

# OpenSSO QA Test Automation

## Multi-Protocol module Testing setup & development

Prepared by Mrudula Gaidhani

### 1. Introduction

This document describes in detail the QA test automation focusing on multi-protocol. 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. For multi-protocol testing, framework assumes that there are four separate deployments of Federated Access Manager 8.0 samples war.
2. All the war's should be either in four different network domains or the cookie names in AMConfig.properties should be different.

There are two ways to achieve this required scenario:

- To have war's on different domains:
  - Consider that the first war is deployed on machineA.domain.com & second war is deployed on machineB.domain.com.
  - Now on the driver (from where this QATest automation will be run) machine change machineA.domain.com to machineA.domainA.com in /etc/hosts file for Solaris. Follow the examples given in the hosts file.  
192.99.999.9 machineA.domainA.com
  - Make sure to update the SP side /etc/hosts file in case IDP has been given a virtual hostname. Failure to do this, will generate UnknownHostException while executing WSFed tests.
- To change the cookie names:
  - Configure the war using  
`http://machineA.domain.com:port/uri/Configurator.jsp`
  - Follow product instructions to proceed with the configuration. After successful configuration, open AMConfig.properties residing under <Config\_Dir>.
  - Change the value of the parameter com.iplanet.am.cookie.name. Make

- sure both war's have different cookie names.
- Now restart the web container.

### 3. Organization of tests

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

- Smoke Tests: SAMLv2 initiated SSO, SLO, IDFF initiated SSO & SLO & WSFed initiated SSO & SLO.

#### 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 multi-protocol
  1. <TEST\_HOME>/xml/testng contains multi-protocol testng xml files
  2. <TEST\_HOME>/resources/ multiprotocol contains all properties files for multi-protocol testing
  3. <TEST\_HOME>/source/com/sun/identity/qatest/ multiprotocol contains all sources of Java implementations for multi-protocol.
  4. Multi-protocol module uses common files of SAMLv2 module, IDFF module, WSFed module. These all common files reside under <TEST\_HOME>/source/com/sun/identity/qatest/common directory.
  5. The specific common functions required by multi-protocol module are in <TEST\_HOME>/source/com/sun/identity/qatest/common/MultiprotocolCommon.java
  6. <TEST\_HOME>/servername1\_servername2/built/classes will contain xml files which will be generated at run time for multi-protocol testing.
3. Multi-protocol tests are divided under different groups such as
  1. ff\_ds\_sec : Flatfile for User management & directory Server as the backend repository with Keystore configured
  2. ds\_ds\_sec : Directory Server for User management and the backend repository with Keystore configured

### 4. Execution details

#### 4.1 How to execute multi-protocol tests?

This section describes how to execute the multi-protocol tests.

1. Deploy Federated Access Manager 8.0 wars on four different machines.
2. Before running the Multi-protocol tests please read Section 2 to make sure either cookie names are different or they are in two different domains.
3. Before executing this test module, user should create four Configurator-<hostname>.properties corresponding to four war deployments.
4. Update Configurator-<hostname>.properties file corresponding to IDP with following details:
  - multiprotocol\_enabled=true
  - idff\_sp=<host3>

- wsfed\_sp=<host4>  
Host2 will be used as SAMLv2 SP.
- 5. Please refer to OpenSSO QATest automation framework document for details. Update following four properties. Properties metaalias, entity\_name will be defaulted to hostname if not specified. In case of IDP, cot will be defaulted to idpcot & for SP, it will be spcot. Property certalias should be mentioned if the keystore is configured & if ff\_ds\_sec or ds\_ds\_sec group needs to be run.
  - metaalias=@COPY\_FROM\_CONFIG@
  - entity\_name=@COPY\_FROM\_CONFIG@
  - cot=@COPY\_FROM\_CONFIG@
  - certalias=testalias
- 6. Change following parameters in <TEST\_HOME>/build.properties file
  1. Change the value of QATEST\_HOME
  2. Change the value of TEST\_MODULE to wsfed
  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 multiprotocol module:  
`ant -DSERVER_NAME1=host1 -DSERVER_NAME2=host2 module`  
 host1 is treated as IDP while host2 is treated as SP

## 4.2 The execution details

1. Run following command to execute wsfed module:  
`ant -DSERVER_NAME1=host1 -DSERVER_NAME2=host2 module`  
 host1 is treated as IDP while host2 is treated as SAMLv2 SP. IDFF SP & WSFed SP are mentioned in host1's configurator properties. Based on the information SAMLv2ConfigData.properties, IDFFConfigData.properties and WSFedConfigData.properties are created at run time. These file will be used by all the tests.
2. OpenSSO war can be just deployed on four different instances. Configuration of all the war's is handled through each protocol's Configure<protocol>.java. These classes always run before any of the tests are executed. That is governed by BeforeSuite annotation. Those classes are picked up from each protocol package.
3. Configure<protocol>.java also creates metadata templates for SP & IDP, and loads on both sides. This will make all war's ready for multiprotocol test execution.
4. The tests are run for different protocols. The protocols are changed in the standard metadata. The parameters are passed through testng.xml. ConfigureWSFedProtocols.java does the metadata manipulation based on these parameters. After the tests are run, original metadata are loaded back to get the system back to the original state. That is governed by BeforeTest & AfterTest annotations.
5. All the testcases inside the java class are executed in random order unless the order is forced by dependsOn annotation.
6. 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, changing metadata values, 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, revert back the metadata

changes. These methods will be run even if the tests fail.

7. Each test is run using the combination of SP & IDP. All the testing will be performed using htmlunit api's. It doesn't depend on AMConfig.properties.
8. After all the testcases are run, Unconfigure<protocol>.java are run. It deletes the entities created on SP & IDP along with Circle of trusts. It brings system back to original state.
9. WSFedTestData.properties contains the data for all the tests. This file mainly contains the result messages after Single-Sign on, Single Logout, etc. If these messages are changed in the product then the user should update this file. But if message change is only for single testcase then it should go in the individual properties file. The individual properties file will override the data specified in WSFedTestData.properties file.

## 5. Tests details

### 5.1 Current Tests classes:

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

Test Class	Properties files used	Description
MultiprotocolSmokeTests.java	MultiprotocolSmkokeTest.properties IDFFConfigData.properties IDFFTestData.properties SAMLv2ConfigData.properties SAMLv2TestData.properties WSFedConfigData.properties WSFedTestData.properties	It contains SP & IDP initiated Single Sign-On, Single Logout requests. If the testcase fails, all the dependent testcases will be skipped.

### 5.2 Properties Files

- Each test class has corresponding properties file. These files mainly contain the user details used for testing. These details can be changed if needed.

### 5.3 Common Files

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

1. MultiProtocolCommon.java  
This file contains the common methods related to SAMLv2, WSFed, etc. The common methods include getting metadata from the htmlPage.
2. WSFedCommon.java  
This file contains the WSFed related common methods such as to get xml's for SSO, SLO, termination, etc.
3. IDFFCommon.java  
This file contains the IDFF related common methods such as to get xml's for SSO, SLO, termination, etc.
4. SAMLv2Common.java  
This file contains the SAMLv2 related common methods such as to get xml's for

SSO, SLO, termination, etc.

5. WebTest package:

As WSFed related tests mostly use http based actions, htmlunit is used. The common package is com.sun.identity.qatest.common.webtest. This takes a input as a xml file, which contains url to go to, inputs to fill in (if any) and action to perform. It also takes a result string. Based on the occurrence of this result, an assert with pass or fail will be added. Sample xml is as follows:

```
<url href="http://machineA.domainA.com:81/idp">
<form name="Login" buttonName="" >
<input name="IDToken1" value="idp11" />
<input name="IDToken2" value="sidp11" />
<result text="Federation Access Manager" />
</form>
</url>
```

Test code is as follows:

```
xmlfile = baseDir + "SPSLOSOAPdefaultRS.xml";
WSFedCommon.getxmlSPSLO(xmlfile, configMap, "soap");
log(Level.FINE, "SPSLOSOAPdefaultRS", "Run " + xmlfile);
task = new DefaultTaskHandler(xmlfile);
page = task.execute(webClient);
```

## 6. Interpreting the MultiProtocol automated testing results

After execution of multiprotocol test module, the results will be located under

<REPORT\_DIR>/<HOSTNAME\_1>\_<HOSTNAME\_2>/<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>-multiprotocol" 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>-multiprotocol". 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>

E>/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>/multiprotocol .output file.

## 7. Debugging the multiprotocol automated test failures

- To learn more about the specific tests click on the link “<EXECUTION\_MODE>-multiprotocol”. 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.
- Also look at the debug logs on IDP and SP to see if any exceptions are reported.

## 8. How to add new testcases?

This section describes how to add new testcases to the existing or new java class. WSFed 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 multiprotocol related testng xml files.
5. Run multiprotocol 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.
4. Tests shouldn't use any command line utility, client sdk api, deployment related global properties.
5. Write a common method in MultiprotocolCommon class to generate such xml's. So that other test also can use it. These xml's should go under <QATEST\_HOME>/built/classes directory. Each xml should have a unique name prefixed with test name.