



NetIQ® Identity Validator Validator 2.0 User Guide

October 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	7
About NetIQ Corporation	9
1 Identity Validator 2.0 Overview	11
How Identity Validator 2.0 Solves Business Challenges	11
Features of Identity Validator 2.0	12
Identity Validator 2.0 Architecture	12
2 Planning to Install Identity Validator 2.0	15
Operating Systems	15
Supported Browsers	15
Prerequisites	15
3 Installing Identity Validator 2.0	17
Installing Identity Validator 2.0 on Windows	17
Installing Identity Validator 2.0 on Linux or Mac OS	18
4 Test Suites	21
Importing Test Suite	21
Creating Test Suite	22
Create New Connection.	23
Adding Environment and Variable	24
Java Script Library	25
Importing Test Suite from Template	26
5 Test Cases	29
Creating Test Case	29
About Setup	29
About Test	29
About Cleanup	30
Create New Test Case	30
Setup Process.	31
Test	31
Cleanup.	32
Grouping the Tests.	33
Reordering Tests, Connections, and Actions	33
Monitoring a Running Test	33
Creating Template	33
6 Scheduler	35
Notifier Configuration.	35

Create New Schedule	36
Run a Schedule	37
Edit a Schedule	38
Delete a Schedule	38

7 Connectors 41

Active Directory Connector	41
Assert Password	41
Set a Password	42
Reset Password on Administrative Setup	42
Create Group in AD	42
Objects Created in AD are in Disabled State	42
Possible Error Codes	43
eDirectory Connector	43
Structured Attributes	43
Grant or Revoke and Check an Entitlement	44
Possible Error Messages	45
Execute Connector	45
Generic Actions Connector	46
HTTP Connector	48
Authentication	49
Header Params	49
JDBC Connector	50
LDAP Based Connectors	52
Retrieve a Value	52
Create Object	52
Setting dates	53
Deleting Objects	53
Bulk Create Objects	53
Save or Restore Object	54
Troubleshooting Save and Restore	54
Remote Connector	54
SMTP Connector	55
Text File Connector	55
User App Connector	56
Web Service Listener Connector	57
Static Endpoints	57

8 Best Practices and Common Questions 59

Saving Test Suite Changes	59
Testing Connections	59
Naming Variable while Executing Test	60
Using Variables and Environments	60
Using Templates	61
Changing the Default Values for Retry Interval and Retry Count	61
Avoid the Pause Actions	61
Running Unit Tests against Integration Tests	62
Using Identity Validator 2.0 in a Development Group	62
Accessing Identity Validator 2.0 from a Remote Workstation	62
Changing Password	63

9 Troubleshooting	65
Unable to Log in to Identity Validator 2.0 After Sign-out.	65

About this Book and the Library

The *NetIQ Identity Validator 2.0 User Guide* provides information on how to install and administer the 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_netiql/office/locations.asp (https://www.microfocus.com/en-us/contact?utm_source=utility_nav)
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 (https://www.microfocus.com/en-us/support)
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 Identity Validator 2.0 Overview

Identity Validator 2.0 is a testing tool for your Identity Management solutions. It is a modernized administration application that is used for testing the Identity Manager drivers. Identity Validator 2.0 communicates with various target applications such as Active Directory, eDirectory, JDBC, HTTP. Using the Identity Validator 2.0 tool, users can generate events in one application and validate them in other applications. Identity Validator 2.0 helps in validating the Identity Management drivers by automating the tasks. This helps the user in eliminating redundant tasks and speeds up the Identity Manager solution deployment.

- ♦ [“How Identity Validator 2.0 Solves Business Challenges” on page 11](#)
- ♦ [“Features of Identity Validator 2.0” on page 12](#)
- ♦ [“Identity Validator 2.0 Architecture” on page 12](#)

How Identity Validator 2.0 Solves Business Challenges

Most organizations have their identities stored on multiple systems. In this case, managing identities and monitoring user activity in physical and virtual environments are important.

The Identity Manager helps in synchronizing the identities from one system to another systems. While creating or deploying an Identity Manager driver you need to verify if the synchronization of identities are happening correctly or not. It requires lot of time and effort to connect to all the required systems separately and verify manually if the synchronization worked for each test case. For example, a customer with 30-40 driver applications, or sometimes more, can easily spend hours to confirm the synchronization.

Identity Validator 2.0 helps the user in testing by connecting to multiple systems and verifying the status of identities. Once the test cases are configured, you can run them for various deployment environments. Identity Validator 2.0 is a preferred testing tool that sets up connections with multiple-end applications and automates repetitive tasks.

Identity Validator 2.0 comes with the following business solutions:

- ♦ Tests Identity Manager Driver Development.
- ♦ Validates the Identity Manager Drivers.
- ♦ Create events in one application and validate in all other applications.
- ♦ Sets up connections with multiple-end applications and automates repetitive tasks.
- ♦ Saves time for the Consultants.
- ♦ Speeds up the Identity Manager Solution Deployment.

Features of Identity Validator 2.0

The overview of the features of Identity Validator 2.0 are as follows:

- ♦ Create a new Test Suite.
- ♦ Import Test Suites from the file system.
- ♦ Adds Connections with multiple applications and automates repetitive tasks.
- ♦ Configure the connection properties and Test connection
- ♦ Write Tests and run the Tests.
- ♦ Displays Summary and Detailed logs available for the users.
- ♦ Create Templates and associate with one or more Tests.
- ♦ Import or export Tests, Templates.
- ♦ Use of Variables.
- ♦ The **Scheduler** feature: This feature automates the Test Cases as per planned schedule.
- ♦ Download general and detailed Report for test case execution.

Identity Validator 2.0 Architecture

Identity Validator 2.0 is a testing utility, that allows 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. Identity Validator 2.0 is a self-contained stand-alone product that requires no other software other than a supported operating system and browser.

Identity Validator 2.0 server is an embedded Jetty HTTP Server and uses Apache Jersey. It is launched via scripts from the command line. Once launched, the default URL to access Identity Validator 2.0 is:

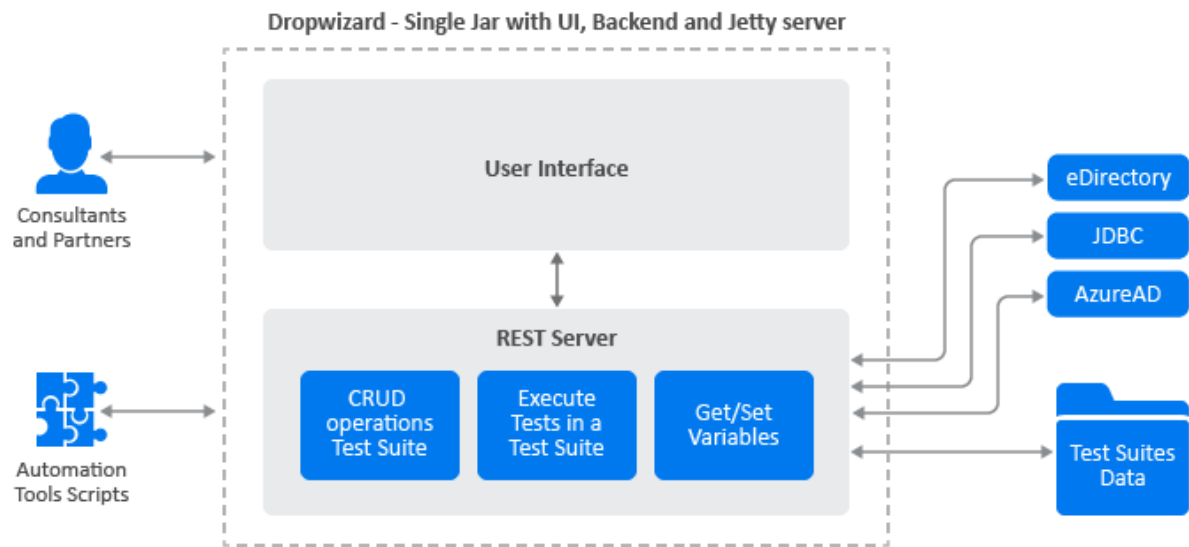
```
https://< ip address>:<port >/validator
```

For example:

```
https://localhost:8380/validator
```

The client app is HTML and Javascript using Angular. It communicates with the Identity Validator 2.0 server via REST calls.

Figure 1-1 Identity Validator 2.0 Architecture Diagram



2 Planning to Install Identity Validator 2.0

This section gives information on the system requirements and prerequisites for installing Identity Validator 2.0.

For latest in scope features, refer: [Identity Validator 2.0 Release Notes. \(https://www.netiq.com/documentation/identity-validator-2.0/identity-validator-2.0-releasenotes/data/identity-validator-2.0-releasenotes.html\)](https://www.netiq.com/documentation/identity-validator-2.0/identity-validator-2.0-releasenotes/data/identity-validator-2.0-releasenotes.html)

- ♦ “Operating Systems” on page 15
- ♦ “Supported Browsers” on page 15
- ♦ “Prerequisites” on page 15

Operating Systems

Following are the supported operating systems:

- ♦ Microsoft Windows 10.
- ♦ RedHat Enterprise Linux (RHEL 7.5, RHEL 7.6, and RHEL 8).
- ♦ SUSE Linux Enterprise Server (SLES15 SP2, and SP3).
- ♦ Mac OS 10.14 (Mojave).

Supported Browsers

Following are the supported browsers:

- ♦ Chrome (v100 and above).
- ♦ Firefox (105.0.1).
- ♦ Safari (V15.6.1 and above).

Prerequisites

Following are the prerequisites for Identity Validator 2.0 installation:

- ♦ Ensure that the Java 1.8 (latest update) is installed on your machine. Also make sure that the Java command is accessible from the console.
- ♦ Extracted `IDM_Validator_2.0.1.zip` file.
- ♦ Get the Identity Validator License Key and rename it as `license.dat`, and then save it in the `config` folder.
- ♦ By default the `IDM_Validator_2.0.1.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 available certificate to the keystore.

3 Installing Identity Validator 2.0

This section gives information on installing Identity Validator 2.0.

- ♦ [“Installing Identity Validator 2.0 on Windows” on page 17](#)
- ♦ [“Installing Identity Validator 2.0 on Linux or Mac OS” on page 18](#)

IMPORTANT: It is recommended to use Validator with a firewall. Make sure that you do not install Validator on a centralized production server. Note that the individual deploying Validator in a production environment is responsible for any security issues that might arise due to this.

Installing Identity Validator 2.0 on Windows

Before installing, ensure that your system meets the [Prerequisites](#). To install the Identity Validator 2.0 on the Windows platform, perform the following steps:

NOTE: Before installing Identity Validator 2.0, ensure that the Java path is added in the system variables.

- 1 Download the `IDM_Validator_2.0.1.zip` file that contains Identity Validator 2.0 installation files.
- 2 Extract the downloaded `IDM_Validator_2.0.1.zip` file.
- 3 Navigate to the extracted folder `IDM_Validator_2.0.1`.
- 4 Copy Identity Validator 2.0 `license.dat` to `config` folder.
- 5 Run the `create-db_win.bat` file.

NOTE: You must execute the `Create-db` file only once to create the initial database.

Creating DataBase for Identity Validator 2.0:

5a When prompted, enter the DataBase password of your choice.

5b Re-enter the DataBase password to confirm.

Create administrator password for Identity Validator 2.0 login:

5c Enter Login password of your choice.

5d Re-enter Login password to confirm.

6 Run the `start-validator_win.bat` file.

7 On an Internet browser open the following URL:

`https://<ip address>:<port>/validator`

For example:

`https://localhost:8380/validator`

The Identity Validator 2.0 application is launched.

NOTE: For the Identity Validator 2.0 API documentation open the following URL:

`https://<ip address>:<port>/api/validator/v1/swagger#`

For example:

`https://localhost:8380/api/validator/v1/swagger#`

Installing Identity Validator 2.0 on Linux or Mac OS

Before installing, ensure that your system meets the [Prerequisites](#). To install the Identity Validator 2.0 on the Linux or Mac OS platform, perform the following steps:

NOTE: Before installing the Identity Validator 2.0, ensure that the Java path is added in the system variables.

- 1 Download the `IDM_Validator_2.0.1.zip` file that contains Identity Validator 2.0 installation files.
- 2 Extract the downloaded `IDM_Validator_2.0.1.zip` file.
- 3 Navigate to the extracted folder `IDM_Validator_2.0.1`.
- 4 Copy Identity Validator `license.dat` to config folder.
- 5 Run the following create-db shell script:

`./create-db_mac_linux.sh`

NOTE: You must run the `./create-db_mac_linux.sh` file only once to create the initial database.

Creating DataBase for Identity Validator 2.0:

- 5a When prompted, enter the DataBase password of your choice.
 - 5b Re-enter the DataBase password to confirm.
 - Create administrator password for Identity Validator 2.0 login:
 - 5c Enter Login password of your choice.
 - 5d Re-enter Login password to confirm.
- 6 Run the following Start shell script:

`./start-validator_mac_linux.sh`

- 7 On an Internet browser open the following URL:

`https://<ip address>:<port>/validator`

For example:

`https://localhost:8380/validator`

The Identity Validator 2.0 application is launched.

NOTE: For the Identity Validator 2.0 API documentation open the following URL:

`https://<ip address>:<port>/api/validator/v1/swagger#`

For example:

`https://localhost:8380/api/validator/v1/swagger#`

4 Test Suites

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

- ♦ [“Importing Test Suite” on page 21](#)
- ♦ [“Creating Test Suite” on page 22](#)
- ♦ [“Create New Connection” on page 23](#)
- ♦ [“Adding Environment and Variable” on page 24](#)
- ♦ [“Java Script Library” on page 25](#)
- ♦ [“Importing Test Suite from Template” on page 26](#)

NOTE: It is important to click **Save**  whenever the you make any modification on the Test Suites. If you miss to Save, the modifications will not be updated.

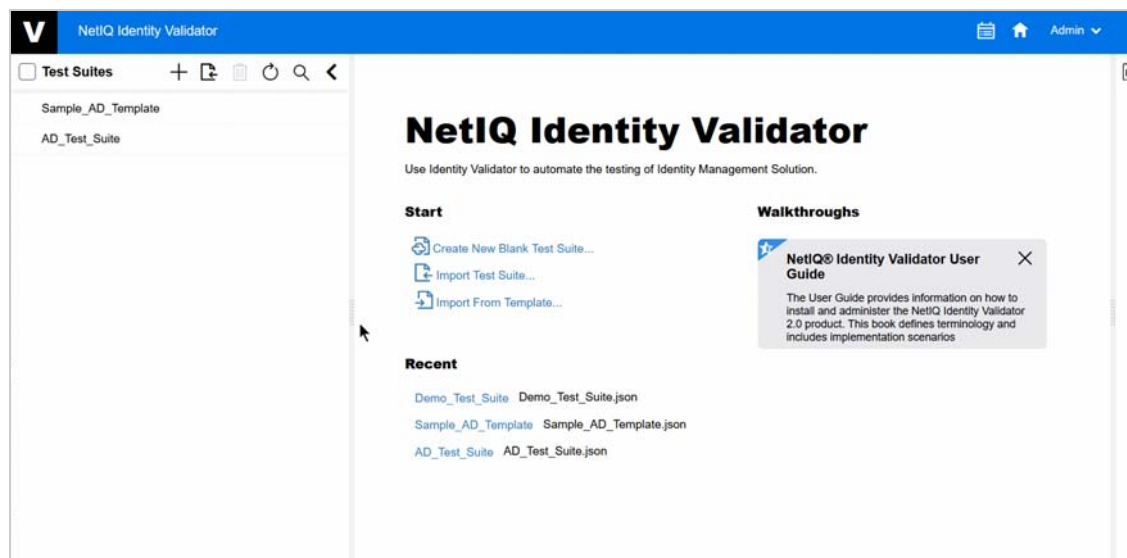
Importing Test Suite

The following procedure explains how to import a Test Suite:

NOTE: The user can use their own existing Test Suites that they already have and import them.

- 1 On the Test Suite pane click **Import Test Suite**.
The **File explorer tab** appears. Navigate to the folder where your files are present.
- 2 Select the Test Suite that needs to be imported, then click **Open**.
Result: The **Importing from New Test Suite** form will appear.
- 3 Rename the imported Test Suite as per your requirement, click .
- 4 Click **Save** .

Figure 4-1 Import Test Suite



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 automates repetitive tasks. The users can automate the testing of Identity Manager environments.

On the NetIQ Identity Validator **Test Suites** tab you can perform the following tasks:

- ♦ View the list of Test Suites.
- ♦ Create new Test Suite (Either by selecting existing template or a new Test Suite).
- ♦ Delete Test Suites
- ♦ View/Modify the details of Test suite.

To create a Test Suite:

- 1 On the Test Suite pane either click  or click **Create New Blank Test Suite**.

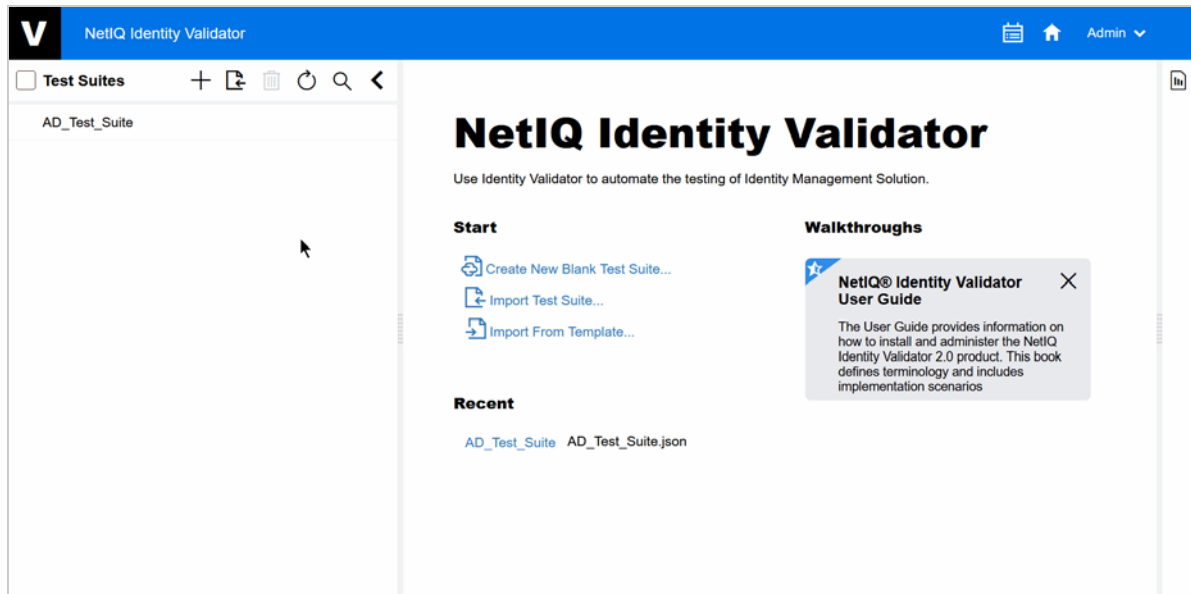
The **Create Test Suite** screen appears.

- 2 Give a Test Suite name.
- 3 Give a brief Description of the Test Suite.
- 4 Click .

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 4-2 Create Test Suite



Create New Connection

On the **Connections** tab the user can view the list of Connections. The user can also Add new connections manually or by importing the connectors from Identity Manager. Other activities that can be performed here are:

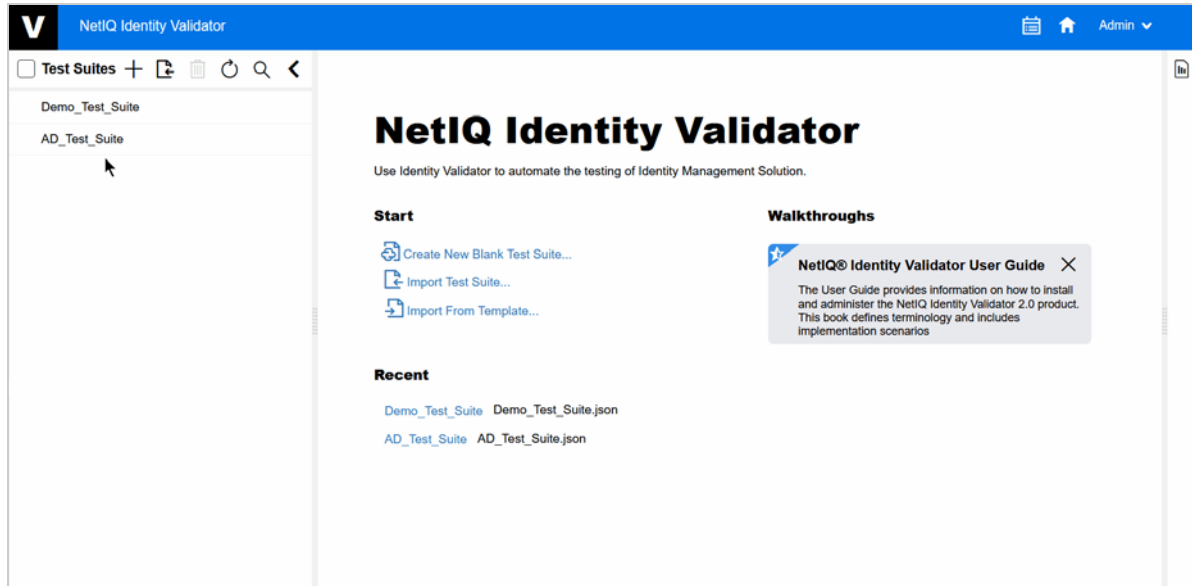
- ♦ View/Modify the Connection information
- ♦ Delete Connections
- ♦ Test Connections.

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

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

The **Reports and Logs** displays the Test Connection status.

Figure 4-3 Create New Connection



Adding Environment and Variable

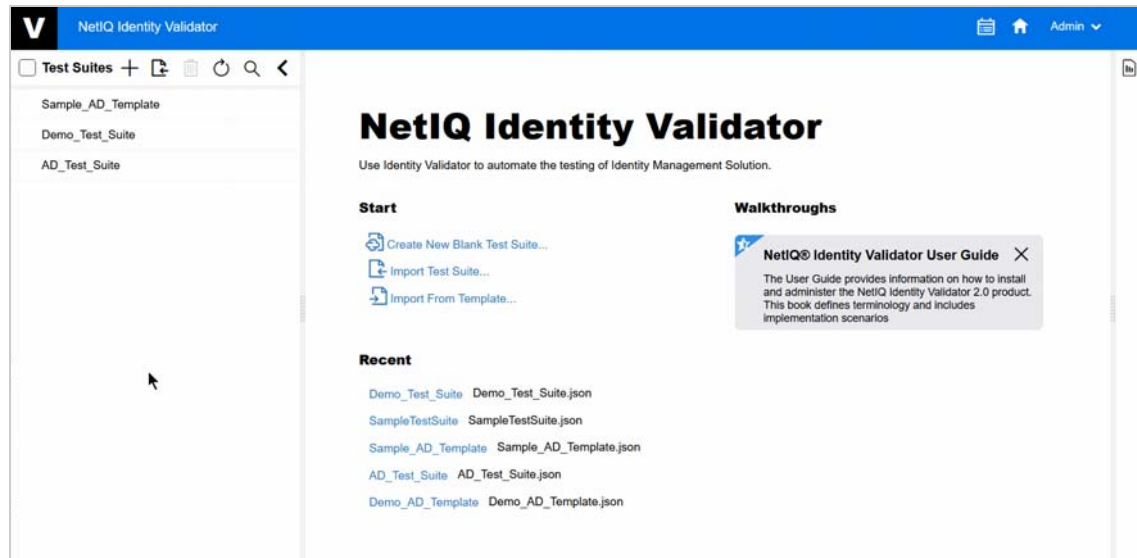
Whenever you want to reuse a particular value or variable, it can be accessed from the **Variable** tab. On the **Variable** tab you can add a variable, delete a variable, and cleanup a variable. You can also add new **Environments** and add Variables.

To add variables, perform the following steps:

- 1 On the **Test Suite** menu, select the Environment for which you want to add a Variables.
- 2 On the **Variables** tab click  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 4-4 Adding a Variable



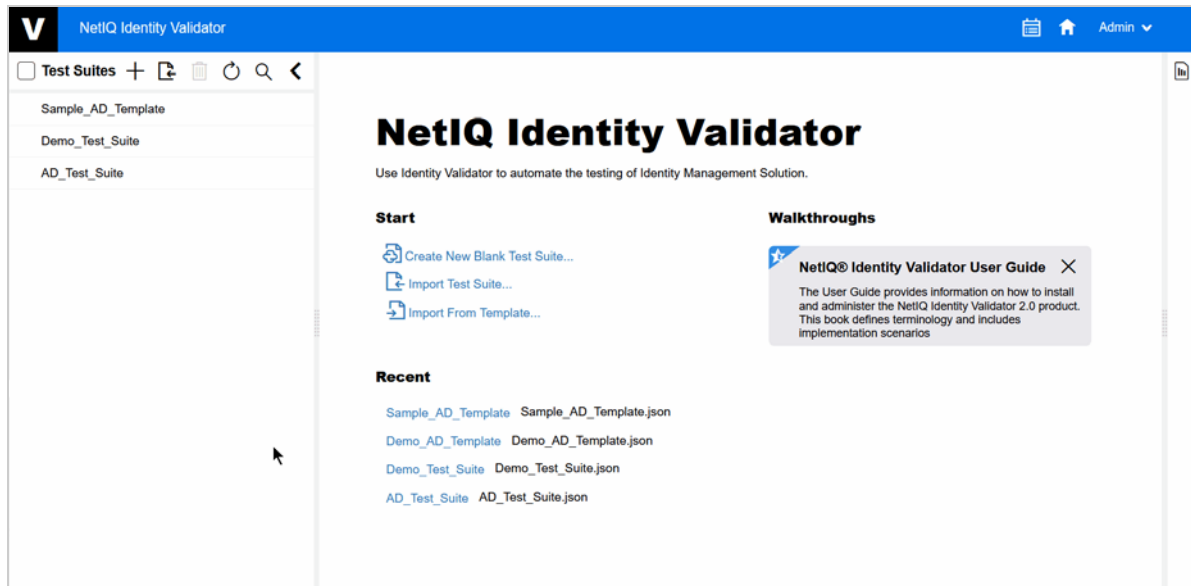
Java Script Library

The following key information helps you while executing JS Library:

- ♦ The content of the Javascript Library is be available to all tests.
- ♦ Javascript expressions can be executed in any field within a Test Action using the `js:` directive.
- ♦ You may use Identity Validator variables within a Javascript expression. Note that you must include the variable token in quotes if the variable value is a string.
- ♦ To use your own Java classes, place your custom jar into the `lib/ext` folder and restart Identity Validator 2.0, and then import your custom classes using the `importPackage` statement. For example: `importPackage(org.jsasypt.util.text)`
- ♦ See the Generic Actions Connector Documentation for more information.

The **Java Script Library** tab helps the users to create Java script functions in the 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 that is available.

Figure 4-5 JS Library



Importing Test Suite from Template

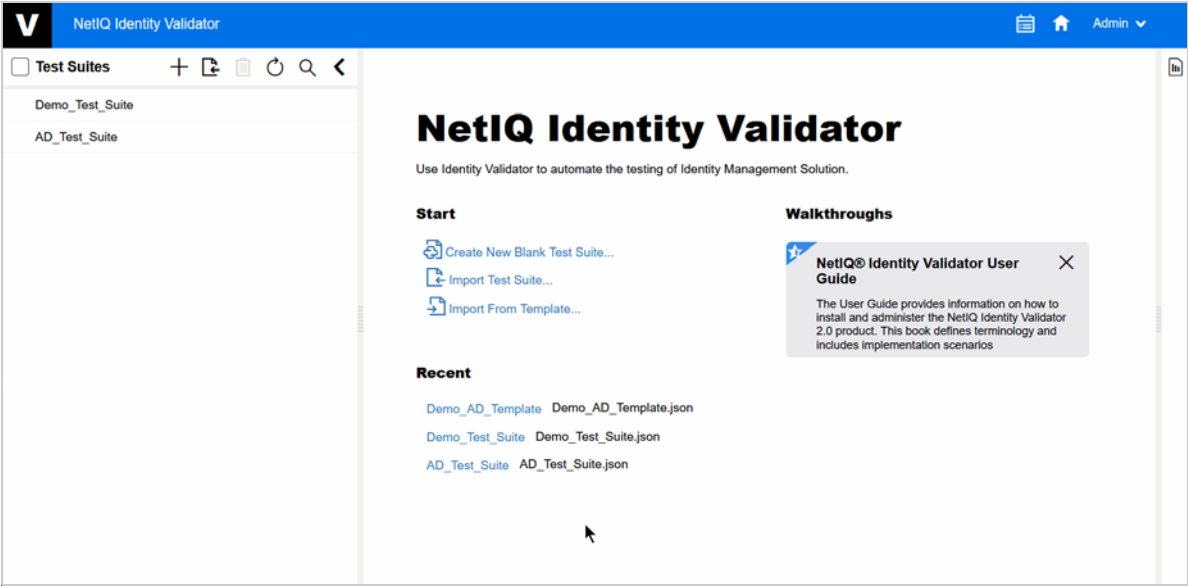
Select a Template from the list of default templates, and create a Test Suite which has connections and variables predefined.

To create a Test Suite by importing a template, perform the following steps:

- 1 On the Homepage, click **Import From Template**.
- 2 Select a given template and click **Import**.
The Create Test Suite screen appears.
- 3 Give a Test Suite name.
- 4 Give a brief Description of the Test Suite.
- 5 Click **Create**.

A new Test Suite is created under the Test Suite pane.

Figure 4-6 Import Test Suite from Template



5 Test Cases

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

- ♦ [“Creating Test Case” on page 29](#)
- ♦ [“Creating Template” on page 33](#)

NOTE: Remember to **Save**  any modification that you make to the Test Suites.

Creating Test Case

Each test contains three sections, Setup, Test, and 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. These sections are described in detail:

About Setup

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

The cleanup section in the test gets the environment ready for the next tests. However, there are scenarios when a test might fail before reaching the cleanup stage. So, as a setup to each test it is recommended that you clean everything such as deleting the objects used for tests.

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 object that needs to be tested. 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

When the Tests succeed, 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 New Test Case

On this page the user can perform 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 be performed.

To create a new Test Case, perform the following steps:

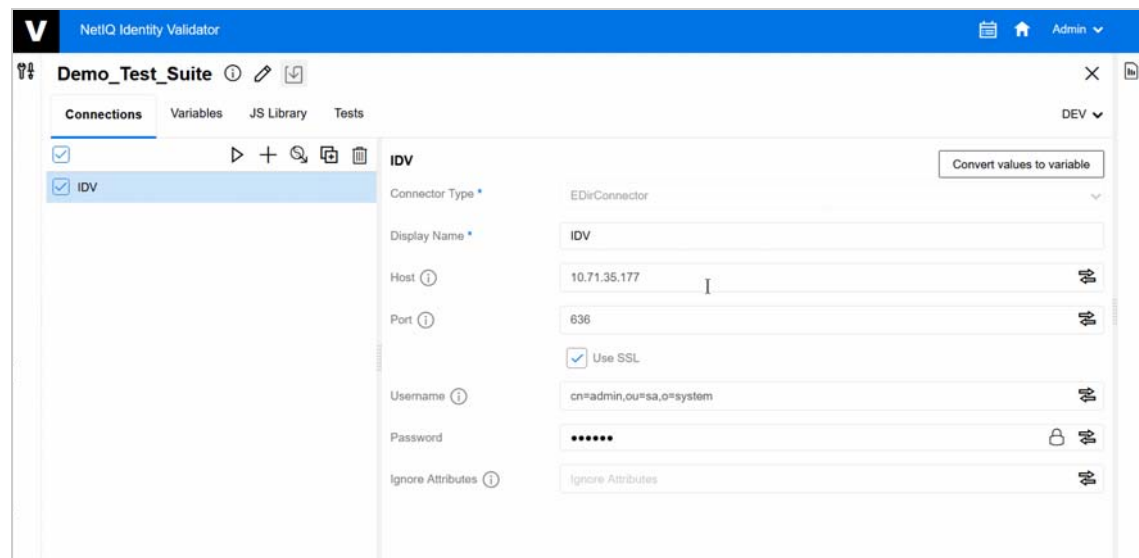
- 1 On the **Tests** tab, click **+** to create a new Test Case.

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.

A new Test Case is created.

- 2 Select **New Test**.
- 3 Click **Edit**  to change the Test Suite **Name** and **Description**.
- 4 Click **Save** and then click global Save .

Figure 5-1 Create New Test Case



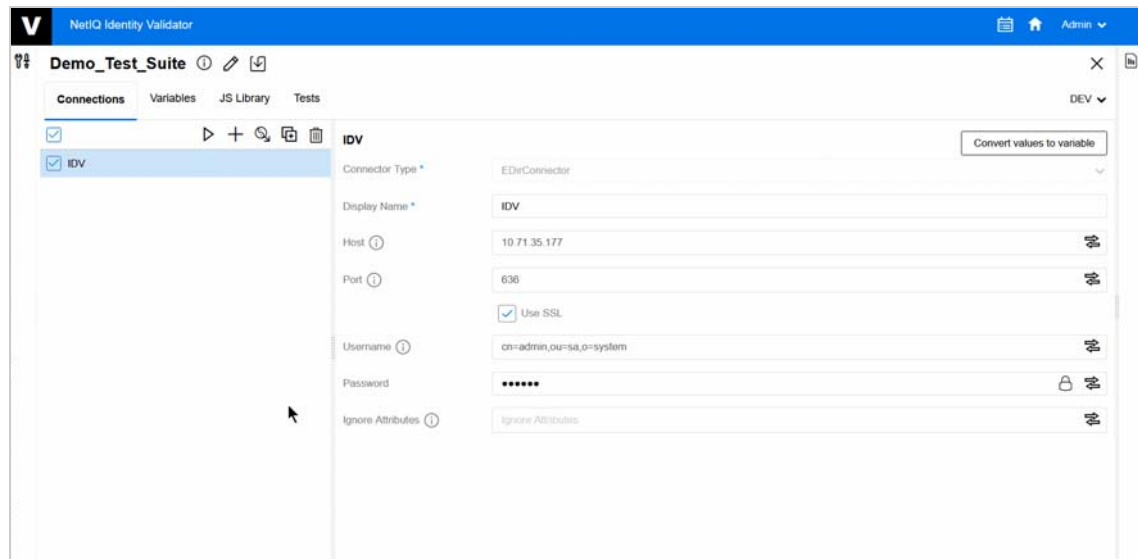
Setup Process

To establish a Setup, perform the following steps:

- 1 On the **Tests** tab, click **Setup** and then .
 - 2 Click  for drop-down.
 - 3 Select **Connection** and **Action**, and then fill all the details as per your requirements such as connection, action, description and so on.
 - 4 Click **Save** , and then Run .
- Reports and Logs** will appear.
- 5 Click  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 5-2 Setup Tab

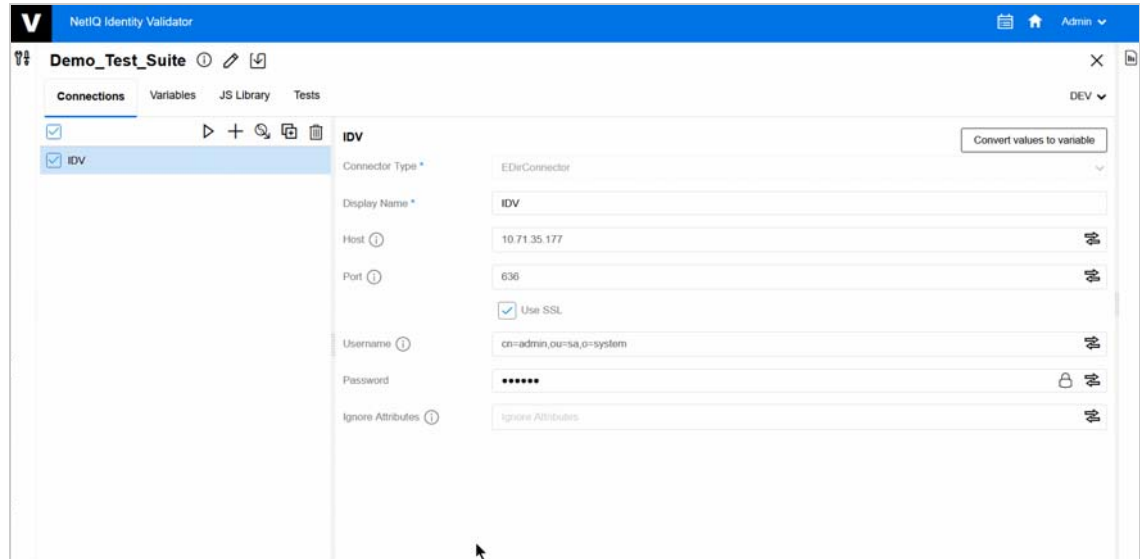


Test

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

- 1 On the **Tests** sub-tab > **Test** >  to add a new Test.
 - 2 Fill all the details as per your requirements such as connection, action, description, Object FDN and so on.
 - 3 Click **Save** , and then Run .
 - 4 Click **Arrow**  to minimize the **Test** tab.
- Reports and Logs** will appear.

Figure 5-3 Test Sub-tab

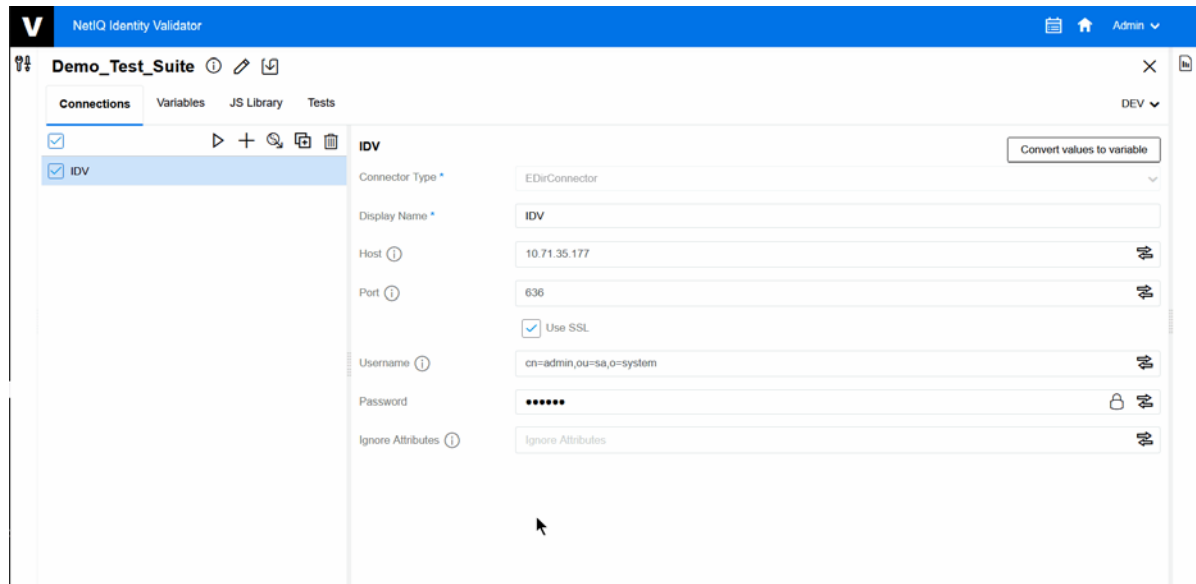


Cleanup

- 1 On the **Tests** tab > **Cleanup** > click **Add** .
- 2 Fill all the other details such as Connection, Action, Description, Test to Run, and so on.
- 3 Click **Save** , and then click **Run** .

The **Reports and Logs** screen will appear. This screen has all the details such as, the Object that is created in the **Setup**, if the **Test** is performed Adding Attributes and Values, if the **Cleanup** is deleting the Attributes. You can also download the reports by clicking the **Download** button.

Figure 5-4 Cleanup



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

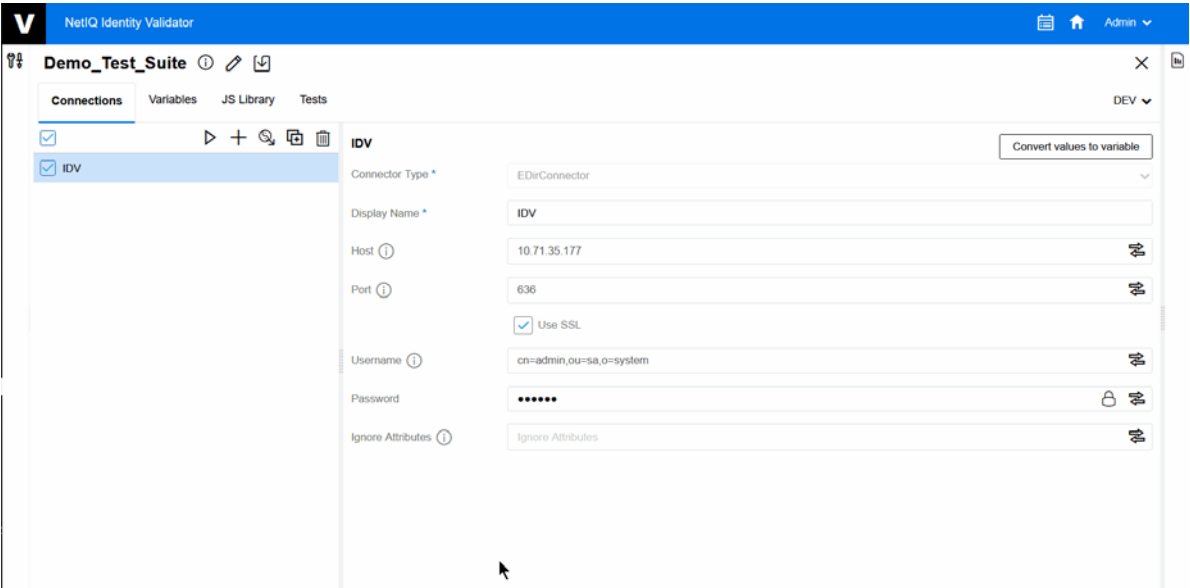
The Results tab will show a live view of the current log file. To stop the auto-refresh click on the **Stop Auto Refresh** icon. If you want to view the log from the OS, all logs are available in the validator\logs folder.

Creating Template

To create a new Template, perform the following steps:

- 1 On the Tests tab, click **Template** >  .
A new Template is created.
- 2 Select the new **Template**.
- 3 Click **Edit**  to change the Template's **Name** and **Description** as per your requirement.
- 4 Click **Save**.
- 5 Click global **Save**  .
The **Saved** message appears.

Figure 5-5 Create New Template



6 Scheduler

Scheduler helps to automate the running of the Test Cases and sends results through email according to a set time interval that you can create. This section gives information about operating the scheduler in Identity Validator 2.0.

NOTE: **Scheduler OFF** toggle helps to enable or disable all the schedules in the Scheduler.

NOTE: The **Auto start** toggle helps to run all the enabled schedules when Identity Validator 2.0 is starting.

- ♦ [“Notifier Configuration” on page 35](#)
- ♦ [“Create New Schedule” on page 36](#)
- ♦ [“Run a Schedule” on page 37](#)
- ♦ [“Edit a Schedule” on page 38](#)
- ♦ [“Delete a Schedule” on page 38](#)

Notifier Configuration

Notifier Configuration is required for Simple Mail Transfer Protocol (SMTP) server’s mailing service.

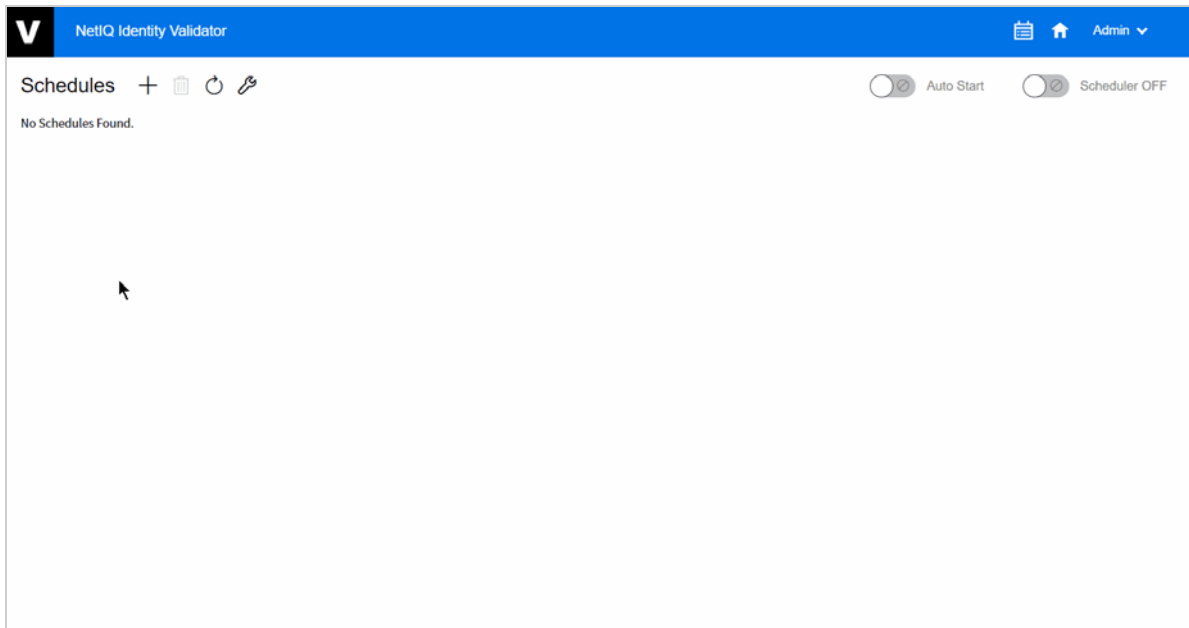
The following the steps explain how to configure Notifier Configuration:

- 1 On the Home page, click the Scheduler .
- 2 On the Schedules page, click Settings .
- 3 Fill the SMTP host field which is a mandatory.
- 4 Fill in all the other fields as per your requirements.

NOTE: You can use HTML to create template for Subject and Body fields. Click  to use active tokens, which will be replaced by their respective values.

- 5 Click **Finish**.

Figure 6-1 Notifier Configuration



Create New Schedule

NOTE: Before creating a schedule, ensure to configure the [Notifier Configuration](#).

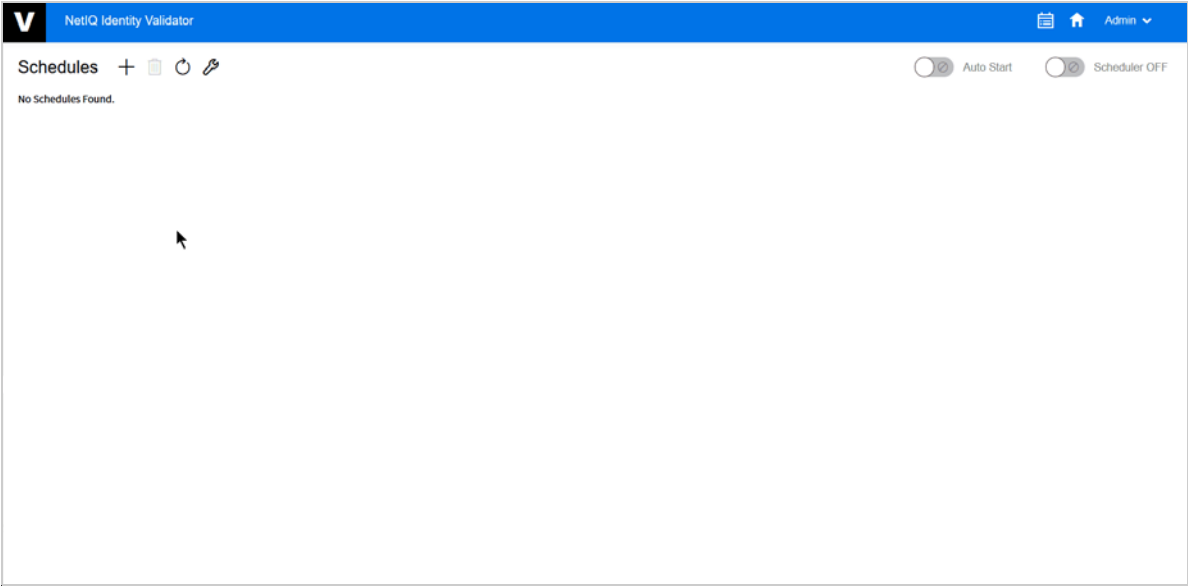
The following procedure explains how to create a schedule:

- 1 On the Home page, click the Scheduler .
- 2 On the Schedules page click Add  to create a new schedule.
- 3 Click **Select Test Suites** and select the Test Suites and Environment that you want to schedule.
- 4 Fill in all the other details as per your requirements.

NOTE: Recipient, Subject, and Body field will use default details from Notifier Configuration if not configured in the schedule.

- 5 (Optional) Click **Attach Log** toggle to get the log of Test Suites that runs through email.
- 6 (Optional) Click **Enable Schedule** toggle to enable the schedule that is created.
- 7 Click **Create**.

Figure 6-2 Create New Schedule



Run a Schedule

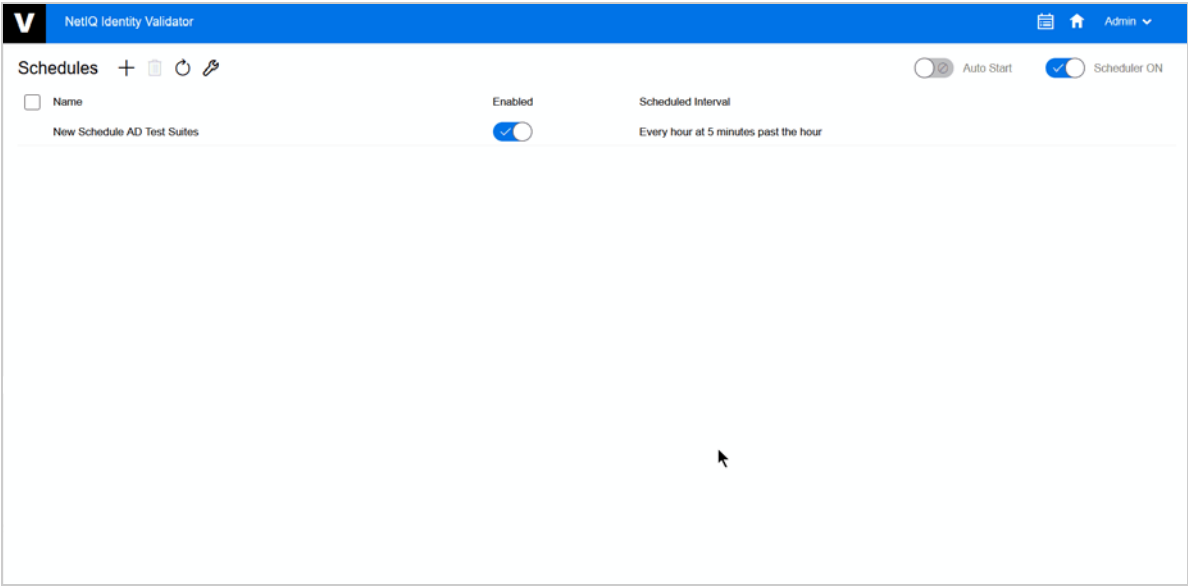
The following procedure explains how to run a schedule:

- 1 On the Home page, click the Scheduler .
- 2 On the Schedules page, click on the Schedule that you want to run.

NOTE: Make sure that the schedule is enabled.

- 3 Click  to run schedule.

Figure 6-3 Run a Schedule

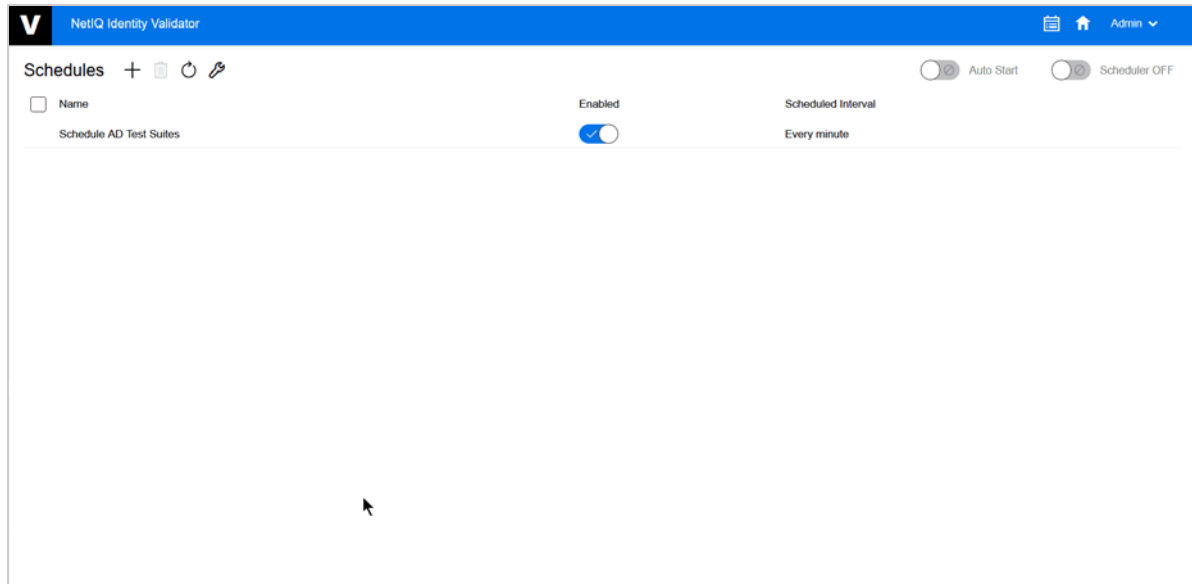


Edit a Schedule

The following procedure explains how to edit a schedule:

- 1 On the Home page, click Scheduler .
- 2 On the Schedules page, click on the Schedule that you want to edit.
- 3 On the **Schedule Details** pane, edit the details as per your requirement.
- 4 Click .

Figure 6-4 Edit Schedule

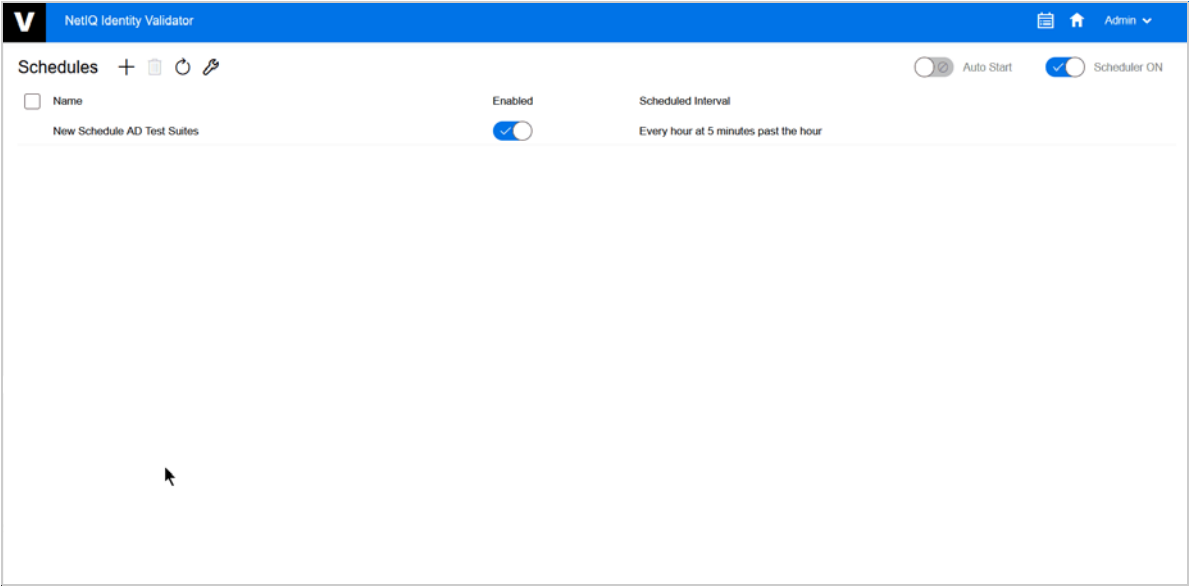


Delete a Schedule

The following procedure explains how to delete a schedule:

- 1 On the Home page, click the Scheduler .
- 2 On the Schedules page, click on the Schedule that you want to delete.
- 3 Click .
- 4 A popup appears to confirm, click **Delete**.

Figure 6-5 Delete a Schedule



7 Connectors

This section gives information about the Connectors that are in use:

- ♦ [“Active Directory Connector” on page 41](#)
- ♦ [“eDirectory Connector” on page 43](#)
- ♦ [“Execute Connector” on page 45](#)
- ♦ [“Generic Actions Connector” on page 46](#)
- ♦ [“HTTP Connector” on page 48](#)
- ♦ [“JDBC Connector” on page 50](#)
- ♦ [“LDAP Based Connectors” on page 52](#)
- ♦ [“Remote Connector” on page 54](#)
- ♦ [“SMTP Connector” on page 55](#)
- ♦ [“Text File Connector” on page 55](#)
- ♦ [“User App Connector” on page 56](#)
- ♦ [“Web Service Listener Connector” on page 57](#)

Active Directory Connector

Review the following considerations before using the Active Directory connector:

- ♦ Ensure the DN of the bind user is in LDAP format. For example, `cn=Administrator, cn=Users, dc=domain, dc=com`.
- ♦ Provide the IP as Host whenever Active Directory is configured.
- ♦ Make sure the default users container in Active Directory is `cn=Users`, and not `ou=Users`.
For more information about Actions, see [LDAP Based Connectors](#).

Assert Password

The Assert password method has been modified to try a bind as the user with the specified password to avoid this problem. Be careful while entering the password because multiple retries can lead to 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.

Active Directory has two possible password attributes exposed via LDAP.

- ♦ `userPassword` (as all LDAP systems should).
- ♦ `unicodePwd`.

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

Set a Password

To set the password you need an SSL connection to AD. It is recommended to set the password using the Set Password action. Note that you cannot create a password during object creation or user creation. So, use the Set Password action after the user is created.

Reset Password on Administrative Setup

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.

Create Group in AD

To create a group in AD, follow this procedure:

- 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

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.

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.

eDirectory Connector

While creating eDir connector, you have the option to import connections from any IDM driver sets configured on the eDir server. This button will not work until you have a good connection with the eDir server. Once the connection is successful, click the (name the button) to import all connection information from all driver sets configured on the server. You cannot import passwords and delete any unwanted connections created by this feature.

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

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

For more information about Actions, see [LDAP Based Connectors](#).

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, the 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 and Check an Entitlement

We do not have an **Entitlement** action yet, but you can still do it by adding the correct attributes.

If you want to tweak this for your specific entitlement the steps are as follows:

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 Identity Validator 2.0 to artificially add an entitlement you do it the same way by adding an attribute with the specific value. This is same as revoke or 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.

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 Identity Validator 2.0 server.

- ♦ Executing commands on Windows:

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

Example: 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>
```

Example: 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.

There are some issues observed with the 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 (Java Virtual Machine), 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 normal shell command-line such as piping(|), redirection(>), spaces in names, and memory between command do not work.

If your command do not work:

1. Make sure the command works on the command line on the Identity 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 the ability 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 - (Optional) Specify your own Javascript functions.
 2. Under the **Test Suite** tab, create your own Javascript Library with any function you wish to use in your tests.

NOTE: To assist in debugging the script, use `console.log()` to output messages to the Identity Validator console.

Using custom Java libraries:

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() }
```

3. 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.

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.

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.

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 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.

HTTP Connector

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 such as Query String, Form data, or Payload Data, and get a response. You can then compare the response with any string that would indicate success.

The HTTP connector consists of the following authentication types:

- ♦ Basic Auth:

If you provide values for the *Basic Auth User* and *Password* when setting up the connection, two auth headers are 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.

- ♦ 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.

- ♦ The connector work with RBPM REST calls:

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. Enter the username, and NOT an LDAP DN, and the password in the connection properties. Don't use Italics. Check the guide when Italics is used and change all instances of Italics to normal text or whatever is applicable.

Header Params

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

Accept: application/json

Otherwise it returns error: 415 unsupported media type.

Usage of the URL suffix field: 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 have to manually encode them. See http://www.w3schools.com/tags/ref_urlencode.asp. You may replace spaces with a + sign if the space is in the query string.

Specify header data: Specify the key/value pairs in the Header Data table within the action.

Specifying form data: Specify the key/value pairs in the Form Data table within the action. Note that this is only applicable for the POST method action.

Using header data, form data, and normal payload raw data together by HTTP call: 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: The SOAP is about 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. Paste your SOAP doc in the Raw Data field.

Expected Results field: If you leave it blank the action always succeeds because 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.

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).

JDBC Connector

The following procedure explains the 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 Identity Validator install directory (lib/jars).
2. Restart the Identity Validator 2.0 server.
3. Within Identity 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. Note the following information for migration.

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. Identity Validator 2.0 uses the value from the first row returned. If your SELECT statement returns multiple columns and or rows, Identity Validator 2.0 uses 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/jars 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 User name and Password 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/jars 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

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.

Connector Actions: The actions that are available in the test or template:

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 (Fully Distinguished Name) FDN and Set Variable.* This action search the tree from a specific **Base DN** for an object whose attributes have values specified by you. 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, if you do not populate, the search may not work on some LDAP servers.

- ♦ *Get Values and Set Variable.* This action sets 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 Imports Fields from the Object link.

If you have an existing object in the directory that is similar to the object you are creating in Identity 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 helps you understand how the date format should be in LDAP.

The LDAP date should be in YYYYMMDDHHMMSSZ format.

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

For deleting Objects using wild-cards, 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, and so on.

Save or Restore Object

Save or 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 ignores 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 and Restore

If an attribute is saved but cannot be restored, you may see 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 means that an attribute contains binary data that cannot be represented in Json so the Json parser throws an error. This attribute can be ignored.
- ♦ 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 allows 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 might ignore that attribute as well.

Remote Connector

This connector is a generic connector which allows 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.

SMTP Connector

- ♦ The SMTP Connector has two purposes:
 - ♦ An SMTP Listener (Server) that receives SMTP emails and saves the emails into the file system. You can then confirm that an email has been received.
 - ♦ Send an SMTP email to any SMTP server. You can 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 Identity 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 for cases where your platform does not allow port 25 to be used by a non-root process. If you need to change the port but still need to allow access from clients that require port 25, you can configure port forwarding on your platform.

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

IMPORTANT: At present they are not working.

```
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 should be present in the same server where Identity Validator is installed. However, you have the option to point at a new network share for the file. To access the files, choose either a file name, which explicitly selects a single file or a file pattern, where all files that match the expression pattern are considered.

Instances where you don't know the filename in a directory because there is no filename but has a pattern. For example, a timestamp you can specify a filename pattern to 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) tests the connection and automatically imports the https certificate into the `/keystore/truststore` file.

The following scenarios provide a few examples when the error `java.lang.RuntimeException` is observed.:

Note that the **Object Browser** button 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.

- ♦ During work flow operation: Here the exception happens when the application is unable to read exception information from stream while executing action method approve Request.

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 Identity Validator, but "reason" was not on the approval form.

- ♦ Bad content: Here the exception happens when you try to start a work flow. This can be because of an invalid URL used in the UserApp server.

You will see the following error message:

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:`

- ♦ Invalid entry: Here a runtime exception happens, when for a field in the approval form you specify that field name as an "attribute" in the Approve Request action.

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 Identity Validator 2.0, you receive the RuntimeException.

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

1. In Designer, click the approval activity in the workflow.

2. Click 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.
- ♦ Start and approve workflow: 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 try 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, revoke, and assert Roles and Resources. You can also do this manually using the [eDirectory connector](#) (Path syntax actions) if the actions in this connector is 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 is later 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 starts any POST or PUT to any endpoint creates a GET endpoint with the same URL in the listener. This GET endpoint returns 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 then creates 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 is removed.

Once the listener stops, 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`.

8

Best Practices and Common Questions

This section provides you the best practices that you can follow to help you 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.

These are some of the best practices that can help you while using the Identity Validator 2.0:

- ♦ [“Saving Test Suite Changes” on page 59](#)
- ♦ [“Testing Connections” on page 59](#)
- ♦ [“Naming Variable while Executing Test” on page 60](#)
- ♦ [“Using Variables and Environments” on page 60](#)
- ♦ [“Using Templates” on page 61](#)
- ♦ [“Changing the Default Values for Retry Interval and Retry Count” on page 61](#)
- ♦ [“Avoid the Pause Actions” on page 61](#)
- ♦ [“Running Unit Tests against Integration Tests” on page 62](#)
- ♦ [“Using Identity Validator 2.0 in a Development Group” on page 62](#)
- ♦ [“Accessing Identity Validator 2.0 from a Remote Workstation” on page 62](#)
- ♦ [“Changing Password” on page 63](#)

Saving Test Suite Changes

It is important to click **Save**  whenever you make any modification on Test Suites. If you do not Save, the modifications will not be updated.

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 **Run** button. If it fails, look at the **Results** tab for detailed error messages.

Some connectors like the User App connector, require the SSL certificate to be in the Identity Validator 2.0 truststore file. By testing the connection to the User App server, it auto-imports the certificate thus allowing the connector to connect via ssl (https).

Naming Variable while Executing Test

Create a naming pattern for the objects your tests generate. This makes it easier to track and deconflict more than one user 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 that you use a Variable for your object names which might 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](#).

Using Variables and Environments

Variables can act as constants, much like Identity Manager Global Constant Values (GCV) 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, Identity Validator 2.0 requires that all variables be declared before using them.

By using Variables in your tests, you can make your tests more flexible and easy 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 the **Covert Values to Variables**. This allows you to utilize the Variable Environments feature so you can easily change the test to access servers in different environments, such as Dev, QA, and Prod.

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. 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 (for example, `${idx}` which is resolved to "2" as in `var1=${var2}`).

Referencing variables can be done in any text field in Identity Validator 2.0. 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 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, Identity Validator 2.0 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 Identity Validator 2.0 header.

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 or Test, or 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.

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 is recommended that you take a backup of the file before editing it.

Avoid the 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 and 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.

So, to add a one minute delay you can add an appropriate Assert Attribute or Value exists, and then set the retry count to 10, and the retry interval to 6000 Millisecond, 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.

Running Unit Tests against Integration Tests

Have different test suites for different types of testing. Unit testing of an Identity Manager drivers functionality is a different use case than integration testing for the end to end solution.

In the Unit test, you can test small pieces of functionality, depending on the circumstance.

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 Identity Validator 2.0 in a Development Group

When using Identity Validator 2.0 and sharing test suite files among a group of developers and testers, consider the following tips:

- ♦ Ensure to limit the number of workstations that require access, as the Identity Validator 2.0 supports authentication for a single administrator account.
- ♦ 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 or 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.

Accessing Identity Validator 2.0 from a Remote Workstation

To access Identity Validator 2.0 from a remote workstation you must modify the configuration file.

You may want to do this if the server running the Identity Validator 2.0 does not have a supported browser, or to allow several users to access the Identity Validator 2.0 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.

Changing Password

You can change the password only if you require. To change the Identity Validator 2.0 password, perform the following steps:

- 1 On the Home page click **Admin > Profile**.
- 2 Enter the new password that you want to change to.
- 3 Click **Save**.

The **Profile Updated Successfully** message appears.

9 Troubleshooting

This section provides useful information for troubleshooting problems with Identity Validator 2.0

Unable to Log in to Identity Validator 2.0 After Sign-out

Issue: After signing out from Identity Validator 2.0, the user is unable to log in again.

Workaround: The user must clear cache data from the browser and retry log in.

