Installation Guide

/ OpenIDM 4

Latest update: 4.0.0

Mark Craig
Lana Frost
Paul Bryan
Andi Egloff
Laszlo Hordos
Matthias Tristl
Mike Jang

ForgeRock AS
201 Mission St., Suite 2900
San Francisco, CA 94105, USA
+1 415-599-1100 (US)
www.forgerock.com

Abstract

Guide to installing and evaluating OpenIDM. OpenIDM offers flexible, open source services for automating management of the identity life cycle.

Table of Contents

Preface

This guide shows you how to install core OpenIDM services for identity management, provisioning, and compliance. Unless you are planning a throwaway evaluation or test installation, read the *Release Notes* before you get started.

1. Who Should Use This Guide

This guide is written for anyone installing OpenIDM to manage identities, and to ensure compliance with identity management regulations.

The guide covers the install and removal (uninstall) procedures that you theoretically perform only once per version. It aims to provide you with at least some idea of what happens behind the scenes when you perform the steps.

You do not need to be an OpenIDM wizard to learn something from this guide, though a background in identity management and maintaining web application software can help. You do need some background in managing services on your operating systems and in your application servers. You can nevertheless get started with this guide, and then learn more as you go along.

If you have a previous version of OpenIDM installed, see "*OpenIDM Compatibility*" in the *Release Notes* before installing this version.

2. Formatting Conventions

Most examples in the documentation are created in GNU/Linux or Mac OS X operating environments. If distinctions are necessary between operating environments, examples are labeled with the operating environment name in parentheses. To avoid repetition file system directory names are often given only in UNIX format as in `/path/to/server`, even if the text applies to `C:\path\to\server` as well.

Absolute path names usually begin with the placeholder `/path/to/`. This path might translate to `/opt/`, `C:\Program Files\`, or somewhere else on your system.

Command-line, terminal sessions are formatted as follows:

```
$ echo $JAVA_HOME
/path/to/jdk
```

Command output is sometimes formatted for narrower, more readable output even though formatting parameters are not shown in the command.

Program listings are formatted as follows:

```
class Test {
    public static void main(String [] args)  {
        System.out.println("This is a program listing.");
    }
}
```

3. Accessing Documentation Online

ForgeRock publishes comprehensive documentation online:

- The ForgeRock Knowledge Base offers a large and increasing number of up-to-date, practical articles that help you deploy and manage ForgeRock software.

  While many articles are visible to community members, ForgeRock customers have access to much more, including advanced information for customers using ForgeRock software in a mission-critical capacity.

- ForgeRock product documentation, such as this document, aims to be technically accurate and complete with respect to the software documented. It is visible to everyone and covers all product features and examples of how to use them.

4. Using the ForgeRock.org Site

The ForgeRock.org site has links to source code for ForgeRock open source software, as well as links to the ForgeRock forums and technical blogs.

If you are a *ForgeRock customer*, raise a support ticket instead of using the forums. ForgeRock support professionals will get in touch to help you.

Chapter 1

Installing OpenIDM Services

This chapter covers the tasks required to install and start OpenIDM.

Note

ForgeRock documentation includes a separate Samples Guide. Once you read through the first two chapters of this document, you can use the Samples Guide to test OpenIDM in about 20 different configurations.

1.1. Before You Run OpenIDM

This section covers what you need to know before running OpenIDM.

1.1.1. Java Environment

For details of the supported Java Environment, see the Java Environment in the *Release Notes*.

On Windows systems, you must set the `JAVA_HOME` environment variable to point to the root of a valid Java installation. The following steps indicate how to set the `JAVA_HOME` environment variable on Windows Server 2008 R2. Adjust the steps for your specific environment:

1. Locate your JRE Installation Directory. If you have not changed the installation path for the Java Runtime Environment during installation, it will be in a directory under `C:\Program Files\Java\`.
2. Select Start > Control Panel > System and Security > System.
3. Click Advanced System Settings.
4. Click Environment Variables.
5. Under System Variables, click New.
6. Enter the Variable name (`JAVA_HOME`) and set the Variable value to the JRE installation directory, for example `C:\Program Files\Java\jre7`.
7. Click OK.

1.1.2. Application Container

OpenIDM services run in an OSGi container with an embedded Servlet container, and an embedded noSQL database. By default the OSGi container is Apache Felix (Felix). The default Servlet container is Jetty. For OpenIDM 4, the only supported configuration is running the services in Felix and Jetty.

1.2. Installing and Running OpenIDM

Follow the procedures in this section to install and run OpenIDM in UNIX and Windows environments.

To Install OpenIDM Services

Follow these steps to install OpenIDM:

1. Make sure you have an appropriate version of Java installed:

   ```
   $ java -version
   java version "1.7.0_79"
   Java(TM) SE Runtime Environment (build 1.7.0_79-b15)
   Java HotSpot(TM) 64-Bit Server VM (build 24.79-b02, mixed mode)
   ```

   For a description of the Java requirements, see "*Before You Install OpenIDM Software*" in the *Release Notes*.

2. Download enterprise software releases through the ForgeRock BackStage site. ForgeRock enterprise releases are thoroughly validated builds for ForgeRock customers who run OpenIDM in production deployments, and for those who want to try or test with release builds.

3. Unpack the contents of the .zip file into the install location:

   ```
   $ cd /path/to
   $ unzip ~/Downloads/openidm-4.0.0.zip
   Archive:  openidm-4.0.0.zip
     inflating: openidm/.checksums.csv
      creating: openidm/bundle/
    extracting: openidm/bundle/openidm-audit-4.0.0
   .jar
   ...
   ```

4. (Optional) By default, OpenIDM listens for HTTP and HTTPS connections on ports 8080 and 8443, respectively. To change the default port, edit your project's `conf/boot/boot.properties` file. For more information, see "*Ports Used*" in the *Integrator's Guide*.

5. (Optional) Before running OpenIDM in production, replace the default OrientDB repository, provided for evaluation, with a supported JDBC repository.

   For more information, see "*Installing a Repository For Production*".

To Start OpenIDM Services

To run OpenIDM as a background process, see "*Starting and Stopping OpenIDM*" in the *Integrator's Guide*.

Follow these steps to run OpenIDM interactively:

1. Start the Felix container, load all OpenIDM services, and start a command shell to allow you to manage the container:

   - Start OpenIDM (UNIX):

     ```
     $ ./startup.sh

     Using OPENIDM_HOME:   /path/to/openidm
     Using PROJECT_HOME:   /path/to/openidm
     Using OPENIDM_OPTS:   -Xmx1024m -Xms1024m
     Using LOGGING_CONFIG: -Djava.util.logging.config.file=/path/to/openidm/conf/logging.properties
     Using boot properties at /path/to/openidm/conf/boot/boot
     .properties
     -> OpenIDM ready
     ```

   - Start OpenIDM (Windows):

     ```
     C:\> cd \path\to\openidm
     C:\> startup.bat

     "Using OPENIDM_HOME:   \path\to\openidm"
     "Using PROJECT_HOME:   \path\to\openidm"
     "Using OPENIDM_OPTS:   -Xmx1024m -Xms1024m -Dfile.encoding=UTF-8"
     "Using LOGGING_CONFIG: -Djava.util.logging.config.file=\path\to\openidm\conf\logging.properties"
     Using boot properties at \path\to\openidm\conf\boot\boot
     .properties
     -> OpenIDM
      ready
     ->
     ```

   At the OSGi console `->` prompt, you can enter commands such as **help** for usage, or **ps** to view the bundles installed. To see a list of all the OpenIDM core services and their states, enter the following command:

```
-> scr list
   Id   State          Name
[  16] [active       ] org.forgerock.openidm.endpoint
[  17] [active       ] org.forgerock.openidm.endpoint
[  37] [active       ] org.forgerock.openidm.endpoint
[  36] [active       ] org.forgerock.openidm.endpoint
[  18] [active       ] org.forgerock.openidm.endpoint
[  38] [active       ] org.forgerock.openidm.endpoint
[  39] [active       ] org.forgerock.openidm.endpoint
[  40] [active       ] org.forgerock.openidm.endpoint
[  32] [active       ] org.forgerock.openidm.endpoint
[  19] [active       ] org.forgerock.openidm.config.enhanced
[  49] [active       ] org.forgerock.openidm.selfservice.userupdate
[   3] [unsatisfied  ] org.forgerock.openidm.datasource.jdbc
[   7] [active       ] org.forgerock.openidm.http.context
[  23] [active       ] org.forgerock.openidm.info
[  41] [active       ] org.forgerock.openidm.info
[  24] [active       ] org.forgerock.openidm.info
```

```
[  35] [active        ] org.forgerock.openidm.provisioner.openicf.connectorinfoprovider
[  44] [unsatisfied   ] org.forgerock.openidm.provisioner.salesforce
[  28] [active        ] org.forgerock.openidm.maintenance.update.log
[  26] [active        ] org.forgerock.openidm.maintenance.updatemanager
[   5] [active        ] org.forgerock.openidm.repo.orientdb
[  34] [active        ] org.forgerock.openidm.openicf.syncfailure
[   8] [active        ] org.forgerock.openidm.api-servlet
[   2] [active        ] org.forgerock.openidm.config.enhanced.starter
[   0] [active        ] org.forgerock.openidm.security
[  25] [active        ] org.forgerock.openidm.maintenance.update
[  10] [active        ] org.forgerock.openidm.audit
[  57] [unsatisfied   ] org.forgerock.openidm.schedule
[  52] [active        ] org.forgerock.openidm.servletfilter.registrator
[  11] [active        ] org.forgerock.openidm.auth.config
[   4] [unsatisfied   ] org.forgerock.openidm.repo.jdbc
[  55] [active        ] org.forgerock.openidm.workflow
[  33] [unsatisfied   ] org.forgerock.openidm.provisioner.openicf
[  15] [active        ] org.forgerock.openidm.managed
[  22] [active        ] org.forgerock.openidm.health
[  31] [active        ] org.forgerock.openidm.provisioner
[  42] [active        ] org.forgerock.openidm.internal
[  27] [active        ] org.forgerock.openidm.maintenance.update.config
[  43] [active        ] org.forgerock.openidm.provisioner.salesforce.confighelper
[  56] [active        ] org.forgerock.openidm.taskscanner
[  21] [active        ] org.forgerock.openidm.external.rest
[  50] [active        ] org.forgerock.openidm.ui.context
[  51] [active        ] org.forgerock.openidm.ui.context
[  46] [active        ] org.forgerock.openidm.selfservice.kbaservice
[   9] [active        ] org.forgerock.openidm.router
[  58] [active        ] org.forgerock.openidm.scheduler
[  20] [unsatisfied   ] org.forgerock.openidm.external.email
[  30] [active        ] org.forgerock.openidm.policy
[   6] [active        ] org.forgerock.openidm.cluster
[  13] [active        ] org.forgerock.openidm.sync
[  45] [active        ] org.forgerock.openidm.script
[  14] [active        ] org.forgerock.openidm.recon
[  53] [active        ] org.forgerock.openidm.servletfilter
[  54] [active        ] org.forgerock.openidm.servletfilter
[  48] [unsatisfied   ] org.forgerock.openidm.selfservice
[  47] [active        ] org.forgerock.openidm.selfservice.kba
[  12] [active        ] org.forgerock.openidm.authentication
[   1] [active        ] org.forgerock.openidm.config.manage
[  29] [active        ] org.forgerock.openidm
.maintenance
->
```

A default startup does not include certain configurable services, which will indicate an `unsatisfied` state until they are included in the configuration. As you work through the sample configurations described later in this guide, you will notice that these services are active.

Startup errors and messages are logged to the console by default. You can also view these messages in the log files at `/path/to/openidm/logs`.

2. Alternatively, you can manage the container and services from the Apache Felix Web Console.

   Use these hints to connect to the Apache Felix Web Console:

- Default URL: https://localhost:8443/system/console
- Default user name: `admin`
- Default password: `admin`

Select Main > Components to see OpenIDM core services and their respective states.

To Stop the OpenIDM Services

1. You can stop OpenIDM Services from the `->` prompt in the OSGi console, or through the Apache Felix Web Console. Both of these options stop the Felix container:
   - In the OSGi console, enter the **shutdown** command at the `->` prompt:

     ```
     -> shutdown
     ...
     $
     ```

   - In the Apache Felix Web Console, select Web Console > System Information to stop the container.
2. On Unix systems, you can stop OpenIDM by using the **shutdown.sh** script, located in the `/path/to/openidm` directory:

   ```
   $ ./shutdown.sh
   ./shutdown.sh
   Stopping OpenIDM (31391)
   ```

Note

If you want to set up OpenIDM on a read-only volume, read "*Installing OpenIDM on a Read-Only Volume*".

To Install OpenIDM as a Windows Service

You can install OpenIDM to run as a Windows service so that the server starts and stops automatically when Windows starts and stops. You must be logged in as an administrator to install OpenIDM as a Windows service:

Note

On a 64-bit Windows server, you must have a 64-bit Java version installed to start the service. If a 32-bit Java version is installed, you will be able to install OpenIDM as a service, but starting the service will fail.

Before you launch the `install-service.bat` file, which registers the `OpenIDM` service within the Windows registry, make sure that your `JAVA_HOME` environment variable points to a valid 64-bit version of the JRE or JDK.

If you have already installed the service with the `JAVA_HOME` environment variable pointing to a 32-bit JRE or JDK, delete the service first, then reinstall the service.

1. Unpack the OpenIDM .zip file, as described previously, and change to the *`install-location`*`\bin` directory:

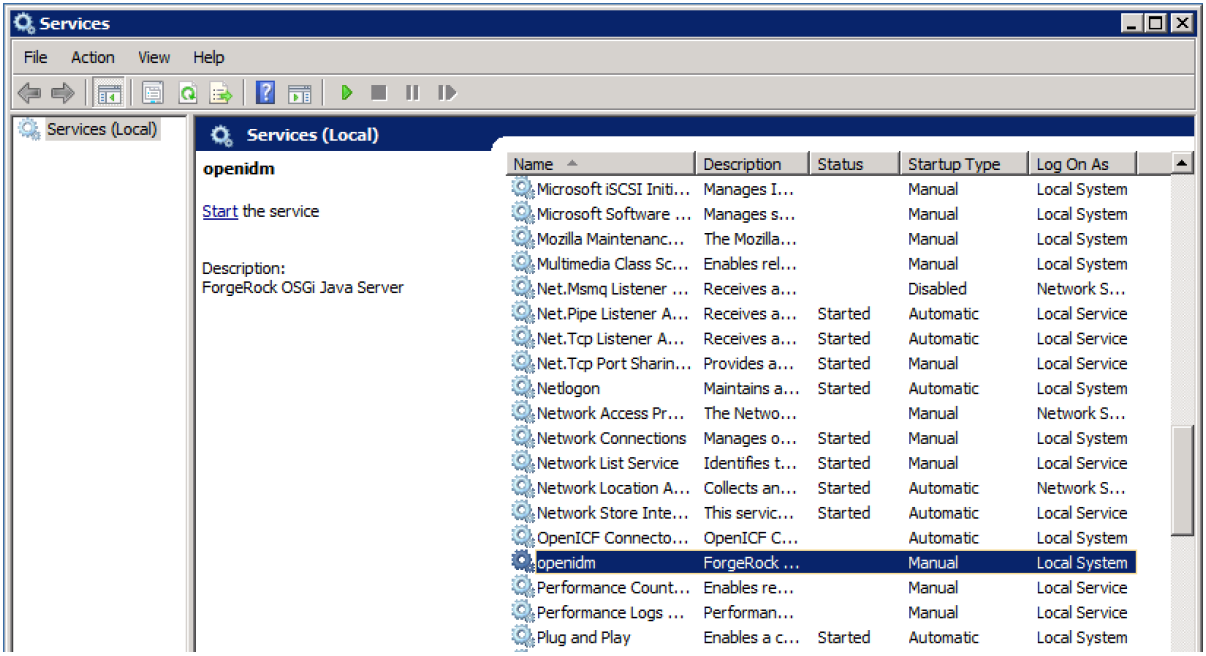
```
C:\>cd openidm\bin
C:\openidm\bin>
```

2. Run the `install-service.bat` command, specifying the name the service should run as:

```
C:\openidm\bin>install-service.bat openidm
ForgeRock Launcher Java Service successfully installed as "openidm" service
```

3. Use the Windows Service manager to manage the OpenIDM service.

Running OpenIDM as a Windows Service



4. Change the user account for this service from the default (`local system`) account to an account with administrative privileges. The `local system` account has limited permissions and an OpenIDM service that runs with this account will encounter problems during synchronization.

   To change the user account:

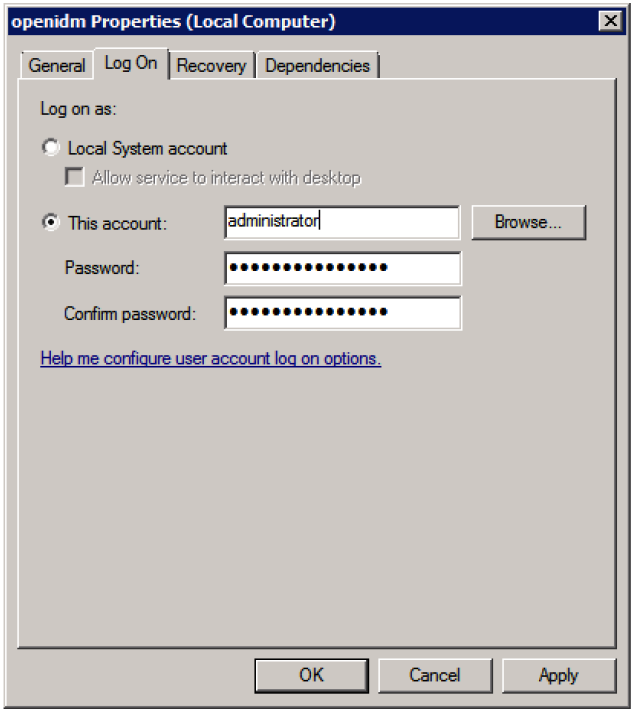
   a. Double click the `openidm` service in the Windows Service manager.

b. Select the Log On tab.

c. Select This Account and browse for an Active Directory administrative account.

d. Enter the password for the administrative account.

Changing the Service User Account



e. Click Apply to save the changes.

5. Use the Windows Service Manager to start, stop, or restart the service.

6. To uninstall the OpenIDM service stop the service, then run the following command:

```
C:\install-location\openidm\bin>launcher.bat /uninstall openidm
...
 Service "openidm" removed successfully
...
```

1.3. Getting Started With the OpenIDM REST Interface

OpenIDM provides RESTful access to users in the OpenIDM repository. To access the OpenIDM repository over REST, you can use a browser-based REST client, such as the *Simple REST Client* for Chrome, or *RESTClient* for Firefox. Alternatively you can use the **curl** command-line utility that is included with most operating systems. For more information about **curl**, see https://github.com/bagder/curl.

OpenIDM is accessible over the regular and secure HTTP ports of the Jetty Servlet container, 8080 and 8443.

If you want to run **curl** over the secure port, 8443, you must either include the **--insecure** option, or follow the instructions in "Restrict REST Access to the HTTPS Port" in the *Integrator's Guide*. You can use those instructions with the self-signed certificate that is generated when OpenIDM starts, or with a `*.crt` file provided by a certificate authority.

In numerous cases, **curl** commands to the secure port are depicted with a `--cacert self-signed.crt` option. Instructions for creating that `self-signed.crt` file are shown in "Restrict REST Access to the HTTPS Port" in the *Integrator's Guide*.

If you would rather use **curl** to connect to the regular HTTP port, omit the `--cacert self-signed.crt` file and point to a regular Jetty HTTP URL such as `http://localhost:8080/openidm/...`.

Note

All RESTful command-line examples in this guide, as depicted with **curl**, are based on the default configuration of OpenIDM. If you change configuration files in directories such as `openidm/conf` and `openidm/script`, you might need to modify the RESTful commands to reflect those changes.

Most of the examples in this guide use client-assigned IDs when creating resources, as it makes the examples easier to read.

In general, server-assigned UUIDs are better in production, as they can be generated easily in clustered environments.

For some versions of Mac OS X, the stock version of the **curl** command with the **--cacert** option may lead to error messages. You may use the **-k** or **--insecure** options as a workaround.

1. Access the following URL to obtain the JSON representation of all users in the OpenIDM repository:

   ```
   $ curl \
    --header "X-OpenIDM-Username: openidm-admin" \
    --header "X-OpenIDM-Password: openidm-admin" \
    --request GET \
    http://localhost:8080/openidm/managed/user/?_queryId=query-all-ids
   ```

   When you first install OpenIDM with an empty repository, no users exist.

2. Create a user `joe` by sending a RESTful POST.

   The following **curl** commands create the user `joe` in the repository.

- Create `joe` (UNIX):

```
$ curl \
 --cacert self-signed.crt \
 --header "Content-Type: application/json" \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request POST \
 --data '{
 "userName":"joe",
 "givenName":"joe",
 "sn":"smith",
 "mail":"joe@example.com",
 "telephoneNumber":"555-123-1234",
 "password":"TestPassw0rd",
 "description":"My first user",
 "_id":"joe"
 }' \
 https://localhost:8443/openidm/managed/user?_action=create
{
  "_id": "joe",
  "_rev": "1",
  "userName": "joe",
  "givenName": "joe",
  "sn": "smith",
  "mail": "joe@example.com",
  "telephoneNumber": "555-123-1234",
  "description": "My first user",
  "accountStatus": "active",
  "effectiveRoles": [],
  "effectiveAssignments": []
}
```

- Create `joe` (Windows):

```
C:\> curl ^
 --cacert self-signed.crt ^
 --header "Content-Type: application/json" ^
 --header "X-OpenIDM-Username: openidm-admin" ^
 --header "X-OpenIDM-Password: openidm-admin" ^
 --request POST ^
 --data "{
 \"userName\":\"joe\",
 \"givenName\":\"joe\",
 \"sn\":\"smith\",
 \"mail\":\"joe@example.com\",
 \"telephoneNumber\":\"555-123-1234\",
 \"password\":\"TestPassw0rd\",
 \"description\":\"My first user\"
 \"_id\":\"joe\"
 }" ^
 https://localhost:8443/openidm/managed/user?_action=create
```

3. Fetch the newly created user from the repository with a RESTful GET:

```
$  curl \
 --cacert self-signed.crt \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 https://localhost:8443/openidm/managed/user/joe
{
  "_id": "joe",
  "_rev": "1",
  "userName": "joe",
  "givenName": "joe",
  "sn": "smith",
  "mail": "joe@example.com",
  "telephoneNumber": "555-123-1234",
  "description": "My first user",
  "accountStatus": "active",
  "effectiveRoles": [],
  "effectiveAssignments": []
}
```

4. Notice that more attributes are returned for user `joe` than the attributes you added in the previous step. The additional attributes are added by a script named `onCreate-user-set-default-fields.js` that is triggered when a new user is created. For more information, see "Managed Object Configuration" in the *Integrator's Guide*.

   When you create a user some attributes might be *required* by the policy that is associated with that user. These are listed in the `conf/policy.json` file.

1.3.1. Format REST Output for Readability

With all **curl**-based REST calls, OpenIDM returns the JSON object all on one line.

Without a bit of help, the JSON output is formatted all on one line. One example is shown below, and it is difficult to read:

```
{"mail":"joe@example.com","sn":"smith","passwordAttempts":"0",
"lastPasswordAttempt":"Mon Apr 14 2014 11:13:37 GMT-0800 (GMT-08:00)",
"address2":"","givenName":"joe","effectiveRoles":["openidm-authorized"],
"password":{"$crypto":{"type":"x-simple-encryption","value":{"data":
"OBFVL9cG8uaLoo1N+SMJ3g==","cipher":"AES/CBC/PKCS5Padding","iv":
"7rlV4EwkwdRHkt19F8g22A==","key":"openidm-sym-default"}}},"country":"",
"city":"","_rev":"1","lastPasswordSet":"","postalCode":"","_id":"joe3",
"description":"My first user","accountStatus":"active","telephoneNumber":
"555-123-1234","roles":["openidm-authorized"],"effectiveAssignments":{},
"postalAddress":"","stateProvince":"","userName":"joe3"}
```

At least two options are available to clean up this output.

The standard way to format JSON output is with a JSON parser such as jq. You would "pipe" the output of a REST call to **jq**, as follows:

```
$ curl \
--cacert self-signed.crt
 \
--header "X-OpenIDM-Username: openidm-admin"
 \
--header "X-OpenIDM-Password: openidm-admin"
 \
--request GET \
"https://localhost:8443/openidm/managed/user/joe" \
| jq .
```

The ForgeRock REST API includes an optional `_prettyPrint` request parameter. The default value is `false`. To use the ForgeRock REST API to format output, add a parameter such as `?_prettyPrint=true` or `&_prettyPrint=true`, depending on whether it is added to the end of an existing request parameter. In this case, the following command would return formatted output:

```
$ curl \
--cacert self-signed.crt
 \
--header "X-OpenIDM-Username: openidm-admin"
 \
--header "X-OpenIDM-Password: openidm-admin"
 \
--request GET \
"https://localhost:8443/openidm/managed/user/joe?_prettyPrint=true"
```

Note that most command-line examples in this guide do not show this parameter, although the output is formatted for readability.

1.4. OpenIDM User Interfaces

OpenIDM supports configuration from Web-based user interfaces, called the UI in the OpenIDM documentation set.

OpenIDM includes UIs at two different endpoints, `/` and `/admin`. We refer to the administrative tools available at each endpoint as the Self-Service UI and the Administrative UI (Admin UI), respectively.

The Self-Service UI allows regular (non-administrative) users to update parts of their profile, such as passwords and addresses. For more information, see "Configuring User Self-Service" in the *Integrator's Guide*. When these features are enabled, anonymous users can self-register and regular users can reset their own passwords. For more information, see "Working With the Self-Service UI" in the *Integrator's Guide*.

In addition, administrative users can configure and manage workflows in the Self-Service UI. For more information, see "Managing Workflows From the Self-Service UI" in the *Integrator's Guide*.

In essence, the Self-Service UI supports day-to-day administrative tasks.

In contrast, the Admin UI allows an administrator to define the overall OpenIDM system configuration. Administrators would access the Admin UI to learn OpenIDM, during initial system setup, and when they identify new requirements.

Unlike the Self-Service UI, the Admin UI allows you to configure connections to external data stores, as well as the way OpenIDM reconciles information between internal and external data stores.

When OpenIDM is running on the localhost system, you can access these UIs at `https://localhost:8443/` and `https://localhost:8443/admin`, respectively.

1.5. About the OpenIDM Repository

OpenIDM comes with an internal noSQL database, OrientDB, for use as the internal repository out of the box. This makes it easy to get started with OpenIDM. OrientDB is not supported for production use, however, so use a supported JDBC database when moving to production.

To query the internal noSQL database, download and extract OrientDB (version 1.7.10). You will find the shell console in the `bin` directory. Start the OrientDB console, using **console.sh** or **console.bat**, and connect to the running OpenIDM instance, with the **connect** command:

```
$ cd /path/to/orientdb-community-1.7.10/bin
$ ./console.sh

OrientDB console v.1.7.10 (build @BUILD@) www.orientechnologies.com
Type 'help' to display all the commands supported.

Installing extensions for GREMLIN language v.2.5.0
orientdb> connect remote:localhost/openidm admin admin
Connecting to database [remote:localhost/openidm] with user 'admin'...OK

orientdb>
```

When you have connected to the database, you might find the following commands useful:

info

Shows classes and records

select * from managed_user

Shows all users in the OpenIDM repository

select * from audit_activity

Shows all activity audit records

This table is created when there is some activity.

select * from audit_recon

Shows all reconciliation audit records

This table is created when you run reconciliation.

You can also use OrientDB Studio to query the default OrientDB repository. After you have installed and started OpenIDM, point your browser to http://localhost:2480/. The default database is `openidm` and the default user and password are `admin` and `admin`. Click Connect to connect to the repository.

To change the default password, use the following POST request on the `repo` endpoint:

```
$ curl \
 --cacert self-signed.crt \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request POST \
 "https://localhost:8443/openidm/repo?_action=updateDbCredentials&user=admin&password=newPassword"
```

You must restart OpenIDM for the change to take effect.

This command updates both the repository and the repository configuration file.

Chapter 2

Installing a Repository For Production

By default, OpenIDM uses OrientDB for its internal repository so that you do not have to install a database in order to evaluate OpenIDM. Before using OpenIDM in production, however, you must replace OrientDB with a supported JDBC repository.

In production environments, OpenIDM 4 supports the use of the following internal repositories:

- MySQL
- MS SQL
- PostgreSQL
- Oracle Database
- IBM DB2 Database

For details of the supported versions, see "*Before You Install OpenIDM Software*" in the *Release Notes*.

This chapter describes how to set up OpenIDM to work with each of these supported repositories, and lists the minimum rights required for database installation and operation. For information about the general JDBC repository configuration, and how to map OpenIDM objects to JDBC database tables, see "*Managing the OpenIDM Repository*" in the *Integrator's Guide*.

2.1. Minimum Database Access Rights

In general, OpenIDM requires minimal access rights to the JDBC repository for daily operation. The first time OpenIDM is started, however, a number of tables must be created, for use by the workflow engine.

This section lists the minimum permissions required at each stage, and suggests a strategy for restricting database access in an OpenIDM installation.

The JDBC repository used by OpenIDM requires only one *relevant* user - the service account that is used to create the tables. Generally, the details of this account are configured in the repository connection file (`datasource.jdbc-default.json`). By default, the username and password for this account are `openidm` and `openidm`, regardless of the database type.

All other users are created by the *database-type*/conf/openidm.sql script. The openidm user account must have read and write access to all the openidm tables that are created by this script. In addition, the account must be able to *create* tables the first time OpenIDM starts up. These tables are specific to the Activiti workflow engine. After the Activiti tables have been created, you can then remove the *create* of the openidm user.

The following steps suggest a method of restricting database access to the required minimum.

1. Give the openidm user (or whatever user you configure in datasource.jdbc-default.json) the following permissions:

   SELECT, UPDATE, INSERT, DELETE, CREATE TABLE, ALTER TABLE

2. Run the OpenIDM script (openidm.sql) that corresponds to your database type.

3. Configure OpenIDM to use your JDBC repository (as described in the subsequent sections of this chapter).

4. Start OpenIDM for the first time.

   The tables that are required by the Activiti workflow engine are created in the database.

5. When the startup has completed, stop OpenIDM.

6. Change the permissions of the openidm user (or whatever user you configure in datasource.jdbc-default.json) to the following:

   SELECT, UPDATE, INSERT, DELETE

7. Restart OpenIDM.

2.2. To Set Up OpenIDM With MySQL

After you have installed MySQL on the local host and *before starting OpenIDM for the first time*, set up OpenIDM to use the new repository, as described in the following sections.

This procedure assumes that a password has already been set for the MySQL root user:

1. Download MySQL Connector/J, version 5.1 or later from the MySQL website. Unpack the delivery, and copy the .jar into the openidm/bundle directory:

   ```
   $ cp mysql-connector-java-version-bin.jar /path/to/openidm/bundle/
   ```

2. Make sure that OpenIDM is stopped:

   ```
   $ cd /path/to/openidm/
   $ ./shutdown.sh
   OpenIDM is not running, not stopping.
   ```

3. Remove the OrientDB configuration file (`repo.orientdb.json`) from your project's `conf/` directory. For example:

```
$ cd /path/to/openidm/my-project/conf/
$ rm repo.orientdb.json
```

4. Copy the MySQL database connection configuration file (`datasource.jdbc-default.json`) and the database table configuration file (`repo.jdbc.json`) to your project's `conf` directory:

```
$ cd /path/to/openidm/
$ cp db/mysql/conf/datasource.jdbc-default.json my-project/conf/
$ cp db/mysql/conf/repo.jdbc.json my-project/conf/
```

5. Import the data definition language script for OpenIDM into MySQL:

```
$ cd /path/to/mysql
$ ./bin/mysql -u root -p < /path/to/openidm/db/mysql/scripts/openidm.sql
Enter password:
$
```

This step creates an `openidm` database for use as the internal repository, and a user `openidm` with password `openidm` who has all the required privileges to update the database:

```
$ ./bin/mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 18
Server version: 5.5.19 MySQL Community Server
 (GPL)
...
mysql> use openidm;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;

+---------------------------+
| Tables_in_openidm         |
+---------------------------+
| auditaccess               |
| auditactivity             |
| ...                       |
| schedulerobjectproperties |
| schedulerobjects          |
| security                  |
| securitykeys              |
| uinotification            |
+---------------------------+
19 rows in set (0.00 sec)
```

The table names are similar to those used with OrientDB.

6. Update the connection configuration in `datasource.jdbc-default.json` to reflect your MySQL deployment. The default connection configuration is as follows:

```
{
    "driverClass" : "com.mysql.jdbc.Driver",
    "jdbcUrl" : "jdbc:mysql://localhost:3306/openidm?allowMultiQueries=true&characterEncoding=utf8",
    "databaseName" : "openidm",
    "username" : "openidm",
    "password" : "openidm",
    "connectionTimeout" : 30000,
    "connectionPool" : {
        "type" : "bonecp"
    }
}
```

When you have set up MySQL for use as the OpenIDM internal repository, start OpenIDM to check that the setup has been successful. After startup, you should see that `repo.jdbc` is `active`, whereas `repo.orientdb` is `unsatisfied`:

```
$ cd /path/to/openidm
$ ./startup.sh
Using OPENIDM_HOME:   /path/to/openidm
Using OPENIDM_OPTS:   -Xmx1024m -Xms1024m
Using LOGGING_CONFIG:
-Djava.util.logging.config.file=/path/to/openidm/conf/logging.properties
Using boot properties at /path/to/openidm/conf/boot/boot.properties
-> scr list

Id   State          Name
[  19] [active       ] org.forgerock.openidm.config.starter
[  23] [active       ] org.forgerock.openidm.taskscanner
[   8] [active       ] org.forgerock.openidm.external.rest
[  12] [active       ] org.forgerock.openidm.provisioner.openicf.connectorinfoprovider
[  15] [active       ] org.forgerock.openidm.ui.simple
[   1] [active       ] org.forgerock.openidm.router
[  22] [active       ] org.forgerock.openidm.scheduler
[  14] [active       ] org.forgerock.openidm.restlet
[   7] [unsatisfied  ] org.forgerock.openidm.external.email
[  18] [unsatisfied  ] org.forgerock.openidm.repo.orientdb
[   6] [active       ] org.forgerock.openidm.sync
[   3] [active       ] org.forgerock.openidm.script
[   5] [active       ] org.forgerock.openidm.recon
[   2] [active       ] org.forgerock.openidm.scope
[  10] [active       ] org.forgerock.openidm.http.contextregistrator
[  20] [active       ] org.forgerock.openidm.config
[   0] [active       ] org.forgerock.openidm.audit
[  21] [active       ] org.forgerock.openidm.schedule
[  17] [active       ] org.forgerock.openidm.repo.jdbc
[  16] [active       ] org.forgerock.openidm.workflow
[  13] [active       ] org.forgerock.openidm.provisioner.openicf
[   4] [active       ] org.forgerock.openidm.managed
[   9] [active       ] org.forgerock.openidm.authentication
[  11] [active       ] org.forgerock.openidm.provisioner
```

2.3. To Set Up OpenIDM With MS SQL

These instructions are specific to MS SQL Server 2012 R2 Standard Edition, running on a Windows Server 2012 R2 system. Adapt the instructions for your environment.

When you install Microsoft SQL Server, note that OpenIDM has the following specific configuration requirements:

- During the Feature Selection installation step, make sure that at least SQL Server Replication, Full Text Search, and Management Tools - Basic are selected.

  These instructions require SQL Management Studio so make sure that you include Management Tools in the installation.

- During the Database Engine Configuration step, select Mixed Mode (SQL Server authentication and Windows authentication). OpenIDM *requires* SQL Server authentication.

- TCP/IP must be enabled and configured for the correct IP address and port. To configure TCP/IP, follow these steps:

  1. Navigate to SQL Server Configuration Manager.
  2. Expand the SQL Server Network Configuration item and select "Protocols for MSSQLSERVER".
  3. Check that TCP/IP is Enabled.
  4. Select the IP Addresses tab and set the addresses and ports on which the server will listen.

     For this sample procedure, scroll down to IPAll and set TCP Dynamic Ports to 1433 (the default port for MS SQL).
  5. Click OK.
  6. Restart MS SQL Server for the configuration changes to take effect.

     To restart the server, select SQL Server Services in the left pane, double click SQL Server (MSSQLSERVER) and click Restart.
  7. If you have a firewall enabled, ensure that the port you configured in the previous step is open for OpenIDM to access MS SQL.

After you have installed MS SQL on the local host, install OpenIDM, if you have not already done so, but *do not start* the OpenIDM instance. Import the data definition and set up OpenIDM to use the MS SQL repository, as described in the following steps:

1. Use SQL Management Studio to import the data definition language script for OpenIDM into MS SQL:

   a. Navigate to SQL Server Management Studio.

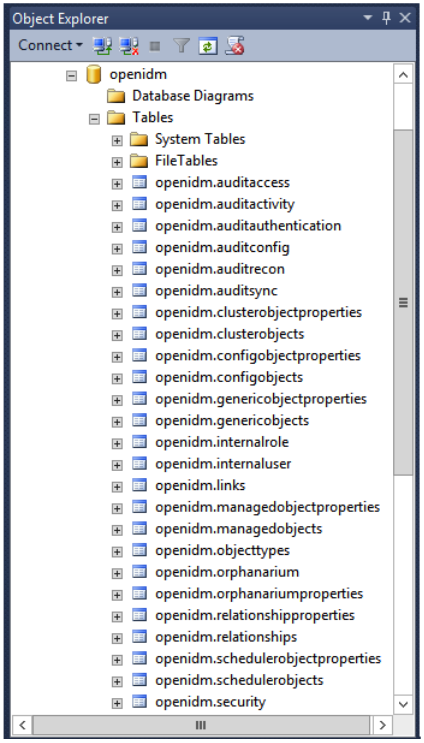
   b. On the Connect to Server panel, select Windows Authentication and click Connect.

c. Select File > Open > File and navigate to the OpenIDM data definition language script (`path\to\openidm\db\mssql\scripts\openidm.sql`). Click Open to open the file.

d. Click Execute to run the script.

2. This step creates an `openidm` database for use as the internal repository, and a user `openidm` with password `Passw0rd` who has all the required privileges to update the database. You might need to refresh the view in SQL Server Management Studio to see the `openidm` database in the Object Explorer.

   Expand Databases > openidm > Tables. You should see the OpenIDM tables in the openidm database, as shown in the following example.



   The table names are similar to those used with OrientDB.

3. OpenIDM requires an MS SQL driver that must be created from two separate JAR files. Create the driver as follows:

   a. Download the JDBC Driver 4.1 for SQL Server (`sqljdbc_4.1.5605.100_enu.tar.gz`) from Microsoft's download site. The precise URL might vary, depending on your location.

Run the downloaded executable file; it should extract multiple files, include Java archive files, to a dedicated folder.

Extract the executable Java archive file (`sqljdbc41.jar`) from the dedicated folder, using 7-zip or an equivalent file management application.

Copy the Java archive file to `openidm\db\mssql\scripts`.

b. Download the `bnd` Java archive file (bnd-1.50.0.jar) that enables you to create OSGi bundles. For more information about `bnd`, see http://bnd.bndtools.org/.

Copy the file to `openidm\db\mssql\scripts`.

c. Your `openidm\db\mssql\scripts` directory should now contain the following files:

```
bnd-1.50.0.jar  openidm.sql  sqljdbc4.bnd  sqljdbc4.jar
```

d. Bundle the two JAR files together with the following command:

```
C:\> cd \path\to\openidm\db\mssql\scripts
./> java -jar bnd-1.50.0.jar wrap -properties sqljdbc4.bnd sqljdbc41.jar
```

This step creates a single `.bar` file, named `sqljdbc41.bar`.

e. Rename the `sqljdbc41.bar` file to `sqljdbc41-osgi.jar` and copy it to the `openidm\bundle` directory:

```
./> ren sqljdbc41.bar sqljdbc41-osgi.jar
./> copy sqljdbc41-osgi.jar \path\to\openidm\bundle
```

4. Remove the default OrientDB repository configuration file (`repo.orientdb.json`) from your project's configuration directory. For example:

```
C:\> cd \path\to\openidm\my-project\conf\
.\> del repo.orientdb.json
```

5. Copy the database connection configuration file for MS SQL (`datasource.jdbc-default.json`) and the database table configuration file (`repo.jdbc.json`) to your project's configuration directory. For example:

```
C:\> cd \path\to\openidm
.\> copy db\mssql\conf\datasource.jdbc-default.json my-project\conf\
.\> copy db\mssql\conf\repo.jdbc.json my-project\conf\
```

6. Update the connection configuration in `datasource.jdbc-default.json` to reflect your MS SQL deployment. The default connection configuration is as follows:

```
{
    "driverClass" : "com.microsoft.sqlserver.jdbc.SQLServerDriver",
    "jdbcUrl" : "jdbc:sqlserver://
localhost:1433;instanceName=default;databaseName=openidm;applicationName=OpenIDM",
    "databaseName" : "openidm",
    "username" : "openidm",
    "password" : "openidm",
    "connectionTimeout" : 30000,
    "connectionPool" : {
        "type" : "bonecp"
    }
}
```

Specifically, check that the host and port match what you have configured in MS SQL.

When you have completed the preceding steps, start OpenIDM to check that the setup has been successful. After startup, you should see that `repo.jdbc` is `active`, whereas `repo.orientdb` is `unsatisfied`:

```
C:> cd \path\to\openidm
./> startup.bat
"Using OPENIDM_HOME:   \path\to\openidm"
"Using OPENIDM_OPTS:   -Xmx1024m -Xms1024m -Dfile.encoding=UTF-8"
"Using LOGGING_CONFIG:
-Djava.util.logging.config.file=\path\to\openidm\conf\logging.properties"
Using boot properties at \path\to\openidm\conf\boot\boot
.properties
-> scr list
Id   State          Name
...
[ 18] [unsatisfied ] org.forgerock.openidm.repo
.orientdb
...
[ 17] [active      ] org.forgerock.openidm.repo
.jdbc
...
```

2.4. To Set Up OpenIDM With Oracle Database

When implementing an Oracle database for OpenIDM, confer with an Oracle DBA when creating the database schema, tables, and users. This section assumes that you have configured an Oracle Database with *Local Naming Parameters (tnsnames.ora)* and a service user for use by OpenIDM.

Import the OpenIDM schema using the data definition language script (`/path/to/openidm/db/oracle/scripts/openidm.sql`), as the appropriate schema owner user.

When you have run the script, you should be able to query the `internaluser` table. The query should return two records (`openidm-admin` and `anonymous`). The output here has been formatted for legibility:

```
SQL> select * from internaluser;

OBJECTID     openidm-
admin
--------------------------------------------------------------------------------
REV          0
--------------------------------------------------------------------------------
PWD          openidm-
admin
--------------------------------------------------------------------------------
ROLES        [ { "_ref" : "repo/internal/role/openidm-admin" },
               { "_ref" : "repo/internal/role/openidm-
authorized" } ]
--------------------------------------------------------------------------------

OBJECTID     anonymous
--------------------------------------------------------------------------------
REV          0
--------------------------------------------------------------------------------
PWD          anonymous
--------------------------------------------------------------------------------
ROLES        [ { "_ref" : "repo/internal/role/openidm-
reg" } ]
--------------------------------------------------------------------------------
```

Before you start OpenIDM, create an Oracle DB driver from two separate jar files and set up the OpenIDM repository configuration for Oracle DB, as follows:

1. Download the Oracle JDBC driver for your Oracle DB version from Oracle Technology Network and place it in the `openidm/db/oracle/scripts` directory:

   ```
   $ ls /path/to/openidm/db/oracle/scripts
   ojdbc7_g.jar    openidm.sql
   ```

2. Create a bind file and edit it to match the version information for your JDBC driver.

   You can use the sample bind file located in `openidm/db/mssql/scripts`. Copy the bind file to the same location as the JDBC driver:

   ```
   $ cd /path/to/openidm/db
   $ cp mssql/scripts/sqljdbc4.bnd oracle/scripts
   $ ls oracle/scripts
   ojdbc7_g.jar    openidm.sql    sqljdbc4.bnd
   ```

   The JDBC driver version information for your driver is located in the `Specification-Version` property in the MANIFEST file of the driver:

   ```
   $ cd /path/to/openidm/db/oracle/scripts
   $ unzip -q -c ojdbc7_g.jar META-INF/MANIFEST.MF
   ...
   Specification-Vendor: Sun Microsystems Inc.
   Specification-Title: JDBC
   Specification-Version: 4
   .0
   ...
   ```

Edit the bind file to match the JDBC driver version:

```
$ more sqljdbc4.bnd
...
version=4.0
Export-Package: *;version=${version}
Bundle-Name: Oracle JDBC Driver 4.0 for SQL Server
Bundle-SymbolicName: Oracle JDBC Driver 4.0 for SQL Server
Bundle-Version: ${version}
```

3. Download the `bnd` Java archive file (bnd-1.50.0.jar) that enables you to create OSGi bundles. For more information about `bnd`, see http://bnd.bndtools.org/.

   Place the `bnd` Java archive file in the same directory as the JDBC driver, and the bind file:

   ```
   $ ls /path/to/openidm/db/oracle/scripts
   bnd-1.50.0.jar    ojdbc7_g.jar    openidm.sql    sqljdbc4.bnd
   ```

4. Change to the directory in which the script files are located and run the following command to create the OSGi bundle:

   ```
   $ cd /path/to/openidm/db/oracle/scripts
   $ java -jar bnd-1.50.0.jar wrap -properties sqljdbc4.bnd ojdbc7_g.jar
   Dec 10, 2013 9:53:28 AM java.util.prefs.FileSystemPreferences$1 run
   INFO: Created user preferences directory.
   ojdbc7_g.jar 984 0
   ```

   A new `.bar` file has now been created:

   ```
   $ ls
   bnd-1.50.0.jar  ojdbc7_g.bar  ojdbc7_g.jar  openidm.sql  sqljdbc4.bnd
   ```

5. Move the `.bar` file to the `openidm/bundle` directory and rename it with a `.jar` extension. The actual name of the file is unimportant:

   ```
   $ mv ojdbc7_g.bar /path/to/openidm/bundle/ojdbc7_g-osgi.jar
   ```

6. Remove the default OrientDB configuration file (`repo.orientdb.json`) from your project's configuration directory. For example:

   ```
   $ rm /path/to/openidm/my-project/conf/repo.orientdb.json
   ```

7. Copy the database connection configuration file for Oracle (`datasource.jdbc-default.json`) and the database table configuration file (`repo.jdbc.json`) to your project's configuration directory. For example:

   ```
   $ cd /path/to/openidm/
   $ cp db/oracle/conf/datasource.jdbc-default.json my-project/conf/
   $ cp db/oracle/conf/repo.jdbc.json my-project/conf/
   ```

8. Update the connection configuration in `datasource.jdbc-default.json` to reflect your Oracle database deployment. The default connection configuration is as follows:

```
{
    "driverClass" : "oracle.jdbc.OracleDriver",
    "jdbcUrl" : "jdbc:oracle:thin:@//HOSTNAME:PORT/DEFAULTCATALOG",
    "databaseName" : "openidm",
    "username" : "openidm",
    "password" : "openidm",
    "connectionTimeout" : 30000,
    "connectionPool" : {
        "type" : "bonecp"
    }
}
```

The `"jdbcUrl"` corresponds to the URL of the Oracle DB listener, including the service name, based on your configured Local Naming Parameters (tnsnames.ora). It should be whatever is appropriate for your environment.

The `DEFAULTCATALOG` should match the user who "owns" the tables. If your schema owner is `openidm`, the `DEFAULTCATALOG` should also be `openidm`. This will cause OpenIDM to generate queries such as `"SELECT objectid FROM openidm.internaluser"`.

The `"username"` and `"password"` corresponds to the credentials of the service user that connects from OpenIDM.

When you have set up Oracle database for use as the OpenIDM internal repository, start OpenIDM to check that the setup has been successful. On startup, a number of INFO messages are output, as the predefined queries are processed.

After startup, you should see that `repo.jdbc` is `active`, whereas `repo.orientdb` is `unsatisfied`:

```
$ cd /path/to/openidm
$ ./startup.sh
Using OPENIDM_HOME:   /path/to/openidm
Using OPENIDM_OPTS:   -Xmx1024m -Xms1024m
Using LOGGING_CONFIG:
-Djava.util.logging.config.file=/path/to/openidm/conf/logging.properties
Using boot properties at /path/to/openidm/conf/boot/boot
.properties
....
-> scr list
   Id   State          Name
...
[   2] [unsatisfied  ] org.forgerock.openidm.repo
.orientdb
...
[   3] [active       ] org.forgerock.openidm.repo
.jdbc
...
```

2.5. To Set Up OpenIDM With PostgreSQL

This procedure assumes that PostgreSQL (version 9.3 or later) is installed and running on the local host.

Before starting OpenIDM for the first time, set up OpenIDM to use a PostgreSQL repository, as described in the following procedure:

1. OpenIDM includes a script (`path/to/openidm/db/postgresql/scripts/createuser.pgsql`) that sets up an `openidm` database and user, with a default password of `openidm`. The script also grants the appropriate permissions.

   Edit this script if you want to change the password of the `openidm` user, for example:

   ```
   $ more /path/to/openidm/db/postgresql/scripts/createuser.pgsql
   create user openidm with password 'mypassword';
   create database openidm encoding 'utf8' owner openidm;
   grant all privileges on database openidm to openidm;
   ```

2. As the `postgres` user, execute the `createuser.pgsql` script as follows:

   ```
   $ psql -U postgres < /path/to/openidm/db/postgresql/scripts/createuser.pgsql
   CREATE ROLE
   CREATE DATABASE
   GRANT
   ```

3. Execute the `openidm.pgsql` script as the new `openidm` user that you created in the first step:

   ```
   $ psql -U openidm < /path/to/openidm/db/postgresql/scripts/openidm.pgsql

   CREATE SCHEMA
   CREATE TABLE
   CREATE TABLE
   CREATE TABLE
   CREATE INDEX
   CREATE INDEX
   ...
   START TRANSACTION
   INSERT 0 1
   INSERT 0 1
   COMMIT
   CREATE INDEX
   CREATE INDEX
   ```

   Your database has now been initialized.

4. Remove the OrientDB repository configuration file (`repo.orientdb.json`) from your project's configuration directory. For example:

   ```
   $ rm /path/to/openidm/my-project/conf/repo.orientdb.json
   ```

5. Copy the database connection configuration file for PostgreSQL (`datasource.jdbc-default.json`) and the database table file (`repo.jdbc.json`) to your project's configuration directory. For example:

   ```
   $ cd /path/to/openidm
   $ cp db/postgres/conf/datasource.jdbc-default.json my-project/conf/
   $ cp db/postgres/conf/repo.jdbc.json my-project/conf/
   ```

6. If you changed the password in step 1 of this procedure, edit the `datasource.jdbc-default.json` file to set the value for the `"password"` field to whatever password you set for the `openidm` user. For example, if you changed the connection password to `mypassword`, edit the file as follows:

```
$ more conf/datasource.jdbc-default.json
{
    "driverClass" : "org.postgresql.Driver",
    "jdbcUrl" : "jdbc:postgresql://localhost:5432/openidm",
    "databaseName" : "openidm",
    "username" : "openidm",
    "password" : "mypassword",
    "connectionTimeout" : 30000,
    "connectionPool" : {
        "type" : "bonecp"
    }
}
```

7. PostgreSQL is now set up for use as the OpenIDM internal repository.

   Start OpenIDM to check that the setup has been successful. After startup, you should see that `repo.jdbc` is `active`, whereas `repo.orientdb` is `unsatisfied`:

```
-> OpenIDM ready
scr list
Id   State          Name
...
[   4] [unsatisfied  ] org.forgerock.openidm.repo
.orientdb
...
[   3] [active       ] org.forgerock.openidm.repo
.jdbc
..
.
->
```

8. If you are using the default project configuration, run the `default_schema_optimization.pgsql` script to create the required indexes. This script must be executed by a user with SUPERUSER privileges, so that the extension can be created. By default, this is the `postgres` user.

   The file includes extensive comments on the indexes that are being created:

```
$ psql -U postgres openidm < /path/to/openidm/db/postgresql/scripts/default_schema_optimization.pgsql
CREATE INDEX
CREATE INDEX
CREATE INDEX
CREATE INDEX
CREATE INDEX
CREATE INDEX
```

2.6. To Set Up OpenIDM With IBM DB2

This section makes the following assumptions about the DB2 environment. If these assumptions do not match your DB2 environment, adapt the subsequent instructions accordingly.

- DB2 is running on the localhost, and is listening on the default port (50000).
- The user `db2inst1` is configured as the DB2 instance owner, and has the password `Passw0rd1`.

This section assumes that you will use basic username/password authentication. For instructions on configuring Kerberos authentication with a DB2 repository, see "Configuring OpenIDM for Kerberos Authentication With a DB2 Repository".

Before you start, make sure that OpenIDM is stopped.

```
$ cd /path/to/openidm/
$ ./shutdown.sh
OpenIDM is not running, not stopping.
```

Set up OpenIDM to use the DB2 repository, as described in the following steps.

1. Create a bundled DB2 JDBC driver, and copy it to the `openidm/bundle` directory, as follows:

   a. Download the DB2 JDBC driver for your database version from the IBM download site and place it in the `openidm/db/db2/scripts` directory.

   Use either the `db2jcc.jar` or `db2jcc4.jar`, depending on your DB2 version. For more information, see the DB2 JDBC Driver Versions.

   ```
   $ ls /path/to/db/db2/scripts/
   db2jcc.jar  openidm.sql
   ```

   b. Create a bind file and edit it to match the version information for your JDBC driver.

   You can use the sample bind file located in `openidm/db/mssql/scripts`. Copy the bind file to the same location as the JDBC driver.

   ```
   $ cd /path/to/openidm/db
   $ cp mssql/scripts/sqljdbc4.bnd db2/scripts/
   $ ls db2/scripts
   db2jcc.jar  openidm.sql  sqljdbc4.bnd
   ```

   The JDBC driver version information for your driver is located in the `Specification-Version` property in the MANIFEST file of the driver.

   ```
   $ cd /path/to/openidm/db/db2/scripts
   $ unzip -q -c db2jcc.jar META-INF/MANIFEST.MF
   Manifest-Version: 1.0
   Created-By: 1.4.2 (IBM Corporation)
   ```

   Edit the bind file to match the JDBC driver version.

   ```
   $ more sqljdbc4.bnd
   ...
   version=1.0
   Export-Package: *;version=${version}
   Bundle-Name: IBM JDBC DB2 Driver
   Bundle-SymbolicName: com.ibm.db2.jcc.db2driver
   Bundle-Version: ${version}
   ```

   c. Download the `bnd` Java archive file (bnd-1.50.0.jar) that enables you to create OSGi bundles. For more information about `bnd`, see http://bnd.bndtools.org/.

Place the `bnd` Java archive file in the same directory as the JDBC driver, and the bind file.

```
$ ls /path/to/openidm/db/db2/scripts
bnd-1.50.0.jar  db2jcc.jar  openidm.sql  sqljdbc4.bnd
```

d. Change to the directory in which the script files are located and run the following command to create the OSGi bundle.

```
$ cd /path/to/openidm/db/db2/scripts
$ java -jar bnd-1.50.0.jar wrap -properties sqljdbc4.bnd db2jcc.jar
Oct 01, 2015 11:50:56 PM java.util.prefs.FileSystemPreferences$1 run
INFO: Created user preferences directory.
db2jcc 1149 0
```

A new `.bar` file has now been created.

```
$ ls
bnd-1.50.0.jar  db2jcc.bar  db2jcc.jar  openidm.sql  sqljdbc4.bnd
```

e. Move the `.bar` file to the `openidm/bundle` directory and rename it with a `.jar` extension. The actual name of the file is unimportant.

```
$ mv db2jcc.bar /path/to/openidm/bundle/db2jcc-osgi.jar
```

2. Remove the default OrientDB configuration file (`repo.orientdb.json`) from your project's configuration directory. For example:

```
$ rm /path/to/openidm/my-project/conf/repo.orientdb.json
```

3. Copy the database connection configuration file for DB2 (`datasource.jdbc-default.json`) and the database table configuration file (`repo.jdbc.json`) to your project's configuration directory. For example:

```
$ cd /path/to/openidm/
$ cp db/db2/conf/datasource.jdbc-default.json my-project/conf/
$ cp db/db2/conf/repo.jdbc.json my-project/conf/
```

4. Edit the `connection` property in the repository configuration file to match your DB2 environment.

   Update the connection configuration in `datasource.jdbc-default.json` to reflect your DB2 deployment. The default connection configuration is as follows:

```
{
    "driverClass" : "com.ibm.db2.jcc.DB2Driver",
    "jdbcUrl" : "jdbc:db2://HOSTNAME:PORT/dopenidm:retrieveMessagesFromServerOnGetMessage=true;",
    "databaseName" : "sopenidm",
    "username" : "openidm",
    "password" : "openidm",
    "connectionTimeout" : 30000,
    "connectionPool" : {
        "type" : "bonecp"
    }
}
```

5. Create a user database for OpenIDM (`dopenidm`).

```
$ db2 create database dopenidm
```

6. Import the data definition language script for OpenIDM into your DB2 instance.

```
$ cd /path/to/openidm
$ db2 -i -tf db/db2/scripts/openidm.sql
```

The database schema is defined in the `SOPENIDM` database.

7. You can show the list of tables in the repository, using the **db2 list** command, as follows:

```
$ db2 LIST TABLES for all

 Table/View                      Schema          Type  Creation
 time
------------------------------- --------------- ----- --------------------------
AUDITACCESS                     SOPENIDM        T     2015-10-01-11.58.04.313685
AUDITACTIVITY                   SOPENIDM        T     2015-10-01-11.58.03.671342
AUDITAUTHENTICATION             SOPENIDM        T     2015-10-01-11.58.02.159573
AUDITCONFIG                     SOPENIDM        T     2015-10-01-11.58.03.307248
AUDITRECON                      SOPENIDM        T     2015-10-01-11.58.02.526214
AUDITSYNC                       SOPENIDM        T     2015-10-01-11.58.02.936434
CLUSTEROBJECTPROPERTIES         SOPENIDM        T     2015-10-01-11.58.05.968933
CLUSTEROBJECTS                  SOPENIDM        T     2015-10-01-11.58.05.607075
CONFIGOBJECTPROPERTIES          SOPENIDM        T     2015-10-01-11.58.01.039999
CONFIGOBJECTS                   SOPENIDM        T     2015-10-01-11.58.00.570231
GENERICOBJECTPROPERTIES         SOPENIDM        T     2015-10-01-11.57.59.583530
GENERICOBJECTS                  SOPENIDM        T     2015-10-01-11.57.59.152221
INTERNALUSER                    SOPENIDM        T     2015-10-01-11.58.04.060990
LINKS                           SOPENIDM        T     2015-10-01-11.58.01.349194
MANAGEDOBJECTPROPERTIES         SOPENIDM        T     2015-10-01-11.58.00.261556
MANAGEDOBJECTS                  SOPENIDM        T     2015-10-01-11.57.59
.890152
...
```

The table names are similar to those used with OrientDB.

When you have set up DB2 for use as the OpenIDM internal repository, start OpenIDM to check that the setup has been successful. After startup, you should see that `repo.jdbc` is `active`, whereas `repo.orientdb` is `unsatisfied`.

```
$ cd /path/to/openidm
$ ./startup.sh
Using OPENIDM_HOME:   /path/to/openidm
Using OPENIDM_OPTS:   -Xmx1024m -Xms1024m
Using LOGGING_CONFIG:
-Djava.util.logging.config.file=/path/to/openidm/conf/logging.properties
Using boot properties at /path/to/openidm/conf/boot/boot.properties
-> scr list

Id   State          Name
[  19] [active       ] org.forgerock.openidm.config.starter
[  23] [active       ] org.forgerock.openidm.taskscanner
[   8] [active       ] org.forgerock.openidm.external.rest
[  12] [active       ] org.forgerock.openidm.provisioner.openicf.connectorinfoprovider
```

```
[  15] [active        ] org.forgerock.openidm.ui.simple
[   1] [active        ] org.forgerock.openidm.router
[  22] [active        ] org.forgerock.openidm.scheduler
[  14] [active        ] org.forgerock.openidm.restlet
[   7] [unsatisfied   ] org.forgerock.openidm.external.email
[  18] [unsatisfied   ] org.forgerock.openidm.repo.orientdb
[   6] [active        ] org.forgerock.openidm.sync
[   3] [active        ] org.forgerock.openidm.script
[   5] [active        ] org.forgerock.openidm.recon
[   2] [active        ] org.forgerock.openidm.scope
[  10] [active        ] org.forgerock.openidm.http.contextregistrator
[  20] [active        ] org.forgerock.openidm.config
[   0] [active        ] org.forgerock.openidm.audit
[  21] [active        ] org.forgerock.openidm.schedule
[  17] [active        ] org.forgerock.openidm.repo.jdbc
[  16] [active        ] org.forgerock.openidm.workflow
[  13] [active        ] org.forgerock.openidm.provisioner.openicf
[   4] [active        ] org.forgerock.openidm.managed
[   9] [active        ] org.forgerock.openidm.authentication
[  11] [active        ] org.forgerock.openidm.provisioner
```

2.6.1. Configuring OpenIDM for Kerberos Authentication With a DB2 Repository

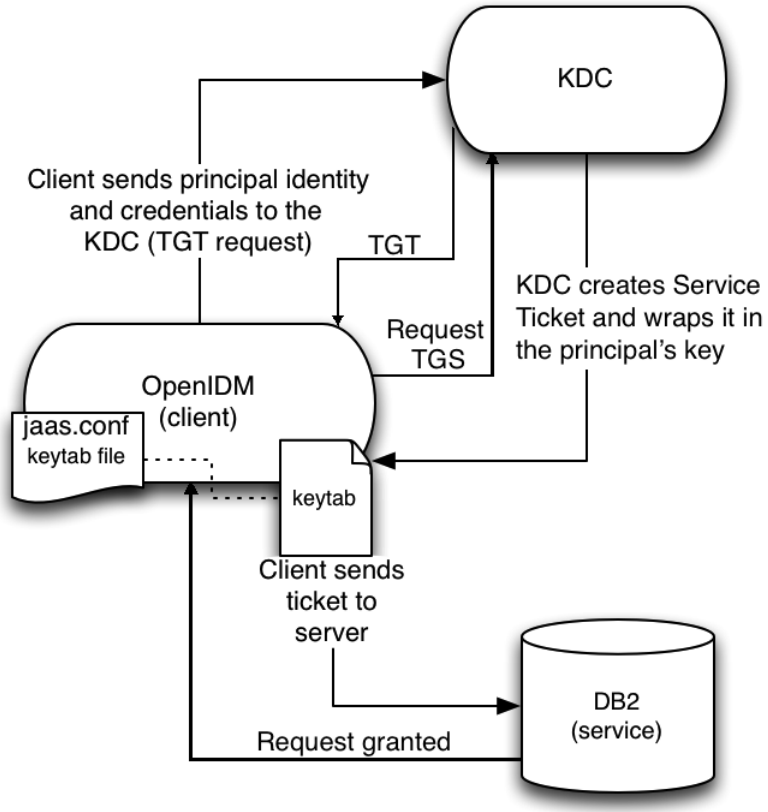
By default, OpenIDM uses the username and password configured in the repository connection configuration file (`conf/datasource.jdbc-default.json`) to connect to the DB2 repository. You can configure OpenIDM to use Kerberos authentication instead.

In this scenario, OpenIDM acts as a *client* and requests a Kerberos ticket for a *service*, which is DB2, through the JDBC driver.

This section assumes that you have configured DB2 for Kerberos authentication. If that is not the case, follow the instructions in the corresponding DB2 documentation before you read this section.

The following diagram shows how the ticket is obtained and how the keytab is referenced from OpenIDM's `jaas.conf` file.

Using Kerberos to Connect to a DB2 Repository



To configure OpenIDM for Kerberos authentication:

1. Create a keytab file, specifically for use by OpenIDM.

   A Kerberos keytab file (`krb5.keytab`) is an encrypted copy of the host's key. The keytab enables DB2 to validate the Kerberos ticket that it receives from OpenIDM. You must create a keytab file on the host that OpenIDM runs on. The keytab file must be secured in the same way that you would secure any password file. Specifically, only the user running OpenIDM should have read and write access to this file.

   Create a keytab for DB2 authentication, in the file `openidm/security/idm.keytab/`:

   ```
   $ kadmin -p kadmin/admin -w password
   $ kadmin: ktadd -k /path/to/openidm/security/idm.keytab db2/idm.example.com
   ```

2. Make sure that the DB2 user has read access to the keytab.

3. Copy the DB2 Java Authentication and Authorization Service (JAAS) configuration file to the OpenIDM `security` directory:

```
$ cd path/to/openidm
$ cp db/db2/conf/jaas.conf security/
```

   By default, OpenIDM assumes that the keytab is in the file `openidm/security/idm.keytab` and that the principal identity is `db2/idm.example.com@EXAMPLE.COM`. Change the following lines in the `jaas.conf` file if you are using a different keytab:

```
keyTab="security/idm.keytab"
principal="db2/idm.example.com@EXAMPLE.COM"
```

4. Adjust the authentication details in your DB2 connection configuration file (`conf/datasource.jdbc-default.json`). Edit that file to remove `password` field and change the username to the instance owner (`db2`). The following excerpt shows the modified file:

```
{
    ...
    "databaseName" : "sopenidm",
    "username" : "db2",
    "connectionTimeout" : 30000,
    ...
}
```

5. Edit your project's `conf/system.properties` file, to add the required Java options for Kerberos authentication.

   In particular, add the following two lines to that file:

```
db2.jcc.securityMechanism=11
java.security.auth.login.config=security/jaas.conf
```

6. Restart OpenIDM.

Chapter 3
Removing and Moving OpenIDM Software

This chapter shows you how to uninstall OpenIDM software and to move an existing install to a different location.

To Remove OpenIDM Software

1. (Optional) Stop OpenIDM services if they are running, as described in "To Stop the OpenIDM Services".
2. Remove the file system directory where you installed OpenIDM software:

```
$ rm -rf /path/to/openidm
```

3. (Optional) If you use a JDBC database for the internal repository, you can drop the `openidm` database.

To Move OpenIDM Software

If you want to move OpenIDM to a different directory, you do not have to uninstall and reinstall. To move an existing OpenIDM instance, follow these steps:

1. Shut down OpenIDM, as described in "To Stop the OpenIDM Services".
2. Remove the `felix-cache` directory:

```
$ cd path/to/openidm
$ rm -rf felix-cache
```

3. Move the files:

```
$ mv path/to/openidm path/to/new-openidm
```

4. Start OpenIDM in the new location:

```
$ cd path/to/new-openidm
$ ./startup.sh
```

Chapter 4

Updating OpenIDM

The update process is largely dependent on your deployment and on the extent to which you have customized OpenIDM. Engage ForgeRock Support Services for help in updating an existing deployment.

- For information on migrating a deployment from OpenIDM 3.1.0 to OpenIDM 4, see "Migrating from OpenIDM 3.1 to OpenIDM 4".
- For information on migrating a deployment from OpenIDM 3.0.0 to 3.1.0, see *Migrating to OpenIDM 3.1.0* in the *OpenIDM 3.1.0 Installation Guide*.
- For information on migrating a deployment from OpenIDM 2.1.0 to 3.0.0, see *Migrating to OpenIDM 3.0.0* in the *OpenIDM 3.0.0 Installation Guide*.

4.1. Migrating from OpenIDM 3.1 to OpenIDM 4

The steps outlined in this section will help you take advantage of the new functionality offered in OpenIDM 4, while preserving your custom configuration where possible. Before you start, read through the changes made for OpenIDM 4 in "*OpenIDM Compatibility*" in the *Release Notes*. Some of these changes might affect your existing deployment.

Note

Updates from OpenIDM 3.1 to OpenIDM 4 are still a manual process. Starting with OpenIDM 4, you can update OpenIDM with UI or CLI tools, as described in "OpenIDM Update Process".

To perform a migration from OpenIDM 3.1 to OpenIDM 4, follow these steps. For the purposes of this procedure, the path to the existing instance of OpenIDM 3.1 instance is defined as `/path/to/openidm-3.1`. In contrast, the path to the OpenIDM 4 is defined as `/path/to/openidm-4`:

1. Download and extract the OpenIDM 4 server.
2. Stop your existing OpenIDM 3.1.0 server, if it is running:

```
$ cd /path/to/openidm-3.1
$ ./shutdown.sh
Stopping OpenIDM (81491)
```

3. Backup: Save your current deployment. Archive the `openidm` directory.

4. Boot properties: On the OpenIDM 4 server, edit the `conf/boot/boot.properties` file to match any customizations that you made on your OpenIDM 3.1 server. Specifically, check the following elements:

   - The HTTP, HTTPS, and mutual authentication ports are specified in the `conf/boot/boot.properties` file. If you changed the default ports in your OpenIDM 3.1 deployment, make sure that the corresponding ports are specified in this file.

   - Check that the keystore and truststore passwords match the current passwords for the keystore and truststore of your OpenIDM 3.1 deployment.

   Depending on the level of customization you have made in your current deployment, it might be simpler to start with your OpenIDM 3.1 `boot.properties` file, and copy all new settings from that file to the version associated with OpenIDM 4. However, as a best practice, you should keep all configuration customizations (including new properties and changed settings) in a single location. You can then copy and paste these changes as appropriate.

5. **Updating `logging.properties`**

   Recent security fixes prevent Jetty from logging sensitive data, such as passwords. Verify that your `conf/logging.properties` file includes the following excerpt (and add the excerpt if necessary) to prevent unnecessary data from being logged:

   ```
   # Logs the output from Jetty
     # Sets the following Jetty classes to INFO level by default because if logging is set to FINE or
    higher,
     # sensitive information can be leaked into the logs
     org.eclipse.jetty.server.HttpChannel.level=INFO
     org.eclipse.jetty.server.HttpConnection.level=INFO
     org.eclipse.jetty.server.HttpInput.level=INFO
     org.eclipse.jetty.http.HttpParser.level=INFO
     org.eclipse.jetty.io.ssl.SslConnection.level=INFO
   ```

   This configuration logs request data at `INFO` level, preventing data such as password changes from being logged. In situations where you *need* to log all data (for example, if you are debugging an issue in a test environment) change the settings here to `FINE` or `FINEST`. For example:

   ```
   org.eclipse.jetty.server.HttpConnection.level=FINE
   ```

6. Security files: Copy the contents of your OpenIDM 3.1 `security/` folder to the OpenIDM 4 instance:

   ```
   $ cd /path/to/openidm-4
   $ cp -r /path/to/openidm-3.1/security .
   ```

   Examine the following excerpt from the `boot.properties` file. OpenIDM automatically prepends the locations of the `keystore.jceks` and `truststore` files with the installation directory.

```
...
openidm.keystore.type=JCEKS
openidm.truststore.type=JKS
openidm.keystore.provider=
openidm.keystore.location=security/keystore.jceks
openidm.truststore.location=security/truststore
```

7. Scripts: Migrate any custom scripts or default scripts that you have modified to the scripts directory of your OpenIDM 4 instance. In general, custom and customized scripts should be located in the `openidm-3.1/script` directory on the OpenIDM 3.1 deployment:

   - For custom scripts, review "*OpenIDM Compatibility*" in the *Release Notes*. If you're confident that the scripts will work as intended on OpenIDM 4, then copy these scripts to the new instance. For example:

     ```
     $ cd /path/to/openidm-4
     $ cp /path/to/openidm-3.1/script/my-custom-script.js script/
     ```

   - If you modified an existing OpenIDM 3.1 script, compare the default versions of the OpenIDM 3.1 and OpenIDM 4 scripts. If you verify that nothing has changed, review what you've customized against "*OpenIDM Compatibility*" in the *Release Notes*. If you're confident that your changes will work as intended, then copy the customized scripts to the new `openidm-4/script` directory. For example:

     ```
     $ cd /path/to/openidm-4
     $ cp /path/to/openidm-3.1/script/policy.js script/
     ```

   - If a default script has changed since the 3.1 release, copy the modified script to the `openidm-4/script` directory. For example:

     ```
     $ cd /path/to/openidm-3.1
     $ cp bin/default/script/linkedView.js script/
     ```

     Check that your customizations work as expected, then port the changes for OpenIDM 4 to the new script in the `openidm-4/script` directory.

8. Provisioner files: Modify any customized provisioner configurations in your existing project to point to the connectors that are provided with OpenIDM-4. Specifically, make sure that the `"connectorRef"` properties reflect the new connectors, where applicable. For example:

   ```
   {
        "bundleName": "org.forgerock.openicf.connectors.ldap-connector",
        "bundleVersion": "[1.4.0.0,2.0.0.0)",
        "displayName": "LDAP Connector",
        "connectorName": "org.identityconnectors.ldap.LdapConnector"
   },
   ```

   Alternatively, copy the connector .jars from your existing installation into the `openidm/connectors/` folder of the new installation.

9. Complete the OpenIDM 4 installation, as described in "*Installing OpenIDM Services*".

10. As there is no automated way to migrate a customized configuration to OpenIDM 4, we recommend the following strategy:

    - Start with the default 3.1 configuration.
    - For each configuration file that you have customized, use a file comparison tool such as the UNIX **diff** utility to assess the differences between your customized file and the OpenIDM 4 file.
    - Based on the results of the **diff**, use either your existing file as a base and port the OpenIDM 4 changes to that file, or vice versa. Ultimately, you want to preserve your customizations but ensure that you are up to date with the latest default configuration. All files should end up in the `openidm-4/conf` directory.
    - Pay particular attention to the `conf/repo.jdbc.json` file in your existing deployment. Start with a **diff** between your OpenIDM 3.1 instance and your OpenIDM 4 instance of this file. For a more complete analysis of these differences, see "Changes in Database Schema" in the *Release Notes*.

      If you have customized this file, make sure that these customizations are ported to the corresponding file updated deployment. For example, if you have defined any new queries, add these queries to the OpenIDM 4 instance of the `repo.jdbc.json` file.

11. Migrate your internal user data, managed objects, and reconciliation and audit data, if required.

    When you migrate this data, note the following points:

    - Database connection information has been moved from `repo.jdbc.json` into a separate file, `datasource.jdbc-default.json`
    - The database audit logging schema has been upgraded significantly for OpenIDM 4:
      - OpenIDM 4 includes new `auditauthentication` and `auditconfig` tables.
      - To support additional audit data, OpenIDM 4 includes extensive changes to existing audit tables: `auditaccess`, `auditactivity`, `auditrecon`, and `auditsync`. These changes include a number of additional fields.

        For more information, see "*Using Audit Logs*" in the *Integrator's Guide*.

        If you have used auditing extensively for versions of OpenIDM before OpenIDM 4, read through your repository configuration file, `repo.jdbc.json`. Compare the six different OpenIDM 4 audit tables with comparable tables in your deployment.

        In many cases, it may be fastest to start new audit tables after a migration to OpenIDM 4.
    - In the `link` table, OpenIDM 4 includes a new `linkQualifier` column. When updating your repository, add `default` as values in that column.

Your data migration strategy might vary, depending on your repository. You can either migrate your existing 3.1.0 database, or start with a new OpenIDM-4 database and import your existing data.

Warning

Before migrating an existing database, review the differences in the `repo.jdbc.json` files with a DBA. Before running database migration commands, have that DBA review schema scripts in the following directory: `/path/to/openidm-4/db/repo/scripts`. These scripts typically include a file such as `openidm.sql`.

To migrate an existing database:

- Use the approved schema scripts from the noted OpenIDM-4 `db/repo/scripts` directory. Apply the changes from that script to your existing database.

  Use the `--force` option in MySQL (or an equivalent option for your repository type) to create any new tables.

To start with a new database:

- Set up a clean repository, using the appropriate schema scripts from the new OpenIDM-4 instance, in the following directory: `/path/to/openidm-4/db/repo/scripts`).
- Use a schema comparison tool and adjust the tables in your existing repository to match the schema in the new repository.
- Export your existing data to the new repository.

12. If you are using the UI, clear your browser cache after the migration. The browser cache contains files from the previous OpenIDM release, that might not be refreshed when you log in to the new UI.
13. Start OpenIDM-4:

```
$ cd /path/to/openidm-4
$ ./startup.sh
```

14. Test that your existing clients and scripts are working as intended.

4.2. OpenIDM Update Process

OpenIDM must be running when you launch an update (using the UI or the CLI). The server is placed in maintenance mode during the update. For more information, see "Placing an OpenIDM Instance in Maintenance Mode".

OpenIDM 4 includes MD5 checksums for each file. When updating, these checksums allow OpenIDM to verify file changes.

The update process compares the current checksum of every file with:

- The checksum of the file as originally installed.
- The checksum of any file included in the update or patch.

Based on that comparison, and the directory where the file is stored, OpenIDM subdivides the possible results into the categories shown here:

Update Actions on Files

Directory	Existing File	Extension	New File from the Update/Patch
`openidm/bundle`, `openidm/bin`, *`project-dir`*`/conf`	Backup saved, with the noted extension	-old-*unix_time*	Written to the target directory
Other directories	Still used after update	-new-*unix_time*	Written to the target directory, with the noted extension

Because of the nature of Java archives in the `openidm/bundle` directory, OpenIDM updates all files in that directory, even if they have not changed.

Note

The `unix_time` is the number of seconds since the `Unix Epoch` of January 1, 1970.

For a list of checksums, review the `openidm/.checksums.csv` file. It contains a list of checksums for every original file in your `openidm/` directory.

You need to copy update archives, in zip format, to the `openidm/bin/update` directory. OpenIDM creates that directory during the start process.

4.2.1. Updating OpenIDM 4 via the UI

With OpenIDM, you can install minor or major updates through the UI. To do so, navigate to `https://localhost:8443/admin`, click Configure > System Preferences, and click the Update tab.

As noted in the UI, the `openidm/bin/update` directory is used to contain updates loaded in zip format. You can copy files to that directory while OpenIDM is running. Refresh the browser, and the updates appear.

The instructions are intuitive. When you select an update, OpenIDM gives you your first chance to cancel with the following message:

```
When you confirm, OpenIDM allows running jobs to finish.
Scheduled jobs are paused and OpenIDM goes into maintenance mode.
```

During the update process, these messages may appear:

```
Pausing scheduler...
All scheduled jobs complete, scheduler is shut down.
Successfully entered maintenance mode.
Getting update preview.
Success
```

You should see an `Installation Preview` screen, with a list of affected files, in the following categories:

Preview of File Updates

Status	Description
Unexpected	Existing file not from the original installation
Nonexistent	A file in the new installation that does not exist in the current installation. This is always the status for *versioned* files, such as the `openidm-*.jar` files in the `openidm/bundle/` directory.
Deleted	OpenIDM replaces the file during the update
Changed	May be replaced or preserved; depends on the directory location as described in "Update Actions on Files"
Unchanged	File not changed from the original. If replaced during an update, this file is not backed up.
Preserved	OpenIDM retains the customized file during the update. If available, OpenIDM adds a newer file with a timestamp extension.

From the Installation Preview screen, you can select Cancel or Install Update.

When you click Install Update, you may be presented with a license. If so, you will have to accept it to continue.

You should see an "Installing Update" screen, which lists the files that are being updated. The installation process may take several minutes, especially if you are updating all of the files in your OpenIDM deployment.

Note

When you see the "Successfully Installed" screen, OpenIDM displays a 30 second countdown. This gives you an opportunity to click Download Update Report. If the timer allows, you can then click Restart Now.

Depending on your configuration, the restart process may take several minutes.

When the process is complete, an installation report appears, detailing all files that have affected in some way. You should see a list of changes that OpenIDM has made to files. You should also see a link to download the list, in JSON format.

The installation report sub-divides files into three categories:

Updated Files: What Happened

Status	Description
Replaced	Obsolete files (including files with an older version number) are updated; missing files are replaced. Any files that were changed are backed up.
Preserved	Your custom version of this file is saved; the update writes a new version of this file in the same directory, with a `new-unix_time` extension.
Applied	The update changed the configuration file. No backup is made, as this change is essential to OpenIDM.

Note

If you download the update report, you can format it with a tool such as `http://jsonprettyprint.com`, or redirect it through the **jq** filter with a command such as:

```
$ cat downloaded_file | jq .
```

4.2.2. Updating OpenIDM 4 via the CLI

You can also apply the update via the **cli.sh** (UNIX) and the **cli.bat** (Windows) scripts. For more information on these scripts, see "*OpenIDM Command-Line Interface*" in the *Integrator's Guide*.

The following command updates the local system with the `openidm-new.zip` binary:

```
$ ./cli.sh update
 \
--acceptLicense
 \
--user openidm-admin:openidm-admin
 \
--url http://localhost:8080/openidm \
openidm-new.zip
```

The `update` subcommand takes the following options:

`--acceptLicense`

Automatically accept the license shown in `/path/to/openidm/legal-notices/Forgerock_License.txt`. If you omit this option, the update process prompts you to accept or decline the license.

`--log` (optional)

The path to the file that will contain log messages related to the update process.

Default: `logs/update.log`

`--maxJobsFinishWaitTimeMs` (optional)

The maximum time, in milliseconds, that the command should wait for scheduled jobs to finish before proceeding with the update.

Default: `-1`, (the process exits immediately if any jobs are running)

`--maxUpdateWaitTimeMs` (optional)

The maximum time, in milliseconds, that the server should wait for the update process to complete.

Default: `30000` ms

`--quiet` (optional)

Use quiet mode, that is, output messages only to the update log file and not to the console.

`--url`

The URL of the OpenIDM REST service.

To update a remote OpenIDM system, substitute the IP address or hostname of the remote system for `localhost`. If you update a remote system, make sure the update file is available in the `openidm/bin/update` directory of that system.

`--user`

The login credentials (username and password) of an administrative user. Any managed or internal user with an administrative role has the rights to update an OpenIDM installation.

If you do not run the command in quiet mode, messages similar to the following are displayed in the console window where you launched the command:

```
Executing ./cli.sh...
Starting shell in /path/to/openidm
Using boot properties at /path/to/openidm/conf/boot/boot.properties
No license found to accept in this update archive.
Pausing the Scheduler
Scheduler has been paused.
Waiting for running jobs to finish.
All running jobs have finished.
Entering into maintenance mode...
Now in maintenance mode.
Installing the update archive openidm-new.zip
Update procedure is still processing...
Update procedure is still processing...
Update procedure is still processing...
Update procedure is still processing...
Update procedure is still processing...
The update process is complete with a status of COMPLETE
Restarting OpenIDM.
Restart request completed.
```

4.3. Placing an OpenIDM Instance in Maintenance Mode

OpenIDM 4 supports a Maintenance Service that disables non-essential services of a running OpenIDM instance, in preparation for a migration to a later version. When maintenance mode is

enabled, services such as recon, sync, scheduling, and workflow are disabled. The complete list of disabled services is output to the OpenIDM log file.

The router remains functional and requests to the `maintenance` endpoint continue to be serviced. Requests to endpoints that are serviced by a disabled component return the following response:

```
404 Resource endpoint-name not found
```

Before you enable maintenance mode, you should temporarily suspend all scheduled tasks. For more information, see "Pausing Scheduled Tasks" in the *Integrator's Guide*.

You can enable and disable maintenance mode over the REST interface.

To enable maintenance mode, run the following command:

```
$ curl \
 --cacert self-signed.crt \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request POST \
 "https://localhost:8443/openidm/maintenance?_action=enable"
{
  "maintenanceEnabled": true
}
```

When maintenance mode is enabled, you can safely migrate a running OpenIDM instance, without the risk of data corruption.

When the migration is complete, disable maintenance mode as follows:

```
$ curl \
 --cacert self-signed.crt \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request POST \
 "https://localhost:8443/openidm/maintenance?_action=disable"
{
  "maintenanceEnabled": false
}
```

To check whether OpenIDM is in maintenance mode, run the following command:

```
$ curl \
 --cacert self-signed.crt \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request POST \
 "https://localhost:8443/openidm/maintenance?_action=status"
{
  "maintenanceEnabled": false
}
```

If the system is in maintenance mode, the command returns `"maintenanceEnabled": true`, otherwise it returns `"maintenanceEnabled": false`.

Appendix A. Installing OpenIDM on a Read-Only Volume

Some enterprises choose to enhance security of their applications by installing them on a dedicated read-only (ro) filesystem volume. In this appendix, we describe how you can set up OpenIDM on such a volume.

This appendix assumes that you have prepared the read-only volume appropriate for your Linux/UNIX installation environment.

A.1. Preparing Your System

Before you continue, read "*Installing OpenIDM Services*", as well as the prerequisites described in "*Before You Install OpenIDM Software*" in the *Release Notes*.

This appendix assumes that you have set up a regular Linux user named `idm` and a dedicated volume for the `/idm` directory.

Configure the dedicated volume device, `/dev/volume` in the `/etc/fstab` file, as follows:

```
/dev/volume   /idm   ext4   ro,defaults   1,2
```

When you run the **mount -a** command, the `/dev/volume` volume device gets mounted on the `/idm` directory.

You can switch between read-write and read-only mode for the `/idm` volume with the following commands:

```
$ sudo mount -o remount,rw /idm
$ sudo mount -o remount,ro /idm
```

You can confirm the result with the **mount** command, which should show whether the `/idm` volume is mounted in read-only or read-write mode:

```
/dev/volume on /idm type ext4 (ro)
```

Set up the `/idm` volume in read-write mode:

```
$ sudo mount -o remount,rw /idm
```

With the following commands, you can unpack the OpenIDM binary in the `/idm` directory, and give user `idm` ownership of all files in that directory:

```
$ sudo unzip /idm/openidm-4.0.0.zip
$ sudo chown -R idm.idm /idm
```

A.2. Redirect Output Through Configuration Files

In this section, you will modify appropriate configuration files for OpenIDM to redirect data to writable volumes. This procedure assumes that you have a user `idm` with Linux administrative (superuser) privileges.

1. Create an external directory where OpenIDM can send logging, auditing, and internal repository information.

   ```
   $ sudo mkdir -p /var/log/openidm/audit
   $ sudo mkdir /var/log/openidm/logs
   $ sudo mkdir -p /var/cache/openidm/felix-cache
   $ sudo mkdir /var/run/openidm
   ```

   Note

   OpenIDM can help you route audit data to a remote data store. For an example of how OpenIDM sends data to a MySQL repository, review "*Audit Sample - Demonstrating the OpenIDM Audit Capabilities*" in the *Samples Guide*.

2. Give user `idm` ownership of the newly created directories:

   ```
   $ sudo chown -R idm.idm /var/log/openidm
   $ sudo chown -R idm.idm /var/cache/openidm
   $ sudo chown -R idm.idm /var/run/openidm
   ```

   Note

   If you use the unsupported OrientDB repository, you should also set up a writable directory to substitute for *`project-dir`*`/db/openidm`.

3. Open the audit configuration file for your project, *`project-dir`*`/conf/audit.json`.

Make sure `handlerForQueries` is set to `repo`.

Redirect the `logDirectory` property to the newly created `/var/log/openidm/audit` subdirectory:

```
{
   "auditServiceConfig" : {
      "handlerForQueries" : "repo",
      "availableAuditEventHandlers" : [
         "org.forgerock.audit.events.handlers.csv.CSVAuditEventHandler",
         "org.forgerock.openidm.audit.impl.RepositoryAuditEventHandler",
         "org.forgerock.openidm.audit.impl.RouterAuditEventHandler"
      ]
   },
   "eventHandlers" : [
      {
         "name" : "csv",
         "class" : "org.forgerock.audit.events.handlers.csv.CSVAuditEventHandler",
         "config" : {
            "logDirectory" : "/var/log/openidm/audit"
      },
      "events" : [ "access", "activity", "recon", "sync", "authentication", "config" ]
   },
```

4. Open the logging configuration file for your project: *project-dir*`/conf/logging.properties`.

   Find the `java.util.logging.FileHandler.pattern` property and redirect it as shown:

   ```
   java.util.logging.FileHandler.pattern = /var/log/openidm/logs/openidm%u.log
   ```

5. Open the configuration properties file for your project: *project-dir*`/conf/config.properties`.

   Activate the `org.osgi.framework.storage` property. Activate and redirect the `felix.cache.rootdir` property and change them as shown:

   ```
   # If this value is not absolute, then the felix.cache.rootdir controls
   # how the absolute location is calculated. (See buildNext property)
   org.osgi.framework.storage=${felix.cache.rootdir}/felix-cache

   # The following property is used to convert a relative bundle cache
   # location into an absolute one by specifying the root to prepend to
   # the relative cache path. The default for this property is the
   # current working directory.
   felix.cache.rootdir=/var/cache/openidm
   ```

Note

You may want to set up additional redirection. Watch for the following configuration details:

- Connectors. Depending on the connector, and the read-only volume, you may need to configure connectors to direct output to writable volumes.

- Scripts. If you're using Groovy, examine the `conf/script.json` file for your project. Make sure that output such as to the `groovy.target.directory` is directed to an appropriate location, such as `launcher.working.location`

A.3. Additional Details

Do configure a supported repository for OpenIDM, as described in: "*Installing a Repository For Production*".

Disable monitoring of JSON configuration files. To do so, open the *`project-dir`*`/conf/system.properties` file, and activate the following option:

```
openidm.fileinstall.enabled=false
```

You should address one more detail, the value of the `OPENIDM_PID_FILE` in the `startup.sh` and `shutdown.sh` scripts.

For RHEL 6 and Ubuntu 14.04 systems, the default shell is bash. You can set the value of `OPENIDM_PID_FILE` for user `idm` by adding the following line to `/home/idm/.bashrc`:

```
export OPENIDM_PID_FILE=/var/run/openidm/openidm.pid
```

If you have set up a different command line shell, adjust your changes accordingly.

You can now log in again as user `idm`. When you do, your `OPENIDM_PID_FILE` variable should now redirect the OpenIDM process identifier file, `openidm.pid` to the `/var/run/openidm` directory, ready for access by the `shutdown.sh` script.

You need to set up security keystore and truststore files, either by importing a signed certificate or by generating a self-signed certificate. For more information, see "*Securing & Hardening OpenIDM*" in the *Integrator's Guide*.

While the volume is still mounted in read-write mode, start OpenIDM normally:

```
$ ./startup.sh -p project-dir
```

The first startup of OpenIDM either processes the signed certificate that you added, or generates a self-signed certificate.

Stop OpenIDM:

```
-> shutdown
```

You can now mount the `/idm` directory in read-only mode. The configuration in `/etc/fstab` ensures that Linux mounts the `/idm` directory in read-only mode the next time that system is booted.

```
$ sudo mount -o remount,ro /idm
```

You can now start OpenIDM, configured on a secure read-only volume.

```
$ ./startup.sh -p project-dir
```

OpenIDM Glossary

assignment

The definition of an *assignment*, in the context of roles, depends on the following:

- Does the assignment imply what account a user should have?

  For example, if you want to assign a user to (or remove a user from) a role of Airplane Mechanics, the details within that role define what OpenIDM executes on changing the account.

- What happens when a user gets (loses) an assignment?

  OpenIDM may execute a custom `onAssignment` or an `onUnassignment` script.

- What is the assignment operation?

  When an administrator assigns a role to (on unassigns a role from) a user, OpenIDM processes that change with one of the following operations:

  ```
  "assignmentOperation" : "mergeWithTarget"
  ```

  ```
  "unassignmentOperation" : "removeFromTarget"
  ```

Some assignments may include an `entitlement`, where the associated role includes access rights to specified resources.

correlation query

A correlation query specifies an expression that matches existing entries in a source repository to one or more entries on a target

	repository. While a correlation query may be built with a script, it is *not* a correlation script. As noted in "Correlating Existing Target Objects" in the *Integrator's Guide*, you can set up a query definition, such as `_queryId`, `_queryFilter`, or `_queryExpression`, possibly with the help of a `linkQualifier`.
correlation script	A correlation script matches existing entries in a source repository, and returns the IDs of one or more matching entries on a target repository. While it skips the intermediate step associated with a `correlation query`, a correlation script can be relatively complex, based on the operations of the script.
entitlement	An entitlement is a collection of attributes that can be added to a user entry via roles. As such, it is a specialized type of `assignment`. A user or device with an entitlement gets access rights to specified resources. An entitlement is a property of a managed object.
JSON	JavaScript Object Notation, a lightweight data interchange format based on a subset of JavaScript syntax. For more information, see the JSON site.
JWT	JSON Web Token. As noted in the *JSON Web Token draft IETF Memo*, "JSON Web Token (JWT) is a compact URL-safe means of representing claims to be transferred between two parties." For OpenIDM, the JWT is associated with the `JWT_SESSION` authentication module.
managed object	An object that represents the identity-related data managed by OpenIDM. Managed objects are configurable, JSON-based data structures that OpenIDM stores in its pluggable repository. The default configuration of a managed object is that of a user, but you can define any kind of managed object, for example, groups or roles.
mapping	A policy that is defined between a source object and a target object during reconciliation or synchronization. A mapping can also define a trigger for validation, customization, filtering, and transformation of source and target objects.
OSGi	A module system and service platform for the Java programming language that implements a complete and dynamic component model. For a good introduction, see the OSGi site. While OpenIDM services are designed to run in any OSGi container, currently only Apache Felix is supported.
reconciliation	During reconciliation, comparisons are made between managed objects and objects on source or target systems. Reconciliation can result in one or more specified actions, including, but not limited to, synchronization.

resource	An external system, database, directory server, or other source of identity data to be managed and audited by the identity management system.
REST	Representational State Transfer. A software architecture style for exposing resources, using the technologies and protocols of the World Wide Web. REST describes how distributed data objects, or resources, can be defined and addressed.
role	OpenIDM includes two different types of provisioning roles and authorization roles. For more information, see "Working With Managed Roles" in the *Integrator's Guide*.
source object	In the context of reconciliation, a source object is a data object on the source system, that OpenIDM scans before attempting to find a corresponding object on the target system. Depending on the defined mapping, OpenIDM then adjusts the object on the target system (target object).
synchronization	The synchronization process creates, updates, or deletes objects on a target system, based on the defined mappings from the source system. Synchronization can be scheduled or on demand.
system object	A pluggable representation of an object on an external system. For example, a user entry that is stored in an external LDAP directory is represented as a system object in OpenIDM for the period during which OpenIDM requires access to that entry.System objects follow the same RESTful resource-based design principles as managed objects.
target object	In the context of reconciliation, a target object is a data object on the target system, that OpenIDM scans after locating its corresponding object on the source system. Depending on the defined mapping, OpenIDM then adjusts the target object to match the corresponding source object.

Index