Creating Tests From JMS System Transactions

This topic explains how you can use SOAtest to monitor transactions that pass through any JMS system, then generate functional test cases that check the monitored messages.

Sections include:

> **Alternative Test Creation Method**
>
> Another way to create tests is to have SOAtest's recording proxy monitor traffic at one or more JMS, HTTP, or MQ endpoints as an application is exercised. SOAtest "listens" to traffic requests and responses, then builds a traffic file of legitimate request/response pairs. This traffic is then used to generate a test suite that represents the captured behavior in preconfigured SOAP Client or Messaging Client tools. See Creating Tests From Recorded HTTP, JMS or MQ Traffic for details.

Overview

SOAtest can monitor transactions that pass through a JMS, then generate functional test cases that check the monitored messages. In addition to providing visibility into the systems messages, this allows you replay transactions directly from SOAtest and verify that the monitored functionality continues to work as expected.

To achieve this, you tell SOAtest how to connect to your JMS and what destination (topic or queue) messages you want it to monitor, then you prompt it to start monitoring. SOAtest will generate a test suite of Messaging Client tests for each JMS message captured at the specified destination or for all messages within the process flow (if a process tracking topic was used). These tests are preconfigured with the connection parameters, requests, and destination information so that SOAtest can replay the same messages.

SOAtest can generate test clients for the following types of JMS messages:

- javax.jms.TextMessage
- javax.jms.MapMessage
- javax.jms.ObjectMessage
- javax.jms.BytesMessage
- javax.jms.StreamMessage

Prerequisites

See JMS Prerequisites.

Generating Tests from JMS Transactions

To generate tests:

1. Choose the **Other> Java Message Service (JMS)** option in one of the available test creation wizards. For details on accessing the wizards, see:
   - Adding a New .tst File to an Existing Project
   - Adding a New Test Suite
2. In the JMS wizard page, complete the following:
   a. In the **Connection** area, specify your JMS connection settings.
   b. In the **Initial Context** field, specify a fully-qualified class name string, passed to the JNDI javax.naming.InitialContext constructor as a string value for the property named javax.naming.Context.INITIAL_CONTEXT_FACTORY.
   c. In the **Connection Factory** field, specify the JNDI name for the factory This is passed to the lookup() method in javax.naming. InitialContext to create a javax.jms.QueueConnectionFactory or a javax.jms.TopicConnectionFactory instance.
   d. In the **Destination Name** field, specify the topic or queue that you want to monitor.
      - You can specify a regular topic or queue (e.g., the entry or exit of a workflow process), or a special process tracking topic .
   e. In the **Destination Type** field, specify whether the tracking destination is a topic or a queue.
   f. (Optional) In the **Message Selector** field, enter a value to act as a message filter. See Using Message Selector Filters for tips.
   g. If you want SOAtest to use the JMS QueueBrowser API in order to trace messages posted on a JMS queue— without removing them from the queue— enable the **Leave messages on the queue** option. This allows SOAtest to gain visibility into these messages without impacting the transaction.

      > **Caution: Leave messages on the queue**
      >
      > For a discussion of potential complications with this option—and how to avoid them—see JMS Queue Options.

   h. In the JNDI properties table, specify any additional JNDI properties you want applied to this deployment.

3. Click **Next**. SOAtest will start monitoring the messages that match the settings specified in the previous wizard page. If you run another application that sends messages to the bus, those messages will be noted in this panel.
4. When you are ready to stop monitoring, click **Finish**. SOAtest will then create test cases based on the verified messages.

Monitoring Intermediary Messages

In addition to automatically generating functional tests from monitoring the transaction messages that touch JMS endpoints in ESBs or middleware systems, you can also visualize and trace the intra-process JMS messages that take place as part of the transactions that are triggered by the tests, and then dissect them for validation.

For details on how to do this, see Event Monitoring - ESBs, Java Apps, Databases, and other Systems.