. Because this is a duration, we include think times by design, to show how long it took for the entire test to run. To exclude think times, set your think time amount to 0 in your staging document or on your test step.

If you are looking for performance numbers, generate a performance report that will give you response times for each step that does not include think times, which is a more meaningful report for performance tests. Think time is not part of the average step performance time.

For Web 2.0 tests that have think time added to the **Playback Speed** parameter in the Playback Settings, that time is included in both the total execution time and the response times for each step. The reason there is also a think time in the browser is that when you execute a debugging session from the browser, LISA think times are not honored because LISA is not driving the playback, the browser is doing it directly, so this gives you a way to control the test speed from within the browser. The default replay speed is 0x (no think time added). You generally do not need to set the replay speed to anything other than 0x except for debugging, so this should not be an issue.

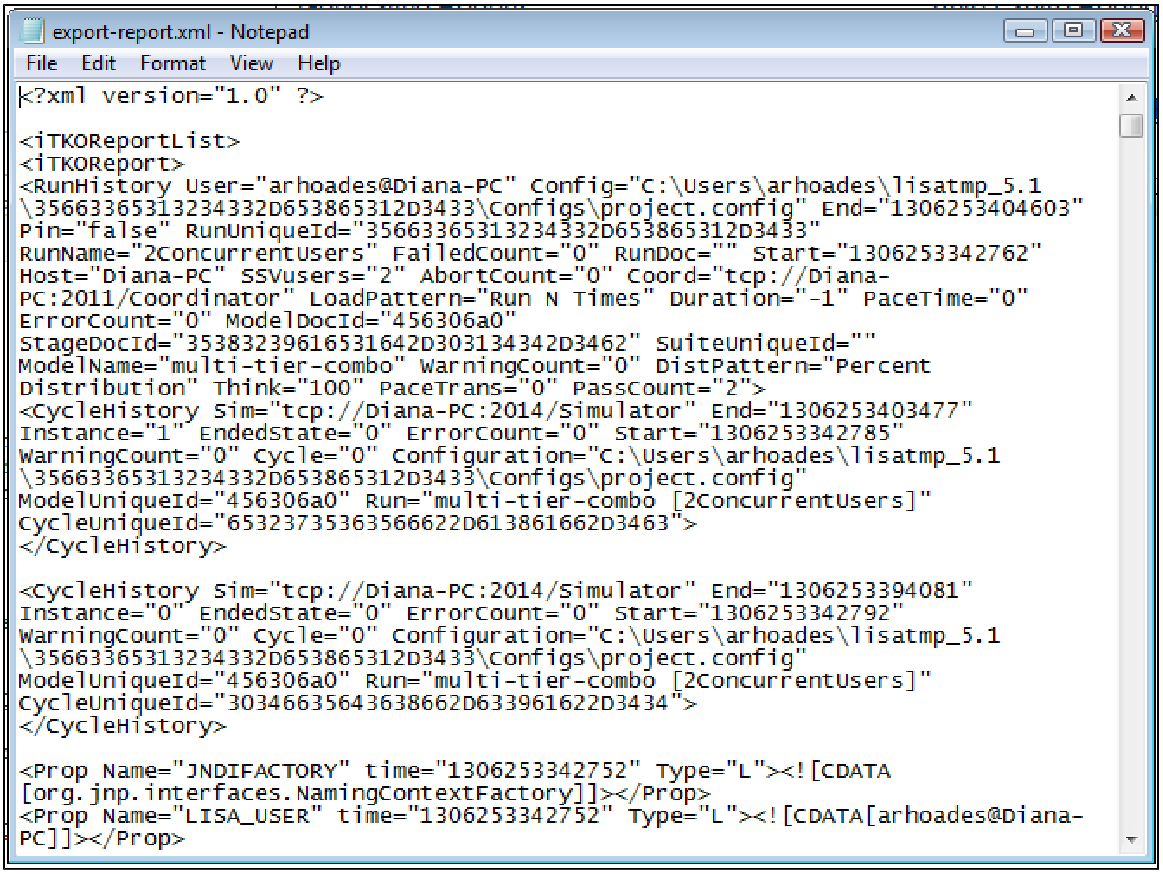
Exporting Reports

In the Reports viewer, the reports can be exported in three formats: XML, Microsoft Office Excel, and Adobe Acrobat PDF.

Exporting Report to XML

If your test suite or staging document specified the XML Report, you will be able to export the report data directly to an XML file. To do this, right-click on the test or suite and select Export. For more information, see Reports - Graphical View. After saving your exported data, you can format or manipulate it as you want.

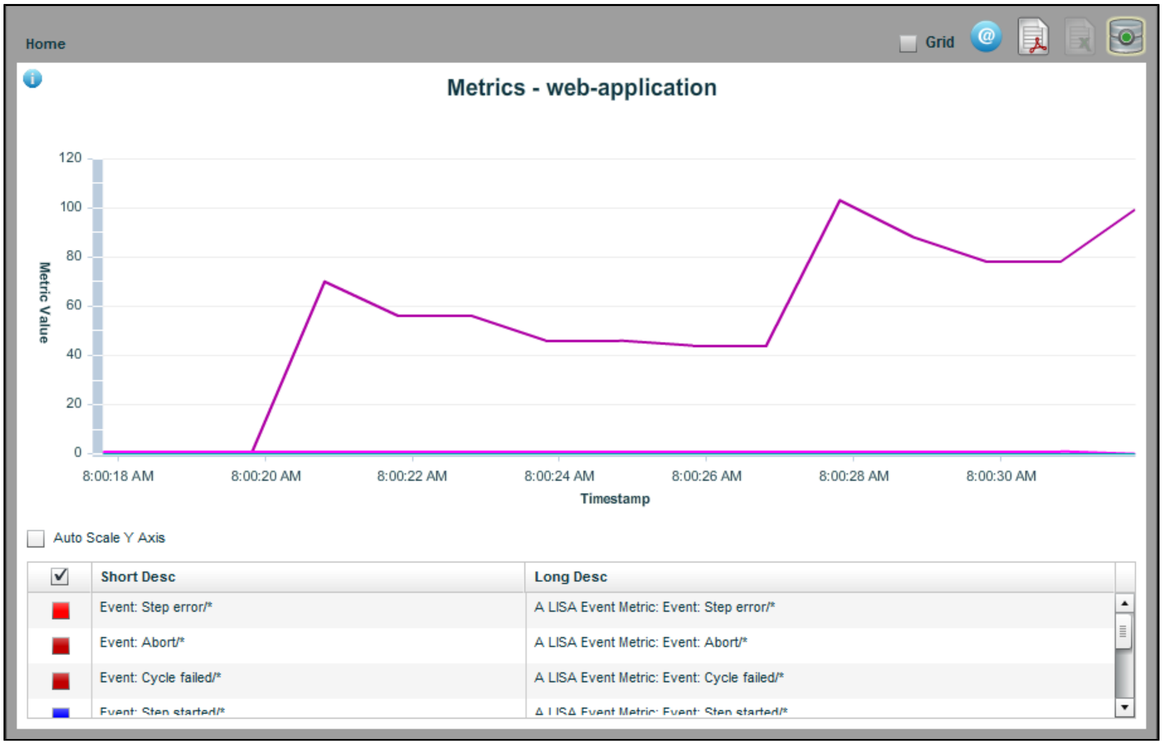
Exporting your report data to XML lets you move reporting data from one registry to another.



```
<?xml version="1.0" ?>

<iTKOReportList>
<iTKOReport>
<RunHistory User="arhoades@Diana-PC" Config="C:\Users\arhoades\lisatmp_5.1
\35663365313234332D653865312D3433\Configs\project.config" End="1306253404603"
Pin="false" RunUniqueId="35663365313234332D653865312D3433"
RunName="2ConcurrentUsers" FailedCount="0" RunDoc="" Start="1306253342762"
Host="Diana-PC" SSVusers="2" AbortCount="0" Coord="tcp://Diana-
PC:2011/Coordinator" LoadPattern="Run N Times" Duration="-1" PaceTime="0"
ErrorCount="0" ModelDocId="456306a0"
StageDocId="35383239616531642D303134342D3462" SuiteUniqueId=""
ModelName="multi-tier-combo" WarningCount="0" DistPattern="Percent
Distribution" Think="100" PaceTrans="0" PassCount="2">
<CycleHistory Sim="tcp://Diana-PC:2014/Simulator" End="1306253403477"
Instance="1" EndedState="0" ErrorCount="0" Start="1306253342785"
WarningCount="0" Cycle="0" Configuration="C:\Users\arhoades\lisatmp_5.1
\35663365313234332D653865312D3433\Configs\project.config"
ModelUniqueId="456306a0" Run="multi-tier-combo [2ConcurrentUsers]"
CycleUniqueId="65323735363566622D613861662D3463">
</CycleHistory>

<CycleHistory Sim="tcp://Diana-PC:2014/Simulator" End="1306253394081"
Instance="0" EndedState="0" ErrorCount="0" Start="1306253342792"
WarningCount="0" Cycle="0" Configuration="C:\Users\arhoades\lisatmp_5.1
\35663365313234332D653865312D3433\Configs\project.config"
ModelUniqueId="456306a0" Run="multi-tier-combo [2ConcurrentUsers]"
CycleUniqueId="30346635643638662D633961622D3434">
</CycleHistory>

<Prop Name="JNDIFACTORY" time="1306253342752" Type="L"><![CDATA
[org.jnp.interfaces.NamingContextFactory]]></Prop>
<Prop Name="LISA_USER" time="1306253342752" Type="L"><![CDATA[arhoades@Diana-
PC]]></Prop>
```

Exporting Report to Excel

You can export all the LISA reports data to an Excel spreadsheet. To export data to Excel file, open the report that you want to export.

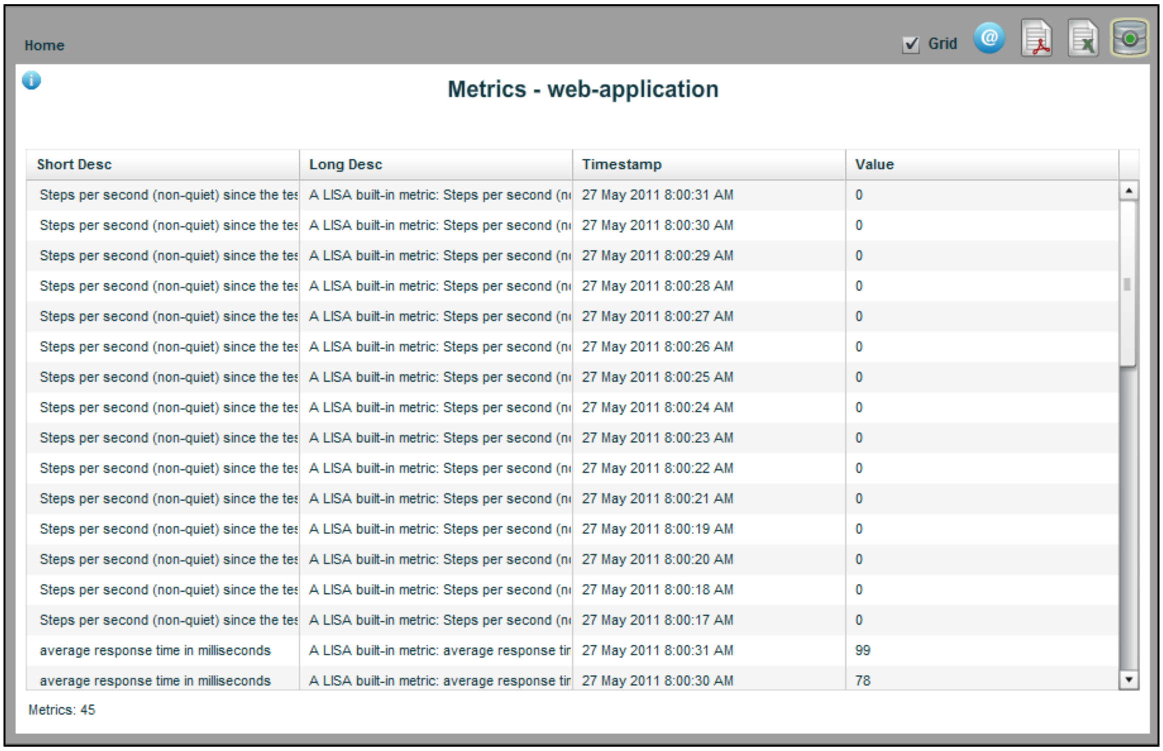


Metrics - web-application

□ Auto Scale Y Axis

☑	Short Desc	Long Desc
🟥	Event: Step error/*	A LISA Event Metric: Event: Step error/*
🟥	Event: Abort/*	A LISA Event Metric: Event: Abort/*
🟥	Event: Cycle failed/*	A LISA Event Metric: Event: Cycle failed/*
🟦	Event: Step started/*	A LISA Event Metric: Event: Step started/*

As you can see here, the Export to Excel icon is disabled. To export a report to Excel, it needs to be viewed in grid view. Select the Grid check box.

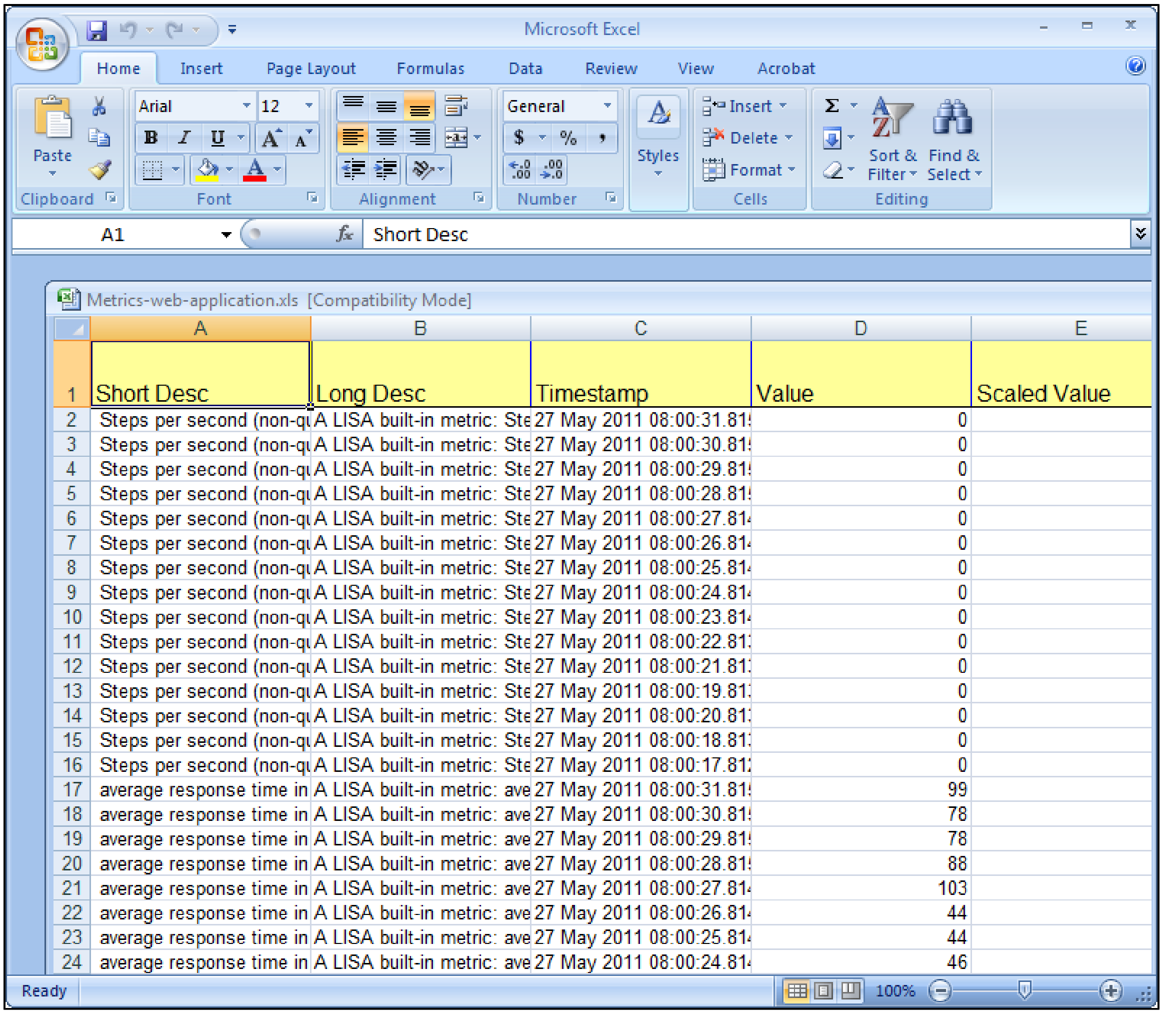


Metrics - web-application

Short Desc	Long Desc	Timestamp	Value
Steps per second (non-quiet) since the tes	A LISA built-in metric: Steps per second (n	27 May 2011 8:00:31 AM	0
Steps per second (non-quiet) since the tes	A LISA built-in metric: Steps per second (n	27 May 2011 8:00:30 AM	0
Steps per second (non-quiet) since the tes	A LISA built-in metric: Steps per second (n	27 May 2011 8:00:29 AM	0
Steps per second (non-quiet) since the tes	A LISA built-in metric: Steps per second (n	27 May 2011 8:00:28 AM	0
Steps per second (non-quiet) since the tes	A LISA built-in metric: Steps per second (n	27 May 2011 8:00:27 AM	0
Steps per second (non-quiet) since the tes	A LISA built-in metric: Steps per second (n	27 May 2011 8:00:26 AM	0
Steps per second (non-quiet) since the tes	A LISA built-in metric: Steps per second (n	27 May 2011 8:00:25 AM	0
Steps per second (non-quiet) since the tes	A LISA built-in metric: Steps per second (n	27 May 2011 8:00:24 AM	0
Steps per second (non-quiet) since the tes	A LISA built-in metric: Steps per second (n	27 May 2011 8:00:23 AM	0
Steps per second (non-quiet) since the tes	A LISA built-in metric: Steps per second (n	27 May 2011 8:00:22 AM	0
Steps per second (non-quiet) since the tes	A LISA built-in metric: Steps per second (n	27 May 2011 8:00:21 AM	0
Steps per second (non-quiet) since the tes	A LISA built-in metric: Steps per second (n	27 May 2011 8:00:19 AM	0
Steps per second (non-quiet) since the tes	A LISA built-in metric: Steps per second (n	27 May 2011 8:00:20 AM	0
Steps per second (non-quiet) since the tes	A LISA built-in metric: Steps per second (n	27 May 2011 8:00:18 AM	0
Steps per second (non-quiet) since the tes	A LISA built-in metric: Steps per second (n	27 May 2011 8:00:17 AM	0
average response time in milliseconds	A LISA built-in metric: average response tir	27 May 2011 8:00:31 AM	99
average response time in milliseconds	A LISA built-in metric: average response tir	27 May 2011 8:00:30 AM	78

Metrics: 45

Now the Export to Excel icon is enabled. Click it to open the Export to file dialog, where you can specify the name of the Excel file. The reporting data will be stored in the saved Excel file.

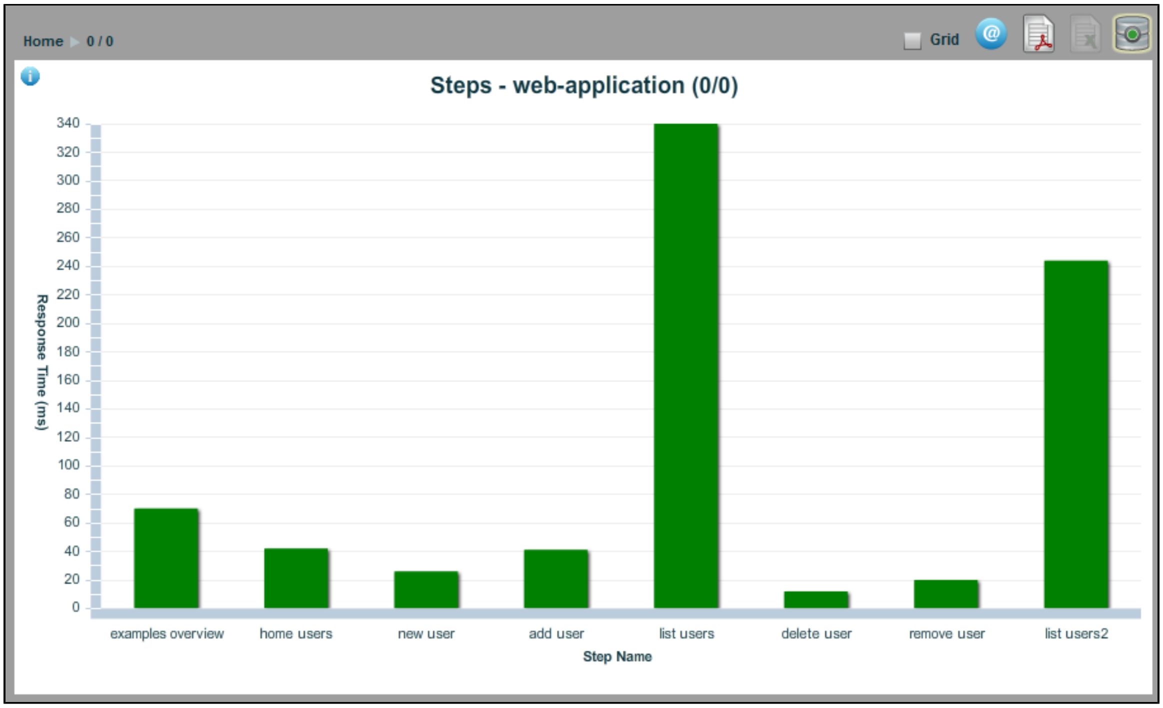


A1 | Short Desc

	A	B	C	D	E
1	Short Desc	Long Desc	Timestamp	Value	Scaled Value
2	Steps per second (non-qu	A LISA built-in metric: Ste	27 May 2011 08:00:31.81	0	
3	Steps per second (non-qu	A LISA built-in metric: Ste	27 May 2011 08:00:30.81	0	
4	Steps per second (non-qu	A LISA built-in metric: Ste	27 May 2011 08:00:29.81	0	
5	Steps per second (non-qu	A LISA built-in metric: Ste	27 May 2011 08:00:28.81	0	
6	Steps per second (non-qu	A LISA built-in metric: Ste	27 May 2011 08:00:27.81	0	
7	Steps per second (non-qu	A LISA built-in metric: Ste	27 May 2011 08:00:26.81	0	
8	Steps per second (non-qu	A LISA built-in metric: Ste	27 May 2011 08:00:25.81	0	
9	Steps per second (non-qu	A LISA built-in metric: Ste	27 May 2011 08:00:24.81	0	
10	Steps per second (non-qu	A LISA built-in metric: Ste	27 May 2011 08:00:23.81	0	
11	Steps per second (non-qu	A LISA built-in metric: Ste	27 May 2011 08:00:22.81	0	
12	Steps per second (non-qu	A LISA built-in metric: Ste	27 May 2011 08:00:21.81	0	
13	Steps per second (non-qu	A LISA built-in metric: Ste	27 May 2011 08:00:19.81	0	
14	Steps per second (non-qu	A LISA built-in metric: Ste	27 May 2011 08:00:20.81	0	
15	Steps per second (non-qu	A LISA built-in metric: Ste	27 May 2011 08:00:18.81	0	
16	Steps per second (non-qu	A LISA built-in metric: Ste	27 May 2011 08:00:17.81	0	
17	average response time in	A LISA built-in metric: ave	27 May 2011 08:00:31.81	99	
18	average response time in	A LISA built-in metric: ave	27 May 2011 08:00:30.81	78	
19	average response time in	A LISA built-in metric: ave	27 May 2011 08:00:29.81	78	
20	average response time in	A LISA built-in metric: ave	27 May 2011 08:00:28.81	88	
21	average response time in	A LISA built-in metric: ave	27 May 2011 08:00:27.81	103	
22	average response time in	A LISA built-in metric: ave	27 May 2011 08:00:26.81	44	
23	average response time in	A LISA built-in metric: ave	27 May 2011 08:00:25.81	44	
24	average response time in	A LISA built-in metric: ave	27 May 2011 08:00:24.81	46	

Exporting Reports to PDF

You can also export the report data to PDF. Open the Graphical view of the report.



To export to PDF, click the Export to PDF icon.

The pre-formatted LISA report that is exported is dependent on the location from which you choose the export. The report will be one of four types, each detailed in Standard LISA Reports.

Changing Reporting Databases

The way you configure LISA to connect to alternate databases has changed in LISA 6.0, which can be seen in **lisa.properties** or **site.properties**. LISA database components are now set by assigning each component to a defined pool configuration like this:

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

By default all of the LISA databases share a common Derby database connection pool as defined here

```
lisadb.pool.common.driverClass=org.apache.derby.jdbc.ClientDriver
lisadb.pool.common.url=jdbc:derby://localhost:1528/database/lisa.db;create=true
lisadb.pool.common.user=rpt
lisadb.pool.common.password=rpt
```

For example if you want to change the reporting database to connect to Oracle, you first define a new database pool configuration

```
lisadb.pool.mypool.driverClass=oracle.jdbc.OracleDriver
lisadb.pool.mypool.url=jdbc:oracle:thin:@//myhost:1521/orcl
lisadb.pool.mypool.user=rpt
lisadb.pool.mypool.password=rpt
```

and then set the reporting component to use that pool.

```
lisadb.reporting.poolName=mypool
```

When you enter the Reporting Portal, you can mouse over the database icon on the upper-right corner of the screen to get the details about the database you are using.

Recorders and Test Generators

There are a variety of methods to record test cases and re-run them.

LISA Test's no-code testing environment allows QA, Development and others to rapidly design and execute functional, unit, regression and load tests against dynamic websites (RIAs).

The product can be used to test rich browser and web user interfaces, and the many building blocks and data residing below the UI. With LISA, all the data and implementation layers the team needs to functionally test can be analyzed, invoked and verified to help ensure requirements are met.

In this section, the following topics are covered:

Recording a Website

Recording a website with LISA can be done in two ways:

You can use **Web Recorder (HTTP Proxy)** when you want to track the path through the website. Steps are created for each HTTP request.

You can use **Web Recorder (DOM Events**) when you want to capture mouse clicks, mouse movements, keys being typed, and so on, and play those events back during test execution.

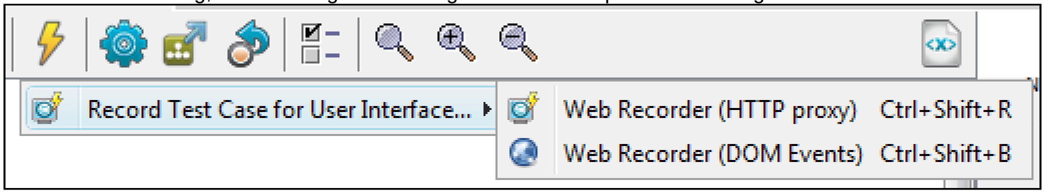
The following sections are available.

Recording a Website via HTTP Proxy

LISA Workstation provides a HTTP recorder to test a website test case using a proxy recorder.

This helps track your path through the website and automatically create test steps for each HTTP request that is generated while recording.

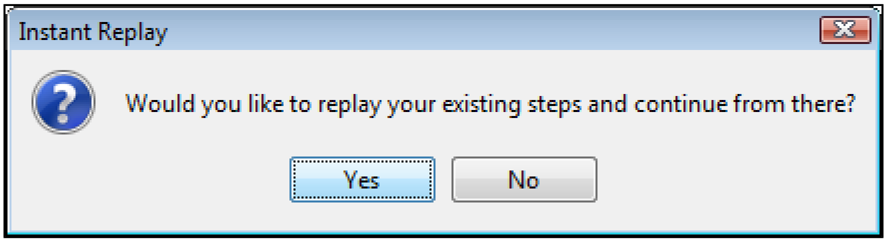
To start the recording, click the Begin Recording ⚡ icon to open the following menu.


Record Test Case for User Interface...
Web Recorder (HTTP proxy) Ctrl+Shift+R
Web Recorder (DOM Events) Ctrl+Shift+B

Click Record Test Case for User Interface > Web Recorder (HTTP Proxy), or from the main menu, select Actions > Record Test Case for User Interface > Web Recorder (HTTP Proxy).

This lets you launch the browser that is used to record and play back HTTP tests.

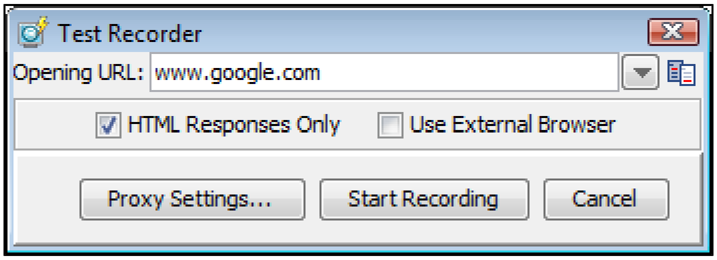
If there is a test case already open in the active tab, it will ask whether you want to replay the tests in the browser.


Instant Replay

Would you like to replay your existing steps and continue from there?

If there is no test case open in LISA Workstation, the Test Recorder window opens, where you enter the name of the website to record.

- Enter the URL for the web page to test.


Test Recorder

Opening URL: www.google.com

☑ HTML Responses Only ☐ Use External Browser

Select your preferences from the following:

- **HTML Responses Only:** Will capture only the HTML responses.
- **Use External Browser:** Will open an external browser window.

Port Usage

By default the LISA Proxy recorder uses port 8010 for recording.

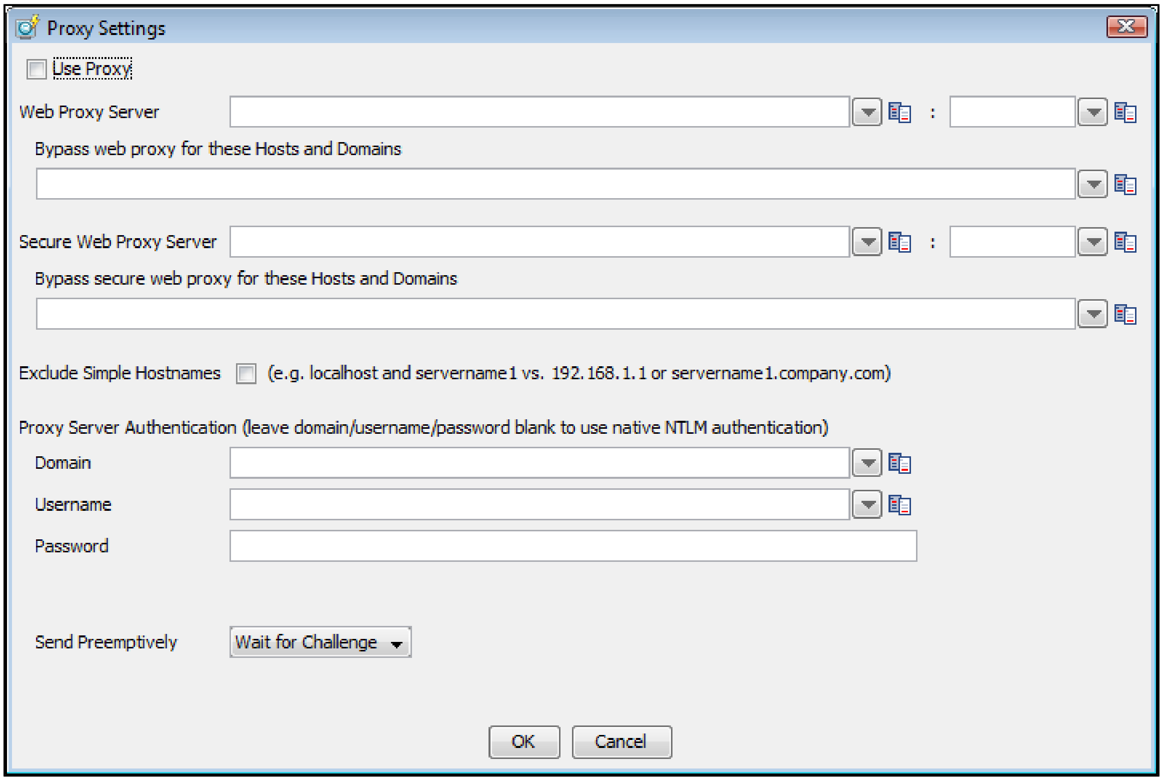
If you do not want to use this port, you can override the setting in your **lisa.properties** file with the following property: **lisa.editor.http.recorderPort=8010**.

To start the recording, click Start Recording to start the Web recorder, or click Proxy Settings to configure the proxy.

The following topics are available.

Configure Proxy

To set the proxy, click the Proxy Settings button to open the Proxy Setting dialog.



☐ Use Proxy

☐ Exclude Simple Hostnames (e.g. localhost and servername1 vs. 192.168.1.1 or servername1.company.com)

Proxy Server Authentication (leave domain/username/password blank to use native NTLM authentication)

Send Preemptively: Wait for Challenge

Proxy Settings
Web Proxy Server
:
Bypass web proxy for these Hosts and Domains
Secure Web Proxy Server
:
Bypass secure web proxy for these Hosts and Domains
Domain
Username
Password
OK
Cancel

- **Use Proxy**: Check this to use proxy settings. This is useful when you want to enter the proxy settings and use it intermittently.
- **Web Proxy Server**: Enter the proxy server hostname (server IP) and respective port number.
  - **Bypass Web Proxy for these hosts and domains**: Enter the host name and domains of those servers for which you want to bypass proxy. Enter a pipe (|) separated list of hosts to be connected to directly and not through a proxy server. A asterisk * can be used as a wildcard character for matching; for example, "*.foo.com|localhost".
- **Secure Web Proxy Server**: Enter the secure proxy settings.
  - **Bypass Web Proxy Server for these hosts and domains**: Enter the host name and domains for which you need to bypass proxy.
- **Exclude simple host names**: Check if you want to exclude simple host names (for example, **localhost** and **servername** compared to **192.168.1.1** or **server1.company.com**.)
- **Proxy server Authentication**: Enter authentication details.
  - Domain: Enter domain name.
  - User name: Enter user name.
  - Password: Enter password.
- **Send Preemtively**: Select **Wait for Challenge/Send Basic** or **Send NTLM.**

> ⚠ Typically the proxy server will challenge any request when authentication is required. If the proxy server does not send the challenge, setting this field will force the authentication header to be set with the first request.

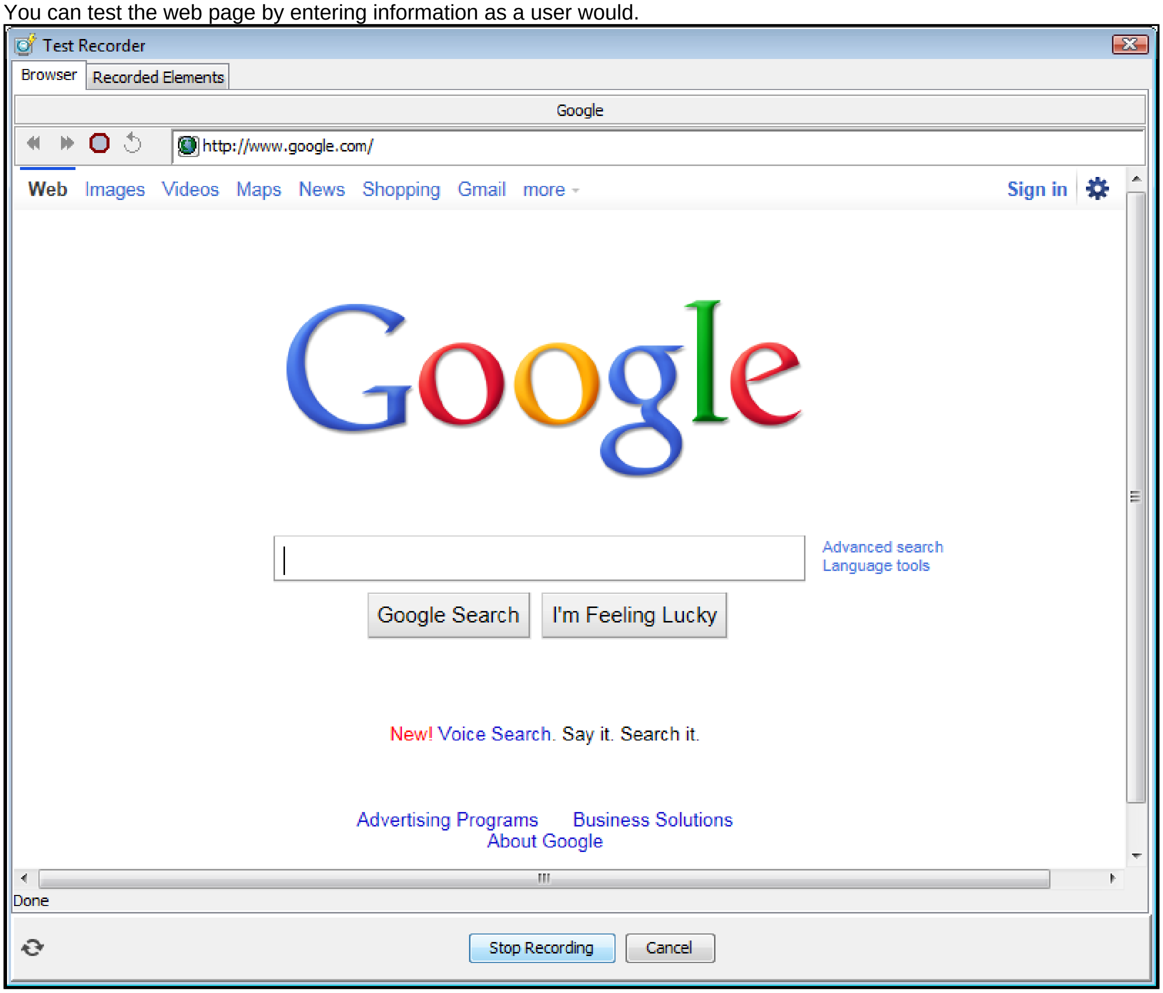
Enter the Proxy configuration information in the dialog and click OK to set the proxy settings.

Start Recording

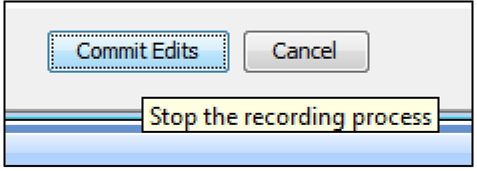
Click Start Recording in the Test Generator to begin recording the test.

The Test Recorder window opens up and displays the web page URL loaded.

You can test the web page by entering information as a user would.



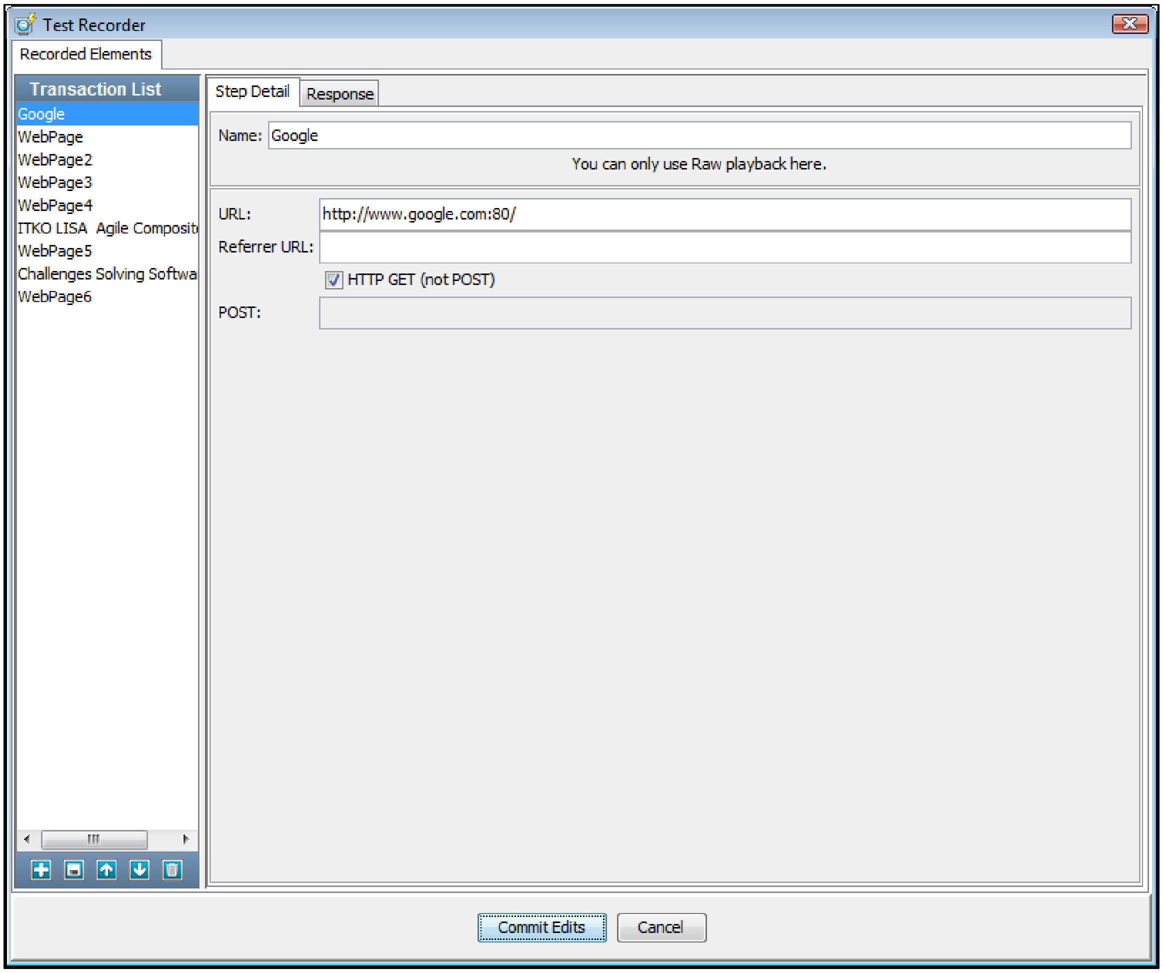
After you are done, you can stop recording your browser activity by clicking Stop Recording at the bottom of the Test Recorder window. You will then see the Recorded Elements screen. To complete recording, click the Commit Edits button.



View Recorded Transactions

After you stop recording, all the recorded transactions are shown in the **Recorded Elements** tab.

All the transactions done are listed in the left panel. The **Step Details** and the **Response** are seen in the right tab.



Name: Google

You can only use Raw playback here.

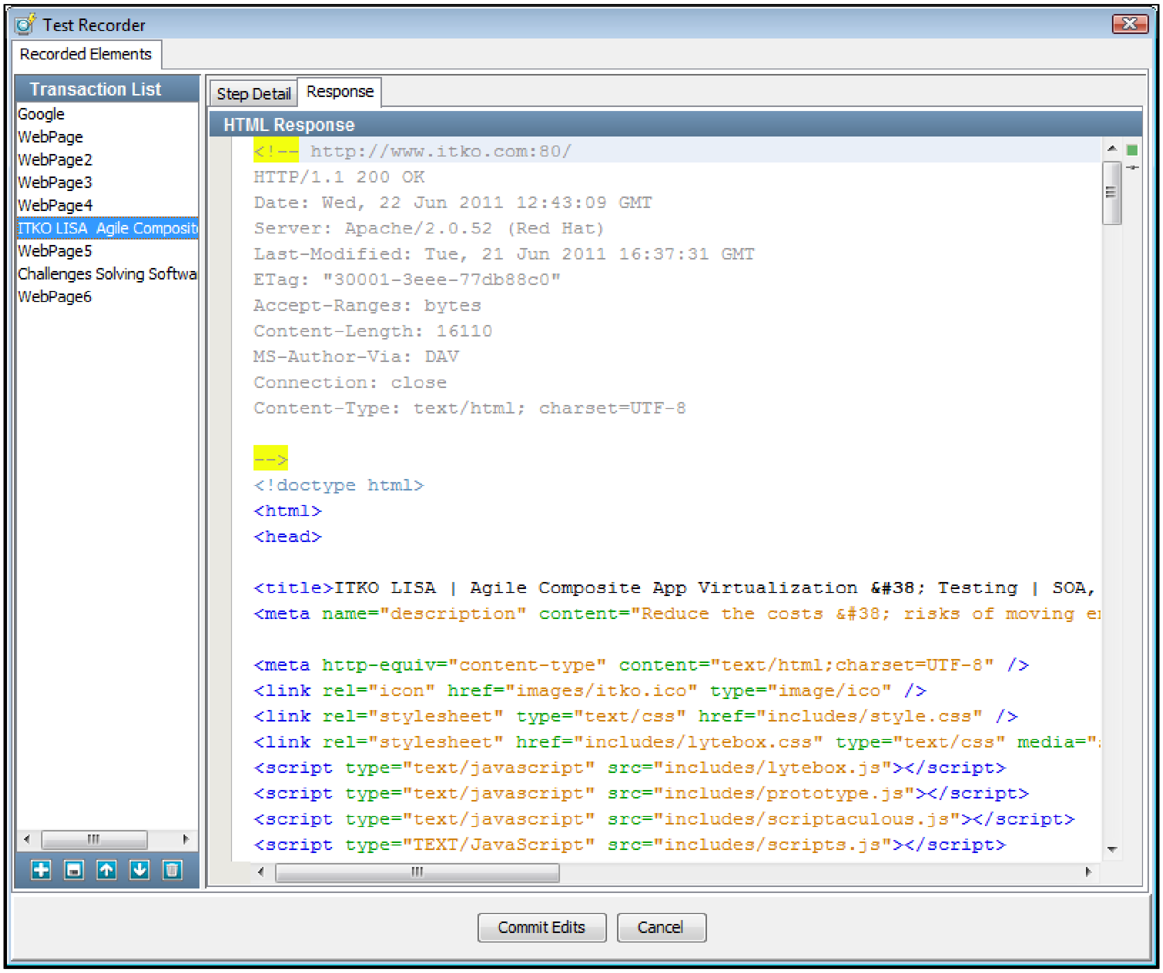
URL: http://www.google.com:80/

Referrer URL:

☑ HTTP GET (not POST)

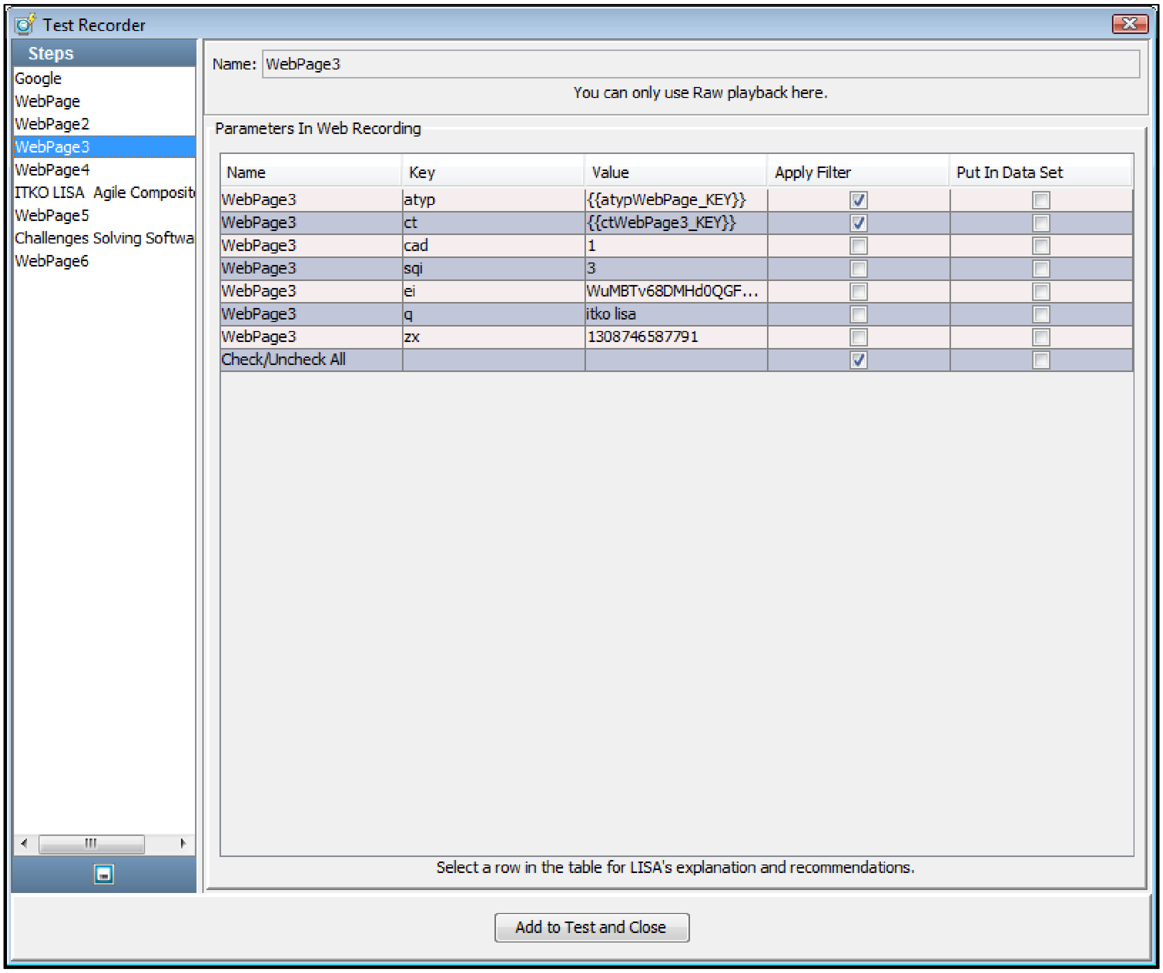
POST:

1. Select the **Response** tab to see the HTML response recorded.



```
<!-- http://www.itko.com:80/
HTTP/1.1 200 OK
Date: Wed, 22 Jun 2011 12:43:09 GMT
Server: Apache/2.0.52 (Red Hat)
Last-Modified: Tue, 21 Jun 2011 16:37:31 GMT
ETag: "30001-3eee-77db88c0"
Accept-Ranges: bytes
Content-Length: 16110
MS-Author-Via: DAV
Connection: close
Content-Type: text/html; charset=UTF-8

-->
<!doctype html>
<html>
<head>

<title>ITKO LISA | Agile Composite App Virtualization &#38; Testing | SOA,
<meta name="description" content="Reduce the costs &#38; risks of moving e

<meta http-equiv="content-type" content="text/html;charset=UTF-8" />
<link rel="icon" href="images/itko.ico" type="image/ico" />
<link rel="stylesheet" type="text/css" href="includes/style.css" />
<link rel="stylesheet" href="includes/lytebox.css" type="text/css" media="
<script type="text/javascript" src="includes/lytebox.js"></script>
<script type="text/javascript" src="includes/prototype.js"></script>
<script type="text/javascript" src="includes/scriptaculous.js"></script>
<script type="TEXT/JavaScript" src="includes/scripts.js"></script>
```

2. Click Commit Edits to commit these transactions in the Test Recorder.



Test Recorder

Steps
- Google
- WebPage
- WebPage2
- WebPage3
- WebPage4
- ITKO LISA Agile Composit
- WebPage5
- Challenges Solving Softwa
- WebPage6

Name: WebPage3

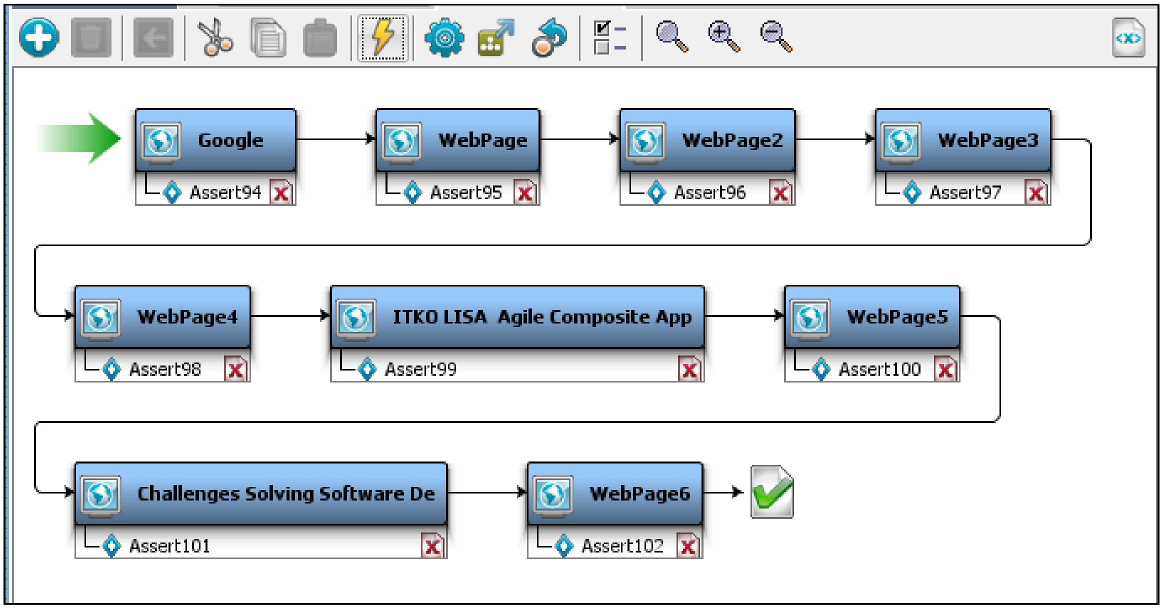
You can only use Raw playback here.

Parameters In Web Recording

Name	Key	Value	Apply Filter	Put In Data Set
WebPage3	atyp	{{atypWebPage_KEY}}	☑	□
WebPage3	ct	{{ctWebPage3_KEY}}	☑	□
WebPage3	cad	1	□	□
WebPage3	sqi	3	□	□
WebPage3	ei	WuMBTv68DMHd0QGF...	□	□
WebPage3	q	itko lisa	□	□
WebPage3	zx	1308746587791	□	□
Check/Uncheck All			☑	□

Select a row in the table for LISA's explanation and recommendations.

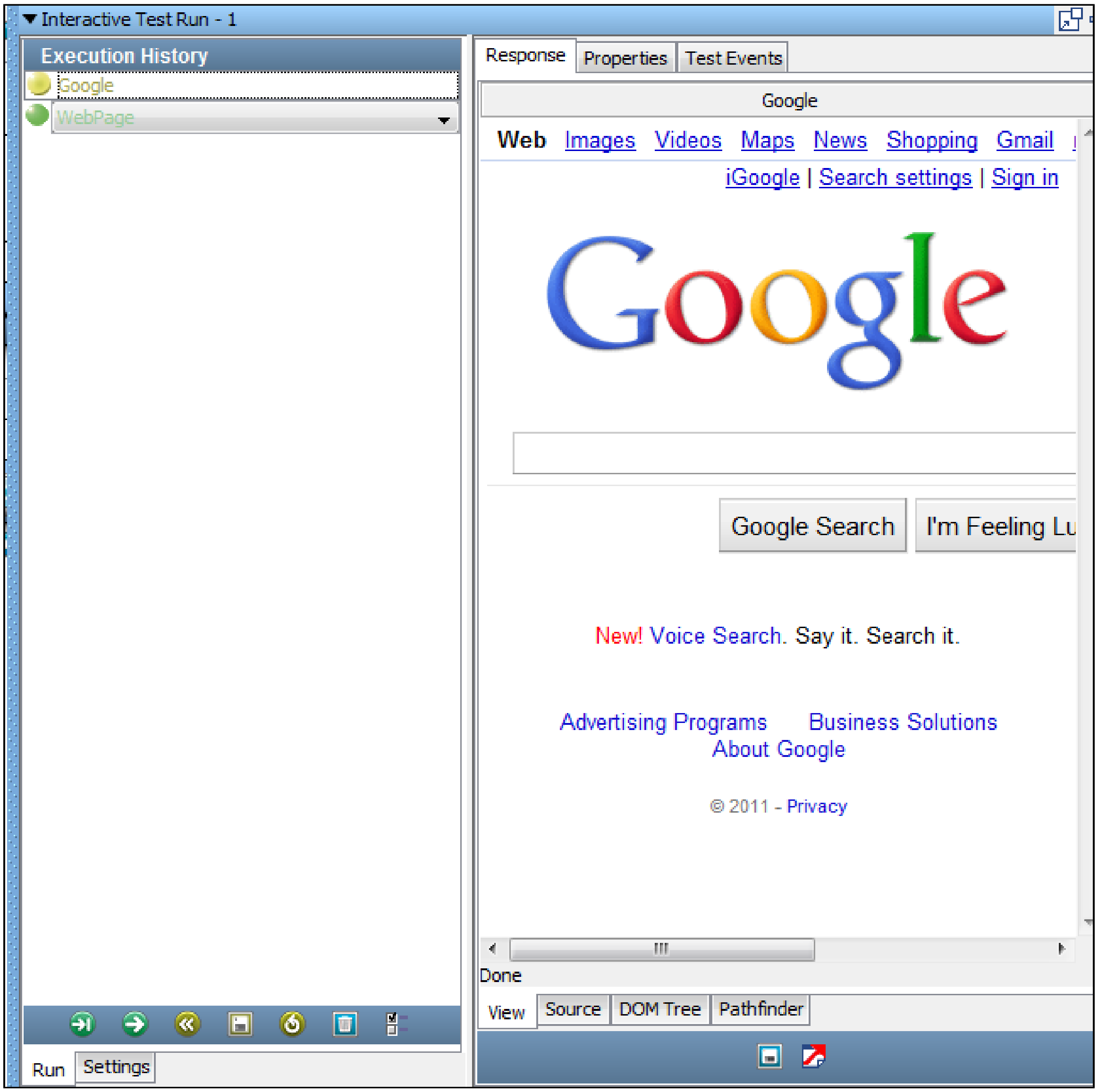
3. In the **Parameters In Web Recording** enter any parameters required by your test.
4. Click Add to Test and Close at the bottom of the Test Recorder window to add transactions as test steps.
5. A test case is created based on your HTTP requests in the LISA workflow. Each step in the test case represents a recorded HTTP request.



View in ITR

You can view all these transactions again when you run this test case in the ITR.

See the **View, Source, DOM Tree** and **Pathfinder** tabs to see more information about the recorded step.



Recording a Website via DOM Events

This feature is covered in a separate user guide. See the *Web 2.0 Guide* for additional information.

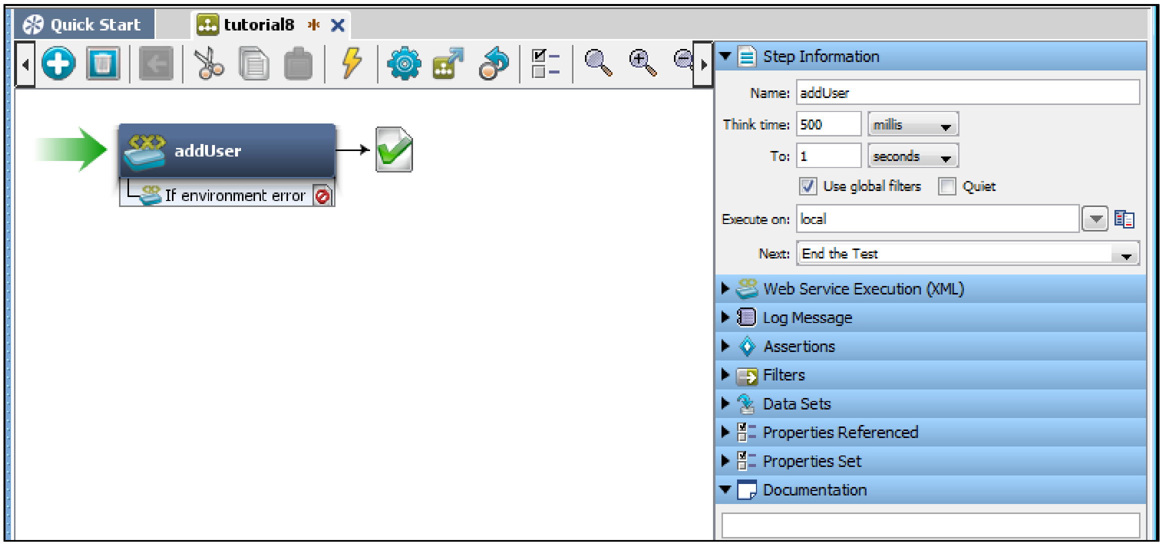
Generating a Web Service

You can create a Web Service (XML) test case. For this, you use the Web Service Execution (XML) step to call web service operations in a test case and test the response and request. These web service operations provide the same functionality as the equivalent method calls in the EJB used in Tutorial 7. Additional information about the Web Service Execution step is available at Web Service Execution (XML) Step.

Make sure you are running the Demo Server (either the local Demo Server, or the ITKO demo server) to use this step.

Creating the Web Service (XML) Step

1. To create a Web Service XML step, open LISA Workstation and right-click in the model editor. Click Add Steps or from the test case toolbar or click the Add Steps icon.
2. Click Web/Web Services> Web Service Execution (XML), to add the step in the Model editor.



Name: addUser

Think time: 500 millis

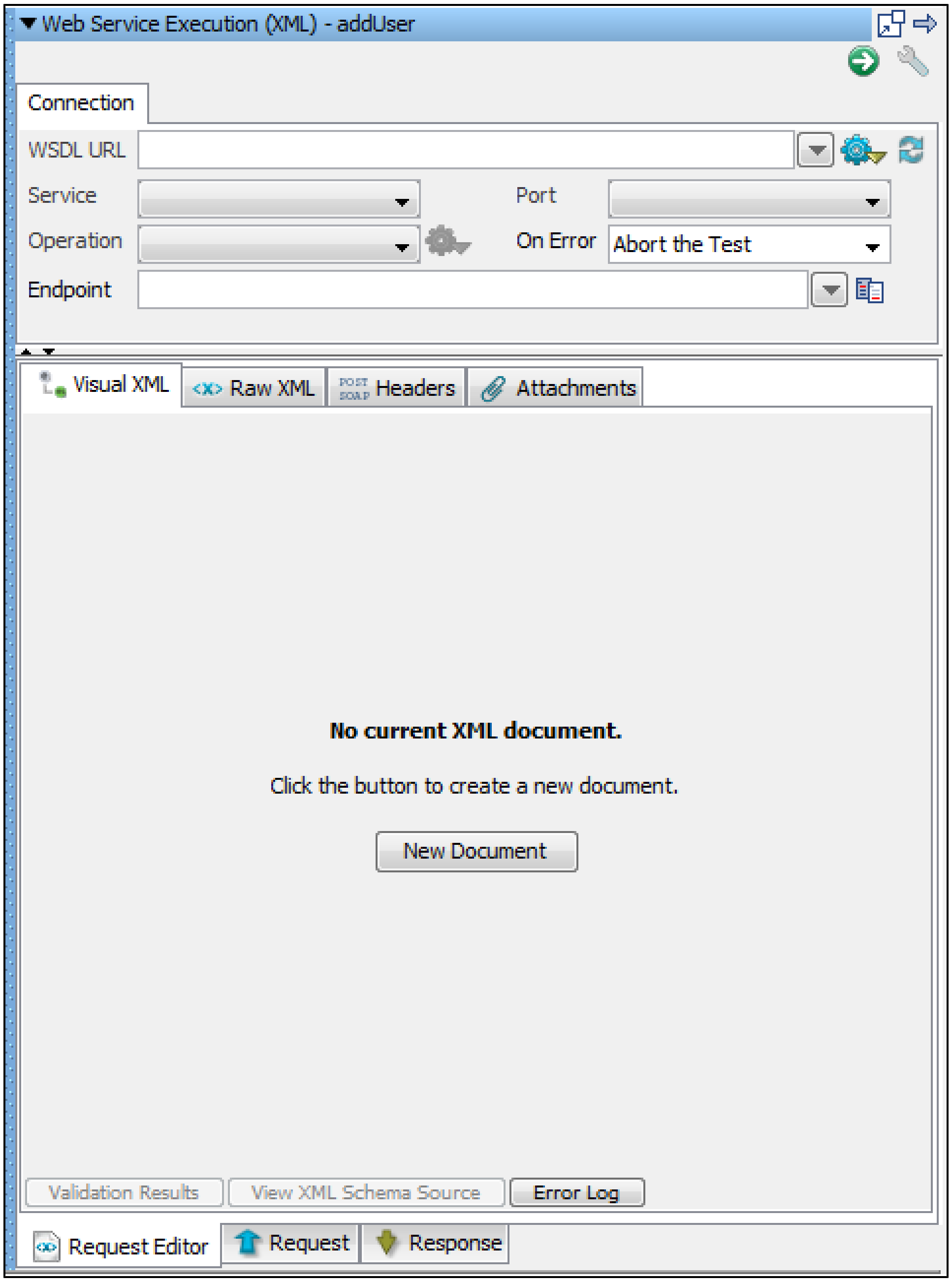
To: 1 seconds

☑ Use global filters □ Quiet

Execute on: local

Next: End the Test

3. A Web Services step is added. Rename the step **AddUser** in the Step Information area.
4. Double-click the **Add User** step to open the **Web Service Execution** editor.

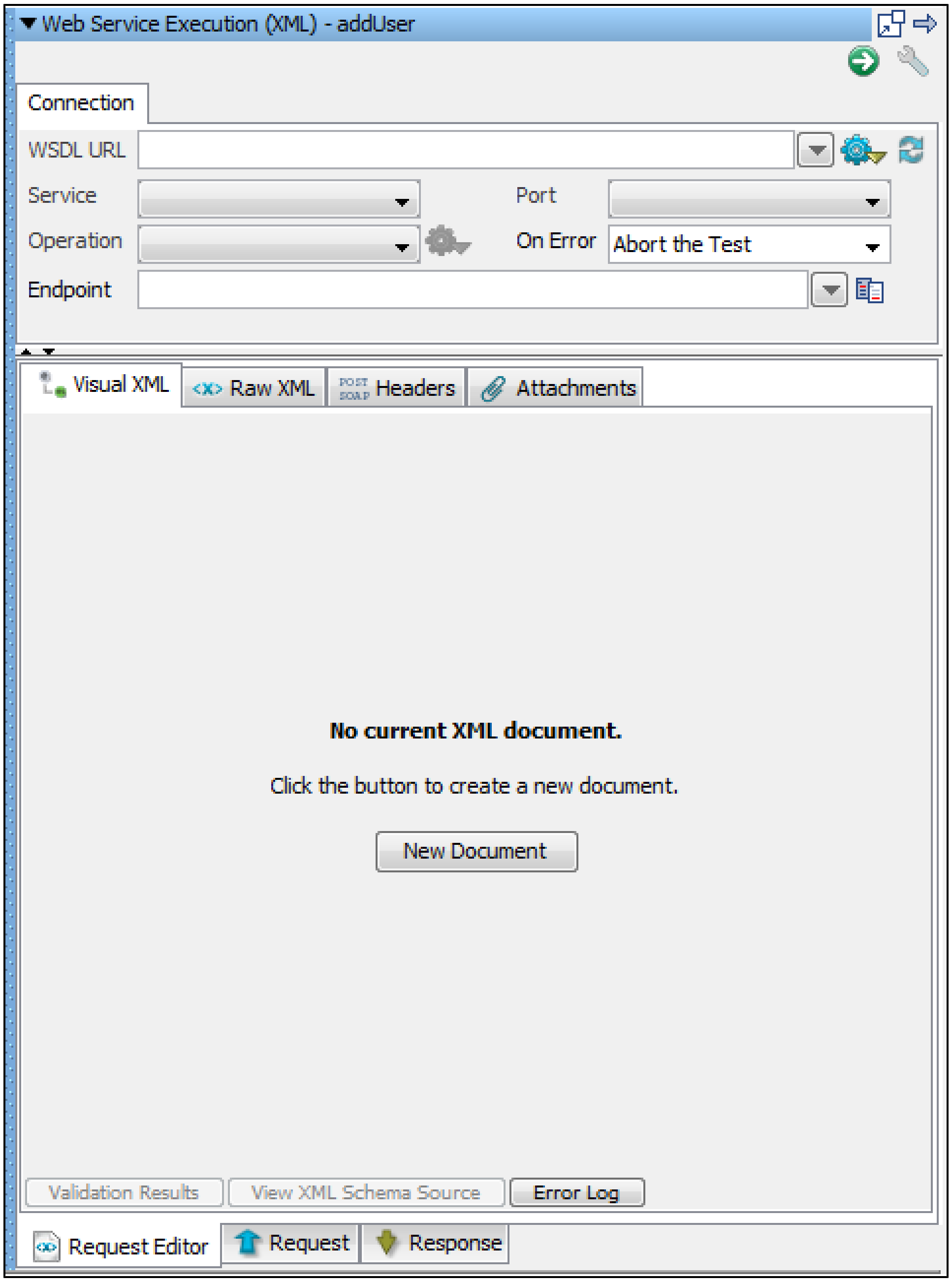


On Error: Abort the Test

No current XML document.

Click the button to create a new document.

5. Click the New Document button to create a new XML document.



On Error: Abort the Test

No current XML document.

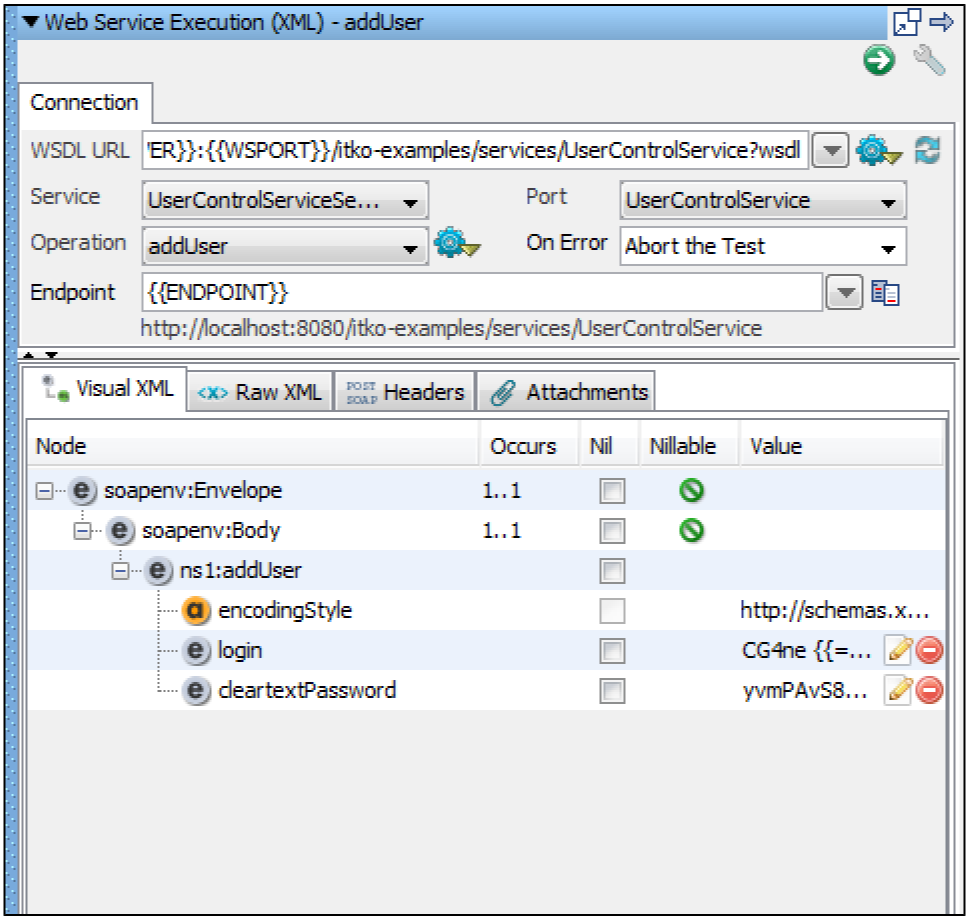
Click the button to create a new document.

Creating the Web Service Client

1. In the WSDL field, enter the location of the WSDL.

```
http://WSSERVER:WSPORT/itko-examples/services/UserControlService?wsdl
```

2. In the Service Name field, enter **UserControlServiceService**. Do not use spaces within the name of the web service.
3. In the Port field, enter **UserControlServiceService**.
4. In the Operation field, select the operation to be tested.
5. In the On Error field, select the action to be taken on test error: **Abort the Test**.



Web Service Execution (XML) - addUser

Connection

WSDL URL: 'ER}}:{{WSPORT}}/itko-examples/services/UserControlService?wsdl

Service: UserControlServiceSe...

Port: UserControlService

Operation: addUser

On Error: Abort the Test

Endpoint: {{ENDPOINT}}

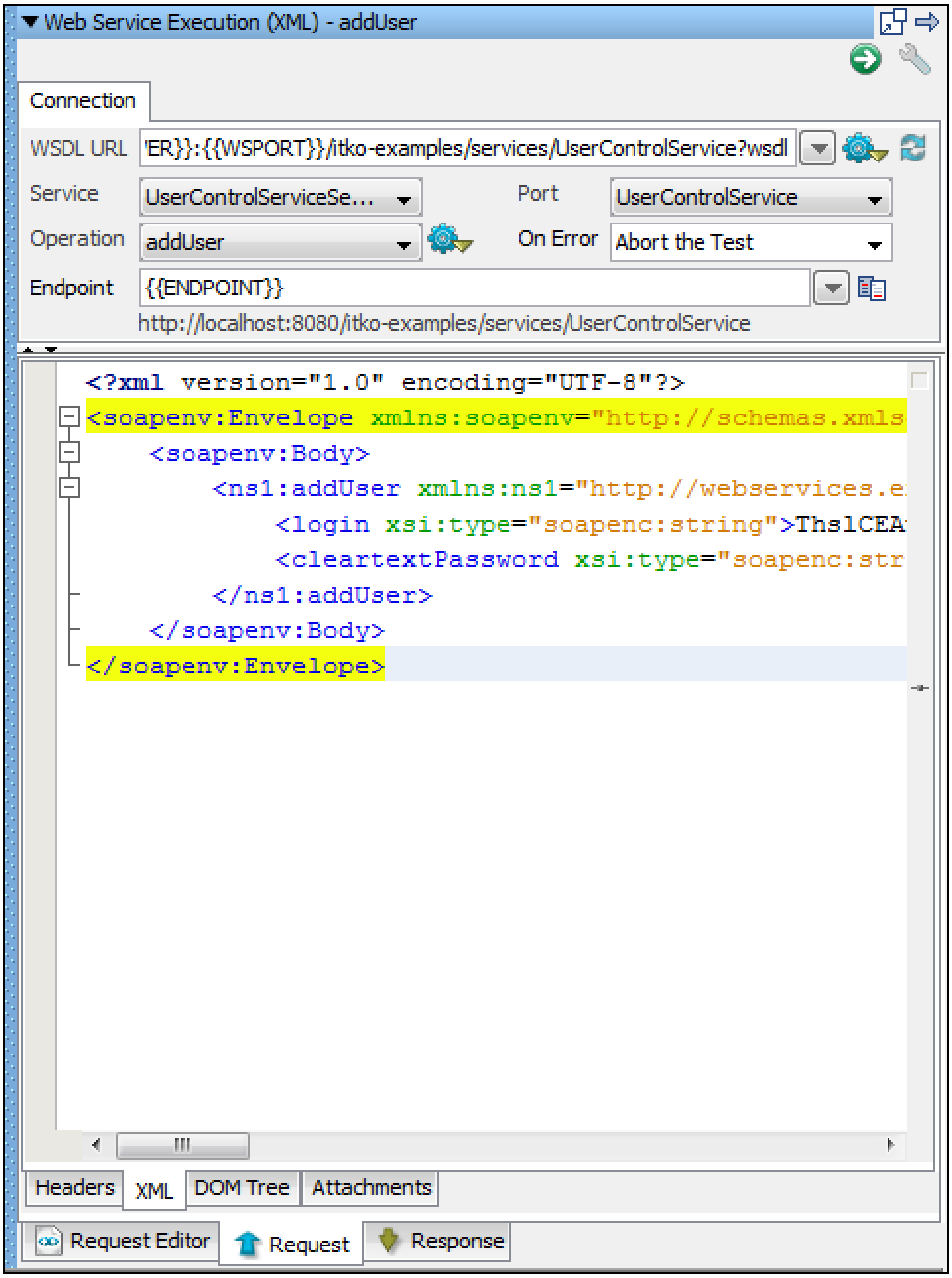
http://localhost:8080/itko-examples/services/UserControlService

Node	Occurs	Nil	Nillable	Value
soapenv:Envelope	1..1	□	⊘	
soapenv:Body	1..1	□	⊘	
ns1:addUser		□		
encodingStyle		□		http://schemas.x...
login		□		CG4ne {{=...
cleartextPassword		□		yvmPAvS8...

Executing the Test Case

To execute the test case:

1. Click the Execute ➔ button on top right. This will execute the test and show us the Request and Response.
2. To view the request upon execution, click the Request tab.



WSDL URL 'ER}}:{{WSPORT}}/itko-examples/services/UserControlService?wsdl

Service UserControlServiceSe...

Port UserControlService

Operation addUser

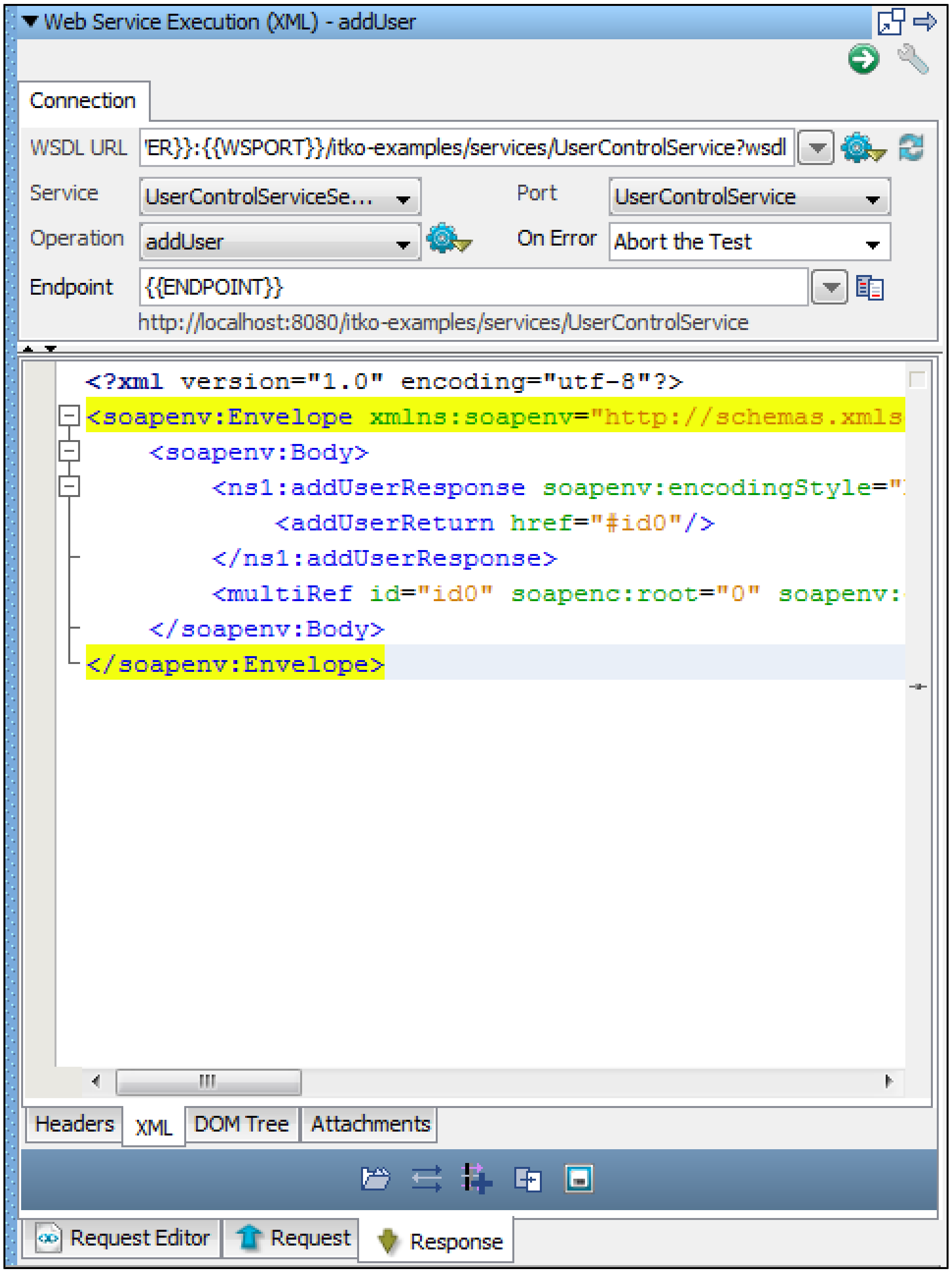
On Error Abort the Test

Endpoint {{ENDPOINT}}

http://localhost:8080/itko-examples/services/UserControlService

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmls
    <soapenv:Body>
        <ns1:addUser xmlns:ns1="http://webservices.e
            <login xsi:type="soapenc:string">ThslCEA
            <cleartextPassword xsi:type="soapenc:str
        </ns1:addUser>
    </soapenv:Body>
</soapenv:Envelope>
```

3. To view the response upon execution, click the **Response** tab.



WSDL URL 'ER}}:{{WSPORT}}/itko-examples/services/UserControlService?wsdl

Service UserControlServiceSe...

Port UserControlService

Operation addUser

On Error Abort the Test

Endpoint {{ENDPOINT}}

http://localhost:8080/itko-examples/services/UserControlService

```
<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmls
    <soapenv:Body>
        <ns1:addUserResponse soapenv:encodingStyle="
            <addUserReturn href="#id0"/>
        </ns1:addUserResponse>
        <multiRef id="id0" soapenc:root="0" soapenv:
    </soapenv:Body>
</soapenv:Envelope>
```

Advanced Features

The following topics are covered:

Using BeanShell in LISA

BeanShell (http://www.beanshell.org/) is a free, open-source, lightweight Java scripting language. It is a Java application that uses the Reflection API to execute Java statements and expressions dynamically. By using BeanShell, you avoid the need to compile class files.

BeanShell lets you type standard Java syntax (statements and expressions) on a command line and see the results immediately. A Swing GUI is also available. BeanShell can also be called from within a Java class; this is how it is used in the product.

BeanShell is used in several places:

- To interpret property expressions
- As the interpreter framework for the Java Script test step
- As the interpreter framework for the Assert by Script Execution assertion

The following topics are available in this chapter.

Using BeanShell Scripting Language

The major difference between BeanShell Java and compiled Java is in the type system. Java is very strongly typed, whereas BeanShell can loosen the typing in its scripting environment. You can, however, impose strict typing in BeanShell if you want.

BeanShell relaxes typing in a natural way so that you can write BeanShell scripts that look like standard Java method code, while on the other hand you can write scripts that look more like a traditional scripting language, such as Perl or JavaScript, while still maintaining the framework of the Java syntax.

If a variable has been typed, then BeanShell will honor and check the type. If a variable is not typed, BeanShell will only signal an error if you attempt to misuse the actual type of the variable.

The following Java fragments are all valid in BeanShell:

```
foo = "Foo";
four = (2+2) * 2 / 2.0;
print(foo + " = " + four);
.
.
hash = new Hashtable();
date = new Date();
hash.put("today", date);
.
.
```

BeanShell lets you declare and then use methods. Arguments and return types can also be loosely typed:

Typed

```
int addTwoNumbers(int a, int b){
return a + b;
}
```

Loosely Typed

```
add (a,b){
return a + b;
}
```

In the second example, the following would work correctly:

```
sumI = add (5,7);
sumS = add("LISA " , "Rocks");
sumM = add ("version ", 2);
```

BeanShell also provides a library of commands that facilitate its use.

A few examples of these commands are:

- **source()**: Read a BeanShell (bsh) script.
- **run()**: Run a bsh script.
- **exec()**: Run a native application.
- **cd()**, **copy()**, and so on: UNIX-like shell commands.
- **print()**: Print argument as a string.
- **eval()**: Evaluate string argument as code.

For more information, see the BeanShell User Guide at http://www.beanshell.org/. You can also get BeanShell, the source code, and the complete Javadoc at the same place.

Using BeanShell as Standalone

BeanShell is available as a standalone interpreter so you can try it outside of LISA. You can download BeanShell from www.beanshell.org. It is a single small JAR file named **bsh-xx.jar** (**xx** is the version number; currently 2.0). Add the JAR file to your classpath.

You can use BeanShell in the following configurations:

- **From a command line**: java bsh.Interpreter [script name] [args]
- **From BeanShell GUI**: java bsh.Console
- **From within a Java class**:

```
Import bsh.Interpreter;
.
.
Interpreter I = new Interpreter();
i.set ("x",5);
i.set("today", new Date());
Date d = (Date)i.get("date");
i.eval("myX = x * 10");
System.out.println(i.get("myX"));
.
```

Using BeanShell in LISA

The BeanShell interpreter is used in the Java Script Execution step and the Assert by Script Execution assertion. Both of these elements also expose LISA Java objects and the current LISA state (properties). This provides a powerful environment for you to use to add custom functionality. The exposed Java objects can be used to both interrogate and modify the current state of the test. For example, you can read, modify, and create LISA properties in your scripts.

As a starting point, become familiar with the **TestExec** class in LISA. Information about TestExec and many other LISA classes can be found in the *Developer's Guide (SDK)*. The SDK includes both a Developer Guide and Javadoc for the LISA classes that you have access to in these scripts.

LISA also uses BeanShell inside property notation when an equal sign is present. For example:

```
{{= new Date()}}
```

This property expression is interpreted using BeanShell.

Using Date Utilities

There are a number of date utility functions as static methods of the **com.itko.util.DateUtils** class. These functions all return the formatted date as a string. You can use these functions in parameter expressions or the JavaScript Execution step.

```
com.itko.util.DateUtils.formatDate(Date date, String format)
com.itko.util.DateUtils.formatCurrentDate(String format)
com.itko.util.DateUtils.formatCurrentDate(int offsetInSec, String format)
com.itko.util.DateUtils.rfc3339(Date date)
com.itko.util.DateUtils.rfc3339()
com.itko.util.DateUtils.rfc3339(int offsetInSec)
com.itko.util.DateUtils.samlDate(Date date)
com.itko.util.DateUtils.samlDate()
com.itko.util.DateUtils.samlDate(int offsetInSec)
```

For example, if you have a web service call that takes a formatted date string, and the server is two minutes slow, you can use

```
=com.itko.util.DateUtils.formatCurrentDate(-120,"yyyy-MM-dd'T'HH:mm:ss.SSSZ")
```

This will generate the string "2007-11-22T13:30:37.545-0500", the current time minus 120 seconds formatted according to these guidelines.

RFC 3339 is slightly different from what the default Java date formatter will generate. If you need a strict RFC 3339 date you can use the rcf3339 functions:

```
=com.itko.util.DateUtils.rfc3339()
```

This will generate the string "2007-11-22T13:30:37.545-05:00".

SAML dates are formatted using the format "**yyyy-MM-dd'T'HH:mm:ss'Z'**". The samlDate functions are just helpers so you do not need to remember that format string when using the formatDate APIs.

For more information, see:

- http://download.oracle.com/javase/1.5.0/docs/api/java/text/SimpleDateFormat.html
- http://tools.ietf.org/html/rfc3339#section-5.6

Class Loader Sandbox Example

The following Java class is a simple example of a class that cannot be run in multi-threaded or multi-user fashion, because it accesses and modifies a static variable:

```
public class NeedsASandbox \{

static \{

System.out.println("This is my static initializer. You will see this many times.");

&nbsp;

static String s;

&nbsp;

public NeedsASandbox() \{\};

&nbsp;

public void setS(String s)\{

this.s = s;

public String getS() \{

return s;
```

This class must run in a class loader sandbox.

Assume, for example, that you were to run this class within LISA and created a load test of ten users. If you do not use the class loader sandbox, you would see the System.out.println phrases only one time, and the value of s would be incorrect. This is because all users will be running within one class loader. This particular class will fail in those circumstances. Presuming that this is the proper function of the application, you will need to use support for the class loader sandbox to make this work properly.

When you create the Class Loader Sandbox Companion, and LISA stages your ten users, you will actually see the text phrase in the code appear ten times. LISA will construct ten separate class loaders and instantiate this class ten times; there will be ten separate instances of the class variable "static string s." This lets your application logic, which is not thread safe, to be run in concurrent user tests.

The Class Loader Sandbox Companion is useful only if the following three conditions are present:

- You are testing a POJO with LISA.
- The POJO has static members.
- You are testing with multiple virtual users.

In-Container Testing (ICT)

In-Container Testing (ICT) using remote proxies in LISA is available in the Enterprise Java Execution and Dynamic Java Execution test nodes.

This feature allows in-container testing of local EJBs and arbitrary Java objects: any objects that are available in the container's classpath for the application being tested.

RMI and EJB are both supported as remote object protocols.

There are two connection modes used by in-container testing (ICT): EJB and RMI.

The following topics are available in this chapter.

Access using EJB (J2EE Container Environments)

To test in-container objects in a standard J2EE container, a stateful session Enterprise JavaBean (EJB) is used. This EJB is bundled as an exploded EAR directory named **lisa-remote-object-manager.ear** and a standalone jarfile **LISARemoteObjectManagerEJB.jar** with the LISA installer, in LISA_HOME/incontainer/ejb, where LISA_HOME is the directory where LISA is installed.

This EJB must be deployed with your J2EE application in the J2EE container and be accessible through JNDI using the name " **LISARemoteObjectManagerEJB**". Deploying an EJB varies depending on the J2EE container being used, and vendor-provided documentation may help if there is a problem deploying the ICT EJB.

If your J2EE application uses an isolated classloader, you must incorporate the ICT EJB into your application by modifying the XML deployment descriptors to include the ICT EJB and its dependencies.

JBoss

1. Test that you can successfully deploy the exploded ICT EAR in JBoss by copying the directory: **$LISA/incontainer/ejb/lisa-remote-object-manager.ear** to **$JBOSS/server/default/deploy** or other appropriate deployment directory.
2. JBoss will recognize the new EAR and deploy it without any errors. Check that you can connect to the EJB from LISA Workstation.
3. After the EAR is successfully deployed, in a standalone configuration, then attempt to integrate the ICT EJB with your J2EE application. This may be as simple as copying the contents of **$LISA/incontainer/ejb/lisa-remote-object-manager.ear** and including them in your existing application EAR. Or create a new EAR that includes your J2EE application combined with these files.

See the application XML deployment descriptor **application.xml** for an idea of how to modify your own application deployment descriptors to include ICT and its dependencies.

The <module> XML elements are used to indicate the presence of the ICT EJB and Java jarfile dependencies.

WebLogic

1. The first test on WebLogic is to deploy the exploded ICT EAR in WebLogic, for example, by using the WebLogic Administrator Console GUI. If your WebLogic Server is in development mode, you can also copy the directory **$LISA/incontainer/ejb/lisa-remote-object-manager.ear** to your Server's **autodeploy** directory and WebLogic will automatically deploy the EAR. You will observe that the EAR is deployed without any errors.
2. Check that you can connect to the EJB from LISA Workstation.
3. After the EAR is successfully deployed in a standalone configuration, attempt to integrate the ICT EJB with your J2EE application. This may be as simple as copying the contents of **$LISA/incontainer/ejb/lisa-remote-object-manager.ear** and including them in your own application EAR. Or create a new EAR that includes your J2EE application combined with these files.

Access using RMI (Custom Java Server and Application Environments)

For custom Java applications, you can modify your application's source code to integrate with ICT. ICT testing can be performed by binding **the LISARemoteObjectManagerRMIServer** remote Server object to the RMI registry. This object will accept connections from LISA Workstation to perform ICT.

For example, the following code can be included in your custom application to bind the ICT remote object Server via RMI:

```
LISARemoteObjectManagerRMIServer remoteObjectManagerServer = new
LISARemoteObjectManagerRMIServer();Registry registry =
LocateRegistry.createRegistry(port);registry.bind("LISARemoteObjectManager",
remoteObjectManagerServer);
```

⚠ The RMI name "**LISARemoteObjectManager**" must be entered exactly as-is for ICT to work correctly.

LISA Workstation will attempt to connect to your application via the RMI URL **rmi://<hostname>:<port>/LISARemoteObjectManager**.

The hostname and port are variables and can be changed in your test application and configured in LISA Workstation.

An example Server application can be found in $LISA/incontainer/rmi/example. Using a console window, you can run the example Server by executing the command "java -jar ExampleServer.jar".

> ⚠ For this command to run successfully, you must be running it from the **$LISA/incontainer/rmi/example** directory. See sample source code in the **$LISA/incontainer/rmi/example/src** directory.

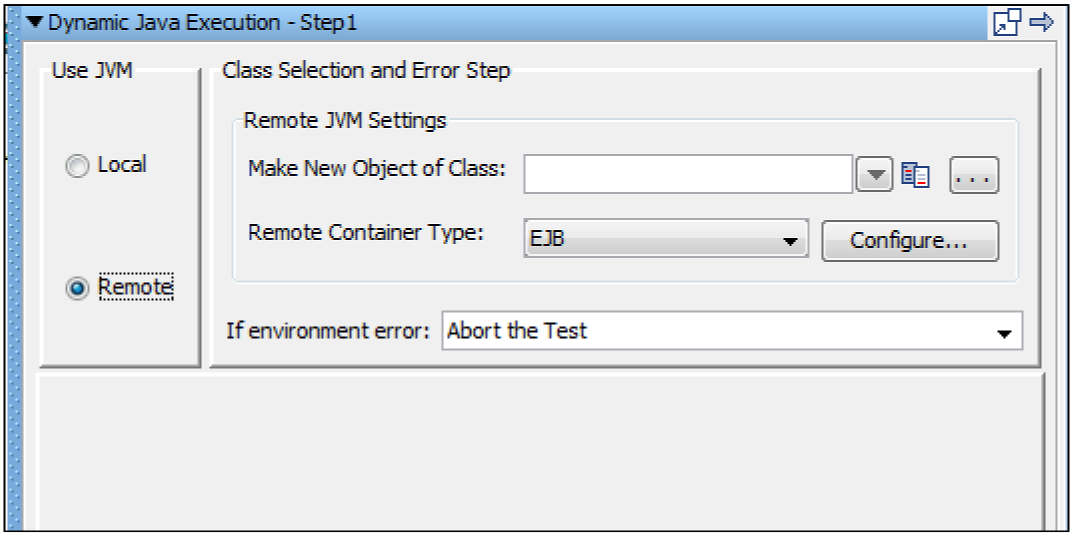
Testing your ICT Installation from LISA Workstation

After the server-side portion of ICT is up and running, you should test that you can connect from LISA Workstation.

To accomplish this, open LISA Workstation and create a new test case named **ICT Connection Test**. Inside of this test case, create a new **Dynamic Java Execution** test step.

In the editor for the new test step, select the **Remote** option.

Set the **Remote Container Type** drop-down list to your connection protocol, either EJB or RMI.



Dynamic Java Execution - Step1

Use JVM

○ Local

● Remote

Class Selection and Error Step

Remote JVM Settings

Make New Object of Class:

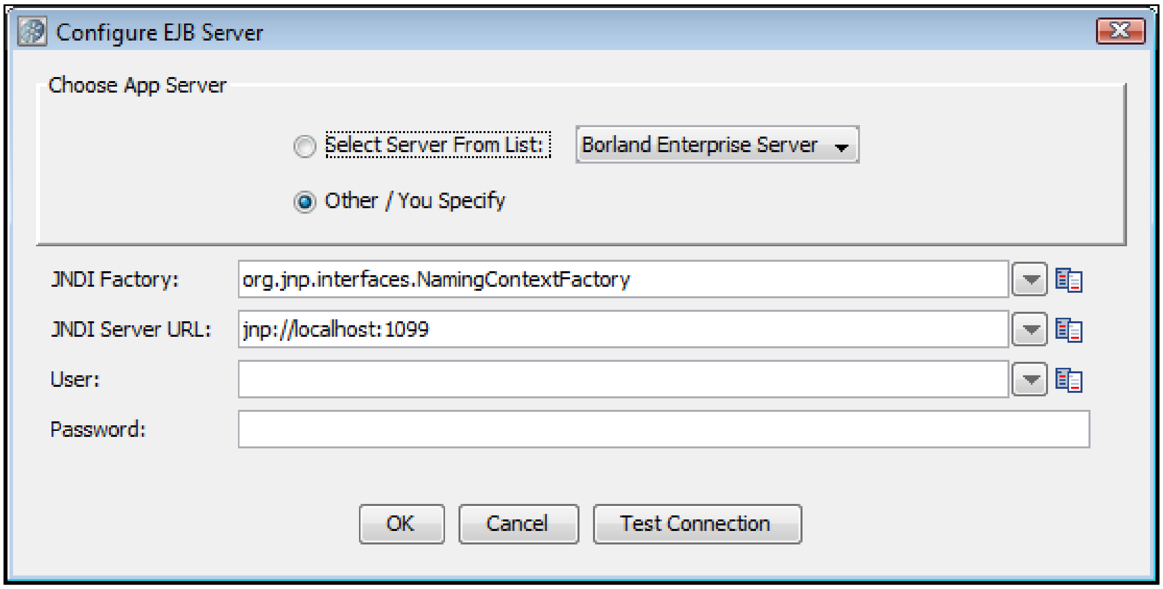
Remote Container Type: EJB

If environment error: Abort the Test

Remote Container Type is EJB

If you are using EJB as the connection protocol, click Configure to enter the configuration settings for your protocol.

For example, fill in the Configure EJB Server dialog with host name or IP address and port number of the J2EE container running ICT.



Configure EJB Server

Choose App Server

○ Select Server From List: Borland Enterprise Server

● Other / You Specify

JNDI Factory: org.jnp.interfaces.NamingContextFactory

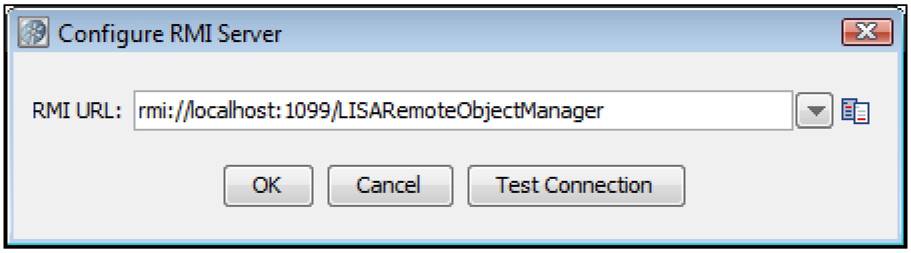
JNDI Server URL: jnp://localhost:1099

User:

Password:

Remote Container Type is RMI

If you are using RMI as the connection protocol, here is an example of the Configure RMI Server dialog that you can use for RMI settings.



RMI URL: rmi://localhost:1099/LISARemoteObjectManager

After you have successfully tested the connection to the ICT Server, enter **java.util.Date** in the **Make New Object of Class** text field and click **Construct/Load Object** to create a new in-container instance of the **java.util.Date** class.

Your installation is complete and you are ready to use ICT.

Dependencies

The ICT Server-side classes depend on the following third-party libraries:

- BeanShell 2.0b4 (bsh-2.0b4-lisa-remote-object-manager.jar).

> ⚠ The original jarfile has been modified so that .bsh scripts are removed. These scripts are known to cause problems with J2EE containers that inspect jarfiles and automatically execute them.

- XStream 1.1.2 (xstream-1.1.2.jar)
- Log4j 1.2.13 (log4j-1.2.13.jar)
- Jakarta Commons Logging 1.0.2 (commons-logging.jar)

These dependencies are intentionally distributed as separate files with LISA ICT jarfiles to make ICT integration easier. Because the application that you are testing may already include some or all of these libraries in its classpath, adding these dependencies to the classpath may not be necessary.

Generating DDLs

Setting the following properties in **local.properties** will create DDL's for Reporting and VSE.

- **eclipselink.ddl-generation=create-tables**
- **eclipselink.ddl-generation.output-mode=sql-script**
- **eclipselink.target-database=Oracle**

For Agent, Pathfinder and CVS, use the following commands for creating DDLs.

- **java -jar LisaAgent.jar -ddl oracle** (generate the Oracle DDL for Pathfinder)
- **java -jar LisaAgent.jar -ddl mysql** (generate the MySQL DDL for Pathfinder)

Appendix A - LISA Property File (lisa.properties)

A property file named **lisa.properties** is used to store initialization and configuration information.

Property files are stored in the LISA install directory.

You can display the contents of this file within LISA Workstation.

> ⚠ Do not add your custom properties to this file, as ITKO reserves the right to replace this file at any time.

To open the lisa.properties file, from the main menu, choose System > Edit LISA Properties.

Comma-separated list of paths for Javadoc and source code

These paths are used to show you class and parameter documentation. The docpath can take directories and URLs that are base paths to the

Javadoc. Here is an example that includes JDK docs from a website, but websites are not recommended because of delays. **lisa.java.docPath= LISA_HOME/examples/javadoc,[http://java.sun.com/j2se/1.3/docs/api/**

- **lisa.java.docPath**=**LISA_HOME/examples/javadoc**
- **lisa.java.sourcePath**=**LISA_HOME/examples/src**

  This sourcepath can take directories as base paths and JAR /zip files of source.

- **lisa.axis.compiler.version**=1.4

  This is lisa.axis.compiler.version 1.4.

System Properties

- **file.encoding**=UTF-8

  Encoding for files read and written by LISA.

- **lisa.supported.jres**=1.5,1.6
- **org.jfree.report.LogLevel**=Error
- **javax.xml.parsers.DocumentBuilderFactory**=org.apache.xerces.jaxp.DocumentBuilderFactoryImpl
- **javax.xml.parsers.SAXParserFactory**=org.apache.xerces.jaxp.SAXParserFactoryImpl
- **javax.xml.transform.TransformerFactory**=org.apache.xalan.processor.TransformerFactoryImpl

Server Properties

- **lisa.net.bindToAddress**=192.168.1.1

  The IP address that we will listen on. By default, we will listen on all our IP addresses. You can restrict this to a particular IP address or set it to 127.0.0.1 or localhost to prevent other computers connecting but allowing local connections.

OS X Properties

- **apple.awt.brushMetalLook**=false
- **apple.awt.brushMetalRounded**=false
- **apple.laf.useScreenMenuBar**=true
- **apple.awt.showGrowBox**=true
- **com.apple.mrj.application.growbox.intrudes**=true
- **com.apple.macos.smallTabs**=true
- **com.apple.mrj.application.apple.menu.about.name**=LISA
- **com.apple.mrj.application.live-resize**=true

LISA Update Notifications

- **lisa.update.every**=1

  Controls how often LISA checks to see whether a newer version is available to download. To disable checking, set the value to blank. The other valid values are 0 (check at every startup), 1 (check once per day), 2 (check every two days), and so on.

- **lisa.update.URL**=**http://www.itko.com/download/ga/**

Basic Defaults

- **lisa.testcase**=test.xml

  Default when running a test from the com.itko.lisa.test.TestCase class directly.

- **lisa.registry**=registry.xml

  Default when running a test from the com.itko.lisa.test.TestCase class directly.

- **lisa.runName**=Ad-hoc Run

  Default when running a test from the com.itko.lisa.test.TestCase class directly.

- **lisa.registryName**=registry

  Default name of the registry to attach to, and the default name of the registry when you start it without a name.

- **lisa.coordName**=coordinator

Coordinator Server default name when started without an explicit name AND when the TestRunner needs one and you do not specify a coordinator server on the command line.

- **lisa.simulatorName**=simulator

  Default name of a simulator daemon if you do not provide one on the command line.

- **lisa.simulatorInstances**=256
- **lisa.vseName**=VSE

  Virtual environment server default name when started without an explicit name.

- **lisa.defaultRegistry.pulseInterval**=30

  Status log interval for registry. The default value is 30 seconds.

- **lisa.coordinator.pulseInterval**=30

  Status log interval for coordinator. The default value is 30 seconds.

- **lisa.simulator.pulseInterval**=30

  Status log interval for simulator. The default value is 30 seconds.

- **lisa.vse.pulseInterval**=30

  Status log interval for VSE. The default value is 30 seconds.

- **lisa.server.projectmap.refresh.pulseInterval**=600

  Time interval after which the LISA servers (coordinator and simulator) refresh the map of project names to file paths.

- **lisa.defaultRegistryConnectionTimeoutSeconds**=90

  Timeout value in seconds that coordinators and simulators will use when connecting to a LISA Registry. A value of 0 indicates an infinite timeout; that is, we will wait forever trying to connect.

- **lisa.regex.helper.tutorial.url**=http://download.oracle.com/javase/tutorial/essential/regex/

  For the Regex helper window, this is the URL to show for the regular expression tutorial.

- **lisa.hooks**=com.itko.lisa.files.SampleHook

  To register hooks with LISA; these are comma-separated.

HTTP Header Keys Properties

These are the default values for header keys in the HTTP support. Remove or change them for all tests to get the benefit of the change, or use TestNode-specific header directives to change them for a test, or even just the execution of one HTTP transaction.

- **ice.browser.http.agent**=Mozilla/4.0

  (compatible; MSIE 6.0; Windows NT 5.0)

- **lisa.http.header.0.key**=Pragma
- **lisa.http.header.0.value**=no-cache
- **lisa.http.header.1.key**=Cache-Control
- **lisa.http.header.1.value**=no-cache
- **lisa.http.header.0.key**=Accept
- **lisa.http.header.0.value**=image/gif, image/x-xbitmap, image/jpeg
- **lisa.http.header.1.key**=Accept-Language
- **lisa.http.header.1.value**=en
- **lisa.http.header.2.key**=Accept-Charset
- **lisa.http.header.2.value**=iso-8859-1,*,utf-8
- **lisa.http.header.3.key**=User-Agent
- **lisa.http.header.3.value**=Mozilla/4.0, MSIE 6.0; Windows NT 5.0

HTTP Field Editor Properties

These are the defaults for fields that show up in the HTTP Field editor. Don't include "Authentication" because it is added by the editor automatically.

- **lisa.gui.http.fieldNames**=Accept,Accept-Language,User-Agent,Connection
- **lisa.webrecorder.textMIMEs**=html,text,magnus-internal,application/pdf

- **lisa.webrecorder.notTextMIMEs**=css,script
- **lisa.webrecorder.alwaysIgnore**=.gif,.jpg,.jpeg,.css,.js,.ico
- **lisa.web.ntlm**=true

  Enables NTLM auth in the Test Runner.

Test Case Execution Parameters

- **lisa.hotDeploy**=/C/Projects/Lisa/custom_classes

  This setting tells the ClassLoader built into LISA where to look for custom classes. The default is $LISA_HOME/hotdeploy.

- **lisa.overloadThreshold**=1000

  The TestNode attempts to determine if the smulator is thrashing by checking the actual amount of time slept in think time as opposed to the amount of think time it was supposed to take. This setting is the amount of additional "slip" in think time that is acceptable before sending a warning TestEvent that the simulator is overloaded. The default of 1000 means that if the simulator sleeps 1 second more than it was supposed to, (presumably because the computer is CPU starved), then raise the TestEvent.

- **lisa.webservices.encode.empty.xmlns**=true

  Some web service server stacks require empty xmlns strings in the SOAP request (for example, jbossWs). Others, such as Amazon's, will not work with empty xmlns strings. Change this property to suit the stack you are calling.

- **lisa.webservices.encode.version**=1.1

  Set the encoding version to force for client stub generation. The default is 1.1.

TestEvent Handling Customizations

- **lisa.perfmon.snmp.port**=1161

  The StatKeeper can load a Perfmon integration class that will wrap or implement a platform-specific monitor, like Windows Perfmon, JMX, SNMP, and others. LISA comes with a Perfmon DLL class and an SNMP class to support producing our stats output to either the Windows Performance monitor OR (not both) as an SNMP agent. For more information, see the docs on SNMP. The usual port for SNMP is 161, but you have to be root for that.

- **lisa.perfmon.class**=com.itko.lisa.stats.snmp.SnmpPerfmon

  When we do have something that can pump native OS data into a performance monitor, implement a class that can push LISA data into that tool and put that class name here.

- **lisa.perfmon.dll**=/c:/Projects/Lisa/PerfmonJNI/LISAPerfmonJNI/Debug/LISAPerfmonJNI.dll

  LISA's Windows Perfmon integration to the StatKeeper has a "DLL" setting for the Windows (native) implementation. Put the full path/file here. You only need this if you are using **PerfmonStatKeeperWindows**.

Simulators use a separate thread and queue to send TestEvents to the coordinator to cut down on the chattiness of RMI. These are the thresholds that cause the deamon thread to push events. We take the minimum of the size or the max-wait (if either we take too long or have too many, we post them).

- **lisa.eventPoolPoll**=250

  How often we check the queue size or see if we have overrun our max wait (in milliseconds)

- **lisa.eventPoolSize**=64

  The maximum size we let it get before sending

- **lisa.eventPoolMaxWait**=1000

  How long we are willing to go with an event being unforwarded

LISA Test Manager/Editor Properties

- **gui.show.memory.status**=false
- **lisa.screencap.delay.seconds**=6
- **lisa.screencap.dir**=`LISA_HOME`/screens
- **gui.show.memory.status**=false
- **lisa.screencap.delay.seconds**=6
- **lisa.screencap.dir**=`LISA_HOME`/screens
- **lisa.screen.cap.prefix**=lisa-screencap-

- **lisa.earsubdir.endingnamepart**=-contents
- **lisa.model.editor.inspector.scale**=0.83

  This property sets the "scale" (primarily, font size) for things in the model and step inspectors of the main model editor. 1.0 is 12pt, so determine a value for this property by dividing the size you want in points by 12. For example, 11pt is 11/12 = 0.92, 10pt is 10/12 = 0.83 (the default), 14pt is 14/12 = 1.17.

- **lisa.stats.decimalFormat**=###,###.#

  Some built-in metrics (steps per second) use a Java DecimalFormat to display floating-point values. If you do not want the decimal point, make this ###### or choose from a range of displays; see http://download.oracle.com/javase/6/docs/api/java/text/DecimalFormat.html.

- **lisa.editor.custJavaNodeEditor.classes**=com.itko.lisa.files.SampleCustJavaNode

  This is the comma-separated list of all custom Java test nodes that you wish to possibly include in your test case.

- **lisa.editor.combined.report.type**=
- **lisa.tm.hh.on**=no

  This drives the formatting of messages to the System Messages window.

- **lisa.gui.log4jfmt**=%-5p - %m%n

  This drives the formatting of messages to the System Messages window.

- **lisa.editor.http.recorderPort**=8010

  The HTTP recorder binds to this port.

- **lisa.editor.proxy.webProxySupport**=on

J2EE server parameters

• **lisa.prefill.jndiNames**=	JBOSS=org.jnp.interfaces.NamingContextFactory
	Weblogic=weblogic.jndi.WLInitialContextFactory
	Websphere=com.ibm.websphere.naming.WsnInitialContextFactory
	Borland Enterprise Server=com.inprise.j2ee.jndi.CtxFactory
	iPlanet/Sun AS=com.sun.jndi.cosnaming.CNCtxFactory

• **lisa.prefill.jndiUrlPrefix**=	JBOSS=jnp://
	Weblogic=t3://
	Websphere=iiop://
	Borland Enterprise Server=iiop://
	iPlanet/Sun AS=iiop://

• **lisa.prefill.jndiDefPort**=	JBOSS=1099
	Weblogic=7001
	Websphere=2809
	Borland Enterprise Server=1099
	iPlanet/Sun AS=1099

• **lisa.prefill.jndiNeedsClass**=	JBOSS=false
	Weblogic=false
	Websphere=true
	Borland Enterprise Server=true
	iPlanet/Sun AS=true

• **lisa.prefill.jndiFactories**=	org.jnp.interfaces.NamingContextFactory weblogic.jndi.WLInitialContextFactory com.ibm.websphere.naming.WsnInitialContextFactory com.webmethods.jms.naming.WmJmsNamingCtxFactory com.tibco.tibjms.naming.TibjmsInitialContextFactory com.inprise.j2ee.jndi.CtxFactory com.sun.jndi.cosnaming.CNCtxFactory fiorano.jms.runtime.naming.FioranoInitialContextFactory

- **lisa.prefill.jndiServerURLs**=	jnp://SERVER:1099&t3://SERVER:7001 iiop://SERVER:PORT tibjmsnaming://SERVER:7222&iiop://SERVER:PORT iiop://localhost:9010&wmjmsnaming://Broker #1@SERVER:PORT/JmsAdminTest
- **lisa.editor.URLTransEditor.protos**=	http,https
- **lisa.editor.URLTransEditor.hosts**=	
- **lisa.editor.URLTransEditor.ports**=	80,443
- **lisa.editor.URLTransEditor.files**=	
- **lisa.prefill.jdbc.names**=	Oracle SQL Server WLS Oracle JDataStore Sybase DB2 MySQL Derby
- **lisa.prefill.jdbc.jdbcDrivers**=	oracle.jdbc.driver.OracleDriver com.microsoft.sqlserver.jdbc.SQLServerDriver com.microsoft.jdbc.sqlserver.SQLServerDriver weblogic.jdbc.oci.Driver com.borland.datastore.jdbc.DataStoreDriver com.sybase.jdbc.SybDriver com.ibm.db2.jcc.DB2Driver org.gjt.mm.mysql.Driver org.apache.derby.jdbc.ClientDriver
- **lisa.prefill.jdbc.jdbcConnectionURLs**=	jdbc:oracle:thin:@SERVER:1521:SIDNAME jdbc:sqlserver://SERVER:PORT;databasename=DBNAME jdbc:microsoft:sqlserver://SERVER:PORT jdbc:weblogic:oracle:TNSNAME jdbc:borland:dslocal:DBNAME jdbc:sybase:Tds:SERVER:PORT/DBNAME jdbc:db2://SERVER:PORT/DBNAME jdbc:mysql://SERVER:PORT/DBNAME jdbc:derby://DBSERVER:DBPORT/DBNAME

Native Browser Information to Use for Internal Rendering

- **lisa.internal.browser.on**=yes
- **lisa.internal.browser.win**=com.itko.lisa.web.ie.IEUtils
- **lisa.internal.browser.osx**=com.itko.lisa.web.jxbrowser.JxBrowserUtils
- **lisa.internal.browser.linux**=com.itko.lisa.web.jxbrowser.JxBrowserUtils
- **lisa.internal.browser.sol-sparc**=null
- **lisa.internal.browser.imgs**=false
- **lisa.internal.browser**=msie

WR type: mozilla, safari, msie

- **lisa.internal.browser.swing.heavy**=false
- **lisa.internal.browser.usejsinviews**=yes
- **lisa.example.wsdls**=http://localhost:8080/itko-examples/services/UserControlService?wsdl

LISA Test Manager/Monitor Properties

- **monitor.events.maxrows**=500

The maximum number of event rows kept in the Test Manager as a test is run (objects kept is actually twice this number).

- **lisa.tm.sysmess.size**=10240

  The system messages window maximum size (memory consumed is twice this value).

LISA Built-in String-generator Patterns

- **lisa.patterns.stringgenerator.types**=&Phone=(DDD)DDD-DDDD, &SSN=DDD-DD-DDDD, &Date=Lll-DD-DDDD, &Zip=D*(5)

JMX Information

• **lisa.jmx.types**=	com.itko.lisa.stats.jmx.JSE5Connection com.itko.lisa.stats.jmx.TomcatConnection com.itko.lisa.stats.jmx.JBossConnection com.itko.lisa.stats.jmx.JSR160RMIConnection com.itko.lisa.stats.jmx.WeblogicConnector com.itko.lisa.stats.jmx.Weblogic9Connector com.itko.lisa.stats.jmx.WebsphereSOAPConnection com.itko.lisa.stats.jmx.ITKOAgentConnection com.itko.lisa.stats.jmx.OracleASConnector

• **lisa.jmx.typeprops**=	com.itko.lisa.stats.jmx.JSE5Connection=LISA_JMX_JSE5
	com.itko.lisa.stats.jmx.TomcatConnection=LISA_JMX_TOMCAT5
	com.itko.lisa.stats.jmx.JBossConnection=LISA_JMX_JBOSS3240
	com.itko.lisa.stats.jmx.JSR160RMIConnection=LISA_JMX_JSR160RMI
	com.itko.lisa.stats.jmx.Weblogic9Connector=LISA_JMX_WLS9
	com.itko.lisa.stats.jmx.WeblogicConnector=LISA_JMX_WLS6781
	com.itko.lisa.stats.jmx.OracleASConnector=LISA_JMX_OC4J
	com.itko.lisa.stats.jmx.WebsphereSOAPConnection=LISA_JMX_WASSOAP5X
	com.itko.lisa.stats.jmx.ITKOAgentConnection=LISA_JMX_ITKOAGENT

You do not generally need anything for these.

- **LISA_JMX_JSR160RMI**=`LISA_HOME`/lib/mx4j.lib
- **LISA_JMX_ITKOAGENT**=`LISA_HOME`/lib/mx4j.lib
- **LISA_JMX_JSE5**=

  If you are running JSE 5, you do not need to change this property. We ship a jboss-client-all that has what you need, assuming the version is right.

- **LISA_JMX_JBOSS3240**=`LISA_HOME`/lib/mx4j.lib{{path.separator}}`LISA_HOME`/hotDeploy/jbossall-client.jar

- **LISA_JMX_TOMCAT5**=`LISA_HOME`/lib/mx4j.lib{{path.separator}}`LISA_HOME`/hotDeploy/mx4j-tools.jar

  This is for Tomcat.

- **LISA_JMX_WLS9**=`LISA_HOME`/hotDeploy/weblogic.jar{{path.separator}}`LISA_HOME`/hotDeploy/wljmxclient.jar
- **LISA_JMX_WLS6781**=`LISA_HOME`/hotDeploy/weblogic.jar

  This needs to change to the location of your weblogic.jar -- no wlclient.jar will not work.

Oracle AS

- **LISA_JMX_OC4J**=`LISA_HOME`/lib/oc4jclient.jar{{path.separator}}`LISA_HOME`/lib/adminclient.jar

IBM Websphere

- **LISA_JMX_WASSOAP5X**=`LISA_HOME`/lib/mx4j.lib

  It is usually easier to use your IBM/WAS CLASSPATH before you start LISA and leave this blank.

- **lisa.alert.email.emailAddr**=lisa@itko.com

  If you use performance monitoring alerts, this is the "from" email address for those alerts.

- **lisa.alert.email.defHosts**=localhost

  This is the email server we will attempt to route emails with (smtp server).

- **lisa.rundoc.builtins** =	com.itko.lisa.files.1user1cycle.stg=Runs the test case once with one simulated user &com.itko.lisa.files.1user1cycle0think.stg=Runs the test case once with one simulated user and no think time &com.itko.lisa.files.1user1min.stg=Runs the test case with one simulated user for one minute, restarting the test as needed &com.itko.lisa.files.1user5min.stg=Stage the test for 5 minutes, restarting as needed &com.itko.lisa.files.1usernonstop.stg=Execute this test until manually stopped &com.itko.lisa.files.5user1min.stg=Execute this test with 5 virtual users for 1 minute

- **lisa.auditdoc.builtins**=com.itko.lisa.files.DefaultAudit.aud
- **lisa.projects.home**=`LISA_HOME`

  This property is used to define all projects home within a server environment.

LISA Test Manager/ITR properties

- **lisa.tm.itr.max.delay.seconds**=5

LISA External Command Shells

- **test.cmde.win.shell**=cmd /c
- **test.cmde.unix.shell**=sh -c
- **test.cmde.Windows.XP.shell**=cmd /c
- **test.cmde.Windows.Vista.shell**=cmd /c
- **test.cmde.Windows.NT.(unknown).shell**=cmd /c

LISA Testing Parameters

- **lisa.props.blankOnMissing**=true

  Property to use if you want LISA to replace `key` with an empty string if key is not in state.

- **lisa.test.custevents**=&101="Custom Event 101"; &102="Custom Event 102"

  Custom events: the first allowed event number is 101, so we have registered two as examples here.

- **lisa.SimpleWebFilter.responseCodeRegEx**=[45] d d
- **lisa.fsss.dateformat**=MM/dd/yyyy hh:mm:ss a

Properties for use by StdSchedulerFactory to create a Quartz Scheduler Instance

Configure Main Scheduler Properties

    - **org.quartz.jobStore.class**=org.quartz.simpl.RAMJobStore
    - **org.quartz.threadPool.class**=org.quartz.simpl.SimpleThreadPool

    - **org.quartz.threadPool.threadCount**=5

- **lisa.meta-refresh.max.delay**=5

Platform-specific Parameters in the TM

- **lisa.tm.exec.unix**=xterm -e {0}
- **lisa.tm.exec.win**=cmd /c start {0}
- **lisa.tm.exec.osx**=open -a /Applications/Utilities/Terminal.app {0}

Properties used by Swing Testing Support

- **lisa.swingtest.client.logging.properties.file**=C:/Lisa/swingtestclient-logging.properties

  Uncomment to use a custom Log4J logging properties file in SwingTestProgramStarter.

License Settings

- **laf.request**=laf/license.do
- **laf.default.url**=https://license.itko.com
- **laf.displaysettings**=true

Properties used by Web 2.0

- **lisa.browser.source.port**=0
- **lisa.browser.target.port**=0

The ports TM and the LISA browser use to communicate: 0 means choose dynamically.

- **lisa.browser.launch.timeout**=10000
- **lisa.browser.max.instances**=25

Reporting JPA Properties

- **rpt.eclipselink.ddl-generation**=create-tables
- **rpt.eclipselink.ddl-generation.output-mode**=database
- **rpt.eclipselink.validateschema**=false
- **perfmgr.rvwiz.whatrpt.autoExpire**=true
- **perfmgr.rvwiz.whatrpt.expireTimer**=30d

To check for expired reports, set autoExpire = true. Set the expiration period: an integer followed by (m=month,w=week,d=day,h=hour). The default expiration period is 30d (30 days).

- **rpt.hibernate.validateschema**=false

Validate the report database schema. By default, only the Registry will validate the schema.

- **lisa.0.registry.local.autoshutdown**=true
- **lisa.8.registry.local.autoshutdown**=true
- **lisa.10.registry.local.autoshutdown**=true

The default behavior is to not automatically shut down the local registry if it was started automatically. The exceptions are LISA Workstation, VSE, and VSE Workstation:

- **lisa.4.registry.local.autoshutdown**=false
- **lisa.5.registry.local.autoshutdown**=false

JUnit and TestRunner should never autoshutdown the registry.

Property used for the Eclipse Connector

- **lisa.eclipse.connector.port**=8546

Property used for example test suites

- **EXAMPLES_HOME**=`LISA_HOME`/examples

VSE Properties

- **lisa.magic.string.min.length**=3

Property used to define the minimum length of argument value in a VSE transaction request that is required to consider that argument for constructing a magic string out of it:

• **lisa.magic.string.word.boundary.type** =	none: Word boundaries don't matter (the default) start: Magic string candidates must start on a word boundary end: Magic string candidates must end on a word boundary both: Magic string candidates must be found as a whole word (that is, a word boundary on both ends)

Property used to define how searches in VSE for magic string content relate to word boundaries.

- **lisa.magic.string.exclusion**=Yes, YES, yes, No, NO, no, true, True, TRUE, false, False, FALSE, __NULL

This property lets you exclude certain strings from being eligible for magic stringing.

- **lisa.magic.string.xml.tags**=false

Should text within XML tags be magic stringed?

- **lisa.vse.server.dir.full.service.name**=false

Uncomment this and set it to true if you have a situation where multiple VSE instances are being run on different computers out of the same install directory.

- **lisa.vse.response.xml.prettyprint**=false

  If we record an XML response in VSE should we format the XML in the service image? The default, false, indicates we will not prettyprint, format, or add line breaks to XML. The XML will be shown as one long string if that is the way it was received.

- **lisa.vse.remember.execution.mode**=true

  Comment this out if you need the VSE to forget across shutdowns the execution modes you set with the VSE dashboard for virtual services.

- **lisa.vse.match.event.buffer.size**=100

  This property controls how many VSE matching related events are buffered. The number of events is per VS model so if you have two VS models deployed with the default event buffer size of 100, then a total of 200 events will be buffered. These events are the source for the "Match" tab of the VS model inspection page in the VSE dashboard.

- **lisa.vse.request.event.set.buffer.size=5**

  This property controls how many inbound VSE requests for which a full set of LISA events is buffered. The number of events is per VS model so if ou have 2 VS models deployed with the default request buffer size of 5, then a total of 10 sets of events (grouped by inbound request) will be buffered. These sets of events are the source for the "Events" tab of theVS model inspection page in the VSE dashboard.

- **lisa.vse.si.text.editor.order**=XMLTextEditor,JSONTextEditor

  Change this to affect the order in which registered VSE text response editors are queried when using auto-detect. Only enough of the "right end" of the class name to make it unique is required. The default text editor is just that; a default, and so does not need to be listed here.

- **lisa.tm.def.min.millis**=500
- **lisa.tm.def.max.millis**=1000

The default think time for new "regular" steps, in milliseconds:

- **lisa.tm.sys.min.millis**=0
- **lisa.tm.sys.max.millis**=0

The default think time for new "system" steps, in milliseconds. System steps include subprocesses, 'continue', 'continue (quiet), 'fail' and 'end.'

- **lisa.numFilters.warning**=100
- **lisa.numAsserts.warning**=100

There are cases where filters and assertions are dynamically added to test steps and in a load test this could mean many thousands of asserts/filters. The following two numbers are the threshold before there is a WARN level message generated in the log.

- **lisa.exception.on.num.exceeded**=true

Should we raise a TestDefException (kills the test) if the threshold is exceeded?

LISA Date-checker Properties

These are the set of properties that are used by the LISA VSE date utilities, to determine which date patterns are going to be considered valid for date sensitivity conversions. Each of the following entries represents a regular expression that will be considered valid as a part of date, by VSE.

<table>
<tr><td>

- **lisa.vse.datechecker.dayregex**=

</td><td>

```
((\[12\]\\d)\|(3\[01\])\|(0?\[1-9\]))
```

</td></tr>
<tr><td>

- **lisa.vse.datechecker.monthnumberregex**=

</td><td>

```
((1\[012\])\|(0 \\d)\|0\[1-9\]\|\[1-9\])
```

</td></tr>
</table>

- **lisa.vse.datechecker.monthalpharegex**=

```
(\\bJAN\\b\|\\bFEB\\b\|\\bMAR\\b\|\\bAPR\\b\|\\bMAY\\b\|\\bJU
```

- **lisa.vse.datechecker.yearlongregex**=

```
\\d\\d\\d\\d
```

- **lisa.vse.datechecker.yearshortregex**=

```
\\d\\d
```

- **lisa.vse.datechecker.timeregex**=

```
(\\s?((\[012\]? \\d)\|(2\[0123\]))\:((\[012345\] \\d)\|(60))\
```

- **lisa.vse.datechecker.time.hhmmssregex**=

```
((\[012\]? \\d)\|(2\[0123\]))\:((\[012345\] \\d)\|(60))\:((\[
d)\|(60))
```

- **lisa.vse.datechecker.time.millisregex**=

```
((\[012\]?\\d)\|(2\[0123\]))\:((\[012345\]\\d)\|(60))\:((\[01
```

- **lisa.vse.datechecker.time.millis.zoneregex**
=

```
((\[012\]?\\d)\|(2\[0123\]))\:((\[012345\] \\d)\|(60))\:((\[0
d)\|(60)) \\.((\\d \\d \\d)\|0) \\s(\[A-Za-z\]\[A-Za-z\]\[A-Z
```

- **lisa.vse.datechecker.wstimestampregex**=

```
\\d\\d\\d\\d-((1\[012\])\|(0\\d)\|0\[1-9\]\|\[1-9\])-((\[12\]
\\d
```

- **lisa.vse.datechecker.ddmmmyyyyregex**=

```
((\[12\]\\d)\|(3\[01\])\|(0?\[1-9\]))(JAN\|FEB\|MAR\|APR\|MAY
```

- **lisa.vse.datechecker.mmmddyyyyregex**=

```
(JAN\|FEB\|MAR\|APR\|MAY\|JUN\|JUL\|AUG\|SEP\|OCT\|NOV\|DEC)(
```

- **lisa.vse.datechecker.yyyyddmmmregex**=

```
\\d\\d \\d \\d((\[12\] \\d)\|(3\[01\])\|(0?\[1-9\]))(JAN\|FEB
```

- **lisa.vse.datechecker.yyyymmmddregex**=

```
\\d\\d\\d\\d(JAN\|FEB\|MAR\|APR\|MAY\|JUN\|JUL\|AUG\|SEP\|OCT
\\d)\|(3\[01\])\|(0?\[1-9\]))
```

- **lisa.vse.datechecker.ddmmmregex**=

```
((\[12\]\\d)\|(3\[01\])\|(0?\[1-9\]))(JAN\|FEB\|MAR\|APR\|MAY
```

- **lisa.vse.datechecker.mmmddregex**=

```
(JAN\|FEB\|MAR\|APR\|MAY\|JUN\|JUL\|AUG\|SEP\|OCT\|NOV\|DEC)(
```

- **lisa.vse.datechecker.ddmmmyyregex** =

```
((\[12\]\\d)\|(3\[01\])\|(0?\[1-9\]))(JAN\|FEB\|MAR\|APR\|MAY
```

Each of the following entries represents a valid pattern of a part of date, in VSE.

- **lisa.vse.datechecker.dayformat**=dd
- **lisa.vse.datechecker.monthnumberformat**=MM
- **lisa.vse.datechecker.monthalphaformat**=MMM
- **lisa.vse.datechecker.yearlongformat**=yyyy
- **lisa.vse.datechecker.yearshortformat**=yy
- **lisa.vse.datechecker.timeformat**=HH:mm:ss
- **lisa.vse.datechecker.time.hhmmssformat**=HH:mm:ss
- **lisa.vse.datechecker.time.millisformat=HH:mm:ss.SSS**
- **lisa.vse.datechecker.time.millis.zoneformat**=HH:mm:ss.SSS z
- **lisa.vse.datechecker.wstimestampformat**=yyyy-MM-dd'T'HH:mm:ss.SSSZ

Following are some of the date patterns that cannot be constructed by using a combination for the patterns that involve at least day, month and year part.

- **lisa.vse.datechecker.mmmddyyyy.separatorformat**=MMM*dd*yyyy
- **lisa.vse.datechecker.mmddyyyy.separatorformat**=MM*dd*yyyy
- **lisa.vse.datechecker.ddmmmyyyy.separatorformat**=dd*MMM*yyyy
- **lisa.vse.datechecker.ddmmyyyy.separatorformat**=dd*MM*yyyy
- **lisa.vse.datechecker.yyyymmmdd.separatorformat**=yyyy*MMM*dd
- **lisa.vse.datechecker.yyyymmdd.separatorformat**=yyyy*MM*dd
- **lisa.vse.datechecker.ddmmmyyyyformat**=ddMMMyyyy
- **lisa.vse.datechecker.mmmddyyyyformat**=MMMddyyyy
- **lisa.vse.datechecker.yyyyddmmmformat**=yyyyddMMM
- **lisa.vse.datechecker.yyyymmmddformat**=yyyyMMMdd
- **lisa.vse.datechecker.ddmmmyyformat**=ddMMMyy
- **lisa.vse.datechecker.ddmmmformat**=ddMMM
- **lisa.vse.datechecker.mmmddformat**=MMMdd

- **lisa.vse.datechecker.seperators**=- /.

  This represents the valid separator characters that can be used within the date formats: for example, 10/15/2001 uses '/' as a separator.

- **lisa.vse.datechecker.top.priorityorder**=lisa.vse.datechecker.wstimestampformat

  Priority order decides what order we should use to match the date pattern. The asterisks will be replaced by the separator characters defined by the "separators" property. For example, MM*dd*yy will generate four date patterns: "MM-dd-yyyy", "MM dd yyyy", "MM/dd/yyyy", "MM.dd.yyyy".

• **lisa.vse.datechecker.date.priorityorder**=	lisa.vse.datechecker.mmmddyyyy.separatorformat&\\ lisa.vse.datechecker.mmddyyyy.separatorformat&\\ lisa.vse.datechecker.ddmmmyyyy.separatorformat&\\ lisa.vse.datechecker.ddmmyyyy.separatorformat&\\ lisa.vse.datechecker.yyyymmmdd.separatorformat&\\ lisa.vse.datechecker.yyyymmdd.separatorformat&\\ lisa.vse.datechecker.mmmddyyyyformat&\\ lisa.vse.datechecker.ddmmmyyyyformat&\\ lisa.vse.datechecker.yyyyddmmmformat&\\ lisa.vse.datechecker.yyyymmmddformat&\\ lisa.vse.datechecker.ddmmmyyformat

- **lisa.vse.datechecker.time.priorityorder** =	lisa.vse.datechecker.time.millis.zoneformat& lisa.vse.datechecker.time.millisformat& lisa.vse.datechecker.time.tenthsformat& lisa.vse.datechecker.time.hhmmssformat
- **lisa.vse.datechecker.bottom.priorityorder**=	lisa.vse.datechecker.mmmddformat& lisa.vse.datechecker.ddmmmformat

- **lisa.vse.datechecker.months**=JAN,FEB,MAR,APR,MAY,JUN,JUL,AUG,SEP,OCT,NOV,DEC

VSE JMS Messaging - Custom JMS Properties to Ignore

- **vse.jms.ignore.proplist**=JMSX.*,JMS_.*

Some JMS platforms include extra custom properties in the JMS messages they deliver. This can interfere with VSE creation. **vse.jms.ignore.proplist** contains a list of properties that should be ignored when recording a JMS service with the VSE messaging recorder. The format is a comma-delimited list of regular expressions. By default it is excluding standard JMS extension properties (JMSX.) and JBoss custom bookkeeping properties (JMS_.).

VSE Recorder Conversation Batch Size

- **lisa.vse.recorder.conversation.batch.size**=10

This defines how many conversations we will process before committing them to the database. If you have a few very large conversations, make this number small. If you have many smaller conversations, make this number large. A batch size of <= 0 is the same as a batch size of 1.

VSE Service Image Database Settings

- **eclipselink.ddl-generation**=create-tables
- **eclipselink.ddl-generation.output-mode**=database

LISA VSE uses the open source EclipseLink JPA provider. By default we will use our Reports database as the repository for service images.

Validate the VSE Schema

- **eclipselink.validateschema**=false

By default only the Registry will validate the schema.

- **isa.eclipselink.query.warn.threshold**=100

If EclipseLink has been configured to do query timings (and it will be by default) then this is the maximum time in milliseconds that a query to the database can take before a WARN level message is raised by the **com.itko.lisa.vse.stateful.model.SqlTimer** logger. If you set that logger's threshold to DEBUG you will see timings for all SELECT statements issued by EclipseLink. Basically this is the first port of call in debugging any VSE performance issues - if the database is local and all the indexes have been created then you should not see anything over about 20 ms. But a database on the network that is not under your control and possibly missing indexes could easily take 100ms or more to return a result and that might make VSE seem slow when in fact it is lightning fast if correctly deployed. In reality, most service images are small enough or have a working set sufficiently small enough to fit into the VSE entity cache, so after some time the number of SELECTs issued by VSE should dwindle to zero, provided the VSE has a large enough heap (the cache maintains soft references by default).

- **lisa.eclipselink.cache.timeout.ms**=10000

By default we cache EclipseLink objects for no longer than 10 seconds. This means that changes made to the service image database using the service image editor will propagate to a running virtual service model in no longer than 10 seconds and on average 5 seconds, assuming the object is already cached by the VSE.

If the object is not already cached the propagation will be immediate.

Ten seconds is a reasonable trade-off in a development environment where you are making changes to service images. In a production environment with a heavily loaded VSE server you might consider revising this number upwards to at least a minute (60000 ms) or even an hour (360000 ms) to reduce the number of SQLs issued and thus increase performance.

Network Port Properties

The network port properties take the form of **lisa.net.APPID.port**.

If you need to change these defaults, override them in the **site.properties** file. Clients make assumptions about what port to connect to their server based on these values.

- **lisa.net.0.port**=2008

  LISA Workstation

- **lisa.net.2.port**=2011

  Coordinator

- **lisa.net.3.port**=2010

  Registry

- **lisa.net.4.port**=2012

  JUnit exec

- **lisa.net.5.port**=2005

  Test Runner (cmdline)

- **lisa.net.6.port**=2007

  Others

- **lisa.net.8.port**=2013

  VSE

- **lisa.net.9.port**=2004

  VSEManager

- **lisa.net.11.port**=2006

  ServiceManager

- **lisa.net.1.port**=2014

  Simulator (we start here so it is easier to have many on the same computer)

Pathfinder Properties

- **lisa.pathfinder.on**=true

Database Properties for Pathfinder

For now we will use a separate pooling mechanism. Passwords in LISA property files are encrypted using AES.

- **lisadb.driver=**org.apache.derby.jdbc.ClientDriver
- **lisadb.url=jdbc:derby://`lisa.server.hostname`**:1528/database/lisa.db;create=true

- **lisadb.username**=rpt
- **lisadb.password_enc=**76f271db3661fd50082e68d4b953fbee
- **lisa.pathfinder.broker.host**=0.0.0.0
- **lisa.pathfinder.broker.port**=2009
- **pathfinderdb.url=jdbc:derby://`lisa.server.hostname`**:1528/database/pathfinder.db;create=true

- **lisa.pathfinder.broker.db=**`lisadb.driver`::`pathfinderdb.url`::`lisadb.username`::`lisadb.password`

- **lisa.webserver.port**=1505

  Our embedded web server.

- **lisa.webserver.host**=0.0.0.0

  Our embedded web server's host. **0.0.0.0** will bind to all local addresses.

These set the values for all of the portal app launchers from LISA Workstation. Currently the TR host name is used for the Host value so we get that dynamically. LISA embeds a native browser so if you have stability issues because of this, you can make all the portal apps be launched externally in your systems browser by setting **lisa.portal.launch.all.external=yes**.

- **lisa.portal.launch.all.external**=yes

  Force the portal to use an external browser.

- **lisa.portal.url.prefix**=http://* (http://*)
- **lisa.portal.root.base.url**=/index.html
- **lisa.portal.cvsdashboard.base.url**=/index.html?lisaPortal=cvsdashboard
- **lisa.portal.pathfinder.console.base.url**=/index.html?lisaPortal=pathfinder
- **lisa.portal.server.console.base.url**=/index.html?lisaPortal=serverconsole
- **lisa.portal.reporting.console.base.url**=/index.html?lisaPortal=reporting
- **lisa.portal.defect.capture.url**=/pathfinder/lisa_pathfinder_agent/defectcapture
- **lisa.portal.save.defect.data.url**=/pathfinder/lisa_pathfinder_agent/saveDefectData
- **lisa.portal.invoke.base.url**=/lisa-invoke
- **lisa.portal.invoke.report.url**=/reports
- **lisa.portal.invoke.server.report.directory**=`lisa.tmpdirlisa.portal.invoke.report.url`

- **lisa.portal.invoke.test.root=`LISA_HOME`**

Reporting Graph View Properties

- **rpt.lisa.graph.view.threshhold**=100
- **rpt.lisa.graph.view.infomessage**=Results more than maximum threshhold. Modify filter to fetch the details.
- **rpt.lisa.graph.scatter.view.threshhold**=10000
- **rpt.lisa.graph.scatter.view.infomessage**=Results more than maximum threshhold. Modify filter to fetch the details.

Async Reporting Support

- **lisa.reporting.useAsync**=false

  If you run load tests and find that the bottleneck in the test cases is writing the events to the reporting database, consider removing everything except metrics from the report generator. If you still see the reporting engine as the bottleneck, consider enabling this property; it will use JMS to send the reporting event and background threads in the simulators and coordinator will write the events to the database asynchronously. This means that your load test will finish before all the events have been written to the database, so the report will not appear for some time (just how long will depend on your test cases and how many events they generate). The simulator queue will typically take the longest to flush; you will get a message at INFO level in the simulator log showing the percentage complete.

  This feature is considered as "advanced usage" for now and is disabled by default. Feedback to support@itko.com is welcome and encouraged.

- **lisa.reporting.step.max.propsused.buffersize**=100
- **lisa.reporting.step.max.propsset.buffersize**=100

  These properties control the collection of statistics for a test step during the execution of a test case. The values specify the maximum number of occurrences that will be recorded for the properties used and properties set. You should not need to change the default values.

- **lisa.threadDump.generate**=true
- **lisa.threadDump.interval**=30
- **lisa.threadDump.loggerName**=threadDumpLogger

  Enable periodic thread dumps. It is a good idea to leave this enabled, it will very efficiently check to see if the threadDumpLogger is at INFO or below and not do anything if it is set to WARN or higher. LISA properties are only read once at startup; the **logging.properties** file is checked every 10 seconds for changes, so all you need to do is leave this alone and set the log level of the threadDumpLogger to INFO and wait for at most 30 seconds to get periodic thread dumps of a running LISA server. These logs are invaluable when debugging performance issues. See the comments in **logging.properties** for more information.

These properties are used to control metrics collection in VSE servers. The metrics collected can be viewed in the web-based VSE dashboard. The values here are the defaults.

- **lisa.vse.metrics.collect**=true

  Main property to turn overall metrics collection on (true) or off (false):

- **lisa.vse.metrics.txn.counts.level**=service

  This property controls the level at which transaction counts are recorded. Options are none (or false), service (the default) or request.

- **lisa.vse.metrics.sample.interval**=5m

This property controls how often transaction rate and response times are sampled.

- **lisa.vse.metrics.delete.cycle**=1h
- **lisa.vse.metrics.delete.age**=30d

These properties control how often old metric data is scanned for and deleted, and what is considered old.

- **lisadb.internal.enabled**=true

  Indicates whether to start the internal Derby database instance in the registry.

- **lisadb.internal.host**=0.0.0.0

  The network interface used by the internal Derby database. The default value 0.0.0.0 indicates that all interfaces are used.

- **lisadb.internal.port**=1528

  The port number that the internal Derby database listens on.

Database Properties

The following components interact with a database: reporting, Agent broker, VSE, and ACL. Typically you would point them all to the same connection pool, but you can define a separate pool for each or mix and match. If you want to define a new pool, just set up properties like **lisa.db.pool.myPool.url**. The underlying pool implementation is the open source c3p0 pool, and the various properties will be passed down. There are many settings to choose from: see http://www.mchange.com/projects/c3p0/index.html#configuration_properties.

- **lisadb.reporting.poolName**=common
- **lisadb.vse.poolName**=common
- **lisadb.acl.poolName**=common
- **lisadb.broker.poolName**=common
- **lisadb.pool.common.driverClass**=org.apache.derby.jdbc.ClientDriver
- **lisadb.pool.common.url**=jdbc:derby://localhost:1528/database/lisa.db;create=true
- **lisadb.pool.common.user**=rpt
- **lisadb.pool.common.password_enc**=76f271db3661fd50082e68d4b953fbee

  Set the password by removing the trailing _enc from the property name and adding =MyPlaintextPassword. The password is automatically encoded at startup.

Extra properties to keep the number of connections to a minimum if we are idle.

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

Other common database settings: copy and paste the relevant user, password and pool size parameters from the common template.

- **lisadb.pool.POOLNAME.driverClass**=oracle.jdbc.OracleDriver
- **lisadb.pool.POOLNAME.url**=jdbc:oracle:thin:@HOST:1521:SID
- **lisadb.pool.POOLNAME.driverClass**=com.ibm.db2.jcc.DB2Driver
- **lisadb.pool.POOLNAME.url**=jdbc:db2://HOST:50000/DBNAME
- **lisadb.pool.POOLNAME.driverClass**=com.microsoft.sqlserver.jdbc.SQLServerDriver
- **lisadb.pool.POOLNAME.url**=jdbc:sqlserver://durry;databaseName=LISA
- **lisadb.pool.POOLNAME.driverClass**=com.mysql.jdbc.Driver
- **lisadb.pool.POOLNAME.url**=jdbc:mysql://HOST:3306/DBNAME

Appendix B - Custom Property Files (local.properties, site.properties)

There are two other property files that you can use for your custom properties:

- **local.properties**
- **site.properties**

Site properties can be stored at the test server, which automatically pushes the site.properties file to all workstations that connect to it.

Properties files are loaded in the following order:

1. System properties

2. lisa.properties
3. local.properties
4. site.properties

> ⚠ When LISA Workstation is started, you are prompted to connect to a registry. After the registry is connected, a site.properties is sent to the LISA Workstation. If you "toggle" the registry and change from Reg1 to Reg2 the site.properties from Reg2 are sent to LISA Workstation.

To use a custom property file

1. Go to the LISA_HOME directory.
2. Copy the _site.properties or _local.properties file and paste it in the same directory.
3. Change the name of the file that you pasted to site.properties or local.properties.

local.properties

lisaAutoConnect=tcp://somehost/Registry

To automatically load site-wide properties from a LISA Registry, uncomment this and make it the right URL to your LISA Registry. Usually hostname/lisa.TestRegistry will work. Your properties in this file will override any properties defined at the site level.

License Properties

laf.server.url=https://license.itko.com

laf.domain=iTKO/LISA/YOURCO

laf.username=YOURUSERNAME

laf.password=YOURPASSWORD

VSE Properties

lisa.vse.deploy.dir=

Lets you override the directory where runtime files are managed. Set this to an absolute directory path. If you do not start it with a drive specification, then it becomes relative to the current directory, which is LISA_HOME. If you are specifying a multi-level directory, you must specify in this format: C:\\Temp\\myVSE.

> ⚠ In properties files, backslashes MUST be doubled to be recognized.

lisa.vse.si.default.text.editor=

Valid values are the list of VSE SI text editor class names found in typemap.properties assigned to the vseTextBodyEditors name. If a custom text editor is written and made available through standard SDK methods, then that class name may also be a value for this property.

The following properties let you have VSE use old-style socket I/O rather than NIO. There are restrictions when using old style socket support.

- The ability of VSE to automatically handle being a proxy in front of SSL cannot be supported. The solution is to create a VSM that is in Live System mode and points to the actual virtual service, with SSL turned on.
- The ability of VSE to still handle plain text traffic, even when the listen step has been configured to use SSL cannot be supported. The solution is to provide two virtual services, one with SSL configured and one not.
- Using old style socket support will not scale as well as the default socket support VSE uses.

lisa.vse.tcp.uses.nio=true

Setting this to false will cause both plain and SSL sockets to use old-style I/O.

lisa.vse.plain.tcp.uses.nio=lisa.vse.tcp.uses.nio

This controls plain sockets only.

lisa.vse.ssl.tcp.uses.nio=lisa.vse.tcp.uses.nio

This controls SSL sockets only.

lisa.vse.execution.mode=EFFICIENT

EFFICIENT uses the most efficient path through a VS model.

TRACK records VSE activity at the VS level. (With the improvements to the VSE dashboard in LISA 6, this is less useful).

LIVE routes requests received by a VS model to a live system (somewhat like a passthrough mode),

VALIDATION uses both VSE and the live system to determine a response. Both are recorded as tracking information and feeds the model healing process. The live response becomes the response of the VS.

DYNAMIC invokes a script or subprocess for every request which must resolve to one of the other 4 modes. This is useful for tracking only certain requests seen by the VS model.

Web 2.0 Properties

lisa.browser.client.user.single=true

Use a single user.

lisa.browser.share.subprocess.state=true

Share state in subprocess.

Property for All Projects Home within a Server Environment

lisa.projects.home=`LISA_HOME`

License properties if using an HTTP proxy server

laf.usehttpproxy.server=true

laf.httpproxy.server=my_proxyserver.com

laf.httpproxy.port=3128

If your proxy server requires credentials, leave blank to use native NTLM authentication.

laf.httpproxy.domain=if needed for NTLM

laf.httpproxy.username=if needed

laf.httpproxy.password=if needed

If there is a license error, also optionally send email: hostname,toAddress,fromAddress.

laf.email.alert=mailhost,recipient@mycompany.com,from@mycompany.com

SDK Properties

LISA SDK examples need the following:

asserts=com.mycompany.lisa.AssertFileStartsWith

com.mycompany.lisa.AssertFileStartsWith=com.itko.lisa.editor.DefaultAssertController,com.itko.lisa.editor.DefaultAssertEditor

filters=com.mycompany.lisa.FilterFileFirstLine

com.mycompany.lisa.FilterFileFirstLine=com.itko.lisa.editor.FilterController,com.itko.lisa.editor.DefaultFilterEditor

nodes=com.mycompany.lisa.node.FTPTestNode

com.mycompany.lisa.node.FTPTestNode=com.mycompany.lisa.node.FTPTestNodeController,com.mycompany.lisa.node.FTPTestNodeEditor

Change the default behavior for validating SSL certificates.

ssl.checkexpiry=false

Should it check the validity dates for the certificate?

ssl.checkcrl=false

Should it check the cert revocation list specified in the certificate?

Enable a client cert and password for SSL (used by both HTTP Step and Raw SOAP Step Execution; also used by Web Service Step Execution if not overridden).

ssl.client.cert.path=a full path to the keystore

ssl.client.cert.pass=password for the keystore (this password will be automatically encrypted when LISA runs)

ssl.client.key.pass=optional password for the key entry if using JKS keystore and key has different password
from keystore (this password will be automatically encrypted when LISA runs)

Override client certificate and password for SSL (**ssl.client.cert.path** and **ssl.client.cert.pass**) for Web Service Step Execution

ws.ssl.client.cert.path=a full path to the keystore

ws.ssl.client.cert.pass=password for the keystore (this password will be automatically encrypted when LISA runs)

ws.ssl.client.key.pass=optional password for the key entry if using JKS keystore and key has different password
from keystore (this password will be automatically encrypted when LISA runs)

HTTP Authorization Properties

These credentials will be automatically encrypted when LISA runs. To reset the values, use the unencrypted property names. If you want to use native NTLM authorization (Windows only), leave these settings commented out.

lisa.http.domain=<domain name> use this for NTLM

lisa.http.user=<username>

lisa.http.pass=<password>

Preemptively send authorization information rather than waiting for a challenge: valid values are basic, ntlm, or negotiate.

lisa.http.preemptiveAuthenticationType=ntlm

NTLMv2 is used by default but by setting this property to true it can be forced to use NTLMv1 when not using native integrated Windows authentication.

lisa.http.forceNTLMv1=true

HTTP Proxy Server Properties

lisa.http.webProxy.host=<machine name or ip>

lisa.http.webProxy.nonProxyHosts=<machine name or ip> – you can also use * as a wildcard

lisa.http.webProxy.port=

lisa.http.webProxy.ssl.host=<machine name or ip>

lisa.http.webProxy.ssl.nonProxyHosts=<machine name or ip> – you can also use * as a wildcard

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=

lisa.http.webProxy.nonProxyHosts.excludeSimple=true

Exclude simple host names from proxy use.

lisa.http.webProxy.preemptiveAuthenticationType=ntlm

Preemptively send authorization information rather than waiting for a challenge: valid values are **basic** or **ntlm**.

lisa.http.timeout.connection=15000

HTTP Timeout (in milliseconds) - To extend the timeout to wait indefinitely, set the values to zero.

lisa.http.timeout.socket=180000

HTTP Timeout (in milliseconds) - To extend the timeout to wait indefinitely, set the values to zero.

XML Serialization settings

lisa.toxml.serialize.mode=NOREFERENCES

NOREFERENCES means a simple tree is rendered. Circular references are not allowed. XPATHREFERENCES means XPATH notation is used for references. Circular references can be used. If there is a circular reference, we will fall back to XPATHREFERENCES but any serialization before v3.6 that might be re-read will fail and may require setting to a default of XPATHREFERENCES.

Workstation management

lisa.ui.admin.tr.control=no

lisa.ws.jms.SoapAction.quoted=false

SOAP over JMS with Tibco BusinessWorks/Active Matrix: BW/AM require that the SoapAction header be quoted. It is possible that in the future, Active Matrix will not require this. If this is the case, uncomment this line.

When LISA serializes a date, time, or dateTime it uses the following formats by default. If you want to change the default formats/timezone you can do that with these properties or dynamically set them during the test case. You can set the format to one that does not comply with the XML schema specification, but this is not recommended (except for negative test cases).

lisa.ws.ser.dateFormat=yyyy-MM-dd

lisa.ws.ser.timeFormat=HH:mm:ss.SSS'Z'

lisa.ws.ser.timeFormat.timeZone=GMT

lisa.ws.ser.dateTimeFormat=yyyy-MM-dd'T'HH:mm:ss.SSS'Z'

lisa.ws.ser.dateTimeFormat.timeZone=GMT

ws.raw.format=true

As of 4.0.5 LISA can format the SOAP response for RAW Soap Steps. If you want the response formatted, set this LISA property to true (this can be done dynamically at runtime also).

lisa.ws.endpoint.fullautoprop=false

As of 5.0.11 the entire WS Endpoint URL is automatically converted into a single LISA property. Set this property to **false** if you want to revert back to the old behavior of converting the URL to use WSSERVER and WSPORT properties.

stats.unix.xml.folder=`LISA_HOME`/umetrics

IP Spoofing

lisa.ipspoofing.interfaces=2-34-56-78-90-AB,eth0,Realtek PCIe GBE Family Controller

Interfaces: a comma-separated list of interfaces that will be used for IP spoofing. These can be named using the MAC address (JDK 1.6+), interface name, or interface display name.

lisa.ui.useNativeFileDialog=true

Force the use of a native file dialog.

Quick Start Recent Items Properties

lisa.prefill.recent=10

lisa.quickstart.recent=5

Specify the maximum number of recent items to show in the Open Recent Quick Start tab (no less than 5) and in Prefill Comboboxes (no less than 10).

lisa-invoke Properties

lisa.portal.invoke.base.url=/lisa-invoke

lisa.portal.invoke.report.url=/reports

lisa.portal.invoke.server.report.directory=`lisa.tmpdirlisa.portal.invoke.report.url`

lisa.portal.invoke.test.root=d:/lisatests/

Server host name Properties

lisa.net.externalIP=localhost

This must be the external IP address or DNS name for the Registry to be accessible remotely. This is referenced in the relevant connection properties. If the value is left at "localhost" it will automatically be overridden in the registry server's external IP address when sent to connecting applications.

LISA Agent and Pathfinder Properties

lisa.pathfinder.on=false

If you are experiencing agent errors and you are not licensed for the LISA agent, consider turning it off here.

IBM WebSphere MQ Properties

The string-value is usually supplied by a data set such as the Unique Code Generator.

lisa.mq.correlation.id=string-value

(runtime value) will set the correlation id for an outgoing MQ message

lisa.mq.correlation.id.bytes=byte[]

(runtime value) will set the correlation id as a byte[] for non-printable values

lisa.mq.message.id=string-value

(runtime value) will set the message id for an outgoing MQ message

lisa.mq.message.id.bytes=byte[]

(runtime value) will set the message id as a byte[] for non printable values

JMS Properties

The string-value is usually supplied by a data set such as the Unique Code Generator.

lisa.jms.correlation.id=string-value

(runtime value) will set the JMSCorrelationID for an outgoing JMS message

lisa.jms.ttl.milliseconds=num-value

(runtime value) will set the time-to-live for an outgoing JMS message

Miscellaneous Properties

lisa.coord.failure.list.size=15

If too many errors are reported in a test, it will cause the coordinator to use up all its available heap space, which will require a restart of the LISA server. This is bad if multiple tests are running, and one test gets staged and throws nothing but errors. This property limits the size of the coordinator failure list.

testexec.lite.longMsgLen=1024

To view the full text of long responses, set this property to the length of your longest response. Run your test, capture what you need, verify that the correct response is indeed being sent back, and then comment out the added line in the local.properties file to revert to the default length of 1024.

java.rmi.server.hostname=<<IP_ADDRESS_OF_SELECTED_NIC>>

In the local.properties file to force communication through a specific NIC on the machine, add:

lisa.xml.xpath.computeXPath.alwaysUseLocalName=false

This property configures whether the XPath local-name() function is always used during XPath generation. The default value is false, meaning that local-name() will only be used when necessary. To generate an XPath that will work regardless of the namespace of an

XML node, set the value of this property to true.

lisa.commtrans.ctstats=true

To capture the information needed to populate the Cumulative HTTP Traffic Summary report, you must enter this property into one of the custom property files.

site.properties

The properties in **site.properties** are sent from the registry to any LISA application or service that connects to it. These properties take precedence over any properties defined locally in **lisa.properties**. However, these properties do not take precedence over properties defined locally in **local.properties** or defined on the command line using -D command line option.

The following components interact with a database: reporting, VSE, ACL and the broker. By default, these components use the **common** connection pool. However, you can define a separate pool for each or mix and match. To define a new connection pool, add properties such as **lisadb.pool.newpool.url**. The underlying pool implementation is the open source c3p0 pool. LISA will pass the various properties down. For information about the c3p0 settings that you can choose from, see http://www.mchange.com/projects/c3p0/index.html#configuration_properties.

These are the default properties used by LISA to connect to the default internal Derby database. To configure LISA to use an external database, see the appropriate **site.properties** file in the **LISA_HOME/database** directory.

lisadb.reporting.poolName=common

lisadb.vse.poolName=common

lisadb.acl.poolName=common

lisadb.broker.poolName=common

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

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

lisadb.pool.common.user=rpt

lisadb.pool.common.password_enc=76f271db3661fd50082e68d4b953fbee

lisadb.pool.common.minPoolSize=0

lisadb.pool.common.initialPoolSize=0

lisadb.pool.common.maxPoolSize=10

lisadb.pool.common.acquireIncrement=1

lisadb.pool.common.maxIdleTime=45

lisadb.pool.common.idleConnectionTestPeriod=5

lisadb.internal.enabled=true

Controls whether the registry starts an internal Derby database.

lisa.pathfinder.on=true

LISA DCM Settings

lisa.dcm.labstartup.min=6

lisa.dcm.lisastartup.min=4

lisa.dcm.lisashutdown.min=2

lisa.dcm.lab.factories=com.itko.lisa.cloud.serviceMesh.ServiceMeshCloudSupport;com.itko.lisa.cloud.vCloud.vCloudDirectorCloudSupport;;

lisa.dcm.SERVICEMESH.baseUri=<url>

lisa.dcm.SERVICEMESH.userId=<userId>

lisa.dcm.SERVICEMESH.password=<password>

lisa.dcm.SERVICEMESH.lab.cache.sec=<int seconds default 300>

```
lisa.dcm.vCLOUD.baseUri=<https://<fqdn>/api/versions>

lisa.dcm.vCLOUD.userId=<userId>

lisa.dcm.vCLOUD.password=<password>

lisa.net.timeout.ms=60000
```