

OpenSSO QA Test Automation

Session module Testing Setup & Development

Prepared by Rahul Misra

1. Introduction

This document describes in detail the QA test automation focusing on Session related tests. 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:

- Setup OpenSSO server

3. Organization of Tests

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

- Validate Global Max Session Time
- Validate Global Max Idle Time
- Validate Realm Max Session Time
- Validate Realm Max Idle Time
- Validate attribute inheritance between Global and Realm. This tests when different values are set at Global and Realm level.
- Setting fixed property in SSO Token. These are the standard predefined properties in each session.
- Setting custom property in SSO Token. These are custom properties which can be set in a session.
- Setting protected property in SSO Token. These are properties which a user can choose to restrict from a change.

Test organization details:

1. The tests are divided into multiple java classes based on features listed above.
2. This is how the directory and file's are laid out for session
 1. <TEST_HOME>/xml/testng contains session testng xml files
 2. <TEST_HOME>/resources/session contains all properties files for sessiontesting
 3. <TEST_HOME>/source/com/sun/identity/qatest/session contains all sources of Java implementations for session.
 4. <TEST_HOME>/<server name>/built/classes will contain xml files which will be generated at run time for session testing.
3. Session tests are divided under different groups such as
 1. ff_ds : Embedded for User management & Directory Server as the service data

- repository
- 2. ds_ds : Directory Server for User management and the service data repository
- 3. ff_ds_sec : Embedded for User management & Directory Server as the service data repository with security enabled
- 4. ds_ds_sec : Directory Server for User management and the service data repository with security enabled

4. Execution details

4.1 How to execute session tests?

This section describes how to execute the session tests.

1. Deploy Federated Access Manager 8.0 war on a host
2. Before executing this test module, user should create a Configurator-<server name>.properties corresponding to server war deployment. Please refer to OpenSSO QATest automation framework document for details.
3. Change following parameters in <TEST_HOME>/build.properties file
 1. Change the value of QATEST_HOME
 2. Change the value of TEST_MODULE to session
 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
4. Run following command to execute session module:
ant -DSERVER_NAME1=<server host name as mentioned in configuration properties file> 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. They prepare the system for the tests to execute. The examples are adding the users, creating policies etc. Methods annotated by @AfterClass are cleanup method. They bring system back to original state. The examples are delete the users created in setup method, delete the policies etc. These methods will be run even if the tests fail.

5. Tests details

5.1 Current Tests classes:

Current Tests in the qatest Framework for session are described in the following table:

Test class	Properties file used	Description
SessionServerTimeConstraintsTest.java	None	Session tests related to max session and idle time at global and realm level.

SessionProperty.java	SessionProperty.properties	Test related to fixed, custom and protected session propertoies
----------------------	----------------------------	---

5.2 Properties Files

Each test class may have a corresponding properties file. These files mainly contain the configuration and execution details used for testing. These details can be changed if needed.

5.3 Common Files

There are multiple common files which are used by session tests.

1. IDCommon.java
This class contains methods related to identity management
2. SMSCommon.java
This class contains methods related to service management

6. Interpreting the Session automated testing results

- After execution of session test module, the results will be located under `<REPORT_DIR>/<SERVER NAME>/<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>-session” 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>-session”. 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>/<SERVER NAME>/<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.

7. Debugging the session automated test failures

- Execute the tests by setting the log level to FINEST in the global configuration properties file.

- To learn more about the specific tests click on the link “<EXECUTION_MODE>-session”. 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>/<SERVER NAME>/<EXECUTION_MODE>/<EXECUTION_DATE_TIME>/logs.
- Also look at the debug logs on server for further details
- Common items to look for
 - Protected properties are mentioned in the AMConfig.properties
 - Session service can be added at the realm level
 - Different session service attributes values can be set at Global and Realm level
 - Default Session service addition at realm has same values as the one's specified in the global configuration

8. How to add new testcases?

This section describes how to add new testcases to the existing or new java class. Session testcases are divided into different features. New feature related tests should be grouped together and added in the new java class. Testcases related to already automated features should be added to existing classes.

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 session related testng xml files.
5. Run session 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 should not have any dependency on existing tests.
2. In testNG, tests from each class are run in random order. Thus it is better not to have dependencies on any other tests. 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.

4. Tests should not use any command line utility.
5. For all operations, one should rely on using API calls through client sdk, as much as possible.
6. Write any reusable functionality as a common method in common classes (IDMCommon, SMSCommon, TestCommon) such that other tests may use it.
7. If there are many common methods specific to session test, create a new class called SessionCommon.java and put all the common methods in that class.