



NetIQ® Identity Validator Administrator and User Guide

September 2022

Legal Notice

For information about NetIQ legal notices, disclaimers, warranties, export and other use restrictions, U.S. Government restricted rights, patent policy, and FIPS compliance, see <https://www.netiq.com/company/legal/>.

Copyright (C) 2022 NetIQ Corporation. All rights reserved.

Contents

About this Book and the Library	5
About NetIQ Corporation	7
1 What is Identity Validator	9
2 Planning to Install Validator 2.0	11
System Requirements and Prerequisites for Validator 2.0 Installation.	11
Operating Systems	11
Supported Browsers	11
Prerequisites	12
Installing the Validator 2.0	12
3 Features and Best Practices	13
Features	13
Grouping the Tests.	13
Reordering Tests, Connections, and Actions	13
Monitoring a Running Test	13
Automatic File Backup.	14
Best Practices	14
Test Object Naming	14
Unit Testing vs Integration Testing	14
Using Validator in a Development Group	15
Using Variables and Environments.	15
Using Templates.	16
Accessing Validator from a Remote Workstation	17
Testing Connections.	17
Changing the default values for Retry Interval and Retry Count	17
Avoid Pause Actions	17
4 Test Suite and Test Case	19
Importing Test Suite	19
Creating Test Suite	19
Create New Connection	20
Creating Test Case	23
Setup/Test/Cleanup Sections	23
About Setup	23
About Test	23
About Cleanup	23
Create Test Case.	24
Setup Process.	24
Test	25
Cleanup.	26

5 Connectors	29
LDAP Based Connectors	29
Connection Information	29
Retrieve a Value	30
Create Object	30
Setting dates	31
Deleting Objects	31
Bulk Create Objects	31
Save/Restore Object	31
Troubleshooting Save & Restore	32
Active Directory Connector	32
Assert Password	33
Cannot Set a Password in AD	33
Objects Created in AD are in Disabled State	33
Admin Password Reset	33
Possible Error Codes	33
Creating a Group in AD	34
eDirectory Connector	34
New Connections	34
Structured Attributes	35
Grant or Revoke an Entitlement and Check	36
Possible Error Messages	36
Remote Connector	37
Execute Connector	37
Generic Actions Connector	38
HTTP Connector	41
Authentication	41
Connection URL	41
Header Params	42
JDBC Connector	42
SMTP Connector	44
Text File Connector	45
User App Connector	45
Web Service Listener Connector	47
Static Endpoints	47

About this Book and the Library

The *Administrator and User Guide* provides information on how to install and administer the NetIQ Identity Validator 2.0 product. This book defines terminology and includes implementation scenarios.

Intended Audience

This guide is intended for network administrators.

About NetIQ Corporation

We are a global, enterprise software company, with a focus on the three persistent challenges in your environment: Change, complexity and risk—and how we can help you control them.

Our Viewpoint

Adapting to change and managing complexity and risk are nothing new

In fact, of all the challenges you face, these are perhaps the most prominent variables that deny you the control you need to securely measure, monitor, and manage your physical, virtual, and cloud computing environments.

Enabling critical business services, better and faster

We believe that providing as much control as possible to IT organizations is the only way to enable timelier and cost effective delivery of services. Persistent pressures like change and complexity will only continue to increase as organizations continue to change and the technologies needed to manage them become inherently more complex.

Our Philosophy

Selling intelligent solutions, not just software

In order to provide reliable control, we first make sure we understand the real-world scenarios in which IT organizations like yours operate — day in and day out. That's the only way we can develop practical, intelligent IT solutions that successfully yield proven, measurable results. And that's so much more rewarding than simply selling software.

Driving your success is our passion

We place your success at the heart of how we do business. From product inception to deployment, we understand that you need IT solutions that work well and integrate seamlessly with your existing investments; you need ongoing support and training post-deployment; and you need someone that is truly easy to work with — for a change. Ultimately, when you succeed, we all succeed.

Our Solutions

- ♦ Identity & Access Governance
- ♦ Access Management
- ♦ Security Management
- ♦ Systems & Application Management
- ♦ Workload Management
- ♦ Service Management

Contacting Sales Support

For questions about products, pricing, and capabilities, contact your local partner. If you cannot contact your partner, contact our Sales Support team.

Worldwide:	www.netiq.com/about_netiq/officelocations.asp
United States and Canada:	1-888-323-6768
Email:	info@netiq.com
Web Site:	www.netiq.com

Contacting Technical Support

For specific product issues, contact our Technical Support team.

Worldwide:	www.netiq.com/support/contactinfo.asp
North and South America:	1-713-418-5555
Europe, Middle East, and Africa:	+353 (0) 91-782 677
Email:	support@netiq.com
Web Site:	www.netiq.com/support

Contacting Documentation Support

Our goal is to provide documentation that meets your needs. If you have suggestions for improvements, click **Add Comment** at the bottom of any page in the HTML versions of the documentation posted at www.netiq.com/documentation. You can also email Documentation-Feedback@netiq.com. We value your input and look forward to hearing from you.

Contacting the Online User Community

Qmunity, the NetIQ online community, is a collaborative network connecting you to your peers and NetIQ experts. By providing more immediate information, useful links to helpful resources, and access to NetIQ experts, Qmunity helps ensure you are mastering the knowledge you need to realize the full potential of IT investments upon which you rely. For more information, visit <http://community.netiq.com>.

1 What is Identity Validator

The Validator server is an embedded Jetty HTTP Server and uses Apache Jersey. It is started via scripts from the command line. Once started, the default URL to access Validator is `https://< ip address>:port >/validator/#/login`.

The client app is HTML and Javascript using Angular13. It communicates with the Validator server via REST calls.

2 Planning to Install Validator 2.0

Validator 2.0 is a testing utility, allowing the user to easily connect to various systems and perform operations. It is a Single Page web Application (SPA) powered by an embedded Jetty HTTP Server listening on port 8380. Validator is a self-contained stand-alone product that requires no other software than a supported operating system and browser.

Validator communicates with various target applications (such as Active Directory, eDirectory, JDBC, HTTP and so on) and helps in validating the Identity Manager driver features. Using Validator the user can generate events in one application and validate in other applications

- ♦ [“System Requirements and Prerequisites for Validator 2.0 Installation” on page 11](#)
- ♦ [“Installing the Validator 2.0” on page 12](#)

System Requirements and Prerequisites for Validator 2.0 Installation

For latest in scope features refer:[NetIQ Identity Console 1.4 Release Notes](#)

This section explains the system requirements and prerequisites for installing Validator 2.0.

- ♦ [“Operating Systems” on page 11](#)
- ♦ [“Supported Browsers” on page 11](#)
- ♦ [“Prerequisites” on page 12](#)

Operating Systems

Following are the supported operating systems:

- ♦ Microsoft Windows 10
- ♦ RHEL 9.0 Versions
- ♦ SLES 15. Latest SP
- ♦ OS X Latest Version

Supported Browsers

Following are the supported browser:

- ♦ Chrome (v26 is minimum)
- ♦ Firefox (v20 is minimum)
- ♦ Safari (v6 is minimum)

Prerequisites

Following are the prerequisites for Validator 2.0 installation.

- ♦ The latest Java 1.8 should be installed in your machine and make sure that the Java command is accessible from the console.
- ♦ Get the license file(`license.dat`) from Validator 2.0 Product Management and save the file inside `config` folder of the extracted Validator 2.0 zip.
- ♦ By default Validator 2.0.zip will provide a `validator.keystore` file with a default SSL certificate. The user should create a new SSL certificate and add that certificate to the keystore. Other option is that the user can import an already available certificate to the keystore.

Installing the Validator 2.0

NOTE: To run Validator 2.0, you must have the Java path added in your system variable. Before installing the Validator 2.0, ensure that Java path is added in the system variables.

To install the Identity Validator 2.0, perform the following steps:

- 1 Download the .zip file that contains Validator 2.0 installation files from:
- 2 Unzip the downloaded .zip file, open Validator 2.0 folder.
- 3 In the Validator folder double click on the `create-db_linux.sh` is for linux or mac OS platform, `create-db_win.bat` for Windows platform.

NOTE: Create-db file should be executed only once for initial database creation for users.

- 4 Close the `create-db_linux.sh` or `create-db_win.bat` file.
- 5 In the Validator folder double click on the `start-validator_linux.sh` is for linux or mac OS platform, `start-validator_win.bat` for Windows platform.
- 6 Open any Internet browser and paste the following URL:

`https://localhost:8380/validator`

The Validator 2.0 application is launched.
- 7 On the Validator login page, enter your user name and password.
- 8 Click **Login** button.

3 Features and Best Practices

There are some features and best practices that helps the user to understand how to Setup, Test, and Cleanup. The purpose of these sections is to break up the test and to make them easily readable and maintainable.

The user must make each test stand-alone, so that it does not rely on any other test to pass. This means the test will setup the environment the way it expects, then perform the tests, and then cleanup the environment for the next test.

- ♦ [“Features” on page 13](#)
- ♦ [“Best Practices” on page 14](#)

Features

The features of Validator 2.0 are as follows:

- ♦ [“Grouping the Tests” on page 13](#)
- ♦ [“Reordering Tests, Connections, and Actions” on page 13](#)
- ♦ [“Monitoring a Running Test” on page 13](#)
- ♦ [“Automatic File Backup” on page 14](#)

Grouping the Tests

When creating a new test, you will notice a check-box at the top of the test editor panel labeled as **Group Header**, when selected, it turns your test into a virtual group header and is highlighted in the left panel. Any tests created below this header are considered part of that group.

Test groups allow you to organize your tests into logical groups that can be collapsed or expanded in the list. Also you can quickly select the tests in the group by checking the box in the group header.

You can re-order tests and group headers using drag and drop.

Reordering Tests, Connections, and Actions

Tests, Connections, and Actions can be arranged by dragging them in any order you wish.

Monitoring a Running Test

The Results tab will show a live view of the current log file. To stop the auto-refresh click on the **Stop Auto Refresh** icon. If you want to view the log from the OS, all logs are in the validator\logs folder..

Automatic File Backup

Validator will create an automatic backup when you save or run a test suite file. The files are stored in a **Backup** folder under the current test suite path. You can restore to a previous version of a backup file by clicking the “Restore a previous version” button. All backup files are listed with the date and time in the filename.

Up to 20 backup versions will be saved for each test suite file. To avoid backups from being overwritten during test suite execution (where the file is saved after each variable update), it will only save the file every 15 minutes if variable values are the only items changed in the file.

Best Practices

These are best practices that can help you while using the Validator 2.0 are as follows:

- ♦ [“Test Object Naming” on page 14](#)
- ♦ [“Unit Testing vs Integration Testing” on page 14](#)
- ♦ [“Using Validator in a Development Group” on page 15](#)
- ♦ [“Using Variables and Environments” on page 15](#)
- ♦ [“Using Templates” on page 16](#)
- ♦ [“Accessing Validator from a Remote Workstation” on page 17](#)
- ♦ [“Testing Connections” on page 17](#)
- ♦ [“Changing the default values for Retry Interval and Retry Count” on page 17](#)
- ♦ [“Avoid Pause Actions” on page 17](#)

Test Object Naming

Consider developing a naming pattern for the objects your tests generate. This will make it easier to track and deconflict more than one tester or test suite at a time in your identity system. For example: Name the test user using the row or item number in a test plan, such as tuser -1 and so on. This allows you to identify the user in a trace if there are issues.

It is also recommended to use a *variable* for your object names which could be defined at the top of each test. This allows you to change the test object name easily if necessary. This method also allows you to use a template test to create and delete test users. See *Using Templates* as well as the *Calculate Variable in the Generic Actions Connector*.

Unit Testing vs Integration Testing

In general, consider having different test suites for different use cases. Unit testing of an IDM drivers' functionality is a different use case than integration testing for the end to end solution.

In the Unit test you want to test small pieces of functionality, (small is ill defined in this context and depends on the circumstances).

In Integration testing, you do not need to retest all the unit tests, rather present the start conditions, and test for the resultant end conditions.

Using Validator in a Development Group

When using Validator and sharing test suite files among a group of developers and testers, consider the following tips:

- ♦ Since Validator does not require authentication, limit it to only specific workstations who should have access. See the *Configuration* section for details.
- ♦ Do not allow multiple people to edit and execute the same test suite file simultaneously. When someone makes a change to a test suite file, it overwrites the file on the server. When a test suite file is executed, any variables that receive values will be written to the test suite file on the server. This means that if two people are executing a file at the same time, the results will not be predictable since variables may contain values from the other execution.
- ♦ Use the *Import/Export* feature found on the **Tests** tab. This allows you to export selected tests, and all connections, and all referenced variables in the test suite file, which can then be imported into another test suite file. This allows you to share connections, variables, and tests with others who can then modify them for their own purposes. They can then be merged into a master test suite file by exporting them from various sources and importing them back into a master test suite file.

NOTE: When exporting a test that uses a template, it will not export the template unless the template is selected.

Using Variables and Environments

Variables can act as constants, much like IDM GCV's, and they can also be assigned with new values during test execution. In order to enforce the disciplined use of Variables and eliminate the side-effects of mistyping variable names and accidentally overwriting values, Validator requires that all variables be declared before using them.

By using Variables in your tests, you can make your tests more flexible and easier to maintain. For example, instead of hard-coding an object name in a test, set a variable at the top of the test that contains the test object cn or dn, then reference that variable throughout your test. You can use this technique for attribute names and values as well, thus allowing you to create templates that use your customized set of variables to accomplish repetitive tasks such as deleting and creating test objects. See *Using Templates*.

Converting connection information to variables can be done on the **Connection Editor** page by clicking on the button at the bottom. This allows you to utilize the Variable Environments feature so you can easily change the test to access servers in different environments, Dev, QA, Prod, and so on. For more information see *Environments* section.

Declaring variables can be done in two places; one is at the Test Suite tab > Variables menu, and in the second is at Variable Browser (**V** icon that appears in various areas of Validator). Variables do not need an initial value if you are simply declaring them so they can be assigned values within a test.

Variable values can contain references to other variables.

Example 1: `var1=${var2}${var3}`

Example 2: `idx=2, var1=${var${idx}}`

IMPORTANT: Variables that contain other variables as in example 2, must be assigned its value in the Variables panel. If you assign a complex value using the Set Variables action, it will store a partially resolved value in the variable (e.x. `${idx}` is resolved to “2” as in `var1=${var2}`).

Referencing variables can be done in any text field in Validator. Most places you may want to insert variables that will have a (V) icon somewhere on the page. This icon brings up a variable browser listing all declared variables. If you want to declare a variable in the browser, click the + icon. When you select a variable to use, it will insert a variable reference in the following format: `${variable name}` in the *last text field* you were editing. If there is no V icon on the page where you want to insert a variable reference, you can do it manually by typing in the same variable reference that the variable browser inserts. If you reference a variable that has not been declared or has been deleted, Validator will treat the reference as literal text. If you attempt to assign a value to an undeclared variable (because the variable was deleted at one point), a pop-up will appear during test execution indicating that the variable does not exist. To fix this, simply declare the variable.

Dynamically assigning values to variables is done in connectors that support such actions. Actions typically have a description of “and set variable” in their names. For example, the LDAP connector has an action “Get Object FDN and Set Variable” to place the located object’s DN into a variable.

Variable Environments (also known as Variable Groups) are used to switch between different testing environments. You can have a set of variables for each environment, such as Dev, QA, and so on. As long as you use variables in your connection definitions and in your tests, you can easily switch environments.

To change the active variable environment, go to the Variables management page and select a different environment. The active environment is displayed in the Validator header. e.g. My Tests - DEV

See the help panel on the Manage Variables page for more information.

Using Templates

Templates are like functions that you can access from any test. You can use templates to do common tasks such as cleanup before a test, or create common test users.

Call a template using the Generic Connector > Run Template action.

To enable a template to be generic, use variables to reference specific things such as object names or attribute names and values. Set these variables at the top of each test before calling the template.

You can call a template within a template if required.

Templates also have Setup/Test/Cleanup sections. These get executed in order, so it does not matter which section you place your actions into.

IMPORTANT: When exporting a test that uses a template, it will not automatically export the template. You must select template tests manually in order to export them with the other selected tests.

Accessing Validator from a Remote Workstation

To access Validator from a remote workstation you must modify the configuration file. See the *Configuration* section for details.

You may want to do this if the server running the Validator does not have a supported browser, or to allow several users to access the Validator server.

IMPORTANT: To avoid accidental overwriting of your test files, only one person at a time should be editing and executing a test suite file.

For tips on how to manage test suite files among several developers and testers, see the section: *Using Validator in a Group*.

Testing Connections

When defining a connection it is recommended to test it to assure it is working. This is done by selecting the connection and clicking the **Play** button. If it fails, look at the Results tab for more detailed error messages.

Some connectors like the User App connector, require the SSL certificate to be in the Validator keystore. By testing the connection to the User App server, it will auto-import the certificate thus allowing the connector to connect via ssl (https).

Changing the default values for Retry Interval and Retry Count

Carefully edit the config/Validator_Schema.json file and search for assertRetry then change the defaultValue for the count and interval. It would be wise to make a backup of the file before editing it.

Avoid Pause Actions

It is normal for the process to take some time. Particularly when there are many connected systems involved, and several events have to flow through a series of drivers.

In some environments, certain tests can take longer than expected. Thus you might be tempted to use Pause or Delay actions to help add some delays for event processing.

However, consider instead asserting that some known attribute exists, since each such action has a Retry Count and Retry Interval setting. These actions will complete, once the result is found.

Thus to add a one minute delay you could add an appropriate Assert Attribute or Value exists, and then set the retry count to 10, and the retry interval to 6000 (milliseconds) and then it will try until it succeeds (thus taking the minimum amount of time required) and continue trying for a full minute. This way your tests will likely run faster than simple hard coded pauses.

4 Test Suite and Test Case

This section gives information about the testing and syncing all the user profiles in Identity Manager environment.

- ♦ [“Importing Test Suite” on page 19](#)
- ♦ [“Creating Test Suite” on page 19](#)
- ♦ [“Creating Test Case” on page 23](#)

Importing Test Suite

The user may want to use their own existing test suites that are provided to them.

The following procedure explains how to import a Test Suite:

- 1 On the Test Suite pane either click **Import**.
The **File explorer** tab appears.
- 2 Select the Test Suite that needs to be imported.
- 3 Click imported Test suite.
- 4 Click **Rename** to rename the Test Suite.
- 5 Click **Save**.

Creating Test Suite

The driver development or driver deployment personnel can create and test the scenarios. The user can set up connections with multiple-end applications and automate repetitive tasks. The users can automate the testing of Identity Manager environments.

On the NetIQ Validator **Test Suites** pane you can perform the following:

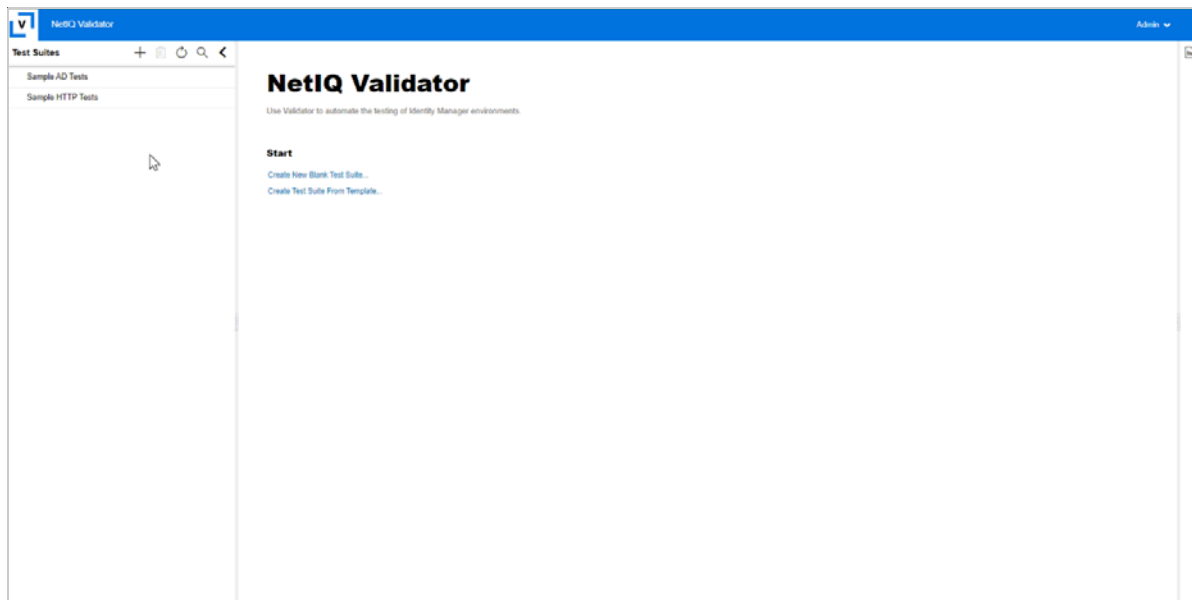
On this tab the user can view the list of Test Suites. Here the user can also create new Test Suite either by selecting existing template or create a new Test Suite. Other activities such as Delete Test suites, View/Modify the details of Test suite are also can be performed.

The following procedure explains how to create a Test Suite:

- 1 On the Test Suite pane either click + or click **Create New Blank Test Suite**.
The **Create Test Suite** screen appears.
- 2 Give a Test Suite name.
- 3 Give a brief Description of the Test Suite.
- 4 Click **Create**.
A new Test Suite is created under the **Test Suite** pane.

After creating the new Test Suite, you can create a new connection to perform a test case.

Figure 4-1 Create the Test Suite



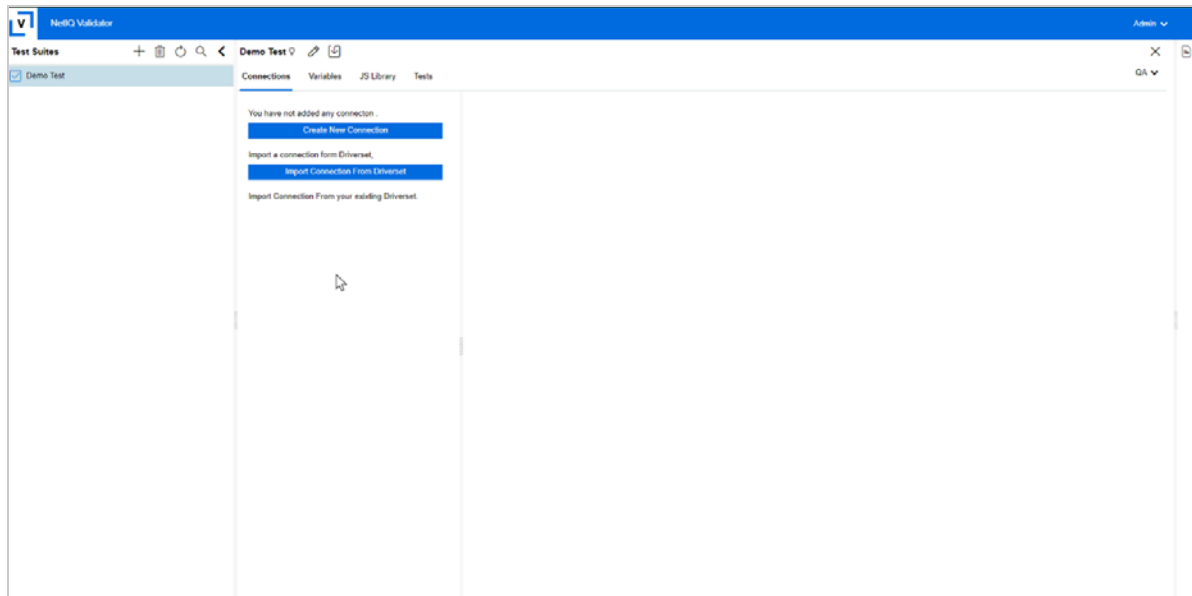
Create New Connection

On this tab the user can view the list of Connections. Here the user can also Add new connections manually or by importing the connectors from Identity Manager. Other activities that can be performed here are: View/Modify the connection information, Delete Connections, and Test Connections.

To create a new connection with the Identity Manager, perform the following:

- 1 Select the Test Suite for which you want to create a new connection.
The **Connections** tab appears.
- 2 Click **Create New Connection**.
- 3 Select the **Connector Type** from the drop-down menu as per your requirement. Fill all the fields that are relevant to that Connector type.
- 4 Click the **Save** button, and then click **Run**.
The **Reports and Logs** displays the Test Connection status.

Figure 4-2 Create New Connection



Adding a Variable

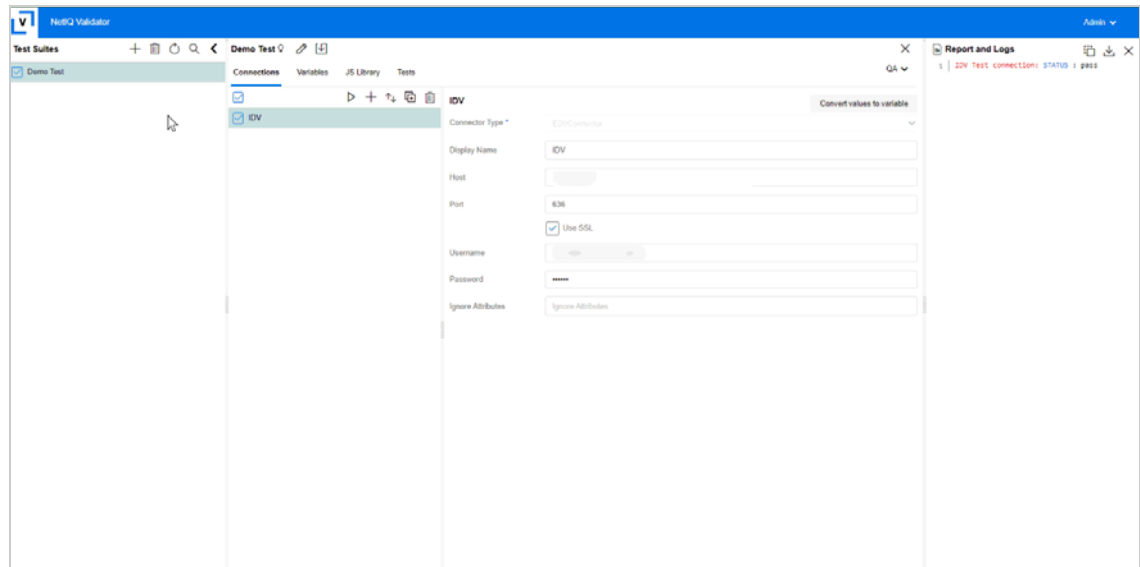
Whenever you want to reuse a particular value or variable, it can be accessed from the variable tab. On the Variable tab you can add a variable, delete a variable, and cleanup a variable. You can also add new **Environments** and add Variables.

Following steps explain how to add Variables:

- 1 Click the Radio button to Select the Environment for which you want to add a Variables.
- 2 Click **Plus** to add Variables.
- 3 Add a Name and Value for the Variable.

We can also insert a value by using a Java Script, Search Directory and Delete the Variables.

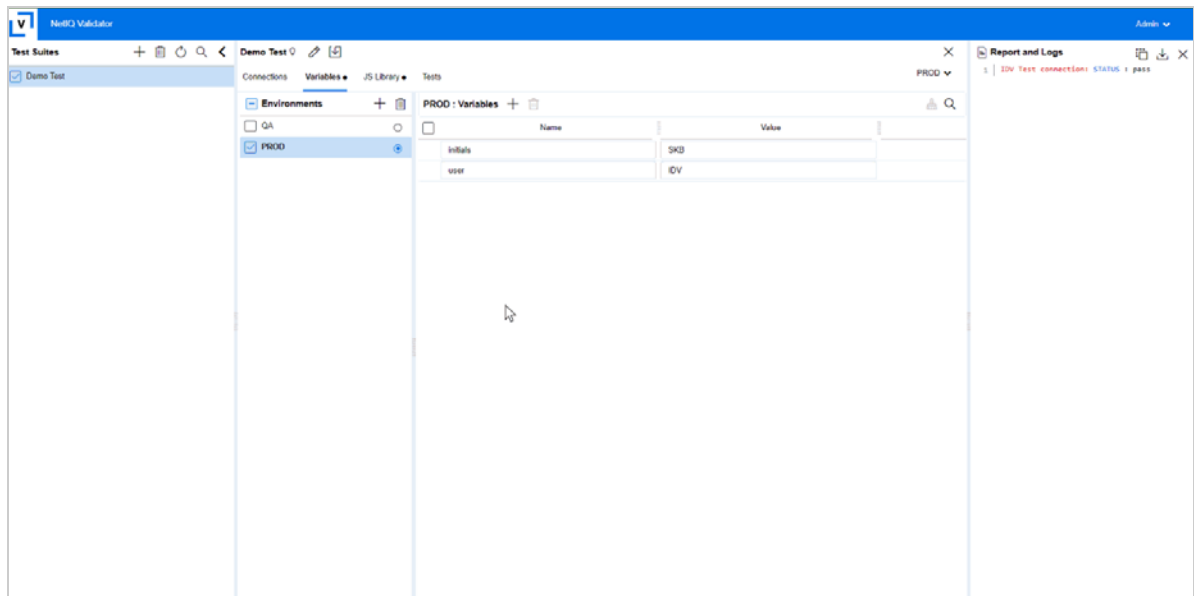
Figure 4-3 Adding a Variable



Java Script Library

On the Java Script Library tab create Java script functions in your test cases. Write the Java script functions and use it anywhere in the test suites. The option to manage Java Script methods in an Editor which will be used in the test suite is available.

Figure 4-4 JS Library



Creating Test Case

Setup/Test/Cleanup Sections

Each test contains three sections; Setup, Test, Cleanup. These sections are executed in order from top to bottom. The purpose of these sections is to logically break up the test to make them more readable and maintainable.

It may be helpful to make each test stand-alone so it doesn't rely on any other test to pass. This means the test will setup the environment the way it expects, then perform the tests, then cleanup the environment so it is clean for the next test. This is the reason for the three test sections described below

About Setup

The user must setup the environment and make sure that everything is in order for the tests to work.

This would include things like deleting the objects used for this test. Ensure that the Assert Object does not exist, so that the delete will succeed. The cleanup section should do this, however a test might fail, and not get to the Cleanup section. Thus as a setup to each test, make sure to clean everything that might be left behind.

If you find that the Setup and Cleanup sections are nearly identical, you can perform these actions in the cleanup section, then use the Generic Actions > Run Cleanup Actions to execute all Cleanup actions.

Also consider placing common cleanup and create tasks in Template tests. Then call the template test (Generic Actions > Run Template). See *Using Templates*.

The setup section can also be used to set any variables referenced later in the test.

About Test

In this section, first create any objects that need to test. Then generate the event that is being tested for, such as changing a password, moving an object, changing a flag attribute, and so on. Then test for the expected results.

Consider placing common tasks such as object creation in a Template.

About Cleanup

Finally, once the test succeeds, cleanup any objects related to the test and reset the directory back as close to starting conditions as possible.

Clean up in all connected systems that might be affected, along with the identity vault.

Create Test Case

On this page the user can perform the activities such as managing Test Case, creating new Test Case, and duplicating Test Case. It is also possible to Import existing Test Case to the test suite. Other activities such as Exporting, deleting, and grouping of Test Cases can also performed.

To create a new test case, perform the following steps:

- 1 Click **Add** to create a new Test Case.

NOTE: There are two options that appear, such as the Test and the Category. The Test option is for adding a Test Case. If you click Category, a folder will be created where you can add all the Test Cases. Categorizing the Test Cases will help in differentiating the Test Cases and identifying them easily.

- 2 Select **Test**

A new Test Case is created.

- 3 Click **New Test**.

- 4 Click **Edit** to change the Test Suite name as per your requirement. Provide a **Name** and **Description**.

Figure 4-5 Test Tab



Setup Process

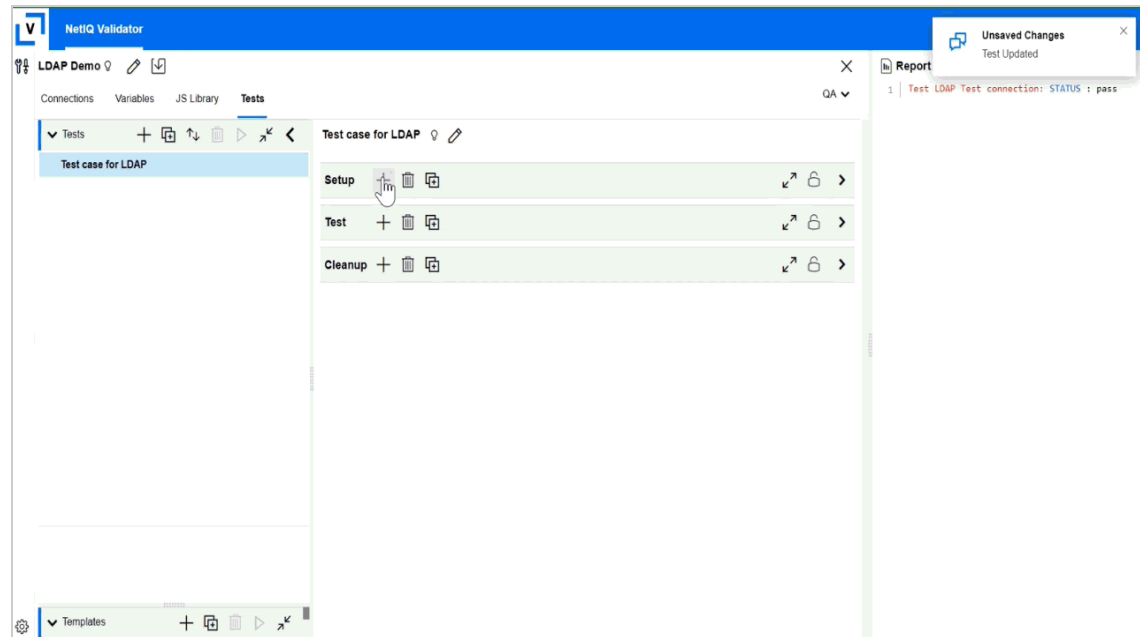
To establish a Setup perform the following steps:

- 1 Click **Add +** icon.
- 2 Click **Arrow >** to take a drop-down.
- 3 Select the **Connection** and **Action**, and then fill all the other details as per your requirements.

- 4 Click **Save** button.
- 5 Click **Arrow >** to minimize the **Setup** tab.

NOTE: The user can perform any number of Test Cases in all the sections such as Setup, Test and Cleanup.

Figure 4-6 Setup Tab

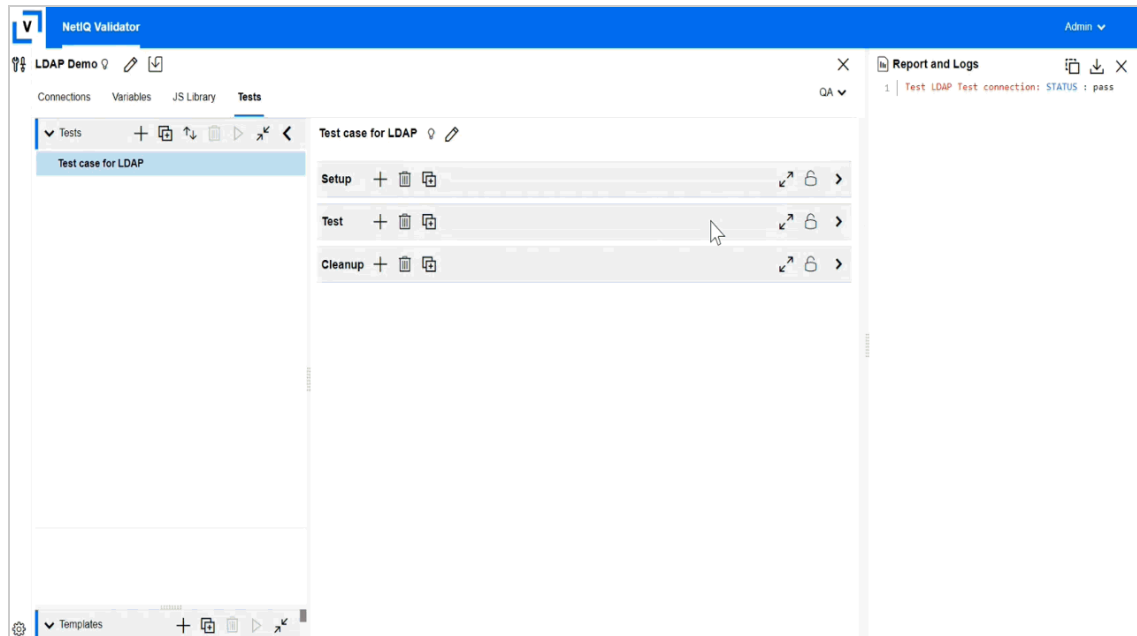


Test

On the **Test** sub-tab you can perform the following actions:

- 1 Click **Plus+** to add a new Test.
- 2 Select the **Connection** and **Action**, and then fill all the other details as per your requirements.
- 3 Click **Save** button.
- 4 Click **Arrow >** to minimize the **Test** tab.

Figure 4-7 Test Sub-tab

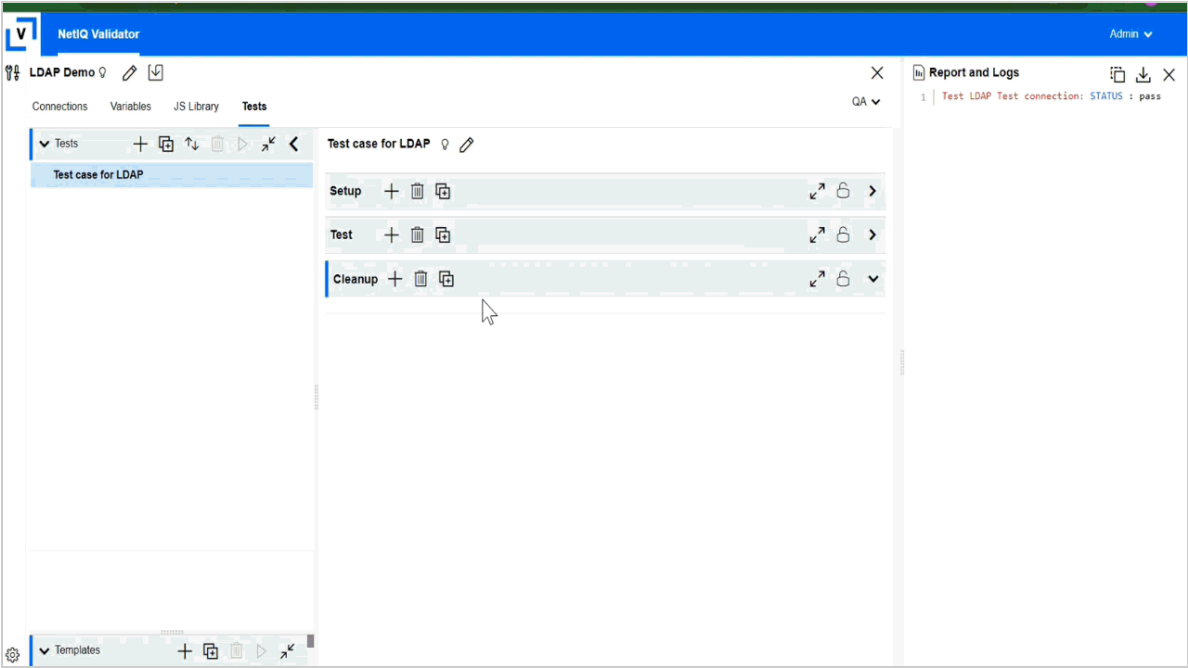


Cleanup

- 1 Click **Add+** to start cleanup.
- 2 Choose **Connections** and **Action**, and then fill all the other details as per your requirements.
- 3 Click **Save**.
- 4 From the **Tests** tab, select the Test, and click **Run**.

The **Reports and Logs** screen will appear. This screen has all the details such as the Object is created in the **Setup**, the **Test** is performing Adding Attributes and Values, the **Cleanup** is deleting the Attributes.

Figure 4-8 Cleanup



5 Connectors

LDAP Based Connectors

LDAP connector helps the user in communicating with LDAP based communication systems. The following connectors use LDAP communication:

- ♦ AD Connector.
- ♦ EDir Connector.
- ♦ LDAP Connector.

Connection Information

Figure 5-1 AD Connector

The screenshot shows a 'New Connection' configuration window. It has a title bar with 'New Connection' and a 'Convert values to variable' button. The form contains the following fields and options:

- Connector Type ***: A dropdown menu with 'AD Connector' selected.
- Display Name**: A text input field with 'New Connection' entered.
- Host**: A text input field with 'Host' entered.
- Port**: A text input field with '389' entered.
- Use SSL**: A checkbox that is currently unchecked.
- Username**: A text input field with 'Username' entered.
- Password**: A text input field with 'Password' entered.
- Ignore Attributes**: A text input field containing a list of LDAP attributes: 'objectGUID, objectSid, dSCorePropagationData, distinguishedName, uSNCreated, uSNChanged, whenChanged, whenCreated, badPasswordTime, badPwdCount, instanceType, las'.

Connector Actions: All the actions that are available in the test/template.

Table 5-1 List of Actions

Action	Description	Example
Get Object FDN and Set Variable	This action will search the tree from a specific Base DN for an object whose attributes have the values you specify. The variable is set to the object FDN. If the object is not found, the variable is simply assigned an empty value, but no error is returned. Note: Be sure to populate the Base DN field or the search may not work on some LDAP servers.	

Action	Description	Example
Get Values and Set Variable	This action will set the variable value to the <i>FIRST</i> value from the specified attribute unless the "Get All Values" box is checked	

Retrieve a Value

Retrieve a value from a directory and use that value in the test.

There are actions that get LDAP values into variables, and those variables can be referenced at any time during test execution. These actions are as follows:

- ♦ *Get Object FDN and Set Variable*. This action will search the tree from a specific Base DN for an object whose attributes have the values you specify. The variable is set to the object FDN. If the object is not found, the variable is simply assigned an empty value, but no error is returned.
Note: Be sure to populate the Base DN field or the search may not work on some LDAP servers.
- ♦ *Get Values and Set Variable*. This action will set the variable value to the *FIRST* value from the specified attribute unless the "Get All Values" box is checked.

"Get All Values" will return all values for the attribute as a Json array that can be referenced in Javascript or the Parse Json action in the Generic connector.

The Parse Json action acts upon the variable as a Json object. For example, to get the first value of a Json array the Json expression would simply be: [0]

To access the variable in Javascript you treat the variable as a Json object. For example, if the variable name is MyVar you would get the first value in the array with: `js:{ ${MyVar}[0] }`

Create Object

Create Object action will Import Fields from the Object link.

If you have an existing object in the directory that is similar to the object you are creating in Validator, you can import all attributes from that object which will automatically populate the Classes and Attributes fields in the Create Object action.

NOTE: If you want to ignore specific attributes during the Import, specify a comma separated list of attribute names (case sensitive) in the connector property *Ignore Attributes*.

There are two import links in this action - you can either browse the tree for an object, or specify an object DN in the Object FDN field.

IMPORTANT: You may need to remove some of the imported classes and attributes since some of them may not be applicable or even writable to the specific directory you are accessing.

Setting dates

Setting dates such as passwordExpirationTime:

Version 1.3 introduced time calculation and conversion actions in the Generic Connector. You can generate LDAP dates for use in LDAP based connectors.

This information is as a general reference about LDAP date formats.

LDAP date format: YYYYMMDDHHMMSSZ

For example: Jan 15, 2011 at 1:05:23 PM is 20110115130523Z for zulu (UTC) time.

To specify time for your own timezone, replace Z with UTC time offset in the form: +|-HHMM where HHMM represents the number of hours and minutes in relation to UTC time.

For example: 20110115130523-0500 is 5 hours behind UTC time which is Eastern Standard Time.

NOTE: Use the Date Calculator in the Generic Actions connector to create and convert dates.

Deleting Objects

Deleting Objects using wild-cards when deleting objects and checking if an object exists

You can use wildcards in the CN value for Delete Objects and Assert Object Exists. For example: cn=testuser*,ou=myou,o=myo would match all users whose CN starts with "testuser". Only * is allowed as a wildcard.

Bulk Create Objects

Bulk Create Objects action allows you to create users in bulk for performance testing. It is similar to the Create Object action but has some additional fields for a starting counter, the number of objects to create, and the delay between each creation.

You can access the current counter value to create unique attribute values, by using the string "CTR" anywhere within the value. For example: JohnsonCTR would result in the value Johnson1, Johnson2, etc.

Save/Restore Object

Save/Restore Object action allows you to take a snapshot of an object and store it in a variable so it can be restored later. The restore is performed by deleting the existing object, then creating a new object based on the saved object.

When saving an object you may wish to ignore certain attributes. These may include system attributes that cannot be written via LDAP to an object, or other attributes you simply wish to ignore. There is a property in the connector configuration called *Ignore Attributes* where you can provide a comma separated list of attributes that will be ignored during an object Import and Save. This attribute names in this list are case sensitive. If this property is left blank, a default list of system attributes is used (mostly for AD objects) to avoid errors during object creation or a restore.

Troubleshooting Save & Restore

If an attribute is saved but cannot be restored, you may see something like the following errors during a restore process:

Error while executing action method saveRestoreObject: java.lang.AssertionError:
com.google.gson.JsonSyntaxException: java.lang.IllegalStateException: Expected BEGIN_OBJECT but was STRING at line 1 column 1

This likely means that an attribute contains binary data that cannot be represented in Json so the Json parser throws an error. This attribute will need to be ignored. We will discuss later how to determine what that attribute may be.

Another possible error is:

Unwilling To Perform (53) Unwilling To Perform LDAPException: Server Message: 0000209A: SvcErr: DSID-031A0FBB, problem 5003 (WILL_NOT_PERFORM), data 0

In this case, the server is indicating that an attribute cannot be written using LDAP.

Determining troublesome attributes may be tricky and is a trial and error process. You can look at the variable containing the saved object by using the Generic Actions Base 64 Decode action. This will allow you to see which attributes were saved, then you can add attributes one at a time to the Ignore Attributes property on the connection until subsequent Save and Restore operations succeed. If an attribute looks like it contains binary data (such as odd characters in the Json) you will want to ignore that attribute as well.

Active Directory Connector

Figure 5-2 New Connection Screen

The screenshot shows a 'New Connection' dialog box. It has a title bar 'New Connection' and a button 'Convert values to variable' in the top right. The form contains the following fields:

- Connector Type *: AD Connector (dropdown menu)
- Display Name: New Connection (text input)
- Host: Host (text input)
- Port: 389 (text input)
- Use SSL: ☐ (checkbox)
- Username: Username (text input)
- Password: Password (text input)
- Ignore Attributes: objectGUID, objectSid, dSCorePropagationData, distinguishedName, uSNChanged, uSNCreated, whenChanged, whenCreated, badPasswordTime, badPwdCount, instanceType, las

- ♦ The DN of the bind user should be in LDAP format. cn=Administrator, cn=Users, dc=domain, dc=com.
- ♦ Host: Whenever the Active Directory is configured, you should provide IP as Host.
- ♦ The default users container in Active Directory is cn=Users, and not ou=Users.

Assert Password

Active Directory has two possible password attributes exposed via LDAP.

1. `userPassword` (as all LDAP systems should)
2. `unicodePwd`

Each AD tree can be configured as to the behavior of these attributes. Please read MSDN articles linked for each.

The Assert password method has been modified to try a bind as the user with the specified password to avoid this problem, and care should be taken to avoid multiple retries as bad password attempts could trigger intruder lockout.

To set the password, you need an SSL connection to AD. In addition, it is best to set the password using the Set Password action as opposed to setting the password during an object create.

Cannot Set a Password in AD

Cannot set a password in AD using Create Object

NOTE: If you cannot set the password in AD using Create Object, then you need an SSL connection to AD to set the password. You cannot set the password during user creation, so use the Set Password action after the user has been created.

Objects Created in AD are in Disabled State

If you try to set `userAccountControl` during object creation, you may get an error “WILL_NOT_PERFORM”.

You must create the user in AD without a password and without setting `userAccountControl`. Once the user is created, use the Set Password action to set the password. This is critical because you cannot enable the account without a password being set on the user.

Then use Replace Attributes and Values to set the appropriate value in “`userAccountControl`”. Set it to a value of 512 for a normal enabled account. See <http://support.microsoft.com/kb/305144> for all possible values for `userAccountControl`.

Admin Password Reset

AD set to require user to reset password on administrative password reset.

This will return an error LDAP 49, subcode 773 ([List of codes](#)) which will be reported in the log, and counts as a login failure, even though it technically indicates the password is correct, but login via LDAP is not allowed until the password is changed.

Possible Error Codes

Possible error codes that might occur on a LDAP bind attempt

- ♦ 525 - user not found

- ♦ 52e - invalid credentials
- ♦ 530 - not permitted to logon at this time
- ♦ 531 - workstation restriction
- ♦ 532 - password expired
- ♦ 533 - account disabled
- ♦ 568 - too many contexts ids (too many group memberships)
- ♦ 701 - account expired
- ♦ 773 - user must reset password

Some of these errors are not technically errors. For example in Assert Password equals, while 525 and 52e are clearly indicators that the password is incorrect, 531 (workstation restriction), 532 (password expired), 533 (account disabled), 701 (account expired), and 773 (user must reset password) only return if the password is correct, and thus in the context of an Assert Password test, mean the password is actually correct.

Creating a Group in AD

The following procedure describes how to create a group in AD.

- 1 Use the “create object” action.
- 2 Enter these two classes:
 - ♦ objectclass=Top
 - ♦ objectclass=group
- 3 Enter these attributes:
 - ♦ cn:MyGroup
 - ♦ dn:cn=MyGroup,ou=groups,dc=whatever,dc=com
 - ♦ sAMAccountName:MyGroup
 - ♦ name:MyGroup

NOTE: When handling CN’s with commas, or specifying the DN, escape the comma with a backslash. For example: cn=Jones\, Fred, ou=user, dc=company, dc=com

eDirectory Connector

New Connections

When creating an eDir connector, there is an option to import connections from any IDM driver sets configured on the eDir server. This button will not work until you must have a good connection to the eDir server. Once the connection tests correctly, clicking this button will import all connection information from all driver sets configured on the server. Passwords cannot be imported. Any unwanted connections created by this feature can be deleted.

Figure 5-3 New Connection

The DN of the bind user should be in LDAP format. cn=admin, ou=system, o=sa

NOTE: If you switch from clear text to SSL via the check box, then change the Port to 636.

Structured Attributes

Path syntax attributes are accessed using the Association/Entitlement/Path actions. The following explanation is provided to help you to understand the syntax. DirXML-Associations, DirXML-EntitlementRef are both Path syntax.

NOTE: For Roles and Resources, use the UserApp Connector.

Path syntax has three components, and in this connector are exposed in LDAP format of volume#nameSpace#path

volume:

DN value in LDAP syntax like cn=DriverName,cn=DriverSet,ou=OrgUnit,o=Org

nameSpace

32 bit integer (usually 0-4 for DirXML-Associations or 0-1 for DirXML-EntitlementRef)

path

A case insensitive string ([Possible DirXML-Association values](#), DirXML-EntitlementRef will be an XML nodeset of varying content).

For DirXML-Entitlement refer:

Use an Assert attribute value that contains the following:

cn=EntitlementName,cn=DriverName,cn=DriverSet,ou=OrgUnit,o=Org#1#

For DirXML-Associations refer:

Use the regex enabled Assert Attribute Value that contains:

cn=DriverName,cn=DriverSet,ou=OrgUnit,o=Org#[0-9].#.+

NOTE: Where [0-9] means 1 to many instances of a number for 0 to 9.

Grant or Revoke an Entitlement and Check

We don't have an "entitlement" action yet but you can still do it by adding the correct attributes.

You'll want to tweak this for your specific entitlement but the steps are more or less:

Make sure DirXML-EntitlementRecipient is added as an objectClass.

To check for an entitlement grant/revoke check the attribute "DirXML-EntitlementRef" for the appropriate value.

To grant an Entitlement:

DirXML-EntitlementRef attribute equals: "cn=TestEntitlement,cn=Loopback Driver,cn=driverset1,o=system#1#<ref><src>RBE</src><id>system\driverset1\Entitlement Policies\Test</id><param>TestValue</param></ref>"

To revoke an Entitlement:

DirXML-EntitlementRef attribute equals:cn=TestEntitlement,cn=Loopback Driver,cn=driverset1,o=system#0#<ref><src>RBE</src><id>system\driverset1\Entitlement Policies\Test</id><param>TestValue</param></ref>

After running the command, check for entitlements such as: work flow, adding role, and so on.

NOTE: If you want validator to artificially add an entitlement you do it the same way by adding an attribute with the specific value. This is same to revoke or to add.

Possible Error Messages

Object Class Violation (-628)

If you attempt to access an attribute that is not available for the object, you will receive Object Class Violation (-628) error.

Do not forget to specify the object class for the object during creation, and any auxiliary classes as needed modifications.

Illegal attribute (-608)

This could occur if you have a single valued attribute and you are trying to add a second value

Remote Connector

- ♦ This connector is a generic connector allowing you to retrieve a file from, send a file to, or execute a command on a remote system
- ♦ The connector defaults to connecting via SSH2, except in the case of the SendFile (sftp) action
- ♦ See the Remote Connector Starter test for samples

Execute Connector

- ♦ Commands are executed as it would be from the system command line (so Windows rules on a Windows host, and Linux rules on a Linux system). These commands are executed on the Validator server.

- ♦ Executing commands Windows

If the command is a Windows executable, you simply call it using the full path.

e.g. `C:\Windows\System32\write.`

If the command is a batch file, or if the command is a command processor (`cmd.exe`) built-in command such as `dir`, `copy`, etc., you need to use the following syntax:

```
cmd /c <command>
```

e.g. `cmd /c dir c:\`

- ♦ Executing commands on Linux or Mac

For Linux and Mac, you enter the command exactly as you would at the command-line.

- ♦ Issues with some commands

Although the Execute Connector documentation says “the command is executed as it would be from the local system command line”, that is only 90% accurate. The back end engine runs in a JVM, meaning, like all Java, the engine is in it’s own machine and OS (JVM truly means Java Virtual *Machine*). Java has limited access to the host OS. The bottom line is normal shell command-line stuff like piping(`|`), redirection(`>`), spaces in names, and memory between command don’t work.

If your command doesn’t work,

1. Make sure the command works on the command line on the Validator server
2. Make sure the command doesn’t need a specific current directory, or you may need to write a script to perform a `CD` and execute the command.
3. Put the command results in a variable, and read it for any error message
4. Put the command in a shell or batch file. Then it should behave exactly like the command line because it is loading the shell.

Generic Actions Connector

- ♦ The Generic Actions Connector provides some general purpose tools that can be used within tests. These include being able to evaluate XPath and Javascript expressions, as well as a date calculator and converter.

- ♦ See the Generic Connector Starter test for samples

- ♦ *Javascript* - Executing Javascript in an action consists of two parts:

1. Javascript Library - specify your own Javascript functions (Optional)

Under the *Test Suite* tab, create your own Javascript Library with any functions you wish to use in your tests.

If you wish to gather a set of standard Javascript functions, place them in a file in the *jslib* folder, then use the *Insert Javascript Library* button to prepend this library into the current window.

Debugging Javascript -

- ♦ To assist in debugging script, use `println()` to output messages to the Validator console.

Using custom Java libraries -

- a. Place any custom Java jar files into the lib/ext folder
- b. *Restart* the Validator server.
- c. Use the `importPackage()` function at the top of the Library. e.g.
`importPackage(org.hasypt.util.text);`
- d. Use your custom classes in that package.

Examples:

For static methods:

```
js:{MyCustomClass.myMethod( )}
```

For non-static methods:

In the Javascript Library create a function to instantiate the class and call your custom method:

```
function myFunc() {  
  
var myObj = new MyCustomClass();  
  
return myObj.myMethod();  
}
```

Then use the directive to call the library function:

```
js:{myFunc( )}
```

2. Use a Javascript directive in any text field in an action.

A JavaScript directive is in the form: `js:{expression}` and can be used in any text field within an action.

IMPORTANT NOTE: Javascript directives cannot be used in Variable values defined in the Variables editor, nor can they be used when defining connection properties. They can only be used in test Actions.

The result value of the directive will replace the directive itself. In other words if you do the following in an attribute value field:

```
js:{"${User Last Name}".substring(0,2)}
```

it will first replace the variable "User Last Name" with the value of the variable.

```
js:("Johnson".substring(0,2))
```

It will then evaluate the Javascript expression and replace the directive with the result, which is the first two characters of the variable value.

Jo

IMPORTANT: The variable replacement process in the parameters to a Javascript function is similar to a text replace function. This means you must use quotes around the variable tokens if you need them treated as strings as shown in the `${User Last Name}` example above. If the variable value is a string and quotes are not used a typical error might be: *"missing ; before statement"*

Example:

```
js:{ myFunc("${User Last Name}") }
```

Quotation mark (") need not be used if the value needs to be treated as a number or a Json object.

Examples:

```
js:{ myFunc(${Json Object}) }  
js:{ myFunc(${My Number}) }
```

- ♦ **JSON** - Parsing JSON variables can be done in Javascript (see the Javascript section above), or by using the Parse JSON and Set Variable action in the Generic Actions connector.

The JSON Expression is Javascript syntax to access data in a JSON variable.

For example, consider the following JSON which is in a variable.:

```
{  
  "person": {  
    "first_name": "John",  
    "last_name": "Smith"  
  }  
}
```

To access the person's first name the JSON expression would be:

`person.first_name`

Or

```
person["first_name"]
```

- ♦ *XPath* - You can use XPath to parse XML documents such as entitlement values, etc.

For example, you have a variable that contains an XML document and you want to retrieve the text value of the “name” element somewhere within the document. The XPath expression would be: `//name`

NOTE: The leading `//` means to traverse the document until you find an element called “name”.

If you want to retrieve an attribute of an element (e.g. age), the XPath statement would be something like: `//person/@age`

Default Namespace Issues

When an XML document has a default namespace (`xmlns=...`), XPath statements in the Linux version will not work if you are referencing nodes within that default namespace. This issue will be addressed in 1.4.2, but a workaround is available on 1.4.1.

Consider the following XML:

```
<mydoc xmlns="http://mydefaultnamespace">
  <person>
    <name>Bob</name>
  </person>
</mydoc>
```

Note that a default namespace (a namespace without a suffix) has been defined in the “mydoc” element and affects all elements below mydoc. One would expect the following XPath expression to access “name”:

```
//person/name
```

This will work on Windows but will not work on Linux because of differences in the Java XML libraries. To make this work on Linux you must modify your XPath statement to the following:

```
//*[local-name()="person"]/*[local-name()="name"]
```

This statement forces the XPath processor to examine the element name without the considering the namespace.

- ♦ *Calculate Variable* - Append and manipulate a counter at the end of a variable

Oftentimes you may want to increment a counter at the end of a username to create unique CN's for objects. This action allows you to do just that.

Let's say you have a username of “TestUser” and you want a counter appended to it. You would use the Calculate Variable action and provide a counter *Value* of 1. The first time it is called it will return the value of “TestUser1”. The next time you call it, the return value would be “TestUser2”, etc. You can provide a positive or negative Value to add or subtract to/from the appended number.

- ♦ *Calculate Date* - Provides a date calculator and date format converter

This action allows you to get today's date or use a variable that contains a date, perform math on it (add or subtract minutes, hours, days, months, etc.) then place the result in a variable in any form you wish.

See the Starter Test for the Generic Actions connector for examples.

HTTP Connector

- ♦ Connector purpose

You can use this connector to test REST or SOAP services. It is a generic connector where you can specify a URL to send data (query string, Form data, or Payload Data) and get a response. You can then compare the response with any string that would indicate success.

- ♦ Authentication

- ♦ Basic Auth

If you provide values for the *Basic Auth User* and *Password* when setting up the connection, two auth headers will be sent with the request.

- ♦ Authorization: Basic <base64 encoded username:password>
 - ♦ RESTAuthorization: <base64 encoded username:password>

- ♦ OAuth2 (limited support)

If you provide a value for the *Access Token URL*, an OAuth2 token will be retrieved from the specified URL before the request, then the following auth header will be sent with the request.

- ♦ Authorization: Bearer <access token>

Request Params are used to configure the OAuth2 authorization. Supported values are:

- ♦ grant_type = password or client_credentials
 - ♦ client_id = provided by the OAuth2 IDP when registering the client.
 - ♦ client_secret = the secret associated with the client_id

When the “password” grant type is used, *Username* and *Password* are required.

Hover over the field labels for additional help.

- ♦ Is there a Listener to receive data from the Subscriber channel of the SOAP driver?

Currently the connector does not provide a listener.

You may consider using SoapUI which does provide a listener.

- ♦ Will the connector work with RBPM REST calls?

Yes. See the following explanation of possible issues that may be encountered.

Authentication

Authentication is done using the RESTAuthorization header which is supplied automatically by the HTTP connector. Simply supply the username (*the username only, NOT an LDAP DN*) and the password in the connection properties. The specified user must be in the user container as specified in the UserApp setup.

Connection URL

Example: <http://localhost:8180/RIS/>

Header Params

In many RBPM REST calls it requires a header value of:

Accept: application/json

otherwise it will return an error: 415 unsupported media type.

- ♦ What is the URL suffix field for?

The value of this field is appended to the URL specified in the connection. With it you can add whatever URL suffix is required for the specific test, including a query string.

NOTE: No part of the URL is encoded automatically. If the URL or Suffix do not work, you may need to manually encode them. (See http://www.w3schools.com/tags/ref_urlencode.asp One item not mentioned in this guide is that you may replace spaces with a + sign if the space is in the query string.)

- ♦ How do I specify header data?

Simply specify the key/value pairs in the Header Data table within the action.

- ♦ Specifying form data

Simply specify the key/value pairs in the Form Data table within the action. Note this is only applicable for the POST method action.

- ♦ Can my HTTP call use all three; header data, form data, and normal payload raw data?

Form data and payload raw data are mutually exclusive and cannot be used together. If both are included in an action, the raw data will be ignored.

- ♦ Using with SOAP Web Services

Typically SOAP is just a matter of sending the right SOAP document (XML) as raw payload data. Authentication can be tricky and may not be possible in this connector depending on the SOAP service you are using. But give it a try. Paste your SOAP doc in the Raw Data field.

- ♦ Expected Results field

If you leave it blank the action will always succeed (since it always matches nothing). You can specify any string in this field that may be returned from the server and the action will succeed. If you use RegEx, you can enter a regular expression in the Expected Results tab.

- ♦ Can I start a UserApp workflow using the HTTP connector?

You can, but you will want to try the User App connector first to be sure it will fit your needs. If you need to do it manually, you can use the RBPM REST services (discussed in an earlier paragraph) or see *Appendix A* for information that will help.

JDBC Connector

The following procedure explains how complete migration to the JDBC Connector. This is a mandatory step.

IMPORTANT: Version 2.0 has removed the built-in connectors for MySQL and Oracle 10G because of licensing issues. If you have existing test suites that use these two connectors, when opening the test suite the migration process will modify it to use the standard JDBC connector.

1. Download the appropriate JDBC Driver jar (MySQL or Oracle10G) and copy it into your Validator install directory (lib/ext). See information below for assistance.
2. Restart the Validator server.
3. Within Validator, edit the connector properties for the former MySQL or Oracle connector. Fill in the Driver Class field with the appropriate value for the JDBC Driver. See information below for assistance.

JDBC Driver Information for MySQL and Oracle 10G

MySQL

- ♦ **JDBC Driver Download:** <https://dev.mysql.com/downloads/connector/j/5.0.html>
- ♦ *Driver Class:* com.mysql.jdbc.Driver
- ♦ *Sample Connection String:* jdbc:mysql://hostname:3306/SampleDatabase

Oracle 10G

- ♦ *JDBC Driver Download:* <http://www.oracle.com/technetwork/apps-tech/jdbc-10201-088211.html>
- ♦ *Driver Class:* oracle.jdbc.driver.OracleDriver
- ♦ *Sample Connection String:* jdbc:oracle:thin:@hostname:port:sid
- ♦ Using the “Get Column Value and Set Variable” action

You must use a SELECT statement in the SQL Statement field that returns a single column. Validator will use the value from the first row returned. If your SELECT statement returns multiple columns and/or rows, Validator will use the value in the first column of the first row.

- ♦ Configure the JDBC Connector to work with Oracle

The following steps will work with any JDBC ‘thin’ client - on the Builder tab:

1. Copy the correct jdbc thin client jar file to the lib/ext directory. If it has a .zip extension, rename it to .jar.
2. Create a new Connector by clicking the Add (plus sign) icon in the Connections Group
3. Select ‘JDBC Connector’ from the Connector Type drop-down
4. In the Connection Editor Section:
 - a. Provide a Display Name for the Connector
 - b. In the ‘Proxy User’ and ‘Proxy Pwd’ fields, provide the user name and password of the database account to be used for authentication
 - c. In the Driver field, provide the class name of the JDBC driver (Ex: ‘oracle.jdbc.driver.OracleDriver’)
 - d. In the Url field, provide the jdbc connection string (Ex: ‘jdbc:oracle:thin:@172.17.2.146:1521:ORCL10G’)
5. In the Connections Group, check the checkbox to the left of the new connection, and click the Test (play button) icon to test the new connection

- ♦ Using a 3rd party JDBC Driver

Place the JDBC Driver JAR in the `lib/ext` directory. If it has a `.zip` extension, rename it to `.jar`.

Common JDBC Drivers:

SQL Server (JTDS Driver option)

- ♦ *JDBC Driver Download:* <https://sourceforge.net/projects/jtds/files/>
- ♦ *Driver Class:* `net.sourceforge.jtds.jdbc.Driver`
- ♦ *Sample Connection String:* `jdbc:jtds:sqlserver://server:1470/SampleDatabase`

SQL Server (MS SQL Server Driver option)

- ♦ *JDBC Driver Download:* <http://msdn.microsoft.com/en-us/sqlserver/aa937724>
- ♦ *Driver Class:* `com.microsoft.sqlserver.jdbc.SQLServerDriver`
- ♦ *Sample Connection String:* `jdbc:sqlserver://server:1470;databaseName=SampleDatabase;`

MySQL

- ♦ *JDBC Driver Download:* <https://dev.mysql.com/downloads/connector/j/5.0.html>
- ♦ *Driver Class:* `com.mysql.jdbc.Driver`
- ♦ *Sample Connection String:* `jdbc:mysql://hostname:3306/SampleDatabase`

Oracle 10G

- ♦ *JDBC Driver Download:* <http://www.oracle.com/technetwork/apps-tech/jdbc-10201-088211.html>
- ♦ *Driver Class:* `oracle.jdbc.driver.OracleDriver`
- ♦ *Sample Connection String:* `jdbc:oracle:thin:@hostname:port:sid`

SMTP Connector

- ♦ Connector Purpose

The SMTP Connector has two purposes:

1. An SMTP Listener (Server) that will receive SMTP emails and save the emails to the file system. You can then assert that an email was received.
2. Send an SMTP email to any SMTP server. You could use this to send an email at the conclusion of a long running test, or send an email to an external system that kicks off a process when an email is received.

- ♦ A typical use case for the listener would be when you need to check if a target system, such as a workflow, is sending emails.

For example: You configure the workflow to send emails to the SMTP Listener in Validator. In a test, you would start the SMTP server then launch a workflow that sends an email to the approver. You then assert that the email is received before approving the workflow. Finally, the SMTP server is stopped at the end of the test.

- ♦ The SMTP Server starts on port 25 by default. This port is configurable in the connector properties in the case that your platform will not allow port 25 to be used by a non-root process. If you need to change the port yet still need to allow access from clients that require port 25, you can configure port forwarding on your platform.

For example, you start the Validator SMTP Server on port 1079, then on SuSE you could use iptables as follows to forward port 25 traffic to port 1079:

WARNING: THESE DO NOT WORK CURRENTLY

```
iptables -t nat -A PREROUTING -p tcp --dport 25 -j REDIRECT --to-port 1079
```

```
iptables -t nat -I OUTPUT -p tcp -d 127.0.0.1 --dport 25 -j REDIRECT --to-ports 1079
```

To list entries in the table:

```
iptables -t nat --line-numbers -n -L
```

Text File Connector

- ♦ The files have to be local to the server where Validator is installed (though you could point at a network share for the file).
- ♦ File name vs File Pattern
Only select one.
- ♦ Either an explicit single file (File Name) or
- ♦ Regular Expression pattern in File Pattern and all files that match that pattern will be considered.

Why are there two ways to access files; via a known filename and via a file pattern?

Oftentimes you won't know the filename in a directory, like if it has a timestamp in the filename for example. In this case you don't specify a filename but instead specify a filename pattern (a regex expression) that will match the files you need to access.

TIP: Hover over the fields when you setup a TextFile connection and it will give you more information.

User App Connector

- ♦ UserApp and https

When you setup the connection you specify https://server:port/IDMProv in the URL.

The Test Connection (Play button in the Connections panel) will test the connection and automatically import the https certificate into the Validator keystore.

- ♦ The Object Browser buttons and the Import Links for form fields require an eDir connection to the IDM server in order to read information from the UserApp driver object.
- ♦ Understanding the error: java.lang.RuntimeException:

When attempting workflow operations you may see the error:

java.lang.RuntimeException: unable to read exception information from stream while executing action method approveRequest

This error is returned by the UserApp API. It is typically returned when you are specifying a piece of data for the approval form that is incorrect, or you don't specify a piece of data that the approval form requires. It can also be returned if the workflow DN is not correct.

To get a more informative error, do the following:

1. Login to UserApp as the user app admin
2. Administration -> Logging -> Change log level of all above logs (set to Debug)
3. Submit

Then run your test again and look in the User App log file:

IdentityManager/rbpm/jboss/server/IDMProv/log/server.log

Look for something like:

APPROVE failed com.novell.soa.af.impl.soap.AdminException={_Reason=Data item [reason] is not defined for activity or provisioning request}

In this case, an attribute of "reason" was included in the Approve Request action in Validator, but "reason" was not on the approval form.

- ♦ Other reasons for: java.lang.RuntimeException (bad content)

You may see the following error:

Error while executing action method startWorkflow:
java.lang.AssertionError: java.lang.RuntimeException:
java.io.IOException: bad content type = text/html; charset=iso-8859-1;
Received Content:

when attempting to start a workflow. This can be caused by an invalid URL to the UserApp server.

- ♦ Other reasons for: java.lang.RuntimeException

You may have a field in the approval form, but if you specify that field name as an "attribute" in the Approve Request action in Validator, you receive the RuntimeException.

Try specifying a default value for the field in the approval activity in the workflow as follows:

1. In Designer, click on the approval activity in the workflow
 2. Click the Data Item Mapping tab -> Pre Activity
 3. Specify two single quotes (apostrophes) for the Source Expression of the Target Form Field that is causing the error. This gives the field an initial empty value which allows the User App api to recognize the field.
- ♦ Starting a workflow and immediately trying to approve it may result in an error:
No task for this process/user while executing action method approveRequest.
The approval tasks allow you to specify a retry count which will allow you to retry until the task becomes available.
If you are trying another action that does not have a retry counter, use a wait action to delay for a few seconds.
 - ♦ This connector allows you to grant and revoke and assert Roles and Resources. You can also do this manually using the eDir connector (Path syntax actions) if the actions in this connector are not sufficient.

Web Service Listener Connector

This Web Service Listener Connector provides a listener to allow an external client to connect to it and POST/PUT information which will later be retrieved by a GET. The only configuration for the connector is a port number. Actions are provided to start and stop the listener, as well as configure extended endpoint properties .

Once the listener has been started, any POST or PUT to any endpoint will create a GET endpoint with the same URL in the listener. This GET endpoint will return the data that was provided in the POST or PUT. In this manner, you can validate the data posted to the listener. Use the HTTP Connector to perform the GET and validate the data.

For example:

An external client does a POST on `http://server:port/myendpoint` with a JSON payload. The listener will then create a GET endpoint of `/myendpoint` and will return the same JSON payload provided by the POST. Use the HTTP connector to perform the GET on `http://localhost:port/myendpoint` and validate the data.

Using the Configuration action, you can set additional endpoint properties such as return data for subsequent POST/PUT or GET requests, Basic Auth, content type of the return data, and return codes. This additional configuration is optional. Hover over any label in this action for assistance.

If the client performs a DELETE on an endpoint, that endpoint will be removed.

Once the listener has been stopped, all endpoints are removed.

Static Endpoints

- ♦ Test Connection: GET `http://server:port/wsconntest`
- ♦ Shutdown listener: GET `http://server:port/shutdown`
- ♦ List Configured Endpoints: GET `http://server:port/wslist`

