

# OpenSSO QA Test Automation

## Notification module Testing setup & development

Prepared by Mrudula Gaidhani

### 1. Introduction

This document describes in detail the QA test automation focusing on notification. It explains the following:

- The Requirements for this module
- How tests are organized.
- Execution details
- Tests in the framework
- Interpreting the report
- Debugging the test failures
- How to add new tests

### 2. Requirements

The primary requirements are:

1. Federated Access Manager war is deployed on the supported web or j2ee container.
2. The datastore plugin should contain “description” attribute under Groups Attribute list and Roles Attribute List. If the datastore is created through the qatest, then these attributes are added. But user can add these manually too. To achieve this, follow these instructions
  1. Login to admin console using amadmin
  2. Go to Access Control tab, then click on the specific realm. After that click on Datastore tab and then click on the datastore.
  3. Scroll down to find Groups Attribute list. Add “description” in this list.
  4. Scroll down to find Roles Attribute list. Add “description” in this list and now save the changes.

### 3. Organization of tests

In the section, we shall discuss how tests are organized. Currently there are testcases for following features

- SMNotification: Various service modification notifications are tested. The modified services are at the global level as well as at the realm level.
  - Platform service
  - Logging service
  - Authentication service at the realm level
  - Authentication service at the global level
  - Session service at the global level

- IDRepoNotification: It tests add, modify and delete notifications for all the supported ID types such as User, Agent, Roles, Groups, Filtered Roles.

#### **Test organization details:**

1. ConfigUnconfig.java starts the jetty server to listen the notifications sent by server to the client. QATest becomes a client application based on the client sdk.
2. This is how the directory and file's are laid out for notification
  1. <TEST\_HOME>/xml/testng contains notification testng xml files
  2. <TEST\_HOME>/resources/ notification contains all properties files for notification testing
  3. <TEST\_HOME>/source/com/sun/identity/qatest/notification contains all sources of Java implementations for notification.
3. Notification tests are divided under different groups such as
  1. ds\_ds : Directory Server for User management and the backend repository
  2. ds\_ds\_sec : Directory Server for User management and the backend repository with Keystore configured

## **4. Execution details**

### **4.1 How to execute notification tests?**

This section describes how to execute the notification tests.

1. Deploy Federated Access Manager 8.0 war on desired web container
2. Change following parameters in <TEST\_HOME>/build.properties file
  1. Change the value of QATEST\_HOME
  2. Change the value of TEST\_MODULE to notification
  3. Change the value of EXECUTION\_MODE to appropriate group name as described in section 3
  4. Change the value of REPORT\_DIR to desired location
5. Run following command to execute notification module:  
`ant -DSERVER_NAME1=host1 module`

### **4.2 The execution details**

1. Framework picks execution of java classes in random order. Plus all the testcases inside the java class are also executed in random order unless the order is forced by dependsOn annotation.
2. Each java class contains multiple tests marked with @Test annotation. Methods annotated by @BeforeClass are setup methods. Notification listener is registered in this method. Methods annotated by @AfterClass are cleanup method. They bring system back to original state. The examples is removing the notification listener. These methods will be run even if the tests fail.
3. If the IDRepoNotification test fails during addition or modification of the identity then cleanup will be executed to make sure the identity is deleted. It will leave the system clean and subsequent run will also produce similar results.

## 5. Tests details

### 5.1 Current Tests classes:

Current Tests in the QATest Framework for notification are described in the following table:

Test Class	Properties files used	Description
IDRepoNotificationTests.java	None	It adds the identity and waits for the notification. After successful completion of this test, that particular identity attribute is modified and a notification is required for successful test. At the end that identity is deleted and notification is received or not is checked.
SMNotificationTests.java	None	This class tests if SM notifications are received for various service modifications at global level as well as at realm level.

## 6. Interpreting the Notification automated testing results

After execution of notification test module, the results will be located under `<REPORT_DIR>/<HOSTNAME_1>/<EXECUTION_MODE>/<EXECUTION_DATE_TIME>`

Open index.html residing in this directory from the browser. It will display the overall result of the test execution with the following details:

- a link named “`<EXECUTION_MODE>-notification`” which points to the detailed test report
- the number of tests which passed
- the number of tests which failed
- the number of test skipped
- a link to the TestNG XML file used in this test run

To learn more about the specific tests click on the link “`<EXECUTION_MODE>-notification`”. In the left frame of the resulting page, the individual results of all the tests which were executed. Passing tests will have a background color of green. Failed tests will have a background color of red.

To find out more information on the results a particular test click on the “Results” link for that test. This will provide you more information about the test such as when the test was executed, the duration of the test in seconds, the test method being executed, and any exception that was thrown during execution of the test.

To view all the log messages which were displayed for a particular test go to the file `<REPORT_DIR>/<HOSTNAME>/<EXECUTION_MODE>/<EXECUTION_DATE_TIME>/logs`. In this file, search for the name of the test of interest. Below the name the log records produced during the three phases of this test's execution, setup, verification, and cleanup, can be viewed.

Execution output is captured in `<REPORT_DIR>/<HOSTNAME>/<EXECUTION_MODE>/<EXECUTION_DATE_TIME>/notification output file`.

## 7. Debugging the notification automated test failures

- To learn more about the specific tests click on the link “`<EXECUTION_MODE>-notification`”. In the left frame of the resulting page, the individual results of all the tests which were executed. Click on the link for the failed tests to see the exception reported.
- The details of each test are logged under `<REPORT_DIR>/<HOSTNAME>/<EXECUTION_MODE>/<EXECUTION_DATE_TIME>/logs`. For each test it will list the XML executed. Open those XML's to debug more.

## 8. How to add new testcases?

This section describes how to add new testcases to the existing or new java class. Notification testcases are divided into different features. New feature related tests should be added in the new java class.

**To add** new testcases in existing class:

1. Before adding new testcase in the existing class, make sure you understand all the existing tests in that class.
2. Logically new testcase should belong to that class.
3. If the class has dependencies between the tests then add a testcase in such a way that it doesn't break the existing dependencies. Plus make to add the new dependency for that test too.
4. Add new properties in the properties file if needed.

**To add** new testcases in new class:

1. Create a new class with appropriate class name. The properties files should have similar name. Please follow naming conventions described in the openSSO QATest document.
2. The class should follow setup, test(s) and cleanup procedures. Cleanup should make sure it restores the server to the default state.
3. Appropriate groups should be assigned to these newly added testcases.
4. Update all the notification related testng xml files.
5. Run notification module with all the tests and make sure all the tests including newly added tests are passing.

**Other general considerations** to be taken while writing new testcases.

1. To add the new test scenario, based on the feature it belongs to, add it to appropriate test class. Ideally new scenario shouldn't have any dependency on existing tests.
2. In testNG, tests from each class are run in random order. Thus its better not to have dependency on other test. In case it is really needed, to control the order you can use dependsOnMethod annotation.
3. Test related setup & cleanup should be done in already existing methods. After cleanup, the system should be brought back to the original state.