Using SOAtest Event Monitor Tool

This topic explains how to configure and apply the Event Monitor tool, which traces the internal events within systems (e.g., ESBs, Java applications, databases, etc.) and allows you to make them part of SOAtest's end-to-end test scenarios. Sections include:

Understanding the Event Monitor

The Event Monitor tool provides visibility into intermediary messages without requiring direct access to the messaging system's interface. Additionally, you can apply the available tool set to validate whether messages satisfy functional expectations.

SOAtest provides built-in support for monitoring many systems, including:

- TIBCO Enterprise Messaging System
- Sonic Enterprise Service Bus systems
- Oracle/BEA Aqualogic Service Bus
- Software AG webMethods Broker
- IBM WebSphere ESB
- Other JMS-based systems
- Java applications
- Relational databases

You can also configure the tool to monitor any custom API-based events source, such as non-JMS systems from other vendors, custom logging frameworks, etc.

> **The Event Monitor should always be positioned as the first test in the test suite.**
>
> If you want to monitor more than one target component (e.g., Java method calls and a JMS topic), then you can add more than one Event Monitor to your test suite and configure each one accordingly.

To configure it, you tell SOAtest how to connect to your system and what you want it to monitor.

When you run the test suite, SOAtest will start the event monitor and then keep it running as the test suite's tests execute. SOAtest monitors and reports the specified events/messages as they occur. Additional tools can then be chained to validate or further process the monitored events/messages.

Tool Configuration

The configuration procedure depends on what type of system you're monitoring. This tool can be used for:

- Monitoring Oracle or BEA AquaLogic Service Bus
- Monitoring Software AG webMethods Broker
- Monitoring Sonic ESB
- Monitoring TIBCO EMS
- Monitoring Other JMS Systems
- Monitoring Java Applications
- Monitoring Databases
- Monitoring a Custom API-Based Events Source

You can monitor more than one target component by adding additional Event Monitors to your test suite and configuring each one accordingly.

Test Suite Execution

Test suite execution is the typical and recommended usage of the Event Monitor tool. Add an Event Monitor tool to your test suite and make it the first test before the test step where you want it to start monitoring. When you select and run the parent test suite, the Event Monitor will start automatically at the appropriate step and continue monitoring while the rest of the tests in the test suite execute. It will stop when the first of the following two events take place:

1. The last test in the test suite (or last row in the data source, if the test suite tests are iterating over a data source) has completed execution, the last event has been retrieved, and the monitoring connection has been destroyed.
2. The maximum monitor execution duration (this value is configured in the tool's Options tab) has been reached.

You do not need to set the Event Monitor's parent test suite to **Tests run concurrently**. SOAtest will automatically recognize the Event Monitor as a special tool and manage its concurrent execution accordingly.

Stand-Alone/Ad-Hoc Execution

You may configure the connection settings of an Event Monitor and execute it on its own while exercising your target systems outside of SOAtest. For example, you may wish to monitor JMS messages in a back-end middleware system while you manually browse the Web interface of your application.

To watch the events being logged in real time, open the **Event Viewer** tab. The Event Monitor tool will run for the duration that is set under Options tab's **M aximum monitor execution duration (milliseconds)** setting. Such adhoc execution is not applicable when the **Custom events source** option is selected with the **Poll after each test execution** pattern.

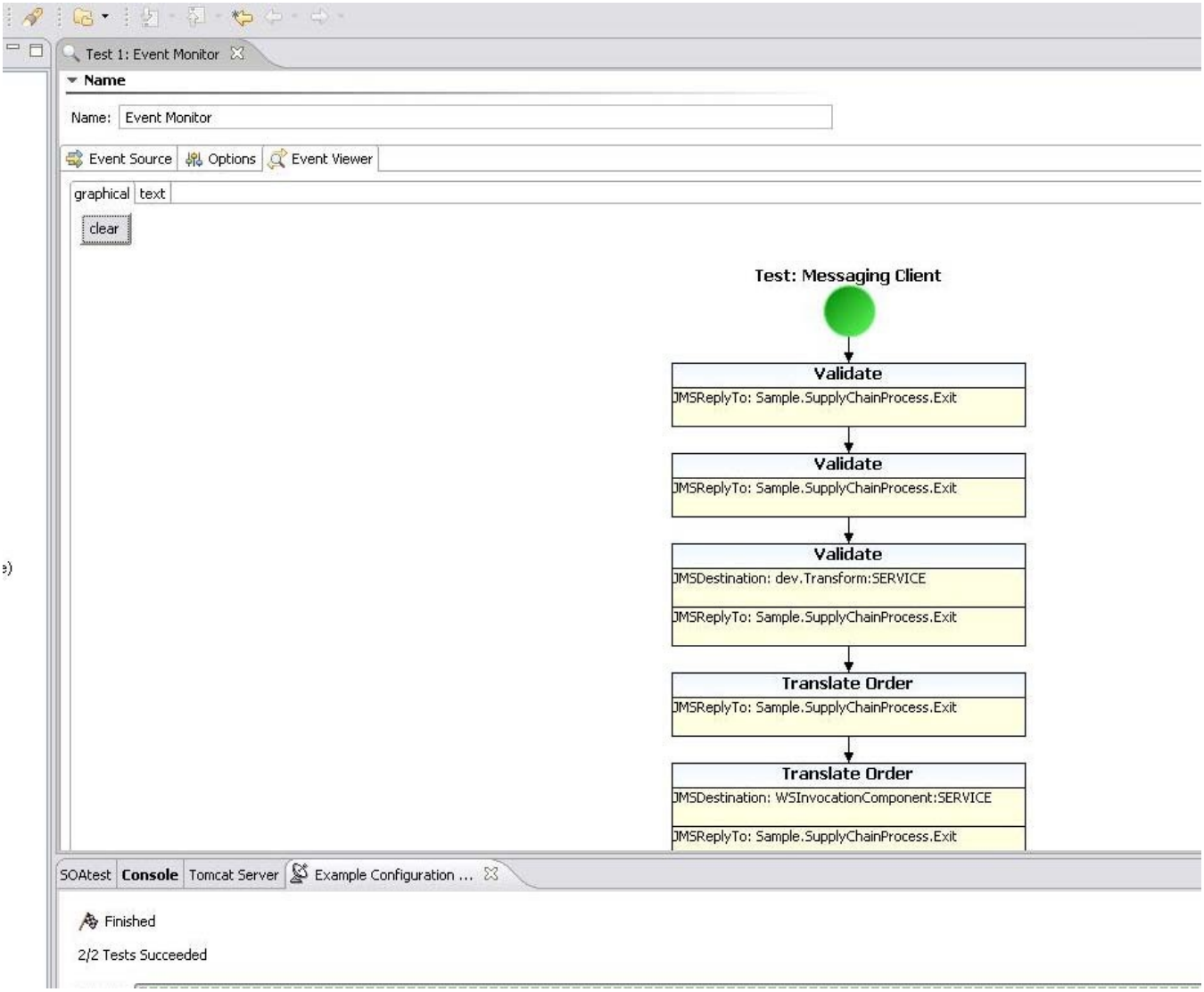
Viewing Monitored Events

Events will be reported and visualized in the Event Viewer tab within the Event Monitor tool configuration panel.

To view monitored events during in real time:

- Keep the **Event Viewer** tab (in the Event Monitor tool's configuration panel) open during test execution.

There are two tabs available: the graphical view and the text view.



In the graphical view, click on an event to see message details.

If you don't see the Graphical View

The graphical view is available by default if you have SOAtest standalone (vs. the plugin) or if your Eclipse has Eclipse GEF installed. You can download and install Eclipse GEF from http://www.eclipse.org/gef/.

Using Data Sources

Event Monitor itself is not parameterizable with a data source. However if the tests inside the test suite that includes the Event Monitor tool are parameterizable, it will start monitoring before the first actual test (following the Event Monitor test) and first data source row executes, and it will stop monitoring when all data sources rows have been used. This ensures that you will obtain a single, seamless log of events regardless of how the tests iterate inside the test suite.

Retrieving Events from Other Platforms

The Event Monitor tool has extensibility hooks that allow it to obtain events from a variety of sources besides the built-in platforms and systems that it supports. See Extensibility API Patterns for details.

Validating Monitored Messages

If the transaction executes correctly, you can add regression controls for monitored messages as described in Configuring Regression Testing. In addition, you can add validations as described in Adding Test Outputs. The available validation tools are listed in Validation Tools.

You can also monitor the system while the transactions execute (for example, trigger the transaction from the Web interface), and then generate tests automatically from these messages. This includes tests for transaction entry/exit points, as well as tests for intermediate parts of the transaction. In this way, you can reproduce issues quickly, create tests with real values rapidly, then isolate the system and create regression tests around the pieces and components of the transaction. For details on how to do this, see:

- Creating Tests From Sonic ESB Transactions
- Creating Tests From TIBCO EMS Transactions
- Creating Tests From JMS System Transactions