NetIQ® Identity Manager

Driver for REST Implementation Guide

February 2017

Legal Notice

For information about NetIQ trademarks, see https://www.netiq.com/company/legal/.

Copyright (C) 2017 NetIQ Corporation. All rights reserved.

Contents

About NetIQ Corporation **5**
About This Guide **7**
About this Book and the Library **9**

1 Understanding the REST Driver **11**

Key Terms . . . 11
Identity Manager . . . 12
Connected System . . . 12
Identity Vault . . . 12
Identity Manager Engine . . . 12
Driver Shim . . . 12
Driver Packages . . . 12
Remote Loader . . . 13
Driver Concepts . . . 13
Introduction . . . 13
How the Driver Works . . . 15
Understanding Driver Operation Data . . . 16
Support for Standard Driver Features . . . 18
Supported Operations . . . 18
Local Platforms . . . 18
Remote Platforms . . . 18
Supporting Driver Authentication . . . 18
Supporting Publish Mode . . . 21
Supporting Identity Manager Engine as a REST EndPoint . . . 21
Synchronizing Information . . . 23
Supporting Entitlements and Permission Collection and Reconciliation Service . . . 23

2 Installing the Driver Files **25**

Prerequisites for Driver Installation . . . 25
Installing the REST Driver Files . . . 25

3 Creating a New Driver Object **27**

Creating the Driver Object in Designer . . . 27
Importing the Current Driver Packages . . . 27
Installing the Driver Packages . . . 28
Configuring the Driver Object . . . 31
Deploying the Driver Object . . . 31
Starting the Driver . . . 32
Activating the Driver . . . 33
Adding Packages to an Existing Driver . . . 33
Creating Custom Entitlements . . . 34

4 Upgrading an Existing Driver **39**

Supported Upgrade Paths . . . 39
What's New . . . 39
What's New in Version 1.0.1.1 . . . 39
What's New in Version 1.0.0.1 . . . 39

Upgrading the Driver 39
. . . . . Upgrading the Installed Packages 39
Applying the Driver Patch 40

5 Customizing the Driver for RESTful Services 43

Using Java Extensions 43
Changing the JSON/XML Payload 43
Using driver-operation-data 43

6 Securing Communication 45

Configuring the Publisher Channel 45
Configuring the Subscriber Channel 46

7 Managing the Driver 49

8 Troubleshooting the Driver 51

Driver Shim Errors 51
Troubleshooting Driver Processes 51
Exception Reported After Upgrading to REST Driver 1.0.0.1 51
Driver Reports an Error When a Password or an Attribute Value Contains the < Character 52

A Driver Properties 53

Driver Configuration 53
Driver Module 54
Authentication 54
Startup Option 54
Driver Parameters 55
ECMAScript 61
Global Configuration 62
Global Configuration Values 62
Password Synchronization 62
Permission Collection and Reconciliation 63

B Using Java Extensions 65

Overview 65
Creating and Configuring Java Extensions 66

C Trace Levels 69

D Supported JSON Format 71

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. The documentation for this product is available on the NetIQ Web site in HTML and PDF formats on a page that does not require you to log in. If you have suggestions for documentation improvements, click **Add Comment** at the bottom of any page in the HTML version 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

NetIQ Communities, 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, NetIQ Communities 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 community.netiq.com.

About This Guide

This guide explains how to install and configure the Identity Manager Driver for REST. The guide includes the following information:

- Chapter 1, “Understanding the REST Driver,” on page 11
- Chapter 2, “Installing the Driver Files,” on page 25
- Chapter 3, “Creating a New Driver Object,” on page 27
- Chapter 4, “Upgrading an Existing Driver,” on page 39
- Chapter 5, “Customizing the Driver for RESTful Services,” on page 43
- Chapter 6, “Securing Communication,” on page 45
- Chapter 7, “Managing the Driver,” on page 49
- Chapter 8, “Troubleshooting the Driver,” on page 51
- Appendix A, “Driver Properties,” on page 53
- Appendix B, “Using Java Extensions,” on page 65
- Appendix C, “Trace Levels,” on page 69
- Appendix D, “Supported JSON Format,” on page 71

Audience

This guide is intended for administrators implementing Identity Manager, application server developers, Web services administrators, and consultants. You should also have an understanding of DSML/SPML, REST,JSON and HTML.

Feedback

We want to hear your comments and suggestions about this manual and the other documentation included with this product. Please use the User Comments feature at the bottom of each page of the online documentation, or go to www.novell.com/documentation/feedback.html and enter your comments there.

Documentation Updates

For the most recent version of this document, see the Identity Manager Drivers Documentation Web site (http://www.netiq.com/documentation/identity-manager-46drivers/index.html).

Additional Documentation

For information on Identity Manager, see the Identity Manager Documentation Web site (http://www.netiq.com/documentation/identity-manager-46drivers).

About this Book and the Library

The *Identity Manager Driver for REST Implementation Guide* explains how to install and configure the Identity Manager Driver for REST.

Intended Audience

This book provides information for administrators implementing Identity Manager, application server developers, Web services administrators, and consultants, who also have an understanding of DSML/SPML, REST, JSON and HTML.

Other Information in the Library

The library provides the following information resources:

Identity Manager Setup Guide

Provides overview of Identity Manager and its components. This book also provides detailed planning and installation information for Identity Manager.

Designer Administration Guide

Provides information about designing, testing, documenting, and deploying Identity Manager solutions in a highly productive environment.

User Application: Administration Guide

Describes how to administer the Identity Manager User Application.

User Application: User Guide

Describes the user interface of the Identity Manager User Application and how you can use the features it offers, including identity self-service, the Work Dashboard, role and resource management, and compliance management.

User Application: Design Guide

Describes how to use the Designer to create User Application components, including how to work with the Provisioning view, the directory abstraction layer editor, the provisioning request definition editor, the provisioning team editor, and the role catalog.

Identity Reporting Module Guide

Describes the Identity Reporting Module for Identity Manager and how you can use the features it offers, including the Reporting Module user interface and custom report definitions, as well as providing installation instructions.

Analyzer Administration Guide

Describes how to administer Analyzer for Identity Manager.

Identity Manager Common Driver Administration Guide

Provides information about administration tasks that are common to all Identity Manager drivers.

Identity Manager Driver Guides

Provides implementation information about Identity Manager drivers.

1 Understanding the REST Driver

REST (**Representational State Transfer**) is an HTTP-based protocol used for Internet communication. REST is the widely emerging standard for applications across World Wide Web, Software as a Service (SaaS) applications, distributed systems, cloud-based services, web services and other business critical applications. A RESTful service is implemented using the HTTP protocol and the principles of REST.

The Identity Manager driver for REST enables identity provisioning and data synchronization between an Identity Vault and any RESTful service.

The driver is not targeted to a specific Web service. The driver is a generic shim that handles the HTTP transport of data between an Identity Vault and a RESTful service. For this driver, a RESTful service is defined as an application that uses HTTP as the transport protocol. The REST driver provides interfaces to transform events and data between Identity Vault and connected system. The driver also exposes REST endpoints that enables Identity Manager to function as a RESTful service.

The driver provides the following key features:

- Supports Anonymous, Basic, and OAuth2.0 authentication
- Supports XML/JSON based requests between the Identity Manager and any RESTful services
- Provides interfaces to extend driver functionalities
- Exposes the REST endpoints that enables CRUD operation to be done in RESTful way on Identity Vault
- Supports password synchronization
- Supports Permission Collection and Reconciliation Service (PCRS)

This section provides the following information for the REST driver:

- “Key Terms” on page 11
- “Driver Concepts” on page 13
- “Support for Standard Driver Features” on page 18

Key Terms

- “Identity Manager” on page 12
- “Connected System” on page 12
- “Identity Vault” on page 12
- “Identity Manager Engine” on page 12
- “Driver Shim” on page 12
- “Driver Packages” on page 12
- “Remote Loader” on page 13

Identity Manager

NetIQ Identity Manager is a service that synchronizes data among servers in a set of connected systems by using a robust set of configurable policies. Identity Manager uses the Identity Vault to store shared information, and uses the Identity Manager engine for policy-based management of the information as it changes in the vault or connected system. Identity Manager runs on the server where the Identity Vault and the Identity Manager engine are located.

Connected System

A connected system is any system that can share data with Identity Manager through a driver. Any RESTful service is a connected system for this driver.

Identity Vault

The Identity Vault is a persistent database powered by eDirectory and used by Identity Manager to hold data for synchronization with a connected system. The vault can be viewed narrowly as a private data store for Identity Manager or more broadly as a metadirectory that holds enterprise-wide data. Data in the vault is available to any protocol supported by eDirectory, including the NetWare Core Protocol (NCP), which is the traditional protocol used by iManager, and LDAP.

Because the vault is powered by eDirectory, Identity Manager can be easily integrated into your corporate directory infrastructure by using your existing directory tree as the vault.

Identity Manager Engine

The Identity Manager engine is the core server that implements the event management and policies of Identity Manager. The engine runs on the Java Virtual Machine in eDirectory.

Driver Shim

A driver shim is the component of a driver that converts the XML-based Identity Manager command and event language (XDS) to the protocols and API calls needed to interact with a connected system. The shim is called to execute commands on the connected system after the Output Transformation runs. Commands are usually generated on the Subscriber channel but can be generated by command write-back on the Publisher channel.

Driver Packages

The REST driver packages are available on the Package Update site. When you create a driver with packages in Designer, Designer creates a set of policies and rules suitable for synchronizing with the REST driver.

The REST driver packages are:

- **NETQRESTBASE**: Contains the base package for the REST driver.
- **NETQRESTDCFG**: Contains the default packages.
- **NETQRESTJSON**: Contains the default JSON policies for converting XDS to JSON format and vice versa.

- **NETQRESTPWD**: Contains the policies for password synchronization.
- **NETQRESTPCRS** - Contains the policies for Permission Collection and Reconciliation Service for resource onboarding.

Remote Loader

A Remote Loader enables a driver shim to execute outside of the Identity Manager engine (perhaps remotely on a different machine). The Remote Loader is a service that executes the driver shim and passes information between the shim and the Identity Manager engine.

For the REST driver, install the driver shim on the server where the Remote Loader is running. You can choose to use SSL to encrypt the connection between the Identity Manager engine and the Remote Loader. For more information, see "Creating a Secure Connection to the Identity Manager Engine" in the *NetIQ Identity Manager Setup Guide*.

Driver Concepts

This section contains the following information:

- "Introduction" on page 13
- "How the Driver Works" on page 15

Introduction

The following concepts are associated with the REST driver:

- "REST" on page 13
- "JSON" on page 13
- "Resource" on page 14
- "Resource Handler" on page 14
- "URL Placeholder" on page 15
- "XML" on page 15
- "HTTP" on page 15
- "HTTPS" on page 15

REST

REST is an HTTP-based protocol for exchanging messages over the network. Since REST is built on HTTP protocol, it supports `POST, PUT, GET, PATCH, DELETE` methods to communicate with the application logic.

JSON

JSON (**Java Script Object Notation**) is a lightweight data-interchange format. JSON stores information in a Key-Value pair format. The Identity Manager driver for REST uses JSON as a data format for payload transfer. For more information about the JSON format used by the driver, see Appendix D, "Supported JSON Format," on page 71.

Resource

A resource is a user, group, or an object that the driver tries to synchronize with the Subscriber and Publisher channels. To be more precise, a REST resource in the driver is a combination of the REST application schema name and the Resource handler. For example, in the URL `http://ipaddress:port/User`, *User* is an example of a REST resource that can be configured to use *Default* as the Resource Handler. For more information, see “Resources” on page 59. To configure a REST resource, Identity Manager provides Driver Configuration options.

Resource Handler

A Resource handler is the mapping of an Identity Manager operation with the REST method. To configure a Resource handler, Identity Manager provides the Driver Configuration options. For more information, see “Resources” on page 59.

A REST call invokes the REST method mapped with an Identity Manager operation. The REST driver supports two Resource handler modes. They are:

- Default - Uses the default HTTP methods for configuring handlers and for managing operations on respective resources. In this mode, the REST driver chooses the best possible mapping for the corresponding Identity Manager operation. For example, an Identity Manager ADD operation corresponds to a POST method and a MODIFY operation corresponds to PUT method of the REST application.

  The REST driver generates the complete URL of a REST method by combining the Base URL for REST Resources and the Schema Name. For example, *https:url.example.com/users*, where *https:url.example.com* is the base URL and *users* is the schema name. Table 1-1 lists the Identity Manager operations, their corresponding default REST methods and the URLs.

Table 1-1 *Default Resource Handler*

Identity Manager Operation	REST Method	URL
ADD	POST	`http://ipaddress:port/SchemaName<api-version>`
MODIFY	PUT	`http://ipaddress:port/SchemaName/<association><api-version>`
QUERY	GET	`http://ipaddress:port/SchemaName/<association><filter><api-version>`
DELETE	DELETE	`http://ipaddress:port/SchemaName/<association><api-version>`

NOTE: In the GET method, the driver replaces the `<filter>` placeholder by `?search-attr=<searchAttrName1> eq <value1>'` and `<searchAttrName2> eq '<value2'>&read-attr='<readAttr1>'` and '`<readAttr2>`' filter value.

- Custom - Uses the Resource Handler parameters in the Driver Configuration page to customize the driver to suite your deployment scenario. In this mode, the driver generates the complete URL of the REST method by combining the Base URL for REST Resources and the user specific URL in the **URL extension**. For example, *https:url.example.com/users*

URL Placeholder

A URL placeholder is a variable defined in the URL extension within angular brackets. The attribute-value pair in the URL token element of the `driver-operation-data` replaces this placeholder value during the data transfer. For example, consider a sample URL `http://ipaddress:port/SchemaName/<association><api-version>`. During the driver operation, the `<api-version>` URL placeholder is replaced by the value in the element `<url-token api-version="1.0"/>`.

XML

XML (Extensible Markup Language) is a generic subset of Standard Generalized Markup Language (SGML) that allows for exchange of structured data on the Internet.

HTTP

HTTP is a protocol used to request and transmit data over the Internet or other computer network. The protocol works well in an Internet infrastructure and with firewalls.

HTTP is a stateless request/response system because the connection is usually maintained only for the immediate request. The client establishes a TCP connection with the server and sends it a request command. The server then sends back its response.

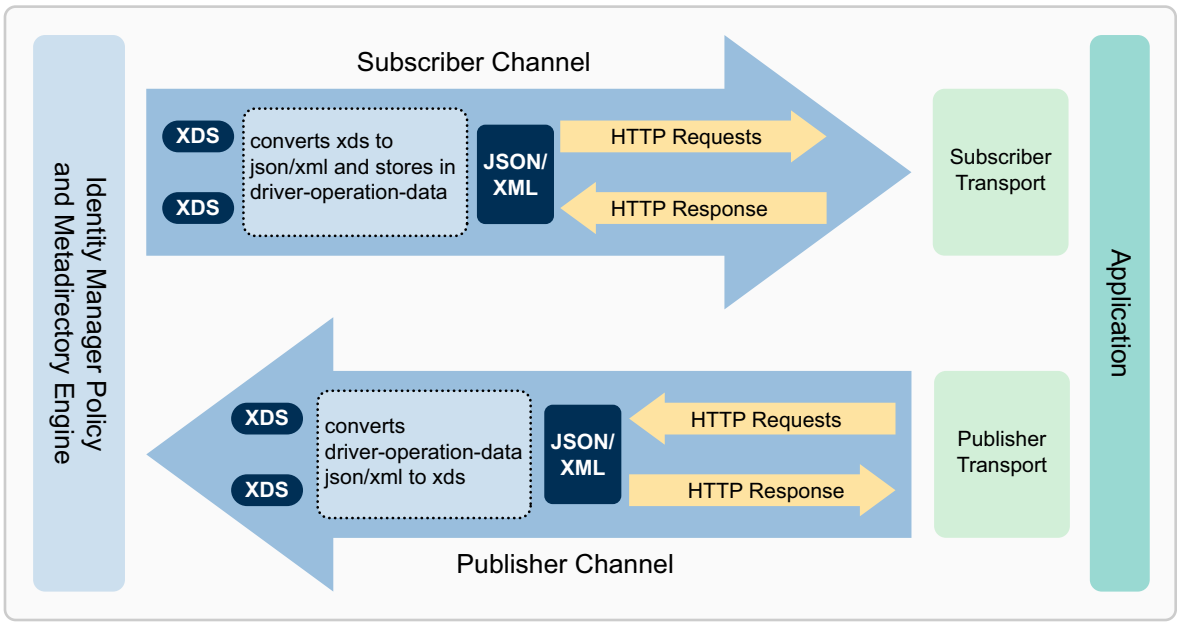
HTTPS

HTTPS is the HTTP protocol over Secure Socket Layer (SSL) as a sub-layer under the regular HTTP application layering. HTTPS encrypts and decrypts user page requests as well as the pages that are returned by the Web server.

How the Driver Works

Figure 1-1 illustrates the data flow between Identity Manager and REST driver:

Figure 1-1 *REST Driver Data Flow*



The Identity Manager engine uses **XDS**, a specialized form of XML, to represent events in the Identity Vault. Identity Manager passes the XDS to the driver policy, which consists of basic policies and DirXML Script.

The driver uses a specialized form of XDS called `<driver-operation-data>`. The `<driver-operation-data>` element encapsulates the metadata and payload for a REST request.

When an event occurs in the Identity Vault, Identity Manager creates an XDS command to represent that event. Identity Manager passes the XDS command to the driver policy. The driver policy transforms that XDS command with an output transformation policy.

This output transformation policy generates the `<driver-operation-data>` that includes commands, URIs, methods, and payload information for the REST request to successfully complete on the Subscriber channel.

When the request completes, the driver processes responses and reports status of the completed operation to the Identity Manager engine or the Identity Vault.

On the Publisher channel, the REST driver receives the REST request in `<driver-operation-data>` format. Using the input policy, the driver converts the request to an XDS event and reports back to the connected system.

Understanding Driver Operation Data

The driver shim applies special handling to Subscriber commands based on an XML element embedded in the command, which appears in the driver shim as `<driver-operation-data>`. The `<driver-operation-data>` element is added to the command from one of the Subscriber channel policies.

The `<driver-operation-data>` element includes the metadata with the class-name, command, REST method, and the URI. The command, the REST method, and the URI is required only if the Resource handlers for the resource are not previously configured in the driver parameters. The `<request>` tag includes the URL-token associations, header content-type, and the data to be transferred. The `<value>` tag includes the JSON payload information.

The <driver-operation-data> element for a REST request includes the following elements:

- **<request>:** Embeds the request information required to make the HTTP call.
- **<url-token>:** Includes the placeholder provided in the driver configuration for resource.For example, during the driver configuration, version is the placeholder added to a resource URL /User/<version>. The attribute-value pair in the URL token element replaces this placeholder. For example, <url-token version="1.0"/>.
- **<header>:** Includes the additional headers that can be added to the REST request in addition to the ones configured in the resource.
- **<value>:** Includes the XML or the JSON payload.

Below is a sample request to add new users with the same common name using the <driver-operation-data> element:

```
<driver-operation-data class-name="User" command="add">
        <request method="put" url="https://172.16.0.0:XXXX/User/rest123">
          <url-token association="rest123"/>
          <header content-type="application/json"/>
          <value>{"CN":[{"value":"rest6789"}],"Full Name":[{"value":"rest6789
rest6789"}],"Given
Name":[{"value":"rest6789"}],","Surname":[{"value":"rest6789"}],"Login
Disabled":[{"value":"true"}]}
          </value>
        </request>
        <request method="put" url="https://172.16.0.0:XXXX/User/rest123">
         <url-token association="rest123"/>
          <header content-type="application/json"/>
          <value>{"CN":[{"value":"rest1234"}],"Full Name":[{"value":"rest1234
rest1234"}],"Given
Name":[{"value":"rest1234"}],","Surname":[{"value":"rest1234"}],"Login
Disabled":[{"value":"true"}]}
          </value>
        </request>
</driver-operation-data>
```

You will get a response similar to the below sample for this request:

```
<input>
   <driver-operation-data class-name="User" command="add" remote-host="172.16.0.0"
url="http://172.16.0.0:XXXX/User">
        <header content-type="application/json"/>
      <response>
        <value>{"association":"noble2","CN":"noble2","Full Name":"noble2","Given
Name":"noble2","nspmDistributionPassword":"novell@123","Surname":"noble2"}
        </value>
      </response>
    </driver-operation-data>
</input>
```

NOTE: The driver retains the <driver-operation-data> between any REST operations, embeds the response in the same <driver-operation-data> and returns the response. A single <driver-operation-data> element is capable of accommodating multiple requests that belong to the same class.

Support for Standard Driver Features

The following sections provide information about how the REST driver supports the standard driver features:

- “Supported Operations” on page 18
- “Local Platforms” on page 18
- “Remote Platforms” on page 18
- “Supporting Driver Authentication” on page 18
- “Supporting Publish Mode” on page 21
- “Supporting Identity Manager Engine as a REST EndPoint” on page 21
- “Synchronizing Information” on page 23
- “Supporting Entitlements and Permission Collection and Reconciliation Service” on page 23

Supported Operations

The REST driver performs the following operations on the Publisher and Subscriber channels:

- **Publisher Channel:** Add, Modify, Delete, and Query operations on User and Group objects, and password synchronization.
- **Subscriber Channel:** Add, Modify, Delete, Migrate, and Query operations on User and Group objects, Password Set/Reset operations only on User objects.

Local Platforms

A local installation is an installation of the driver on the Metadirectory server. The REST driver can be installed on the operating systems supported for the Metadirectory server.

For information about the operating systems supported for the Metadirectory server, see *System Requirements for the Identity Manager Engine* in the *NetIQ Identity Manager Setup Guide*.

Remote Platforms

The REST driver can use the Remote Loader service to run on a server other than the Metadirectory server. The REST driver can be installed on the operating systems supported for the Remote Loader.

For information about the supported operating systems, see *System Requirements for the Remote Loader* in the *NetIQ Identity Manager Setup Guide*.

Supporting Driver Authentication

The REST driver allows you to configure the following authentication methods. By default the REST driver supports Basic authentication method. However, you can change the authentication method using the Driver configuration.

- **Anonymous:** The driver uses anonymous authentication method for authenticating to a RESTful service. On the Subscriber channel, this method allows valid connectivity between the REST driver and any RESTful service that supports anonymous authentication method. On the Publisher channel, the driver allows anonymous access to the Identity Vault for any RESTful service.

IMPORTANT: NetIQ recommends that you do not use anonymous authentication method on the Publisher channel.

- **Basic:** The driver uses the ID and password that you specify during driver configuration for authenticating to the RESTful service. The driver considers the Publisher user credentials as the basic authentication method credentials. In this authentication method, the driver uses these credentials to connect to the endpoints exposed on the Publisher channel.
- **OAuth2.0:** The OAuth 2.0 is an open authentication protocol that enables any third-party application to access data from an HTTP service to share data among various applications. The driver supports OAuth2.0 authentication only on the Subscriber channel.

  The resource owner grants authorization to a client application in cooperation with the authorization server associated with the resource server. When requesting for authorization, the client receives an authorization grant from the resource owner. An authorization grant is an authorization credential representing the resource owner authorization. The two authorization grants supported by the REST driver are resource owner password credentials and client credentials.

  - Client Credentials - Uses the client ID and secret received while registering with the identity provider.
  - Resource Owner Password - Shares the resource owner credentials with the client application. Uses the user name and password of the resource owner as authorization grant to obtain an access token. For example, you can use your Twitter user name and password to log in to a client application.

NOTE: Ensure that you set the appropriate query options while configuring the authorization query in the driver parameters. For more information, see “Subscriber Settings” on page 55.

Figure 1-2 illustrates the client credentials option.

Figure 1-2 *Client Credentials*

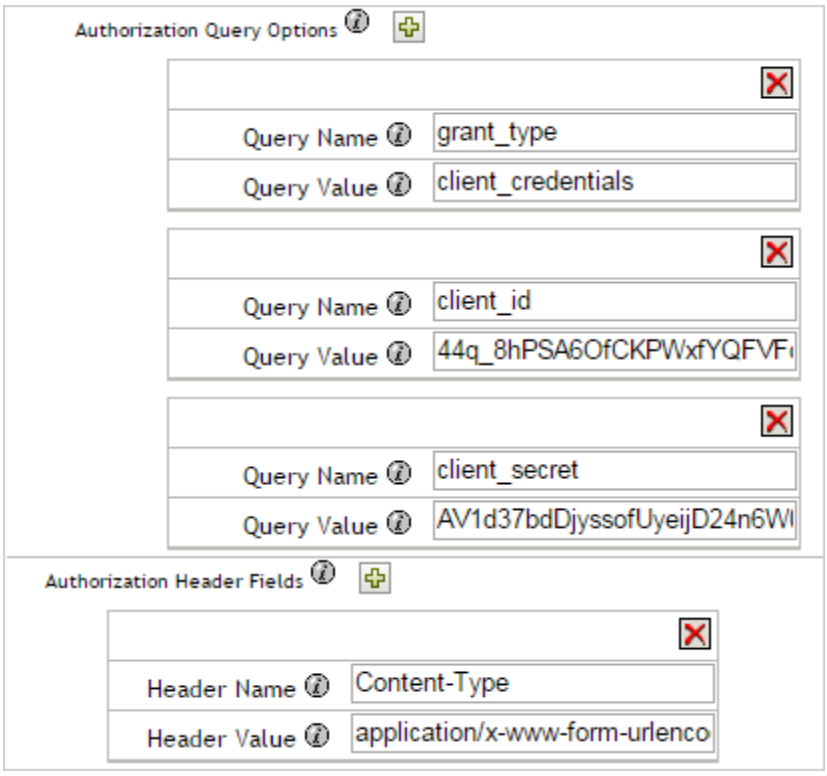
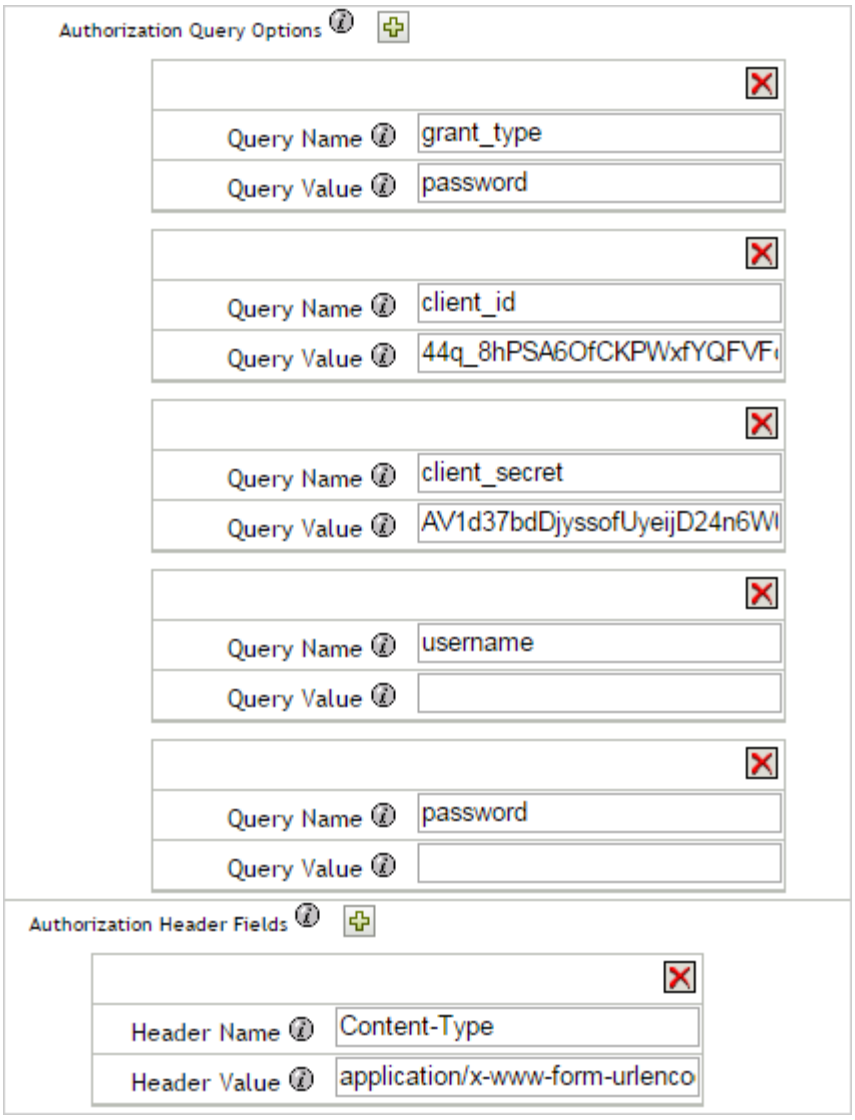


Figure 1-3 illustrates the Resource Owner Password Grant Type option.

Figure 1-3 *Resource Owner Password Grant Type*



Supporting Publish Mode

The Identity Manager driver for REST supports Publish as Publisher option.

If **Publish** is selected, the driver exposes the REST endpoints to receive the events from the connected RESTful service and then pushes the events to the Identity Vault.

Supporting Identity Manager Engine as a REST EndPoint

The REST driver exposes REST endpoints to the Identity Manager engine. This facilitates easy communication between external applications and services with eDirectory and Identity Manager engine via the REST API.

NOTE: The authentication header and content type are mandatory for REST methods.

Table 1-2 lists an example of POST REST method that the driver supports for a User class:

Table 1-2 *POST Method*

METHOD: POST	
User URI	http://ipaddress:port/User
Payload	{"association":"User2","Postal Code":["324324324"],"Surname":["User2"],"CN":["User 2"]}
Authorization	Basic c3lzdGVtL3N5c3RlbQ==
Content-Type	application/json
Response	201 Created

Table 1-3 lists an example of DELETE REST method that the driver supports:

Table 1-3 *DELETE Method*

METHOD: DELETE	
User URI	http://ipaddress:port/User/User2
Payload	Not required
Authorization	Basic c3lzdGVtL3N5c3RlbQ==
Content-Type	application/json
Response	200 OK

Table 1-4 lists an example of PUT REST method that the driver supports:

Table 1-4 *PUT Method*

METHOD: PUT	
User URI	http://ipaddress:portUser/User2
Authorization	Basic c3lzdGVtL3N5c3RlbQ==
Content-Type	application/json
Payload	{"Title":[{"add":["Manager"]}]}
Response	204 No Content

Table 1-5 lists an example of GET REST method that the driver supports:

Table 1-5 *GET Method*

METHOD: GET	
User URI	http://ipaddress:port/User?search-attr=given name eq ‘test*user’ and cn eq ‘test*’&read-attr=title
Payload	Not Applicable
Authorization	Basic c3lzdGVtL3N5c3RlbQ==
Content-Type	application/json
Response	{ "totalResults": 1, "results": [{ "src-dn": "\\GEN-REST1\\system\\servers\\TestUser", "class-name": "User", "Title": ["SE"] }]}

Synchronizing Information

Unlike most other drivers, the REST driver synchronizes protocols instead of objects. The driver includes the following features:

- HTTP transport of data between the Identity Vault and a Web service
- SSL connections using the HTTPS protocol
- Subscriber HTTP and HTTPS proxy servers
- Potential to act as an HTTP or HTTPS listener for incoming connections on the Publisher channel
- Potential extensibility through customized Java code

  For more information, see Appendix B, “Using Java Extensions,” on page 65.

Supporting Entitlements and Permission Collection and Reconciliation Service

The driver supports custom entitlements. In the Role-Based Services, assignments are made based on attributes of a user object. Entitlements standardize a method of recording this information on objects in the Identity Vault.

The REST driver supports Permission Collection and Reconciliation Service (PCRS) to map entitlements to resources and automatically assign those entitlements to users when permissions changes in connected system. The driver updates the Resource Catalog so that it reflects the exact state of user permissions in the connected system.

PCRS also helps in creating and automatically managing the relationship of identities to resource assignments. To map entitlements to resources, the REST driver uses either a CSV file or queries the connected system for entitlement values.

For a rest driver enabled with PCRS, if an administrator assigns a resource to a user in the User Application or in iManager, this change is reflected in the connected system, and similarly, if a connected system administrator makes a change to the user permission, that change is reflected in the Identity Vault and the corresponding resource is updated with the permission assignment.

The PCRS package contains the content necessary for permission collection and reconciliation service. If you want the driver to support permission collection and reconciliation service, ensure that these packages are installed on the driver. You can turn this functionality on or off using the new set of GCVs included with the driver.

To support only entitlements, turn off PCRS and create the entitlement objects using Designer and then redeploy the driver. If PCRS is turned on, the driver supports custom entitlements on both Subscriber and Publisher channels. If PCRS is turned off, the driver supports custom entitlements only on the Subscriber channel. For more information, refer to “Permission Collection and Reconciliation” on page 63.

NOTE: You should enable entitlements for the drivers only if you plan to use the User Application or Role-Based Entitlements with the drivers.

Prerequisites for Permission Collection and Reconciliation Service

To support PCRS, you must have the following:

- Identity Manager 4.6 or later
- Designer for Identity Manager 4.6 or later
- Managed System Gateway driver version 4.0.0.6 and later
- Driver Set Package:
  - Common Settings Advanced Edition Package (NOVLACOMSET 2.0.0 and later)
- Driver Package:
  - Driver-specific entitlements packages for REST driver.
  - Permission Collection and Reconciliation Service package (NOVLCOMPCRS 2.0.0 and later)

    This is the common PCRS package for defining custom entitlements on drivers such as REST, SOAP, and JDBC.
- Set up administrative user accounts and configure a password policy. For more information, see “Setting Up Administrative User Accounts” and “Setting Up Administrative Passwords” in the *NetIQ Identity Manager Driver Administration Guide*.

2 Installing the Driver Files

You can install the REST driver on the Identity Manager server or on a remote server using the Remote Loader.

To install the driver, you first need to install the driver files, install the driver packages, and then modify the driver configuration to suit your environment. This section describes how to install the driver files. For information on installing and configuring driver packages, see Chapter 3, “Creating a New Driver Object,” on page 27.

- “Prerequisites for Driver Installation” on page 25
- “Installing the REST Driver Files” on page 25

Prerequisites for Driver Installation

The installation and configuration process for the driver requires Identity Manager and/or Remote Loader, and Designer for Identity Manager. Before installing the driver, ensure that you download the following software to your Identity Manager environment:

- Identity Manager 4.6 or later
- Identity Manager Designer 4.6 or later

Installing the REST Driver Files

You can install the REST driver files on the Identity Manager server or a remote server that supports Remote Loader configuration. For more information about installing Remote Loader, see “Installing and Managing the Remote Loader” in the *NetIQ Identity Manager Setup Guide*.

3 Creating a New Driver Object

After the REST driver files are installed on the server where you want to run the driver (see Chapter 2, “Installing the Driver Files,” on page 25), you can create the driver in the Identity Vault. You do so by installing the driver packages and then modifying the driver configuration to suit your environment.

The following sections provide instructions to create the driver:

- “Creating the Driver Object in Designer” on page 27
- “Activating the Driver” on page 33
- “Adding Packages to an Existing Driver” on page 33
- “Creating Custom Entitlements” on page 34

Creating the Driver Object in Designer

The Designer tool helps you to create the REST driver object. You need to install the driver packages and them modify the configuration to suit your environment. After you create and configure the driver, you need to deploy it to the Identity Vault and start it.

- “Importing the Current Driver Packages” on page 27
- “Installing the Driver Packages” on page 28
- “Configuring the Driver Object” on page 31
- “Deploying the Driver Object” on page 31
- “Starting the Driver” on page 32

NOTE: NetIQ recommends that you use the new package management features provided in Designer to create the REST driver. You should not create the driver objects by using the new Identity Manager 4.0 and later or configuration files through iManager. This method of creating driver objects is no longer supported.

Importing the Current Driver Packages

The driver packages contain the items required to create a driver, such as policies, entitlements, filters, and Schema Mapping policies. These packages are only available in Designer and can be updated after they are initially installed. You must have the most current version of the packages in the Package Catalog before you can create a new driver object.

To verify that you have the most recent version of the driver packages in the Package Catalog:

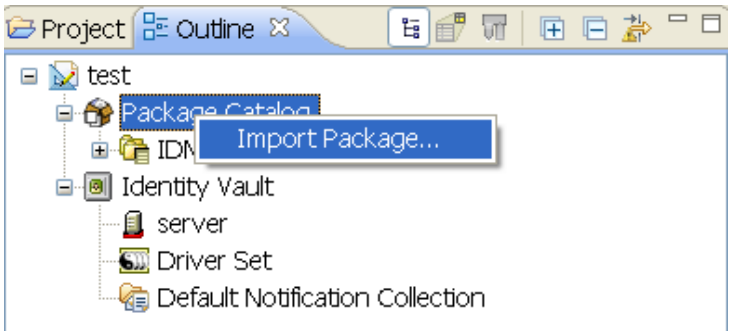
1. Open Designer.
2. In the toolbar, click **Help** > **Check for Package Updates**.
3. Click **OK** to update the packages

   or

   Click **OK** if the packages are up-to-date.

4 In the Outline view, right-click the Package Catalog.

5 Click **Import Package**.



6 Select any REST driver packages

or

Click **Select All** to import all of the packages displayed.

By default, only the base packages are displayed. Deselect **Show Base Packages Only** to display all packages.

7 Click **OK** to import the selected packages, then click **OK** in the successfully imported packages message.

8 After the current packages are imported, continue with "Installing the Driver Packages" on page 28.

Installing the Driver Packages

After you have imported the current driver packages into the Package Catalog, you can install the driver packages to create a new driver.

1 In Designer, open your project.

2 In the Modeler, right-click the driver set where you want to create the driver, then click **New** > **Driver**.

3 Select **REST Base**, then click **Next**.

NOTE: You can only select one base package.

---

4 Select the type of REST driver packages to install, then click **Next**.

5 Select the optional features to install for the REST driver, then click **Next**.

The options are:

Password Synchronization: This packages contains the policies that enable the REST driver to synchronize passwords. If you want to synchronize passwords, verify that this option is selected. For more information, see the *NetIQ Identity Manager Password Management Guide*.

Default JSON Configuration: This package contains the default JSON configurations.

Permission Collection and Reconciliation Services: This package contains policies for quick onboarding of custom entitlements and dynamic resource creation. This package contains GCVs to control the resource mapping. Select this package if you want to enable the entitlement onboarding feature for this driver. For more information about creating custom entitlements, see "Creating Custom Entitlements" on page 34.

For more information about PCRS, see “Synchronizing Permission Changes from the Connected Systems ” in the *NetIQ Identity Manager Driver Administration Guide*.

6 (Conditional) If there are package dependencies for the packages you selected to install, you must install them to install the selected package. Click **OK** to install the package dependency listed.

7 (Conditional) If more than one type of package dependency must be installed, you are presented with these packages separately. Continue to click **OK** to install any additional package dependencies.

8 On the Driver Information page, specify a name for the driver, then click **Next**.

9 On the Install REST Base page, fill in the following fields for the Subscriber options, then click **Next**:

Authentication Method: Select the authentication method for the REST driver.

NOTE: The authentication methods available are **Anonymous**, **Basic**, and **OAuth2.0**. You need to specify additional parameters depending upon the selected authentication method. For more information, see “Driver Configuration” on page 53.

Authorization Header Fields: Click the icon to create authentication header fields.

Truststore file: Specify the path and the name of the keystore file that contains the trusted certificates for the remote server to provide server authentication. For example, `c:\security\truststore`. Leave this field blank when server authentication is not used.

Set mutual authentication parameters: Select **Show** if you want to set mutual authentication information.

- **Keystore file:** Specify the path and the name of the keystore file that contains the trusted certificates for the remote server to provide mutual authentication. For example, `C:\security\keystore`. Leave this field blank when mutual authentication is not used.
- **Keystore password:** Specify the password for the keystore file. Leave this field blank when mutual authentication is not used.

HTTP Connection Timeout: Specify the HTTP connection time out value. The driver waits for the time specified and terminates the HTTP connection. The timeout value must be greater than zero.

Proxy host and port: Specify the host address and the host post when a proxy host and port are used. For example: 192.168.0.0:port. Choose an unused port number on your server. Otherwise, leave this field blank.

HTTP Errors to Retry: Specify the HTTP errors that must return a retry status. Error codes must be a list of integers separated by spaces. For example, 307 408 503 504.

Base URL of the REST Resources: Specify the URL of the REST server or Web service.

NOTE: The **Configure Resources to synchronize** option is not available in the Designer tool to configure the class-name. You should use iManager to configure the resources for the REST driver to start successfully. For example, if user is the class-name, ensure that you specify the **Schema name** as user under Resources section in the Driver Configuration.

---

10 On the Install REST Base page, fill in the following fields for the Publisher options, then click **Next**:

Publisher Setting: Specify the publisher setting for the REST driver.

If you select **Publish**, fill in the following parameters:

Listening IP address and port: Specify the IP address of the server where this driver is installed and the port that this driver listens on. You can specify 127.0.0.1 if there is only one network card installed in the server. Choose an unused port number on your server. For example: 127.0.0.1:port. The driver listens on this address for incoming requests, processes the requests, and returns a result.

Authentication Method: Select the authentication method for the REST driver.

NOTE: The authentication methods available are **Anonymous and Basic**. You need to specify additional parameters depending upon the selected authentication method. For more information, see "Driver Configuration" on page 53.

Polling interval in minutes: Specify the polling interval in minutes. Default is one minute.

KMO name: When this server is configured to accept HTTPS connections, this is the KMO name in eDirectory. The KMO name is the name before the - in the RDN. Leave this field blank when a keystore file is issued or when HTTPS connections are not used.

Keystore file: When this server is configured to accept HTTPS connections, this is the path and the name of the keystore file. For example; `C:\security\keystore`. Leave this field blank when a KMO name is used or when HTTPS connections are not used.

Keystore password: When this server is configured to accept HTTPS connections, this is the keystore file password. Leave this field blank when a KMO name is used or when HTTPS connections are not used.

Server key alias: When this server is configured to accept HTTPS connections, this is the key alias. Leave this field blank when a KMO name is used or when HTTPS connections are not used.

Server key password: When this server is configured to accept HTTPS connections, this is the key alias password (not the keystore password). Leave this field blank when a KMO name is used or when HTTPS connections are not used.

Require mutual authentication: When using SSL, it is common to do only server authentication. However, if you want to force both client and server to present certificates during the handshake process, select **Required**.

Heartbeat interval in minutes: Specify the heartbeat interval in minutes. Leave this field blank to turn off the heartbeat.

11 Fill in the following fields for the Remote Loader information, then click **Next**:

Connect To Remote Loader: Select **Yes** or **No** to determine if the driver will use the Remote Loader. For more information, see *Configuring the Remote Loader and Drivers* in the *NetIQ Identity Manager Setup Guide*.

If you select **No**, skip to Step 12. If you select **Yes**, use the following information to complete the configuration of the Remote Loader:

Host Name: Specify the IP address or DNS name of the server where the Remote Loader is installed and running.

Port: Specify the port number for this driver. Each driver connects to the Remote Loader on a separate port. The default value is 8090.

KMO: Specify the key name of the Key Material Object that includes keys and certificates for SSL. You use this parameter only when an SSL connection exists between the Remote Loader and the Metadirectory engine.

NOTE: When this server is configured to accept HTTPS connections, this is the KMO name in eDirectory. The KMO name is the name before the - in the RDN. Leave this field blank when a keystore file is issued or when HTTPS connections are not used.

Other Parameters: Specify any other parameter required in the connection string. The parameter must be a key-value pair. For example, paraName1=paraValue1.

Remote Loader Password: Specify a password to control access to the Remote Loader. It must be the same password that is specified as the Remote Loader password on the Remote Loader.

Driver Password: Specify a password for the driver to authenticate to the Metadirectory server. It must be the same password that is specified as the Driver Object Password on the Remote Loader.

12 Review the summary of tasks that will be completed to create the driver, then click **Finish**.

13 After you have installed the driver, you must change the configuration for your environment. Proceed to "Configuring the Driver Object" on page 31.

Configuring the Driver Object

After the driver packages are installed, you need to configure the driver before it can run. You should complete the following tasks to configure the driver:

- **Configure the driver parameters:** There are many settings that can help you customize and optimize the driver. The settings are divided into categories such as Driver Configuration, Engine Control Values, and Global Configuration Values (GCVs). Although it is important for you to understand all of the settings, your first priority should be to review the Driver Parameters located on the Driver Configuration page. The Driver Parameters let you configure the publication method and other parameters associated with the Publisher channel.
- **Customize the driver policies and filter:** The driver policies and filter control data flow between the Identity Vault and the application. You should ensure that the policies and filters reflect your business needs. For instructions, see Chapter 5, "Customizing the Driver for RESTful Services," on page 43.
- **Set Up a Secure HTTPS Connection:** The connection between the driver and the RESTful connected system can be configured to use a secure HTTPS connection rather than an HTTP connection.

After completing the configuration tasks, continue with either Creating Custom Entitlements or Deploying the Driver Object.

Deploying the Driver Object

After the driver object is created in Designer, it must be deployed into the Identity Vault.

1 In Designer, open your project.

2 In the Modeler, right-click the driver icon or the driver line, then select **Live > Deploy**.

3 If you are authenticated to the Identity Vault, skip to Step 4; otherwise, specify the following information, then click **OK**:

Host: Specify the IP address or DNS name of the server hosting the Identity Vault.

Username: Specify the DN of the user object used to authenticate to the Identity Vault.

Password: Specify the user's password.

4 Read the deployment summary, then click **Deploy**.

5 Read the message, then click **OK**.

6 Click **Define Security Equivalence** to assign rights to the driver.

The driver requires rights to objects within the Identity Vault. The Admin user object is most often used to supply these rights. However, you might want to create a DriversUser (for example) and assign security equivalence to that user. Whatever rights that the driver needs to have on the server, the DriversUser object must have the same security rights.

6a Click **Add**, then browse to and select the object with the correct rights.

6b Click **OK** twice.

For more information about defining a Security Equivalent User in objects for drivers in the Identity Vault, see the NetIQ Identity Manager Security Guide

7 Click **Exclude Administrative Roles** to exclude users that should not be synchronized.

You should exclude any administrative User objects (for example, Admin and DriversUser) from synchronization.

7a Click **Add**, then browse to and select the user object you want to exclude, then click **OK**.

7b Repeat Step 7a for each object you want to exclude, then click **OK**.

8 Click **OK**.

9 Continue with the next section, Starting the Driver.

Starting the Driver

When a driver is created, it is stopped by default. To make the driver work, you must start the driver. Identity Manager is an event-driven system, so after the driver is started, it won't do anything until an event occurs. You can use iManager or `dxevent` commands to start the driver.

To start the driver using Designer:

1 In Designer, open your project.

2 In the Modeler, right-click the driver icon or the driver line, then select **Live > Start Driver**.

To start the driver using iManager:

To start the driver using iManager:

1 In iManager, click to display the Identity Manager Administration page.

2 Click **Identity Manager Overview**.

3 Browse to and select the driver set object that contains the driver you want to start.

4 Click the driver set name to access the Driver Set Overview page.

5 Click the upper right corner of the driver, then click **Start driver**.

IMPORTANT: When you start the driver for the first time, don't add new users to the Publisher channel until the first polling interval completes because the driver treats all users as existing users and stores them in the change cache without sending them to the Identity Manager engine. It sends the new users to the Identity Manager engine from the next polling interval. Therefore, ensure that new users are added to the Publisher channel after the first polling cycle completes.

Activating the Driver

The Identity Manager driver for REST is part of the Identity Manager Integration Module for Tools. This integration module includes the following drivers:

- Identity Manager driver for Delimited Text
- Identity Manager driver for SOAP
- Identity Manager driver for REST

This integration module requires a separate activation. After purchasing the integration module, you will receive activation details in your NetIQ Customer Center.

If you create a new REST driver in a driver set that already includes an activated driver from this integration module, the new driver inherits the activation from the driver set.

If you create the driver in a driver set that has not been previously activated with this integration module, the driver will run in the evaluation mode for 90 days. You must activate the driver with this integration module during the evaluation period; otherwise, the driver will be disabled.

If driver activation has expired, `ndstrace` displays an error message indicating that you need to reactivate the driver to use it. For information on activation, refer to Activating Identity Manager in the *NetIQ Identity Manager Setup Guide*.

Adding Packages to an Existing Driver

You can add new functionality to an existing driver by adding new packages to it.

1. Right-click the driver, then click **Properties**.
2. Click **Packages**, then upgrade the already installed REST Base package.
   - **2a** Select the package from the list of packages, then click the **Select Operation** cell.
   - **2b** Click **Upgrade** from the drop-down list, then click **Apply**.
   - **2c** Click **OK** to close the Package Management page.

     You can upgrade the Password Synchronization package in a similar way.
3. Click the **Add Packages** icon .
4. Select the packages to install.
5. (Optional) If you want to see all available packages for the driver, clear the **Show only applicable package versions** option, if you want to see all available packages for the driver, then click **OK**.

   This option is only displayed on drivers. By default, only the packages that can be installed on the selected driver are displayed.
6. Click **Apply** to install all of the packages listed with the Install operation.
7. (Conditional) Fill in the fields with appropriate information to install the package you selected for the driver, then click **Next**.
8. Read the summary of the installation, then click **Finish**.
9. Click **OK** to close the Package Management page after you have reviewed the installed packages.
10. Modify the driver configuration settings. See “Configuring the Driver Object” on page 31.
11. Deploy the driver. See “Deploying the Driver Object” on page 31.

12 Start the driver. See “Starting the Driver” on page 32.

13 Repeat Step 1 through Step 9 for each driver where you want to add the new packages.

Creating Custom Entitlements

In an enterprise, entitlement servers as granting agent. Custom entitlements helps to control the permission parameters such as user accounts, group memberships., resources access.

The Permission Collection and Reconciliation Services (PCRS) help you to create entitlements that you can map with resources in the Identity Vault. You can dynamically create resources with custom entitlement populated with permission values of connected system, and permission assignments between Identity Manager resource/entitlement model and connected systems.

Unlike other drivers, the REST packages do not include the default entitlements. However, you can customize the REST driver to support the default entitlements such as useracccount and group membership. The driver also supports other entitlements such as printers, conference rooms, floor access, and so forth.

The Identity Manager driver for REST installed with Permission Collection and Reconciliation Service packages perform the following actions:

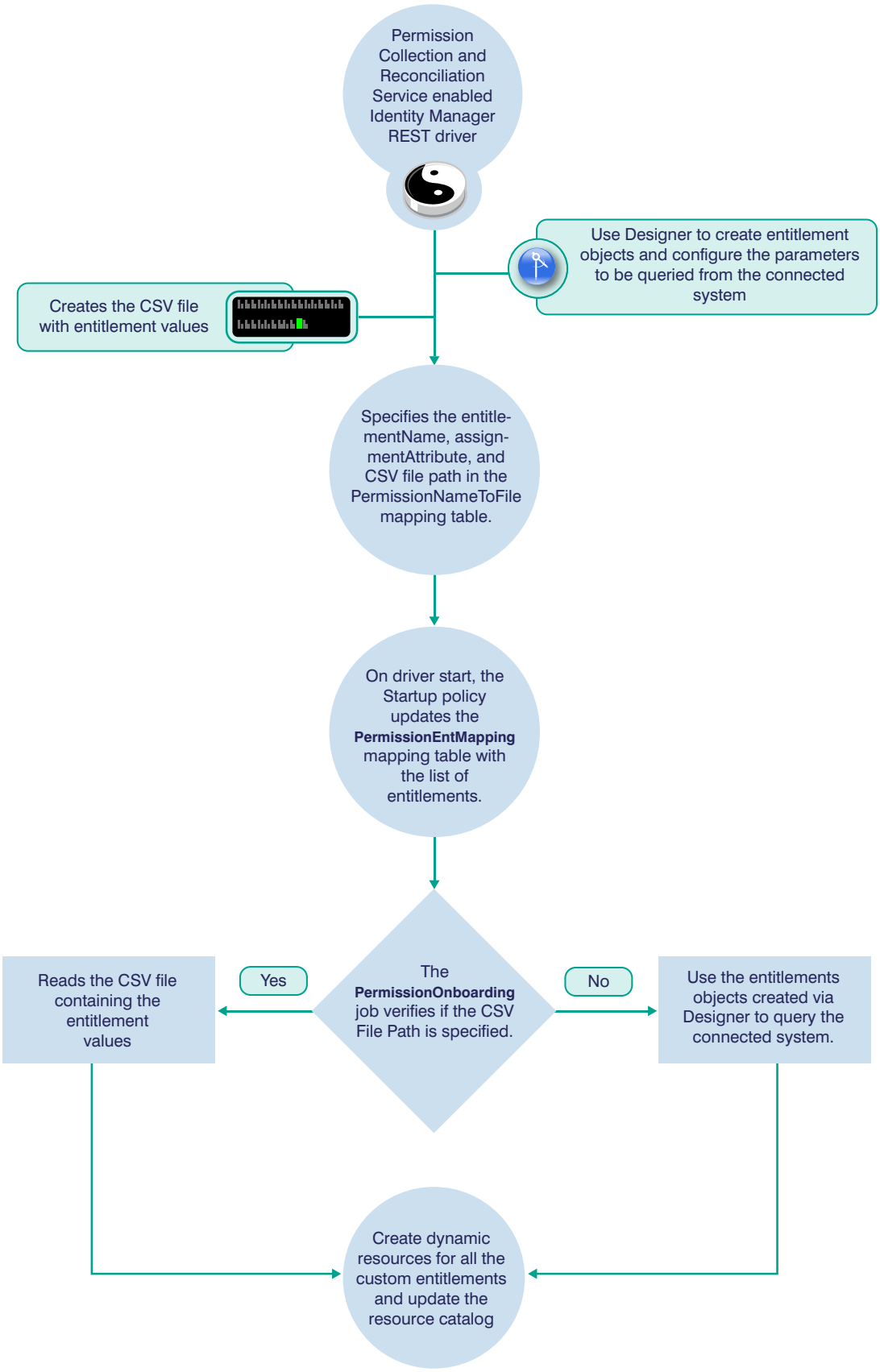
- Provide a way to create custom entitlements and resources specific to your environment.
- Reconcile resource or permission assignments between the Identity Vault and connected systems.

For example, let’s consider a deployment scenario in which the Identity Vault connects with the SCIM server as the target system.

You can define a new role “Manager” in the SCIM server and assign users to this role. You can define the Manager role as a custom entitlement and when the driver returns the custom entitlements from the connected system, all users assigned with this role is created and mapped as resources in the Identity Vault. The REST driver synchronizes the permission updates from the connected system and updates the Identity and Resource Catalog.

Figure 3-1 illustrates the process of creating resources using PCRS:

Figure 3-1 *Permission Collection and Reconciliation Service Flow*



IMPORTANT: In the Driver Configuration, ensure that you configure the resources in the Subscriber Setting. You should configure the Schema name as Entitlement and the required handlers for the PCRS to work. For more information, see “Resources” on page 59.

For creating custom entitlements, you should specify the entitlement value, attribute, and CSV file path in the `PermissionNameToFile` mapping table in the Driver Configuration page. The REST driver uses either the CSV file or query the connected system to populate entitlement values. If the custom entitlements are not dependent on the values from the connected system, use the CSV file to provide the entitlement values to create entitlements and resources. If the custom entitlements are dependent on the values from the connected system, create entitlements objects using Designer and specify the information required for querying the connected system.

In the CSV file, the administrator must provide the entitlement values of the connected system in the specified format that includes the entitlement value, the display name and the description. You must place the CSV file containing the entitlement values on the server where Identity Manager engine is installed. Ensure that you update the **csvFile** column in the `PermissionNameToFile` mapping table with the correct path for the CSV file. The REST driver access the custom entitlement information from the CSV file.

For any REST driver enabled with PCRS, to create custom entitlements, you should create the entitlements objects. The Designer tool helps you to create the entitlement objects. You can define the values that must be queried for from the connected system in these entitlement objects. For example, you can create entitlement objects for users and groups. For more information about creating entitlement objects, see ““Valued Entitlement that Queries an External Application”” in the *NetIQ Designer for Identity Manager Administration Guide*.

The `PermissionOnboarding` job is a standard Identity Manager job and is available in the entitlement package. During the driver startup, the `PermissionOnboarding` job runs and queries the connected system for resource updates. After you create, deploy, and start the driver, the driver automatically reads the `PermissionNameToFile` mapping table. If the CSV file name is available in `PermissionNameToFile` mapping table, the driver consumes the entitlements values from the file and creates the custom entitlements. If the **csvFile** column in the `PermissionNameToFile` mapping table is empty, the driver queries the connected system using the entitlement objects. New resources are created with the entitlement values from the CSV file or values obtained as the query response.

The new custom entitlement and the corresponding resource object is created in the Resource Catalog. When the permission assignments change in connected system, the driver policies consume the modified permission values and update the Resource Catalog.

IMPORTANT

- If you are querying the connected system to create resources, ensure that entitlement objects are configured correctly. If the entitlement object configuration is wrong, the `PermissionOnboarding` job fails to create resources for all entitlements provided in the `PermissionNameToFile` mapping table editor.
- The entitlement assignments are only for a user class. A user resource is eligible to reconcile and synchronize the entitlement changes. However the assignment attributes can differ as per the requirement.

CSV File Format

The REST driver consumes the entitlement information from the CSV file, which is present on the server where Identity Manager is installed. This file must contain values of the connected system permissions in the format specified below. The connected system administrator should maintain a separate CSV file for every custom entitlement.

For example, a CSV file can contain details about granting access to the users for the *ResourcesAccess* entitlement. A CSV file that contains *ResourceAccess* entitlement details represents this information in the following format:

```
Resource A,Google,The google access
Resource B,Twitter,The twitter access
Resource C,Facebook,The facebook access
```

where *Resource A* is the entitlement value, *Google* is the display name in the User Application for the entitlement value *Resource A*, and *The google access* is the description for the entitlement value. This description is displayed in the User Application.

4 Upgrading an Existing Driver

The following sections provide information to help you upgrade an existing driver:

- “Supported Upgrade Paths” on page 39
- “What’s New” on page 39
- “Upgrading the Driver” on page 39

Supported Upgrade Paths

You can upgrade from 1.0 version of the REST driver to 1.0.0.1 and 1.0.11 versions.

What’s New

What’s New in Version 1.0.1.1

This version of the driver does not include any new features.

What’s New in Version 1.0.0.1

This version of the driver does not include any new features.

Upgrading the Driver

The REST driver upgrade process involves upgrading the installed driver packages and updating the driver files.

This section provides general instructions for updating a driver. For information about updating the driver to a specific version, search for that driver patch in the NetIQ Patch Finder Download Page and follow the instructions from the Readme file accompanying the driver patch release.

- “Upgrading the Installed Packages” on page 39
- “Applying the Driver Patch” on page 40

Upgrading the Installed Packages

1 Download the latest available packages.

To configure Designer to automatically read the package updates when a new version of a package is available, click **Windows > Preferences > NetIQ > Package Manager > Online Updates** in Designer. For more information about managing packages, see the *NetIQ Designer for Identity Manager Administration Guide*.

2 Upgrade the installed packages.

2a Open the project containing the driver.

2b Right-click the driver for which you want to upgrade an installed package, then click **Driver > Properties**.

2c Click **Packages**.

If there is a newer version of a package, there is check mark displayed in the Upgrades column.

2d Click **Select Operation** for the package that indicates there is an upgrade available.

2e From the drop-down list, click **Upgrade**.

2f Select the version that you want to upgrade to, then click **OK**.

NOTE: Designer lists all versions available for upgrade.

2g Click **Apply**.

2h (Conditional) Fill in the fields with appropriate information to upgrade the package, then click **Next**.

Depending on which package you selected to upgrade, you must fill in the required information to upgrade the package.

2i Read the summary of the packages that will be installed, then click **Finish**.

2j Review the upgraded package, then click **OK** to close the Package Management page.

For detailed information, see the Upgrading Installed Packages in the *NetIQ Designer for Identity Manager Administration Guide*.

Applying the Driver Patch

The driver patch updates the driver files. You can install the patch as a `root` or `non-root` user.

- "Prerequisites" on page 40
- "Applying the Patch as a Root User" on page 41
- "Applying the Patch as a Non-Root User" on page 41

Prerequisites

Before installing the patch, complete the following steps:

1 Take a back-up of the current driver configuration.

2 (Conditional) If the driver is running with the Identity Manager engine, stop the Identity Vault and the driver instance.

3 (Conditional) If the driver is running with a Remote Loader instance, start the Remote Loader instance and the driver instance.

4 In a browser, navigate to the NetIQ Patch Finder Download Page.

5 Under **Patches**, click **Search Patches**.

6 Specify **Identity Manager *nn* REST DRIVER nn** in the search box.

7 Download and unzip the contents of the patch file to a temporary location on your server.

For example, `IDM45_REST_1001.zip`.

Applying the Patch as a Root User

In a root installation, the driver patch installs the driver files RPMs in the default locations on Linux. On Windows, you need to manually copy the files to the default locations.

1. Update the driver files:
   - **Linux:** To upgrade the existing RPM, log in as `root` and run the following command in a command prompt:

     `rpm -Uvh <Driver Patch File Temporary Location>/linux/netiq-DXMLRESTDrv.rpm`

     For example, `rpm -Uvh <IDM4.5_REST_1001.zip>/linux/netiq-DXMLRESTDrv.rpm`
   - **Windows:** Navigate to the `<Extracted Driver Patch File Temporary Location>\windows` folder and copy the following files to `<IdentityManager installation>\NDS\lib` or `<IdentityManager installation>\RemoteLoader\lib` folder:
     - `RESTCommon.jar`
     - `RESTDriverShim.jar`
     - `RESTUtil.jar`
2. (Conditional) Start the Remote Loader instance.
3. (Conditional) Start the REST driver.

Applying the Patch as a Non-Root User

1. Verify that `<non-root eDirectory location>/rpm` directory exists and contains the file, `_db.000`.

   The `_db.000` file is created during a non-root installation of the Identity Manager engine. Absence of this file might indicate that Identity Manager is not properly installed. Reinstall Identity Manager to correctly place the file in the directory.
2. To set the `root` directory to non-`root` eDirectory location, enter the following command in the command prompt:

   `ROOTDIR=<non-root eDirectory location>`

   This will set the environmental variables to the directory where eDirectory is installed as a non-`root` user.
3. To install the driver files, enter the following command:

   ```
   rpm --dbpath $ROOTDIR/rpm -Uvh --relocate=/usr=$ROOTDIR/opt/novell/eDirectory
   --relocate=/etc=$ROOTDIR/etc --relocate=/opt/novell/eDirectory=$ROOTDIR/opt/
   novell/eDirectory --relocate=/opt/novell/dirxml=$ROOTDIR/opt/novell/dirxml --
   relocate=/var=$ROOTDIR/var --badreloc --nodeps --replacefiles <rpm-location>
   ```

   For example, to install the REST driver RPM, use this command:

   ```
   rpm --dbpath $ROOTDIR/rpm -Uvh --relocate=/usr=$ROOTDIR/opt/novell/eDirectory
   --relocate=/etc=$ROOTDIR/etc --relocate=/opt/novell/eDirectory=$ROOTDIR/opt/
   novell/eDirectory --relocate=/opt/novell/dirxml=$ROOTDIR/opt/novell/dirxml --
   relocate=/var=$ROOTDIR/var --badreloc --nodeps --replacefiles /home/user/
   novell-DXMLRESTDrv.rpm
   ```

5 Customizing the Driver for RESTful Services

The following sections provide information to help you understand the available customization to make the driver connect to any RESTful service:

- “Using Java Extensions” on page 43
- “Changing the JSON/XML Payload” on page 43
- “Using driver-operation-data” on page 43

Using Java Extensions

Use the java extensions to modify a REST request or response before it is submitted or received on the Subscriber or Publisher channels. For more information, see Appendix B, “Using Java Extensions,” on page 65.

Changing the JSON/XML Payload

After you install the default JSON package, you can transform the payload generated in the `<driver-operation-data>` to a format supported by your RESTful service.

Using driver-operation-data

You can use the policies to add a new `<driver-operation-data>` element to the Subscriber channel. The driver processes the `<driver-operation-data>` element irrespective of the configured handlers. For more information, see “Understanding Driver Operation Data” on page 16.

6 Securing Communication

If the remote Web service you are accessing allows HTTPS connections, you can configure the driver to take advantage of this increased security.

IMPORTANT: Only certificates from a Java keystore are accepted. Make sure that the keystore for the certificates is a Java keystore.

The following sections provide instructions for creating a secure connection:

- "Configuring the Publisher Channel" on page 45
- "Configuring the Subscriber Channel" on page 46

Configuring the Publisher Channel

The Publisher channel publishes the information from the RESTful service to the Identity Vault. To establish a secure connection for the Publisher Channel, you need a keystore or a KMO containing a certificate issued by the certificate authority that signed the server's certificate.

1. Create a server certificate in iManager.:
   - **1a** In the **Roles and Tasks** view, click **NetIQ Certificate Server > Create Server Certificate**.
   - **1b** Browse to and select the server object where the REST driver is installed.
   - **1c** Specify a certificate nickname.
   - **1d** Select **Standard** as the creation method, then click **Next**.
   - **1e** Click **Finish**, then click **Close**.
2. Export a self-signed certificate from the certificate authority in eDirectory:
   - **2a** In the **Roles and Tasks** view, click **Directory Administration > Modify Object**.
   - **2b** Select your tree's certificate authority object, then click **OK**.

     It is usually found in the Security container and is named something like *TREENAME CA.Security*.
   - **2c** Click **Certificate > Self Signed Certificate**.
   - **2d** Click **Export**.
   - **2e** When asked if you want to export the private key with the certificate, click **No**, then click **Next**.
   - **2f** Based on the client to be accessing the Web service, select either **File in binary DER format** or **File in Base64 format** for the certificate, then click **Next**.

     If the client uses a Java-based keystore or trust store, then you can choose either format.
   - **2g** Click **Save the exported certificate to a file**.
   - **2h** Click **Save**, then browse to a known location on your computer.
   - **2i** Click **Save**, then click **Close**.
3. Import the self-signed certificate into the client's trust store:

The steps to import the certificate vary depending on the client that connects to the Publisher channel's HTTPS listener. If the client uses a typical Java keystore, you can perform the following steps to create the keystore:

3a Use the keytool executable that is included with any Java JDK.

For more information on keytool, see Keytool - Key and Certificate Management Tool (http://java.sun.com/j2se/1.4.2/docs/tooldocs/windows/keytool.html).

3b Enter the following command at a command prompt:

```
keytool -import -file name_of_cert_file -trustcacerts -noprompt
-keystore filename -storepass password
```

For example:

```
keytool -import -file tree_ca_root.b64 -trustcacerts -noprompt -keystore
dirxml.keystore -storepass novell
```

4 Configure the Publisher channel to use the server certificate you created in Step 1:

4a In iManager, in the **Roles and Tasks** view, click **Identity Manager > Identity Manager Overview**.

4b Locate the driver set containing the REST driver, then click the driver's icon to display the Identity Manager Driver Overview page.

4c In the Identity Manager Driver Overview page, click the driver's icon again, then scroll to **Publisher Settings**.

4d In the **KMO name** setting, specify the certificate nickname you used in Step 1.

5 Click **Apply**, then click **OK**.

Configuring the Subscriber Channel

The Subscriber channel sends information from the Identity Vault to the Web service. To establish a secure connection for the Subscriber channel, you need a trust store containing a certificate issued by the certificate authority that signed the server's certificate. See "Configuring the Publisher Channel" on page 45 for an example.

1 Make sure you have a server certificate signed by a certificate authority.

2 Import the certificate into your trust store or create a new trust store by entering the following command at the command prompt:

```
keytool -import -file name_of_cert_file -trustcacerts -noprompt -keystore
filename -storepass password
```

For example:

```
keytool -import -file tree_ca_root.b64 -trustcacerts -noprompt -keystore
dirxml.keystore -storepass novell
```

For more information on keytool, see Keytool - Key and Certificate Management Tool (http://java.sun.com/j2se/1.4.2/docs/tooldocs/windows/keytool.html).

3 Configure the Subscriber channel to use the trust store you created in Step 2:

3a In iManager, in the **Roles and Tasks** view, click **Identity Manager > Identity Manager Overview**.

3b Locate the driver set containing the REST driver, then click the driver's icon to display the Identity Manager Driver Overview page.

3c On the Identity Manager Driver Overview page, click the driver's icon again, then scroll to **Subscriber Settings**.

3d In the **Keystore File** setting, specify the path to the trust store you created in Step 2.

4 Click **Apply**, then click **OK**.

7 Managing the Driver

As you work with the REST driver, there are a variety of management tasks you might need to perform, including the following:

- Starting, stopping, and restarting the driver
- Viewing driver version information
- Using Named Passwords to securely store passwords associated with the driver
- Monitoring the driver's health status
- Backing up the driver
- Inspecting the driver's cache files
- Viewing the driver's statistics
- Using the DirXML Command Line utility to perform management tasks through scripts
- Securing the driver and its information

Because these tasks, as well as several others, are common to all Identity Manager drivers, they are included in one reference, the *NetIQ Identity Manager Driver Administration Guide*.

8 Troubleshooting the Driver

You can log Identity Manager events by using the Event Auditing Service. Using this service in combination with the driver log level setting provides you with tracking control at a very granular level. For more information, see the *Administrator Guide to NetIQ Identity Reporting*.

This section contains the following information on error messages:

- "Driver Shim Errors" on page 51
- "Troubleshooting Driver Processes" on page 51
- "Exception Reported After Upgrading to REST Driver 1.0.0.1" on page 51
- "Driver Reports an Error When a Password or an Attribute Value Contains the < Character" on page 52

Driver Shim Errors

The following errors might occur in the core driver shim. Error messages that contain a numerical code can have various messages, depending on the application or Web service.

- "Issues with commons-codec-1.3.jar" on page 51

Issues with commons-codec-1.3.jar

Explanation:	The driver initialization fails if installed on Remote Loader set up.
Possible Cause:	Unsupported version of commons-codec-1.3.jar.
Action:	Replace the commons-codec-1.3.jar file with the latest version of commons-codec-1.6.jar shipped along with the driver packages.

Troubleshooting Driver Processes

Viewing driver processes is necessary to analyze unexpected behavior. To view the driver processing events, use DSTrace. You should only use it during testing and troubleshooting the driver. Running DSTrace while the drivers are in production increases the utilization on the Identity Manager server and can cause events to process very slowly. For more information, see "Viewing Identity Manager Processes" in the *NetIQ Identity Manager Driver Administration Guide*.

Exception Reported After Upgrading to REST Driver 1.0.0.1

Issue: After upgrading, the driver reports the following error:

```
Message: Code(-9010) An exception occurred: java.lang.NoSuchMethodError:
org.apache.http.impl.client.HttpClientBuilder.setConnectionManagerShared(Z)Lorg/
apache/http/impl/client/HttpClientBuilder;
at
com.novell.nds.dirxml.driver.rest.HTTPSubscriberConnectionInfo.makeHttpClient(Http
RESTOperations.java:547)
```

Workaround: Include httpclient and httpcore 441 jar files in the REST driver installation directory.

1. Download httpclient 4.4.1 jar file.
2. Download httpcore 4.4.1 jar file.

Driver Reports an Error When a Password or an Attribute Value Contains the < Character

Issue: The JSON converter does not accept the “<” character. However, this issue does not occur with the ">" character:

Workaround: No workaround is available.

A Driver Properties

This section provides information about the Driver Configuration and Global Configuration Values properties for the REST driver. These are the only unique properties for drivers. All other driver properties (Named Password, Engine Control Values, Log Level, and so forth) are common to all drivers. Refer to "Driver Properties" in the *NetIQ Identity Manager Driver Administration Guide* for information about the common properties.

The information is presented from the viewpoint of iManager. If a field is different in Designer, it is marked with a Designer icon.

- "Driver Configuration" on page 53
- "Global Configuration Values" on page 62

Driver Configuration

In iManager:

1. Click to display the Identity Manager Administration page.
2. Open the driver set that contains the driver whose properties you want to edit:
   - **2a** In the **Administration** list, click **Identity Manager Overview**.
   - **2b** If the driver set is not listed on the Driver Sets tab, use the Search In field to search for and display the driver set.
   - **2c** Click the driver set to open the Driver Set Overview page.
3. Locate the driver icon, then click the upper right corner of the driver icon to display the **Actions** menu.
4. Click **Edit Properties** to display the driver's properties page.

   By default, the Driver Configuration page displays.

In Designer:

1. Open a project in the Modeler.
2. Right-click the driver icon or line, then select click **Properties > Driver Configuration.**

The Driver Configuration options are divided into the following sections:

- "Driver Module" on page 54
- "Authentication" on page 54
- "Startup Option" on page 54
- "Driver Parameters" on page 55
- "ECMAScript" on page 61
- "Global Configuration" on page 62

Driver Module

The driver module changes the driver from running locally to running remotely or the reverse.

Java: Use this option to specify the name of the Java class that is instantiated for the shim component of the driver. This class can be located in the `classes` directory as a class file, or in the `lib` directory as a `.jar` file. If this option is selected, the driver is running locally. Select this option to run the driver locally.

The Java class name is: `com.novell.nds.dirxml.driver.rest.RESTDriverShim`

Native: This option is not used with the REST driver.

Connect to Remote Loader: Used when the driver is connecting remotely to the connected system. Designer includes two suboptions:

- **Remote Loader Client Configuration for Documentation**: Includes information on the Remote Loader client configuration when Designer generates documentation for the driver.
- **Driver Object Password**: Specifies a password for the Driver object. If you are using the Remote Loader, you must enter a password on this page. Otherwise, the remote driver does not run. The Remote Loader uses this password to authenticate itself to the remote driver shim.

Name: Displays the java class name.

Driver Object Password: Use this option to set a password for the driver object. If you are using the Remote Loader, you must enter a password on this page or the remote driver does not run. This password is used by the Remote Loader to authenticate itself to the remote driver shim.

Authentication

The authentication section describes the parameters required for authentication to the connected system. This section is not applicable for the Identity Manager driver for REST. The authentication method for REST driver is Anonymous, Basic or OAuth2.0.

Startup Option

The Startup Option section allows you to set the driver state when the Identity Manager server is started.

Auto start: The driver starts every time the Identity Manager server is started.

Manual: The driver does not start when the Identity Manager server is started. The driver must be started through Designer or iManager.

Disabled: The driver has a cache file that stores all of the events. When the driver is set to Disabled, this file is deleted and no new events are stored in the file until the driver state is changed to Manual or Auto Start.

Driver Parameters

The Driver Parameters section lets you configure the driver-specific parameters. When you change driver parameters, you tune driver behavior to align with your network environment.

The parameters are presented by category:

- "Driver Settings" on page 55
- "Subscriber Settings" on page 55
- "Resources" on page 59
- "Publisher Options" on page 60

Driver Settings

Custom Java Extensions: Select **Show** if you have developed custom Java classes to extend the driver shim's functionality. Otherwise, select **Hide**.

- **Document Handling:** Select **Implemented** if you have developed a custom Java class to process data as XML documents. Otherwise, select **None**.
  - **Class:** Specify the class by using a complete package identifier. For example, `com.novell.DocumentModifier`.
  - **Init Parameter:** Specify the parameter to pass to the init() method of the specified class. The init method is responsible for parsing the information contained in this string. Leave this field blank if the configuration string is not required for the class.
- **Schema:** Select **Implemented** if you have developed a custom Java class to provide the application schema to the driver and specify the **Class** and **Init Parameter** values. Otherwise, select **None**.

For more information, see Appendix B, "Using Java Extensions," on page 65.

Subscriber Settings

Authentication Method: Select the method for authentication with the RESTful service. The available options are:

- **Anonymous:** The user name and password is not required in Anonymous authentication method.
- **Basic:** The driver uses the specified ID and password for authentication when processing the requests.
- **OAuth2.0:** The driver uses the specified access token URL, ID and password for authentication when processing the request.

If **Anonymous** is selected, fill in the following parameters:

Parameters	Description
Authorization Header Fields	Click the icon to create authorization header fields. ◆ **Header Name:** If the remote server requires an authentication ID, specify the ID in the field. Otherwise, leave the field empty. ◆ **Header Value:** Specify the authentication password for the remote server if you specified an header name. Otherwise, leave the field empty.
Truststore file	Specify the name and path of the keystore file containing the trusted certificates used when the remote server is configured to provide server authentication. For example, `c:\security\truststore`. Leave this field empty when server authentication is not used.
Set mutual authentication parameter	Specify **Show** to set mutual authentication information. Specify **Hide** to not use mutual authentication. ◆ **Keystore file:** Specify the path and the name of the keystore file that contains the trusted certificates for the remote server to provide mutual authentication. For example, `C:\security\keystore`. Leave this field blank when mutual authentication is not used. ◆ **Keystore password:** Specify the password for the keystore file. Leave this field blank when mutual authentication is not used.
Http Connection Timeout	Specify the HTTP connection timeout value. The driver waits for the time specified and terminates the HTTP connection. The timeout value must be greater than zero.
Proxy host and port	Specify the host address and the host port when a proxy host and port are used. For example: 192.10.1.3:18180. Or, if a proxy host and port are not used, leave this field empty.
HTTP errors to retry	Specify the HTTP errors that must return a retry status. Error codes must be a list of integers separated by spaces. For example, 307 408 503 504.
Base URL for REST Resources	Specify the common part of the REST resource URL. This is the part of the URL remaining after excluding the URL extension of the resource. For example, `http://ipaddress:port/`.

If **Basic** is selected, fill in the following parameters:

Parameters	Description
Authentication ID	Specify the authentication ID used for basic authorization on the HTTP header.
Authentication Password	Specify the authentication password used for basic authorization on the HTTP header.
Authorization Header Fields	Click the icon to create authorization header fields. ◆ **Header Name:** If the remote server requires an authentication ID, specify the ID in the field. Otherwise, leave the field empty. ◆ **Header Value:** Specify the authentication password for the remote server if you specified an header name. Otherwise, leave the field empty.
Truststore file	Specify the name and path of the keystore file containing the trusted certificates used when the remote server is configured to provide server authentication. For example, `c:\security\truststore`. Leave this field empty when server authentication is not used.
Set mutual authentication parameters	Specify **Show** to set mutual authentication information. Specify **Hide** to not use mutual authentication. ◆ **Keystore file:** Specify the path and the name of the keystore file that contains the trusted certificates for the remote server to provide mutual authentication. For example, `C:\security\keystore`. Leave this field blank when mutual authentication is not used. ◆ **Keystore password:** Specify the password for the keystore file. Leave this field blank when mutual authentication is not used.
Http Connection Timeout	Specify the HTTP connection timeout value. The driver waits for the time specified and terminates the HTTP connection. The timeout value must be greater than zero.
Proxy host and port	Specify the host address and the host port when a proxy host and port are used. For example: 192.10.1.3:18180. Or, if a proxy host and port are not used, leave this field empty.
HTTP errors to retry	Specify the HTTP errors that must return a retry status. Error codes must be a list of integers separated by spaces. For example, 307 408 503 504.
Base URL for REST Resources	Specify the common part of the REST resource URL. This is the part of the URL remaining after excluding the URL extension of the resource. For example, `http://ipaddress:port/`.

If **OAuth2.0** is selected, fill in the following parameters:

Parameters	Description
Access Token URL	Specify the URL of the server used for requesting token access.
User Name	Specify the user name for authentication. This parameter is optional.
User Password	Specify the password for authentication. This parameters is optional
Authorization Query Options	Click the icon to create authentication query options for OAuth2.0 authorization method. The supported OAuth authorization types for REST driver are Client Credentials and `Resource Owner Credentials`. You can create any one of these authorization types. ◆ **Query Name:** Specify the name of the query. For example, `grant_type`. You also can configure `client_id`, `client_secret`, and `resource` as query names. ◆ **Query Value:** Specify the value for the query. For example, `client_crendentials` or `password`.
Authorization Header Fields	Click the icon to create authorization header fields. ◆ **Header Name:** If the remote server requires an authentication ID, specify the ID in the field. Otherwise, leave the field empty. ◆ **Header Value:** Specify the authentication password for the remote server if you specified an header name. Otherwise, leave the field empty.
Truststore file	Specify the name and path of the keystore file containing the trusted certificates used when the remote server is configured to provide server authentication. For example, `c:\security\truststore`. Leave this field empty when server authentication is not used.
Set mutual authentication parameters	Specify **Show** to set mutual authentication information. Specify **Hide** to not use mutual authentication. ◆ **Keystore file:** Specify the path and the name of the keystore file that contains the trusted certificates for the remote server to provide mutual authentication. For example, `C:\security\keystore`. Leave this field blank when mutual authentication is not used. ◆ **Keystore password:** Specify the password for the keystore file. Leave this field blank when mutual authentication is not used.

Parameters	Description
Http Connection Timeout	Specify the HTTP connection timeout value. The driver waits for the time specified and terminates the HTTP connection. The timeout value must be greater than zero.
Proxy host and port	Specify the host address and the host port when a proxy host and port are used. For example: 192.10.1.3:18180. Or, if a proxy host and port are not used, leave this field empty.
HTTP errors to retry	Specify the HTTP errors that must return a retry status. Error codes must be a list of integers separated by spaces. For example, 307 408 503 504.
Base URL for REST Resources	Specify the common part of the REST resource URL. This is the part of the URL remaining after excluding the URL extension of the resource. For example, `http://ipaddress:port/`.

Resources

Configure Resources to synchronize: Click the icon to add a class name of the user resource present in application schema.

- **Schema name:** Specify the class name of the user resource in the application schema. For example, Users, Groups, and Entitlement.
- **Configure Handlers:** Select the appropriate customer handlers. The available options are **Default** and **Custom**.

  If you select **Custom**, fill in the following parameters:
- **Rest Handler Details:** Click the icon to add rest custom handler information.
- **URL Extension:** Specify the relative URL extension where the resource is located. The driver shim appends this URL extension to the base URL. The URL extension also includes the necessary URL placeholder. A placeholder is defined as a variable embedded within the URL. The `driver-operation-data` element replaces this with the URL token element during data transformation.

  For example, /Users/<version>. In this example, version is the placeholder and the driver replaces this with the URL token element in the `driver-operation-data` element.

```
<driver-operation-data class-name="User" command="add" method="put"
uri="https://172.16.0.0:XXXX/User/rest123">
        <request>
          <url-token version="1.0"/>
          <header content-type="application/json"/>
          <value>{"CN":[{"value":"rest6789"}],"Full Name":[{"value":"rest6789
rest6789"}],"Given
Name":[{"value":"rest6789"}],","Surname":[{"value":"rest6789"}],"Login
Disabled":[{"value":"true"}]}
          </value>
        </request>
</driver-operation-data>
```

- **Operation:** Select the required operation for Identity Manager operation.

- **Method:** Select the HTTP method to use. The options are: GET, POST,PATCH,PUT, and DELETE.
- **Optional Header Fields:** Click the icon to add optional header name and value.

Publisher Options

Publisher Settings: Specify the publisher settings. You can select either **Publish Mode** or **Poll Mode** as the publisher setting. If **Publish Mode** is selected, the driver pushes the events to the Identity Vault. In the Publish mode, the driver exposes the REST endpoints to receive the events. These events are then pushed to the Identity Vault. If **Poll Mode** is selected, the driver periodically pulls the data from the connected RESTful service.

In **Publish Mode** is selected, fill in the following parameters:

Parameters	Description
Listening IP address and port	Specify the IP address of the server where the REST driver is installed and the port number that this driver listens on.
Authentication Method	Select the authentication method as **Anonymous** or **Basic**. If **Basic** is selected, fill in the following parameters: - **Authentication ID:** Specify the Authentication ID of the remote server to validate incoming requests. If you imported a sample configuration file, this field contains the IP address and port that you specified in the wizard. - **Authentication Password:** Specify the authentication password of the remote server to validate incoming requests.
KMO Name	Specify the KMO name to be used in eDirectory. When the server is configured to accept HTTPS connections, this name becomes the KMO name in eDirectory. The KMO name is the name before the “-” (dash) in the RDN. Leave this field empty when a keystore file is used or when HTTPS connections are not used.
Keystore file	Specify the keystore name and path to the keystore file. This file is used when the server is configured to accept HTTPS connections.
Keystore password	Specify the keystore file password used with the keystore file specified above when this server is configured to accept HTTPS connections.
Server key alias	Specify a Server key alias when this server is configured to accept HTTPS connections. Leave this field empty when a KMO name is used or when HTTPS connections are not used.

Parameters	Description
Server key password	When this server is configured to accept HTTPS connections, this is the key alias password (not the keystore password). Leave this field empty when a KMO name is used or when HTTPS connections are not used.
Require mutual authentication	When using SSL, it is common to do only server authentication. However, if you want to force both client and server to present certificates during the handshake process, you should require mutual authentication.
Heartbeat interval in minutes	Specify the heartbeat interval in seconds. Leave this field empty to turn off the heartbeat.

In **Poll Mode** is selected, fill in the following parameters:

Parameters	Description
Configure Resource for poll	Click the + icon to add a class name of the user resource present in application schema. ◆ **Schema name:** Specify the class name of the user resource in the application schema. ◆ **Service Endpoint:** Specify the service end point of the connected RESTful service for the publisher polling. A generic example is `http://ip:port/schema`. For users: `http://172.16.0.0:port/User?search-attr=`. ◆ **Method:** Select the method. ◆ **Optional Header Fields:** Click the + icon to add optional header name and value.
Polling interval in minutes	Specify the polling interval in minutes. Default is one minute. **NOTE:** The Subscriber Base URL is mandatory for the driver authentication when using the poll mode.
Heartbeat interval in minutes	Specify the heartbeat interval in minutes. Leave this field empty to turn off the heartbeat.

If **Anonymous** is selected, the values you specified for authentication ID and authentication password are cleared.

ECMAScript

Displays an ordered list of ECMAScript resource files. The files contain extension functions for the driver that Identity Manager loads when the driver starts. You can add additional files, remove existing files, or change the order the files are executed.

Global Configuration

Displays an ordered list of Global Configuration objects. The objects contain extension GCV definitions for the driver that Identity Manager loads when the driver is started. You can add or remove the Global Configuration objects, and you can change the order in which the objects are executed.

Global Configuration Values

Global configuration values (GCVs) are values that can be used by the driver to control functionality. GCVs are defined on the driver or on the driver set. Driver set GCVs can be used by all drivers in the driver set. Driver GCVs can be used only by the driver on which they are defined.

The REST driver includes several predefined GCVs. You can also add your own if you discover you need additional ones as you implement policies in the driver.

To access the driver's GCVs in iManager:

1. Click to display the Identity Manager Administration page.
2. Open the driver set that contains the driver whose properties you want to edit:
   - **2a** In the **Administration** list, click **Identity Manager Overview**.
   - **2b** If the driver set is not listed on the **Driver Sets** tab, use the **Search In** field to search for and display the driver set.
   - **2c** Click the driver set to open the Driver Set Overview page.
3. Locate the driver icon, click the upper right corner of the driver icon to display the **Actions** menu, then click **Edit Properties**.

   or

   To add a GCV to the driver set, click **Driver Set**, then click **Edit Driver Set properties**.

To access the driver's GCVs in Designer:

1. Open a project in the Modeler.
2. Right-click the driver icon or line, then select **Properties > Global Configuration Values.**

   or

   To add a GCV to the driver set, right-clickthe driver set icon , then click **Properties > GCVs**.

The global configuration values are organized as follows:

- "Password Synchronization" on page 62
- "Permission Collection and Reconciliation" on page 63

Password Synchronization

These GCVs enable password synchronization between the Identity Vault and the connected system.

In Designer, you must click the icon next to a GCV to edit it. This displays the Password Synchronization Options dialog box for a better view of the relationship between the different GCVs.

In iManager, to edit the Password management options go to **Driver Properties > Global Configuration Values**, and then edit it in your Password synchronization policy tab.

For more information about how to use the Password Management GCVs, see "Configuring Password Flow" in the *NetIQ Identity Manager Password Management Guide*.

Application accepts passwords from Identity Manager: If **True**, allows passwords to flow from the Identity Manager data store to the connected system.

Identity Manager accepts passwords from application: If **True**, allows passwords to flow from the connected system to Identity Manager.

Publish passwords to NDS password: Use the password from the connected system to set the non-reversible NDS password in eDirectory.

Publish passwords to Distribution Password: Use the password from the connected system to set the NMAS Distribution Password used for Identity Manager password synchronization.

Require password policy validation before publishing passwords: If **True**, applies NMAS password policies during publish password operations. The password is not written to the data store if it does not comply.

Reset user's external system password to the Identity Manager password on failure: If **True**, on a publish Distribution Password failure, attempts to reset the password in the connected system by using the Distribution Password from the Identity Manager data store.

Notify the user of password synchronization failure via e-mail: If **True**, notifies the user by e-mail of any password synchronization failures.

Permission Collection and Reconciliation

If you installed the Permission Collection and Reconciliation package, iManager and Designer display the following options. For more information about permission reconciliation feature, and "Synchronizing Permission Changes from the Connected Systems " in the *NetIQ Identity Manager Driver Administration Guide*.

Enable Permissions Collection and Reconciliation: Set the value of this parameter to **true** for allowing permission collection and entitlement assignment. By default, the value is set to **false**, which allows the driver to override any other conditions to reconcile custom entitlements.

Enable Permissions Reconciliation for all Custom entitlements: If the value of this parameter is set to **No**, it allows you to select the custom entitlements for reconciling them. By default, it is set to **Yes**, which allows reconciling of all custom entitlements.

Click the **Add** icon add custom entitlements you want to selectively onboard and specify **Assignment Attribute Name** for them.

B Using Java Extensions

The functionality of the REST driver can be extended by using Java. You use an API defined by Java interfaces to create your own custom Java classes that have access to the data passing through the Subscriber and Publisher channels. These classes read and interpret the data, and, optionally, modify the data.

This section contains the following information on using Java extensions:

- "Overview" on page 65
- "Creating and Configuring Java Extensions" on page 66

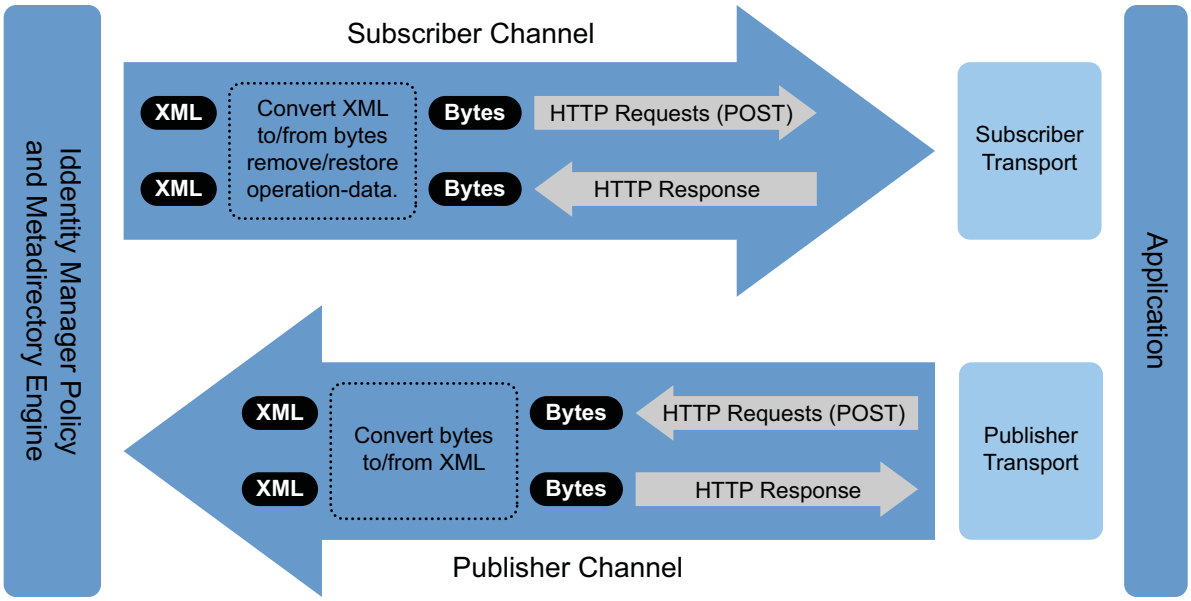
Overview

If the application you are using with the REST driver uses non-XML data that is not supported by the REST driver, you can create Java extensions to convert the non-XML data to the JSON format supported by the REST driver.

As illustrated in Figure B-1, there are five points where functionality can be extended:

- Two in the Subscriber channel
- Two in the Publisher channel
- One to report the application schema

Figure B-1 *Using Java to Extend Functionality*



The REST driver is designed to be flexible and extensible. For the Java programmer who wants to extend or modify the capabilities of the driver, there are programming interfaces that can be used for this purpose. These interfaces should be used only when you need to do transformations that cannot be done in policies or style sheets.

The Javadoc describes these interfaces.

There are two Java interfaces that can be used to extend or customize the driver behavior. They are DocumentModifiers and SchemaReporter.

DocumentModifiers is used to access and to modify the commands and events passing through the driver shim, if this is desired. DocumentModifiers gives you access to the data as XML DOM documents.

The other interface, SchemaReporter, can be used if you have a way of programatically determining the classes and attributes used by the remote Web service. The advantage to this is that creating schema mapping rules is easier if the schema can be dynamically determined.

Creating and Configuring Java Extensions

You should name your class by using any Java package and class name that is convenient for your environment and your organization.

For example, if you were writing your own class that implemented the DocumentModifiers interface, and you named your class *MyDocumentModifiers* within a package called `com.novell.idm`, then you would perform the following steps to compile, jar, and deploy your class:

1. Prepare your environment.

   Make sure you have a current Java Development Kit (JDK) installed on your computer. Visit the Java Web Site if you need to download one.

2. Gather your source code in the proper directory structure as defined by your package naming.

   In the example given above, you would have a `com` directory that contained a `novell` directory that contained an `idm` directory. Within the `idm` directory, you would have a source file named `MyDocumentModifiers.java`.

3. Make sure you have the jar files you need to compile your class.

   At a minimum, you need `RESTUtil.jar`. If you are using XML documents within your class, you also need `nxsl.jar`.

4. Put a copy of the required jar files in a convenient location like the root of your compile directory just outside the `com` directory, then access a system command prompt or shell prompt with that location as the current directory.

5. Compile your class by entering one of the following commands:

   - **For Windows:** `javac -classpath RESTUtil.jar;nxsl.jar com\novell\idm\*.java`
   - **For Linux or UNIX:** `javac -classpath RESTUtil.jar:nxsl.jar com/novell/idm/*.java`

6. Create a Java archive file containing your class by entering one of the following commands:

   - **For Windows:** `jar cvf mydriverextensions.jar com\novell\idm\*.class`
   - **For Linux:** `jar cvf mydriverextensions.jar com/novell/idm/*.class`

7. Place the jar file you created in Step 6 into the same directory that contains the `RESTShim.jar`.

   In Windows, this is often `C:\Novell\NDS\lib`.

8. In iManager, edit the driver settings.

   8a Next to Custom Java Extension, select **Show**.

   8b Next to Document Handling, select **Implemented**.

8c Specify *com.novell.idm.MyDocumentModifiers* as the value for Class and any string as the value for Init Parameter.

The init parameter is the string that is passed to the init method of your class, so you can put any information here that you want to use during your class initialization.

9 Restart the driver.

You can now use your custom class.

C Trace Levels

The driver supports the following trace levels:

Table C-1 *Supported Trace Levels*

Level	Description
0	No debugging
1-3	Identity Manager messages. Higher trace levels provide more detail.
4	Previous level plus Remote Loader, driver, driver shim, and driver connection messages, driver parameters, driver security, driver schema, request and response XML

For information about setting driver trace levels, see “Viewing Identity Manager Processes” in the *NetIQ Identity Manager Driver Administration Guide*.

D Supported JSON Format

The Identity Manager driver for REST queries the exposed RESTful endpoints and the returns the responses in JSON format.

The following is an example of the QUERY response in the supported JSON format.


```
{
    "totalResults": 1,
    "results": [
      {
            "src-dn": "\\SERVER-LINUX-TREE-45\\data\\users\\thomaswagner",
            "class-name": "User",
            "CN": [
                "thomaswagner"
            ],
            "Object Class": [
                "User",
                "Organizational Person",
                "Person",
                "ndsLoginProperties",
                "Top"
            ],
            "Password Allow Change": [
                "true"
            ],
            "Password Minimum Length": [
                "4"
            ],
            "Password Required": [
                "true"
            ],
            "Password Unique Required": [
                "false"
            ],
            "Public Key": [
                "AQAAAAQAAAAgAGAAAADWACc7sIe2QAUFVSU0FG"
            ],
            "Surname": [
                "thomaswagner"
            ],
            "Full Name": [
                "thomaswagner thomaswagner"
            ],
            "Revision": [
                "6"
            ],
            "Given Name": [
                "thomaswagner"
            ],
            "GUID": [
                "OTGey593Bkx1sDkxnsufdw=="
            ],
            "DirXML-Associations": [
                {
```

```
                    "nameSpace": "1",
                    "volume": "\\SERVERL-LINUX-TREE-45\\system\\driverset1\\REST-
DRIVER-PUB",
                    "path": "thomaswagner"
                },
                {
                    "nameSpace": "1",
                    "volume": "\\SERVERL-LINUX-TREE-45\\system\\driverset1\\Data
Collection Service Driver",
                    "path": "39319ECB-9F77-064c-75B0-39319ECB9F77"
                }
            ],
            "creatorsName": [
                "CN=linux-yal5,OU=servers,O=system"
            ],
            "modifiersName": [
                "CN=linux-yal5,OU=servers,O=system"
           ]
        }
    ]
}
```

The following is an example of the ADD request in the supported JSON format.

```
{
    "cn": "Sam2",
    "title": [
        "Sr Engineer",
        "Manager",
        "Mr. "
    ],
    "streetAddress": [
        {
            "component": "566666"
        },
        {
            "component": "area numero",
            "postal code": "566666"
        }
    ]
}
```

The following is an example of the MODIFY request in the supported JSON format.

```
{
    "cn": {
        "remove": "Sam1",
        "add": "Sam"
    },
    "title": {
        "add": [
            "Mr",
            "mr2"
        ]
    },
    "streetAddress": {
        "remove": [
            {
                "component1": "areanumero",
                "postalcode": "566666"
            }
        ],
        "add": [
            {
                "component2": " area numero ",
                "postalcode": "5555"
            }
        ]
    }
}
```

The following is an example of the GET response in the supported JSON format.

```
<nds dtdversion="4.0" ndsversion="8.x">
  <source>
    <product edition="Advanced" version="4.5.0.0">DirXML</product>
    <contact>NetIQ Corporation</contact>
  </source>
  <output>
    <status event-id="0" level="success"><driver-operation-data>
        <header Accept="application/json"/>
        <response>           <value>{"totalResults":2,"results":
[{"keyvalue1":{"NAME":"thomas","VALUE":29},
"keyvalue2":{"NAME":"wagnor","VALUE":30}}, {"search-attr":[{"Surname": "Thomas"
}],"read-attr":["Surname","cn","Given Name"]}]}</value>         </response>
      </driver-operation-data>
    </status>
  </output>
</nds>
```