OpenSSO QA Test Automation
Identity Repository Testing

Prepared by Brian Tran

1. Introduction

This document describes in details the QA test automation for Identity Management test in the following areas:

- The requirement for this module
- How tests are organized
- Executing details
- Tests in the framework
- Interpreting the report
- Debugging the test failures
- How to add new tests
- Global configuration file
- How to configure the datastore

2. Requirements

The requirements and set up for executing IDM test are:

1. Deploy fam.war file on a supported web container
2. Set up QA test framework with all required jar files in lib directory. For more information on QA test framework set up, please refer to OpenSSO QA Test Automation Framework
3. Create a datastore instance for Sun directory server and make sure it is started up successfully (Active Directory and generic LDAP datastores will be supported later)

3. Organization of tests

The following tests are executed for Identity Repository module:

- Create user, agent, role, filtered role, and group
- Update user, agent, role, filtered role, and group
- Search user, agent, role, filtered role, and group
- Delete user, agent, role, filtered role, and group
- Add user member to role and group
- Remove user member from role and group

Test organization details:

1. <TEST_HOME>/xml/testng : This directory contains IDM's testNG XML files to invoke IDM test cases. There are 4 groups in IDM test that represents by 4 XML files in this directory :

- ff_ds-idm-testng.xml : This testNG XML file is used to execute IDM test cases for flat file's user management datastore and directory server's service management datastore

- ff_ds_sec-idm-testng.xml: This testNG XML file is used to execute IDM test cases for flat file's user management datastore and directory server's service management with security enabled
- ds_ds-idm-testng.xml: This testNG XML file is used to execute IDM test cases for directory server's user management datastore and directory server's service management
- ds_ds_sec-idm-testng.xml: This testNG XML file is used to execute IDM test cases for directory server's user management datastore and directory server's service management with security enabled

2. <TEST_HOME>/resources/idm : This directory contains IDM test cases properties files, datastore properties file, and global data properties file
3. <TEST_HOME>/source/com/sun/identity/qatest/common : This directory contains the common classes and methods that are used by IDM test module
4. <TEST_HOME>/source/com/sun/identity/qatest/idm : This directory contains the classes and methods to execute IDM test
5. <TEST_HOME>/<hostname>/built/classes : This directory contains xml files, properties files, and Java classes which will be generated at run time for IDM testing.

4. Executing details

4.1. How to execute IDM tests?

This section describes the steps to execute IDM test module.

1. Deploy Federated Access Manager 8.0 wars on a machine with supported web container.
2. Create and customize Configurator-<hostname>.properties in <TEST_HOME>/resources directory. Please refer to OpenSSO QA Test Automation Framework document for more details and instruction to customize the parameters in this file
3. Edit <TEST_HOME>/build.properties file and change the following parameters:
    - QATEST_HOME to the location where QATest framework is installed
    - TEST_MODULE to idm
    - EXECUTION_MODE to appropriate group name as described in section 3
    - REPORT_DIR to desired location to store test result
4. Go to <TEST_HOME> and run the following command to execute IDM test module ant -DSERVER_NAME1=<hostname> module

    Note: This hostname should match with the hostname in <TEST_HOME>/resources/Configurator-<hostname>.properties file

4.2. The execution details

When IDM test is executed, it will :

1. Check and create a new sub realm if it does not exists
2. Create one or more datastore and different type of datastore depending on the

datastore configuration specified in SMSGlobalDataStore.properties and the datastore number that is specified in parameter dsindex in testNG xml file. Note: if the flag setupidm in this properties is set to false, no datastore is created. This is used for debugging purpose

3. Execute the IDM test cases that are specified in testNG xml file and generate the test results in report directory. For each test case to be executed, it will:
   - Read in number of action items and realm name the test is executed in
   - Go through the loop based on the number of action items to
     - Read in the identity attributes, check if the identity type is supported, and perform the action
     - Verify the result of the action
     - Compare the expected error code, expected error message, and/or expected result with actual data. This is for negative test cases
   - Retrieve all identities in the realm and delete them at the clean up step
4. Delete the datastore that was created previously. Note: if the flag cleanupidm is set to false, the datastore is not deleted. This is used to keep the datastore for debugging purpose
5. Delete sub realm that was created previously *(This feature is not yet implemented)*

5. Tests details

5.1 **Current Tests classes:**

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

Test Class	**Properties files used**	**Description**
IDMTestConfig.java	SMSGlobalConfig.properties, SMSGlobalDataStore.properties	A class which executes the method to create sub realm and datastore in the set up step and delete datastore and sub realm in the clean up step
IDMConstants.java		A class which contains the constants for IDM classes
IdentitiesTest.java	IDMCreateIdentitiesTest.properties, IDMUpdateIdentitiesTest.properties, IDMSearchIdentitiesTest.properties, IDMDeleteIdentitiesTest.properties, IDMAddMembersTest.properties, IDMRemoveMembersTest.properties	A class which reads the test cases properties files and based on the action defined in these properties, it executes different methods to create, update, search, delete, add members, remove member for user, agent, role, filtered role, and group identities

5.2 **Properties Files**

- SMSGlobalConfig.properties : This properties file contains the common parameters for IDM test such as user schema files, user schema attributes, etc.
- SMSGlobalDataStoreConfig.properties : This properties file contains the datastore configuration parameters for Sun directory server, Active Directory server, generic LDAP, and flat file. The instruction to define these parameters will be described in later section
- createIdentitiesTest.properties : This properties file contains the test cases for create user, agent, role, filtered role, and group identities
- updateIdentitiesTest.properties : This properties file contains the test cases for update user, agent, role, filtered role, and group identities
- searchIdentitiesTest.properties : This properties file contains the test cases for search user, agent, role, filtered role, and group identities
- deleteIdentitiesTest.properties : This properties file contains the test cases for delete user, agent, role, filtered role, and group identities
- addMembersTest.properties : This properties file contains the test cases for add user member to role and group identities
- removeMembersTest.properties : This properties file contains the test cases for remove user member from role and group identities

5.3 **Common Files**

- SMSCommon.java : A class which contains the common methods to invoke Service Manager APIs. These methods are used in IDM test to create datastore, list datastore, update datastore, and delete datastore
- SMSConstants.java : A class which contains the constants that are used for SMSCommon class
- LDAPCommon.java : A class which contains the common methods to execute LDAP commands. These methods are used to check LDAP server up, verify scheme loaded, and load the schema
- IDMCommon.java : A class which contains the common methods to invoke Identity Repository APIs. These methods are used to create, update, search, and delete identities such as user, filtered role, role, agent, and group. They are also used to add and remove the member from group and role
- TestCommon.java : A class which contains the common methods for all modules in QA Test Framework
-

6. Interpreting the IDM automated testing results

Test results and reports are located in directory <REPORT_DIR>/<HOSTNAME>/<EXECUTION_MODE>/<EXECUTION_DATE_TIME> with the following files and directory:

- index.html : open this file in any browser to view overall test result with the following details:
  - <execution mode>-idm : displays the details of test report with a list of test cases and the status. For the failed test case, click on the test name to display the error message and exception stack

    - number of **test** cases passed, failed, or skipped
    - link to testng.xml file <execution mode>-idm-testng.xml in <TEST_HOME>/xml/testng directory
- logs : view this file for general and debug information
- idm.output : contains the IDM test output
- debug : contains the client SDK debug log files

7. Debugging the IDM automated test failures

- Make sure the debug mode on client side set to default value, message. If not, edit <TEST_HOME>/resources/AMClient.properties and change the value of parameter com.iplanet.services.debug.level to message. The client SDK debug logs are in debug directory of test result directory. For information about test result directory, please refer to section 5.
- To change the log level of QA Test from FINE to SERVER, CONFIG, FINER or FINEST so it can display more debug information about the test, edit <TEST_HOME>/resources/Configurator-<hostname>.properties and change the value of parameter log_level.
- To display verbose level of testNG test case execution, go to <TEST_HOME>/xml/testng/<group name>-idm-testng.xml and change the value of attribute verbose in tag <suite> to 5 . For example, <suite name="ff_ds-idm" verbose="5">
- For each action performed by IDM test, it will perform the verification step and display a successful or failed message in the logs file in test result directory. For instance, if the action is create, identity type is user, and user name is testuser, IDM test will create an user, check to make sure user exists, and display a message “user testuser is created successfully”. In case of failure, it will display a message “failed to create user testuser”
- Common errors and solutions

Problems	**Solutions**
LDAP Bind Password for the datastore is not appeared on Admin Console's datastore configuration page that causes the test failed.	Edit <TEST_HOME>/resources/Configurator-<hostname>.properties and make sure the value of encryption_key matches with the value of am.encryption.passwd in FAM server's AMConfig.properties file

Problems	Solutions
Data store attributes or parameters that are defined in SMSGlobalDataStore.properties are empty when executing datastore related operation	Edit SMSGlobalDataStore.properties and make sure the parameter are in this format SMSGlobalDataStore<datastore_number>.<attributes_name>.<index_number>=<value> For example, SMSGlobalDataStore0.datastore_name.0=sun_ds if index number 0 is missing after datastore_name, the value will not be read

8. How to add new testcases?

This section describes the details of IDM classes and the instruction how to add new test cases.

8.1 To add a new test scenarios:

1. IDM test case are specified in the properties files in directory <TEST_HOME>/resources/idm. New test cases should be added to one of these files and appended to the existing test cases. Please refer to section 5 for more information of these properties files.
2. The format and description of each parameters in these properties files are. Note: They also available on the header of each properties file.
   <testcase_name><testcase_number>.<action_parameter_key>.<index>=<value> : for test case group action parameters
   or
   <testcase_name><testcase_number>.<common_parameter_key>=<value> : for test case common parameters.
   For example,
   createIdentitiesTest0.action.0=create
   createIdentitiesTest0.realm=testrealm
3. The common parameters for each IDM test case are:
   - realm : realm name that is used to executed IDM test. Datastore should be created in this realm
   - count : number of action group in the test case
   - description : description of the test case
4. The action parameters for each IDM test case are:
   - action : action to be performed for an identity. The available actions are : create, delete, update, search, addmember, removemember
   - name : identity name
   - type : identity type
   - attributes: identity attributes. For create action, if attributes is not defined, it will use default attributes that created by setDefaultIdAttributes method in IdentitiesTest class

- member_name : member name of an identity
- member_type : member type of an identity
- expected_error_code : expected error code to be compared to actual error code
- expected_error_message : expected error message to be compared to actual error message
- expected_result : expected result to be compared to actual result

5. Each IDM test case in this properties can be constructed by one or more action group. You need to specify common parameter first. The next parameter will be the action you want to perform for the test case and it will be executed in order of the index number. Make sure that number of action group matches with value of count parameter.
   For example, you have one test case to create 2 users, create 1 role, add 1 user member to a role, remove this user member from a role, and delete all identities. The test case can be created as following:

   The total of action is 8 with 3 actions to create two users and one role, 1 action to add one member to a role, 1 action to remove one member from a role, and 3 actions to delete three identities
   <testcase_name><testcase_number>.count=8
   <testcase_name><testcase_number>.description=Sample test case

   The datastore and identities will be created in sub realm testsubrealm
   <testcase_name><testcase_number>.realm=/testsubrealm

   The action 0, 1, and 3 will create two users and one role
   <testcase_name><testcase_number>.action.0=create
   [user attributes]
   <testcase_name><testcase_number>.action.1=create
   [user attributes]
   <testcase_name><testcase_number>.action.2=create
   [role attributes]

   The action 3 and 4 will add member to a role and remove memer from a role
   <testcase_name><testcase_number>.action.3=addmember
   [member attributes]
   <testcase_name><testcase_number>.action.4=removemember
   [member attributes]

   The action 5, 6, and 7 will delete all identities that were created previously
   <testcase_name><testcase_number>.action.5=delete
   [user attributes]
   <testcase_name><testcase_number>.action.6=delete
   [user attributes]
   <testcase_name><testcase_number>.action.7=delete
   [role attributes]
6. After adding a new test case in the properties file, go to <TEST_HOME>/xml/testng and edit all <execution mode>-idm-testng.xml file i.e. ff_ds-idm-testng.xml. Append this entry to each file:

```
    <test name="<test name><new test number>">
        <parameter name="testNum" value="<new test number>"/>
        <parameter name="testName" value="<test name matches with properties file
name>"/>
        <groups>
            <run>
                <include name="<group name>"/>
            </run>
        </groups>
        <classes>
            <class name="com.sun.identity.qatest.idm.IdentitiesTest"/>
        </classes>
        </test>
```

7. IDM test cases properties files will be read and executed by IdentitiesTest class. It will read in the value of realm to perform the identity action. It will read in the value of count and use it to perform the number of actions. Depending on the value of action, it will execute different methods to create, update, search, delete, add members,and remove members.

8.2 **To add a new test case action:**

The new test action should be added to IdentitiesTest.java class with the following steps:

- Create a new method for this new action with the input data read by testIdentities method
- If this new method calls requires to call a new methods that will be used by other methods later, put this method in IDMCommon.java class
- Go to method testIdentities and include this new method in the condition to check for testAction.equals(<new action>)

8.3 **To add new test cases in new class:**

- Create a new class with appropriate class name. The properties files that is used by this test class should locate in <TEST_HOME>/resources/idm directory. Please follow naming conventions described in the openSSO QA Test Document for the class name and properties file name
- The new class should follow setup, test(s) and cleanup procedures. Cleanup should make sure it remove the identities and realm that was created during the setup or test execution
- Appropriate groups should be assigned to these newly added test cases.
- Update all the idm related testNG xml files in <TEST_HOME>/xml/testng directory
- Run idm module with all the tests and make sure all the tests including newly added tests are passing.

9. Global configuration file

The global configuration data are specified in SMSGlobalConfig.properties with the following parameters:

- SMSGlobalConfig.schemalist.<datastore type> : a list of user schema files for specific datastore type. Specify full path name where the file locates
- SMSGlobalConfig.schema_attributes.<datastore type> : a list of schema attributes which are used to check for the existing of user schema.   This list corresponds to the list of user schema files for the same datastore type and it should be in sequence order. For example, if schemalist=file1;file2 and attribute=attr1;attr2, attr1 will be used to check for schema in file1 and attr2 will be used for schema in file2. To skip the schema checking, leave the attribute entry empty or don't specify if the attribute is the last entry in the list.

Note: These parameters are used for LDAPv3 datastore only. The datastore type can be ds, ad, or ldap. For multiple entries, separate them by a “;” character.

10. How to configure the datastore

This section describe the details of the parameters in the datastore configuration file. Datastore configuration parameters are specified in <TEST_HOME>/resources/idm/SMSGlobalDataStoreConfig.properties. This configuration file is used by the method in SMSCommon.java class to create, update, list, and delete datastore. There are 4 types of datastore supported by FAM 8.0 but IDM test works only with Sun directory server datastore.

The format of datastore parameters
SMSGlobalDataStore<datastore_number>.<datastore_parameter_key>.<index>=<value>

For example,
SMSGlobalDataStore0.datstore_name.0=idm_sunds_datastore

There is a parameter datastore-count that does not require the index. This parameter is used to determine number of datastore to be performed the datastore action.

The datastore common parameters are:

- datastore-name : datastore name
- datastore-realm : realm name where datastore is created
- datastore-type : datastore type. The value can be : ds for Sun directory server, ad for Active Directory, ldap for generic ldap, and ff for flat file
- datastore-adminid : LDAP server bind dn. This parameter is used for LDAPv3 datastore only and is used to load the datastore user schema.
- datastore-adminpw : LDAP server bind dn passsword. This parameter is used for LDAPv3 datastore only and is used to load the datastore user schema.

The datastore attributes parameters for LDAPv3 datastore are:
These parameters are as same as datastore attributes and that are used to create and update LDAPv3 datastore. For a list of datastore attributes, see SMSConstants.java.
The minimum parameters should be defined for LDAPv3 datastore are:

- SMSGlobalDatastoreConfig0.sun-idrepo-ldapv3-config-ldap-server.0=LDAP server hostname. This is also used to load the user schema
- SMSGlobalDatastoreConfig0.sun-idrepo-ldapv3-config-ldap-port.0=LDAP server port. This is also used to load the user schema
- SMSGlobalDatastoreConfig0.sun-idrepo-ldapv3-config-authid.0=LDAP bind DN that is configured with the datastore. This value can be the same or different with the value of datastore-adminid
- SMSGlobalDatastoreConfig0.sun-idrepo-ldapv3-config-authpw.0=LDAP bind DN password
- SMSGlobalDatastoreConfig0.sun-idrepo-ldapv3-config-organization_name.0=Organization DN. This value needs to be specified if user management datstore is not the same instance with service management datastore
- SMSGlobalDatastoreConfig0.sun-idrepo-ldapv3-config-psearchbase.0=Persistence search base. This value needs to be specified if user management datstore is not the same instance with service management datastore
- SMSGlobalDatastoreConfig0.sun-idrepo-ldapv3-config-ssl-enabled.0=SSL mode

Other datastore attributes can be added to overwrite the default values. For the attribute with multiple values, make sure to include existing values and new values that can be concatenated with a "|" character. For example, SMSGlobalDatastoreConfig0.sun-idrepo-ldapv3-config-group-attributes.0=cn|description|dn|iplanet-am-group-subscribable|objectclass|uniqueMember where description is a new attribute to be added to group attribute list and others are default values.

11. Acronym

Acronym	Name or Term
FAM	Federated Access Manager
IDM	Identity Repository Management
UM	User Management (datastore)
SM	Service Management (datastore)
ff	Flat file
ds	Directory server
SSL	Secure Socket Layer