Installation Guide

/ OpenIDM 5

Latest update: 5.0.1.1

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, updating, and uninstalling OpenIDM software. This software offers flexible services for automating management of the identity life cycle.

Table of Contents

Preface

ForgeRock Identity Platform™ serves as the basis for our simple and comprehensive Identity and Access Management solution. We help our customers deepen their relationships with their customers, and improve the productivity and connectivity of their employees and partners. For more information about ForgeRock and about the platform, see https://www.forgerock.com.

1. About This Guide

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

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

It 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 a complete understanding of OpenIDM software 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 software installed, see "*Compatibility*" in the *Release Notes* before you install 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
Preparing to Install and Run Servers

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

ForgeRock documentation includes a separate Samples Guide. When you have read the first two chapters of this document, use the Samples Guide to test a number of different deployment scenarios.

1.1. Before You Install

This section covers what you need to know before you install 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 runs in an OSGi container with an embedded Servlet container and an embedded noSQL database. By default the OSGi container is Apache Felix (Felix) and the default Servlet container is Jetty. No other configuration is supported.

1.2. Installing and Running Servers

Follow the procedures in this section to install and run OpenIDM. To set up the server on a read-only volume, read "*Installing on a Read-Only Volume*".

To Install OpenIDM 5

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*" in the *Release Notes*.

2. Download OpenIDM from ForgeRock's BackStage site.

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

   ```
   $ cd /path/to
   $ unzip ~/Downloads/IDM-5.0.0.zip
   Archive:  IDM-5.0.0.zip
     inflating: openidm/.checksums.csv
      creating: openidm/bundle/
    extracting: openidm/bundle/openidm-audit-5.0.0
   .jar
   ...
   ```

4. 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 "*Host and Port Information*" in the *Integrator's Guide*.

   The hostname associated with OpenIDM by default is `localhost`. To change the default hostname, edit the `openidm.host` property in your `conf/boot/boot.properties` file.

5. Before running OpenIDM in production, replace the default OrientDB repository, provided or evaluation, with a supported JDBC repository.

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

To Start OpenIDM

To run OpenIDM as a background process, see "*Starting and Stopping the Server*" 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 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 core services and their respective states.

To Stop OpenIDM

You can stop OpenIDM from the `->` prompt in the OSGi console, or through the Apache Felix Web Console. Both of these options stop the Felix container.

1. In the OSGi console, enter the **shutdown** command at the `->` prompt:

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

2. In the Apache Felix Web Console, select Web Console > System Information to stop the container.

3. 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)
   ```

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 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 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 IDM-5.0.0.zip file, as described previously, and change to the `install-location\bin` directory:

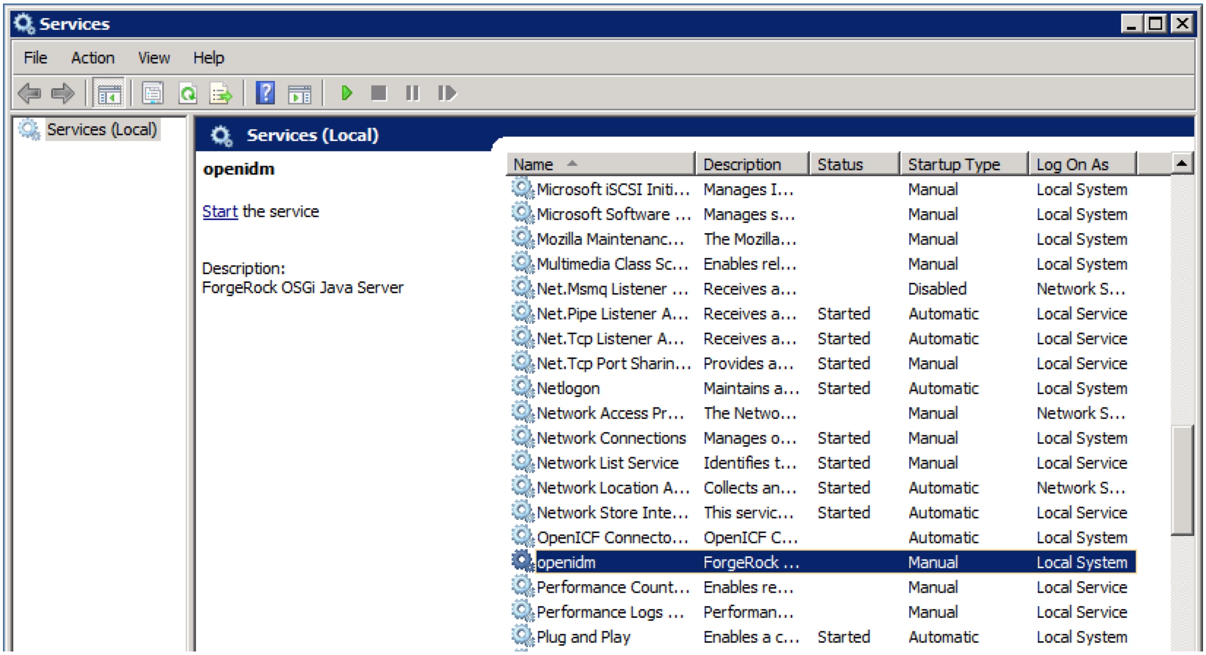
   ```
   C:\>cd openidm\bin
   C:\openidm\bin>
   ```

2. Run the `install-service.bat` command, specifying the name that 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



openidm

Start the service

Description:
ForgeRock OSGi Java Server

Name	Description	Status	Startup Type	Log On As
Microsoft iSCSI Initi...	Manages I...		Manual	Local System
Microsoft Software ...	Manages s...		Manual	Local System
Mozilla Maintenanc...	The Mozilla...		Manual	Local System
Multimedia Class Sc...	Enables rel...		Manual	Local System
Net.Msmq Listener ...	Receives a...		Disabled	Network S...
Net.Pipe Listener A...	Receives a...	Started	Automatic	Local Service
Net.Tcp Listener A...	Receives a...	Started	Automatic	Local Service
Net.Tcp Port Sharin...	Provides a...	Started	Manual	Local Service
Netlogon	Maintains a...	Started	Automatic	Local System
Network Access Pr...	The Netwo...		Manual	Network S...
Network Connections	Manages o...	Started	Manual	Local System
Network List Service	Identifies t...	Started	Manual	Local Service
Network Location A...	Collects an...	Started	Automatic	Network S...
Network Store Inte...	This servic...	Started	Automatic	Local Service
OpenICF Connecto...	OpenICF C...		Automatic	Local System
openidm	ForgeRock ...		Manual	Local System
Performance Count...	Enables re...		Manual	Local Service
Performance Logs ...	Performan...		Manual	Local Service
Plug and Play	Enables a c...	Started	Automatic	Local System

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:

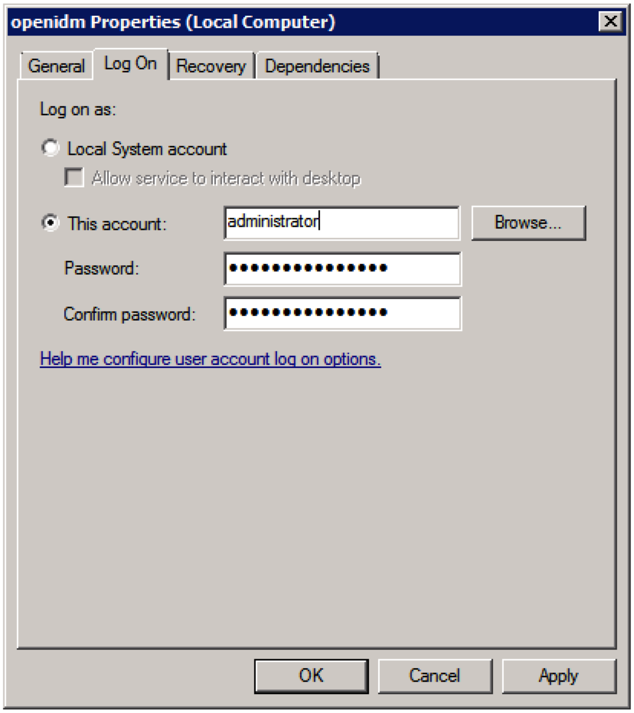
   a. Double click the `openidm` service in the Windows Service manager.

   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



openidm Properties (Local Computer)

Log on as:

○ Local System account

□ Allow service to interact with desktop

● This account: administrator

Password: •••••••••••••••

Confirm password: •••••••••••••••

Help me configure user account log on options.

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 REST Interface

OpenIDM provides RESTful access to users in its repository. To access the repository over REST, you can use a client application like Postman, 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**
>
> Some of the examples in this documentation set use client-assigned IDs when creating resources, as it makes the examples easier to read. In general, immutable server-assigned UUIDs should be used in production.

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 `onCreateUser.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

By default, **curl**-based REST calls return the JSON object 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 can "pipe" the output of a REST call to **jq**, as follows:

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/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 \
 --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

You can manage OpenIDM using Web-based user interfaces, called the UI in this 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 User Access to Workflows" 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 server configuration. Administrators would access the Admin UI to learn about OpenIDM during initial system setup, and when they identify new requirements.

The Admin UI also enables you to configure connections to external data stores, and to specify the reconciliation and synchronization configuration between 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 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. 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 server 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 repository

select * from audit_activity

Shows all activity audit records

This table is created on install and populated when there is any activity on the server.

select * from audit_recon

Shows all reconciliation audit records

This table is created on install and populated when you run a reconciliation operation.

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, the following internal repositories are supported:

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

For details about the supported versions, see "*Before You Install*" 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 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. This section lists the minimum permissions required, 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 SELECT, UPDATE, INSERT, and DELETE permissions on all the openidm tables that are created by this script, and by the scripts that create the tables specific to the Activiti workflow engine.

Important

The scripts that create the Activiti tables are new in OpenIDM 5.0.0. Prior to OpenIDM 5.0.0, these tables were created the first time OpenIDM was started.

2.2. Setting Up a MySQL Repository

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:

```
$ 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
$ 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:

```
$ 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;

mysql> show tables;
+---------------------------+
| Tables_in_openidm         |
+---------------------------+
| auditaccess               |
| auditactivity             |
| auditauthentication       |
| ...                       |
| uinotification            |
| updateobjectproperties    |
| updateobjects             |
+---------------------------+
27 rows in set (0.00 sec)
```

The table names are similar to those used with OrientDB.

Exit the mysql console.

```
mysql> exit
Bye
```

6. Run the three scripts that set up the tables required by the Activiti workflow engine.

   If you are running MySQL 5.6.4 or higher, run the following scripts:

```
$ cd /path/to/mysql
$ mysql -D openidm -u root -p < /path/to/openidm/db/mysql/scripts/activiti.mysql.create.engine.sql
Enter password:
$ mysql -D openidm -u root -p < /path/to/openidm/db/mysql/scripts/activiti.mysql.create.history.sql
Enter password:
$ mysql -D openidm -u root -p < /path/to/openidm/db/mysql/scripts/activiti.mysql.create.identity.sql
Enter password:
```

   If you are running a MySQL version prior to 5.6.4, run the following scripts:

```
$ cd /path/to/mysql
$ mysql -D openidm -u root -p < /path/to/openidm/db/mysql/scripts/activiti.mysql55.create.engine.sql
Enter password:
$ mysql -D openidm -u root -p < /path/to/openidm/db/mysql/scripts/activiti.mysql55.create.history.sql
Enter password:
$ mysql -D openidm -u root -p < /path/to/openidm/db/mysql/scripts/activiti.mysql.create.identity.sql
Enter password:
```

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

   ```
   {
       "driverClass" : "com.mysql.jdbc.Driver",
       "jdbcUrl" : "jdbc:mysql://&{openidm.repo.host}:&{openidm.repo.port}/openidm?
   allowMultiQueries=true&characterEncoding=utf8",
       "databaseName" : "openidm",
       "username" : "openidm",
       "password" : "openidm",
       "connectionTimeout" : 30000,
       "connectionPool" : {
           "type" : "hikari",
           "minimumIdle" : 20,
           "maximumPoolSize" : 50
       }
   }
   ```

   Specify the values for `openidm.repo.host` and `openidm.repo.port` in one of the following ways:

   - Set the values in your project's `conf/system.properties` or `conf/boot/boot.properties` file, for example:

     ```
     openidm.repo.host = localhost
     openidm.repo.port = 3306
     ```

   - Set the properties in the `OPENIDM_OPTS` environment variable and export that variable before startup. You must include the JVM memory options when you set this variable. For example:

     ```
     $ export OPENIDM_OPTS="-Xmx1024m -Xms1024m -Dopenidm.repo.host=localhost -Dopenidm.repo.port=3306"
     $ ./startup.sh
     Executing ./startup.sh...
     Using OPENIDM_HOME:   /path/to/openidm
     Using PROJECT_HOME:   /path/to/openidm
     Using OPENIDM_OPTS:   -Xmx1024m -Xms1024m -Dopenidm.repo.host=localhost -Dopenidm.repo.port=3306
     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 version "5.0.0"
     OpenIDM ready
     ```

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
...
[  18] [unsatisfied  ] org.forgerock.openidm.repo
.orientdb
...
[  17] [active       ] org.forgerock.openidm.repo
.jdbc
...
```

2.3. Setting Up an MS SQL Repository

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.

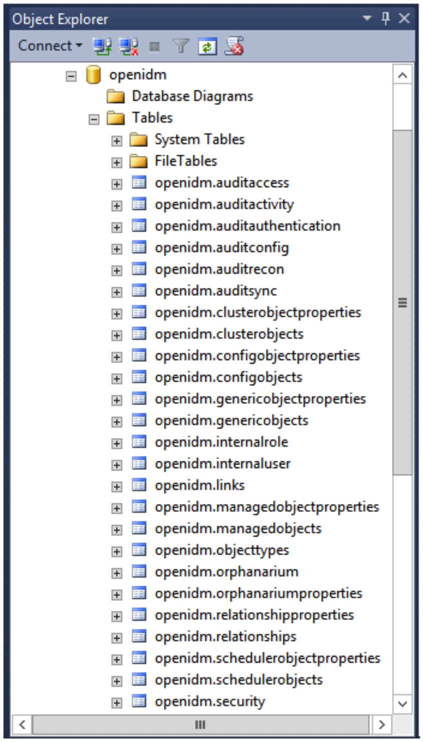
   b. On the Connect to Server panel, select Windows Authentication and click Connect.

   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 `openidm` 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.



openidm
Database Diagrams
Tables
System Tables
FileTables
openidm.auditaccess
openidm.auditactivity
openidm.auditauthentication
openidm.auditconfig
openidm.auditrecon
openidm.auditsync
openidm.clusterobjectproperties
openidm.clusterobjects
openidm.configobjectproperties
openidm.configobjects
openidm.genericobjectproperties
openidm.genericobjects
openidm.internalrole
openidm.internaluser
openidm.links
openidm.managedobjectproperties
openidm.managedobjects
openidm.objecttypes
openidm.orphanarium
openidm.orphanariumproperties
openidm.relationshipproperties
openidm.relationships
openidm.schedulerobjectproperties
openidm.schedulerobjects
openidm.security

The table names are similar to those used with OrientDB.

3. Execute the three scripts that set up the tables required by the Activiti workflow engine:

   You can use the sqlcmd command to execute the scripts, for example:

```
PS C:\Users\Administrator> sqlcmd -S localhost -d openidm ^
    -i C:\path\to\openidm\db\mssql\scripts\activiti.mssql.create.engine.sql
PS C:\Users\Administrator> sqlcmd -S localhost -d openidm ^
    -i C:\path\to\openidm\db\mssql\scripts\activiti.mssql.create.history.sql
PS C:\Users\Administrator> sqlcmd -S localhost -d openidm ^
    -i C:\path\to\openidm\db\mssql\scripts\activiti.mssql.create.identity.sql
```

Note

When you run the activiti.mssql.create.engine.sql script, you might see the following warning in the OpenIDM log:

```
Warning! The maximum key length is 900 bytes. The index 'ACT_UNIQ_PROCDEF' has maximum
length of 1024 bytes. For some combination of large values, the insert/update operation will fail.
```

It is very unlikely that the key length will be an issue in your deployment, and you can safely ignore this warning.

4. 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
      ```

5. 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
   ```

6. 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\
   ```

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

```
{
    "driverClass" : "com.microsoft.sqlserver.jdbc.SQLServerDriver",
    "jdbcUrl" : "jdbc:sqlserver://
&{openidm.repo.host}:&{openidm.repo.port};instanceName=default;databaseName=openidm;applicationName=OpenIDM",
    "databaseName" : "openidm",
    "username" : "openidm",
    "password" : "openidm",
    "connectionTimeout" : 30000,
    "connectionPool" : {
        "type" : "hikari",
        "minimumIdle" : 20,
        "maximumPoolSize" : 50
    }
}
```

   Specify the values for `openidm.repo.host` and `openidm.repo.port` in one of the following ways:

   - Set the values in your project's `conf/system.properties` or `conf/boot/boot.properties` file, for example:

     ```
     openidm.repo.host = localhost
     openidm.repo.port = 1433
     ```

   - Set the properties in the `OPENIDM_OPTS` environment variable and export that variable before startup. You must include the JVM memory options when you set this variable. For example:

     ```
     $ export OPENIDM_OPTS="-Xmx1024m -Xms1024m -Dopenidm.repo.host=localhost -Dopenidm.repo.port=1433"
     $ ./startup.sh
     Executing ./startup.sh...
     Using OPENIDM_HOME:   /path/to/openidm
     Using PROJECT_HOME:   /path/to/openidm
     Using OPENIDM_OPTS:   -Xmx1024m -Xms1024m -Dopenidm.repo.host=localhost -Dopenidm.repo.port=1433
     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 version "5.0.0"
     OpenIDM ready
     ```

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. Setting Up an Oracle Database Repository

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.

1. 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" } ]
-----------------------------------------------------------------------------
```

2. Run the three scripts that set up the tables required by the Activiti workflow engine.

   You can use the Oracle SQL Developer Data Modeler to run the scripts, as described in the corresponding Oracle documentation.

   You must run the following scripts:

   ```
   /path/to/openidm/db/oracle/scripts/activiti.oracle.create.engine.sql
   /path/to/openidm/db/oracle/scripts/activiti.oracle.create.history.sql
   /path/to/openidm/db/oracle/scripts/activiti.oracle.create.identity.sql
   ```

3. Create an Oracle DB driver from two separate jar files and set up the OpenIDM repository configuration for Oracle DB, as follows:

   a. 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
      ```

   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 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}
```

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/oracle/scripts
bnd-1.50.0.jar    ojdbc7_g.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/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
```

e. Move the `.bar` file to the `openidm/bundle` directory and rename it with a `.jar` extension. It does not matter what you name the file:

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

4. 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
```

5. 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/
   ```

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

```
{
    "driverClass" : "oracle.jdbc.OracleDriver",
    "jdbcUrl" : "jdbc:oracle:thin:@//&{openidm.repo.host}:&{openidm.repo.port}/DEFAULTCATALOG",
    "databaseName" : "openidm",
    "username" : "openidm",
    "password" : "openidm",
    "connectionTimeout" : 30000,
    "connectionPool" : {
        "type" : "hikari",
        "minimumIdle" : 20,
        "maximumPoolSize" : 50
    }
}
```

   Specify the values for `openidm.repo.host` and `openidm.repo.port` in one of the following ways:

   - Set the values in your project's `conf/system.properties` or `conf/boot/boot.properties` file, for example:

     ```
     openidm.repo.host = localhost
     openidm.repo.port = 1525
     ```

   - Set the properties in the `OPENIDM_OPTS` environment variable and export that variable before startup. You must include the JVM memory options when you set this variable. For example:

     ```
     $ export OPENIDM_OPTS="-Xmx1024m -Xms1024m -Dopenidm.repo.host=localhost -Dopenidm.repo.port=1525"
     $ ./startup.sh
     Executing ./startup.sh...
     Using OPENIDM_HOME:   /path/to/openidm
     Using PROJECT_HOME:   /path/to/openidm
     Using OPENIDM_OPTS:   -Xmx1024m -Xms1024m -Dopenidm.repo.host=localhost -Dopenidm.repo.port=1525
     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 version "5.0.0"
     OpenIDM ready
     ```

   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 correspond 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. Setting Up a PostgreSQL Repository

This procedure assumes that PostgreSQL is installed and running on the local host. For supported versions, see "Supported Repositories" in the *Release Notes*.

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. Edit the Postgres client authentication configuration file, `pg_hba.conf`. Add the following entries for the following users: `postgres` and `openidm`:

   ```
   local    all    openidm      trust
   local    all    postgres      trust
   ```

3. 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
   ```

4. 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.

5. Run the three scripts that set up the tables required by the Activiti workflow engine:

   ```
   $ psql -d openidm -U openidm < /path/to/openidm/db/postgresql/scripts/activiti.postgres.create.engine
   .sql
   $ psql -d openidm -U openidm < /path/to/openidm/db/postgresql/scripts/activiti.postgres.create.history
   .sql
   $ psql -d openidm -U openidm < /path/to/openidm/db/postgresql/scripts/activiti.postgres.create
   .identity.sql
   ```

6. 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
   ```

7. 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/
   ```

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

```
{
    "driverClass" : "org.postgresql.Driver",
    "jdbcUrl" : "jdbc:postgresql://&{openidm.repo.host}:&{openidm.repo.port}/openidm",
    "databaseName" : "openidm",
    "username" : "openidm",
    "password" : "openidm",
    "connectionTimeout" : 30000,
    "connectionPool" : {
        "type" : "hikari",
        "minimumIdle" : 20,
        "maximumPoolSize" : 50
    }
}
```

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.

Specify the values for `openidm.repo.host` and `openidm.repo.port` in one of the following ways:

- Set the values in your project's `conf/boot/boot.properties` file:

  ```
  openidm.repo.host = localhost
  openidm.repo.port = 5432
  ```

- Set the properties in the `OPENIDM_OPTS` environment variable and export that variable before startup. You must include the JVM memory options when you set this variable. For example:

  ```
  $ export OPENIDM_OPTS="-Xmx1024m -Xms1024m -Dopenidm.repo.host=localhost -Dopenidm.repo.port=5432"
  $ ./startup.sh
  Executing ./startup.sh...
  Using OPENIDM_HOME:   /path/to/openidm
  Using PROJECT_HOME:   /path/to/openidm
  Using OPENIDM_OPTS:   -Xmx1024m -Xms1024m -Dopenidm.repo.host=localhost -Dopenidm.repo.port=5432
  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 version "5.0.0"
  OpenIDM ready
  ```

9. 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
..
.
->
```

10. 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. Setting Up an IBM DB2 Repository

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

```
{
    "driverClass" : "com.ibm.db2.jcc.DB2Driver",
    "jdbcUrl" : "jdbc:db2://&{openidm.repo.host}:&{openidm.repo.port}/
dopenidm:retrieveMessagesFromServerOnGetMessage=true;",
    "databaseName" : "sopenidm",
    "username" : "openidm",
    "password" : "openidm",
    "connectionTimeout" : 30000,
    "connectionPool" : {
        "type" : "hikari",
        "minimumIdle" : 20,
        "maximumPoolSize" : 50
    }
}
```

Specify the values for `openidm.repo.host` and `openidm.repo.port` in one of the following ways:

- Set the values in your project's `conf/system.properties` or `conf/boot/boot.properties` file, for example:

```
openidm.repo.host = localhost
openidm.repo.port = 50000
```

- Set the properties in the `OPENIDM_OPTS` environment variable and export that variable before startup. You must include the JVM memory options when you set this variable. For example:

```
$ export OPENIDM_OPTS="-Xmx1024m -Xms1024m -Dopenidm.repo.host=localhost -Dopenidm.repo.port=50000"
$ ./startup.sh
Executing ./startup.sh...
Using OPENIDM_HOME:   /path/to/openidm
Using PROJECT_HOME:   /path/to/openidm
Using OPENIDM_OPTS:   -Xmx1024m -Xms1024m -Dopenidm.repo.host=localhost -Dopenidm.repo.port=50000
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 version "5.0.0"
OpenIDM ready
```

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.

8. Connect to the openidm database, then run the three scripts that set up the tables required by the Activiti workflow engine:

```
$ db2 connect to dopenidm
$ db2 -i -tf /path/to/openidm/db/db2/scripts/activiti.db2.create.engine.sql
$ db2 -i -tf /path/to/openidm/db/db2/scripts/activiti.db2.create.history.sql
$ db2 -i -tf /path/to/openidm/db/db2/scripts/activiti.db2.create.identity.sql
```

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
[   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

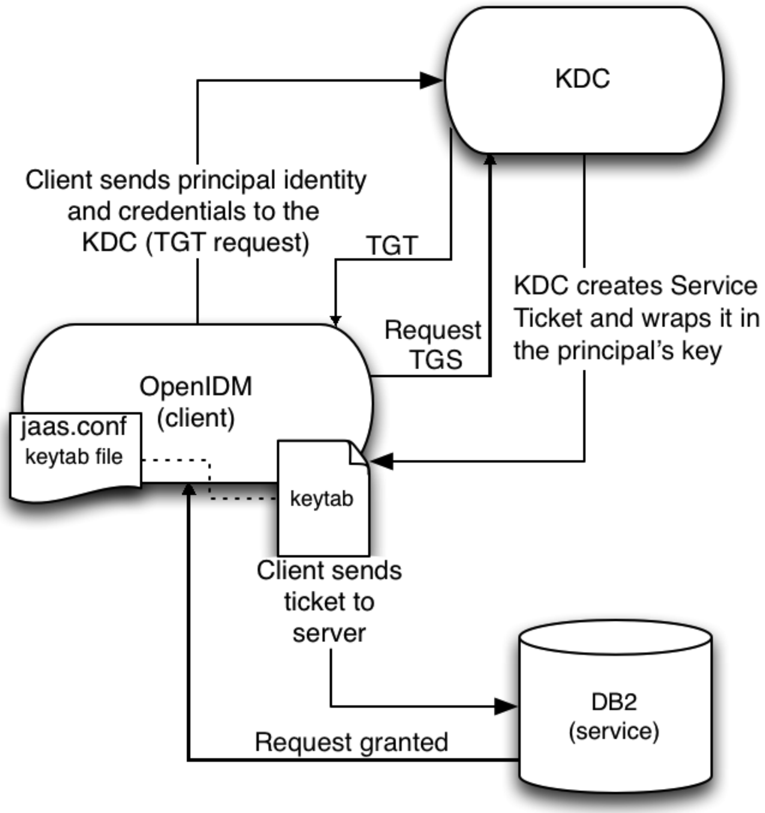
By default, OpenIDM uses the username and password configured in the repository connection configuration file (`conf/datasource.jdbc-default.json`) to connect to the DB2 repository. You can configure OpenIDM to use Kerberos authentication instead.

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


KDC
Client sends principal identity and credentials to the KDC (TGT request)
TGT
Request TGS
KDC creates Service Ticket and wraps it in the principal's key
OpenIDM (client)
jaas.conf
keytab file
keytab
Client sends ticket to server
DB2 (service)
Request granted

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 Server Software

This chapter covers uninstallation and describes how to move an existing installation to a different location.

To Remove OpenIDM

1. (Optional) Stop the service if it is running, as described in "To Stop OpenIDM".
2. Remove the file system directory where you installed the 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

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

1. Shut down the server, as described in "To Stop OpenIDM".
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 the server in the new location:

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

Chapter 4 Updating Servers

This chapter describes the process to update an existing deployment to OpenIDM 5.

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

If you are updating from OpenIDM 4 and above on UNIX/Linux systems, you can take advantage of new OpenIDM update options from the command-line and the Admin UI. For a generic view of the update process, see "An Overview of the Update Process".

4.1. Limitations of the Automated Update Process

You cannot use the automated update process in all OpenIDM deployments. This section lists the limitations:

Phased Update From Previous Versions

You cannot update directly from OpenIDM 4.0 to OpenIDM 5. To update from OpenIDM 4.0, first update to OpenIDM 4.5, as described in *Updating From OpenIDM 4.0 to OpenIDM 4.5* in the *OpenIDM 4.5 Installation Guide*. Then update from OpenIDM 4.5 to OpenIDM 5, as described in "Updating to OpenIDM 5".

If you are updating from a version prior to OpenIDM 4.0, follow the manual update process first:

- To migrate a deployment from OpenIDM 3.1.0 to OpenIDM 4.0, see *Migrating From OpenIDM 3.1 to OpenIDM 4.0* in the *OpenIDM 4 Installation Guide*.
- To migrate 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*.
- To migrate 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*.

Update on Windows Systems

Automated update is not supported on Microsoft Windows systems. If you are updating to OpenIDM 4.5 or 5 on Microsoft Windows, follow the manual update process:

- To migrate a Windows deployment from OpenIDM 4.0 to 4.5, see *Migrating From OpenIDM 4.0 to OpenIDM 4.5 on Windows* in the *OpenIDM 4.5 Installation Guide*.

- To migrate a Windows deployment from OpenIDM 4.5 to OpenIDM 5, see "Migrating to OpenIDM 5 on Windows".

Updating a Clustered Deployment

Updating nodes in a cluster is not presently supported. As a general practice, do not apply the update process to more than one node in a cluster. If you're updating a cluster, follow these steps:

- Redirect client traffic to a different OpenIDM system or cluster.
- Shut down every node in the cluster.
- Update one node.
- Clone the first node to the other OpenIDM instances in that cluster.

Supported Project Directories

The automated update process works for an OpenIDM project directory in the following locations:

- The default OpenIDM project directory, `/path/to/openidm`
- Outside of the OpenIDM directory tree, such as `/home/project` or `/other/hard_drive/idm`

The update process does not support changes to any project directory when configured as a subdirectory of `/path/to/openidm`. That includes the samples listed in the `/path/to/openidm/samples` directory. For more information on the samples, see "*Overview of the Samples*" in the *Samples Guide*.

OpenIDM documentation represents the project directory as *`project-dir`*.

It is an OpenIDM best practice to copy the default project or sample to an external installation *`project-dir`* directory, such as `/path/to/project`. If needed, this is an opportunity to move the *`project-dir`* to such a location, to facilitate the OpenIDM update process.

Apart from the repository update scripts, the update process does not change information related to data and schema.

The update process also generally does not change scripts in your project directories. For a list of files affected during the update process, see "Files Changed During the Update Process".

Supported Repositories Only

Update is supported *only* with the supported JDBC repositories, and not with the default OrientDB repository. As such, repository update scripts are provided only for the supported JDBC repositories.

For instructions on installing a supported repository, see "*Installing a Repository For Production*".

Custom Configuration Files

If you customized configuration files for your deployment of OpenIDM 4.5, such as JSON files, or files related to a custom UI, you'll need to reapply what you customized after the update is complete. For more information, see "Managing Custom Changes After an Update".

4.2. An Overview of the Update Process

This section provides background information on the update process. If you want to jump to the steps required to update from OpenIDM 4.5 to OpenIDM 5, read "Updating to OpenIDM 5".

- "The Update Process in Labels"
- "Files Changed During the Update Process"
- "Command-Line Update Options"
- "Update File States"

4.2.1. The Update Process in Labels

When you start an update, OpenIDM goes through a process to determine the files to be updated, to facilitate repository updates, to move OpenIDM into maintenance mode before implementing an update, and then to restart OpenIDM after the update is complete. Most of these steps happen automatically when you run the update from the UI or CLI. The following basic steps include labels that you might see in log files, if there are problems.

- `PREVIEW_ARCHIVE`: Preview the archive using appropriate checksums, to define the files to be updated.
- `ACCEPT_LICENSE`: Present the appropriate license for the update. If the license is not accepted, the update stops, and you retain all files in your current OpenIDM installation.
- `LIST_REPO_UPDATES`: Identify and save scripts to update the current repository.
- `PAUSING_SCHEDULER`: Pause the scheduler, to prevent new jobs from starting during the update process.
- `WAIT_FOR_JOBS_TO_COMPLETE`: Wait for any currently running jobs to complete.
- `ENTER_MAINTENANCE_MODE`: Set OpenIDM in maintenance mode, which disables non-essential OpenIDM services.
- `INSTALL_ARCHIVE`: Install the update archive into OpenIDM.
- `WAIT_FOR_INSTALL_DONE`: Wait until the installation of the update archive is complete.
- `MARK_REPO_UPDATES_COMPLETE`: Note status when repository updates are complete.
- `EXIT_MAINTENANCE_MODE`: Exit from maintenance mode.

- ENABLE_SCHEDULER: Enable the scheduler. OpenIDM will re-establish any schedules that were previously in place.
- FORCE_RESTART Restart OpenIDM to implement all changes that require it.

4.2.2. Files Changed During the Update Process

OpenIDM 4 and above 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.

When you run the update process, changes are applied to the OpenIDM *project-dir*, either in the default directory, /path/to/openidm, or outside of the OpenIDM directory tree, such as /home/project. If that *project-dir* is outside of the OpenIDM directory tree, there is no MD5 checksum for that file. Based on that information, the update mechanism takes the following actions:

Files in openidm/bundle

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.

New files are written to the target directory.

Existing files are saved with a .old-*unix_time* extension.

Files in openidm/bin and openidm/ui/*/default

New files are written to the target directory.

If the original file was customized, it is saved with a .old-*unix_time* extension.

***project-dir*/script/access.js**

The update process writes access.js files with a .new-*unix_time* extension in the same directory. You *must* merge any changes from the latest version of the .new file into your customized script file. For more information, see "Managing Custom Changes After an Update".

Files in the *project-dir*/conf directory, outside of the /path/to/openidm directory tree

Project configuration files (JSON): Files related to your project are patched with the content of the corresponding version 5 configuration file, regardless of whether they have been customized.

> **Warning**
>
> If you have customized standard OpenIDM JSON configuration files, be careful. Analyze the OpenIDM 5 version of that file. It may include new features that you want. As a best practice, apply each customization

> to the corresponding version 5 file and test the change before you put it into production. For more information, see "Managing Custom Changes After an Update".

System configuration files: (`boot.properties`, `system.properties`, `config.properties`, `logging.properties`, and `jetty.xml`) are written with a `.-new-unix_time` extension in the same directory, regardless of whether they have been customized. You *must* merge any changes from the latest version of the `.new` file into your existing configuration file. For more information, see "Managing Custom Changes After an Update".

Files in the default `project-dir/conf` directory, where `project-dir` is `/path/to/openidm`

Project configuration files (JSON): Files related to your project are overwritten with the content of the corresponding version 5 configuration file, regardless of whether they have been customized.

> **Warning**
>
> If you have customized any standard OpenIDM JSON configuration files, be careful. Analyze the version 5 file. It may include new features that you want. As a best practice, apply and test each custom change to the version 5 configuration file. For more information, see "Managing Custom Changes After an Update".

System configuration files: (`boot.properties`, `system.properties`, `config.properties`, `logging.properties`, and `jetty.xml`) are not patched if they have been customized. Instead, the update process creates configuration files with a `.new-unix_time` extension in the same directory. You *must* merge any changes from these `.new-` files into your customized configuration files. For more information, see "Managing Custom Changes After an Update". If you have not customized these files, the update process replaces the existing configuration file with the corresponding version 5 file.

Files in any other directory

Existing files are overwritten and no backup files are created.

Configuration in the repository

OpenIDM configuration information is stored in your supported repository. The update process overwrites information in that data store.

> **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.

Warning

If you've changed defaults in the boot.properties file, be careful, as this can affect the OpenIDM boot process; refer to Update boot.properties for details.

4.2.2.1. 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
```

4.2.3. The update.json File

The update.json file, in the /path/to/openidm directory, specifies basic configuration parameters for the update process:

```
{
    "update" : {
        "description" : "Full product installation",
        "resource" : "https://backstage.forgerock.com/docs/idm/<current version number>/release-notes",
        "restartRequired" : true
    },
    "origin": {
        "product": "OpenIDM",
        "version": [
            "<current version number>"
        ]
    },
    "destination": {
        "product": "OpenIDM",
        "version": "<update version number>"
    },
    "removeFile" : [
        "<file to be removed>"
    ],
    "filesToBeIgnored" : [
        "security/keystore.jceks",
        "security/realm.properties",
        "security/truststore"
    ]
}
```

If you want to try your own update, make sure that the `origin` and `destination` version numbers match your current and planned update versions of OpenIDM.

The version numbers may be either specific, such as 5.0.0 or a range, such as `[5.0.0.1,5.0.0.3)`. That range supports version numbers `5.0.0.1` and `5.0.0.2` (but not `5.0.0.3`). For a full explanation of how you can specify a range of versions, see "Setting the Connector Reference Properties" in the *Integrator's Guide*.

If you want to delete files during the update process, include them in the `removeFile` code block.

Note

To preserve existing keystore and truststore files, `update.json` includes the default versions of these files in the `filesToBeIgnored` code block.

If you want to keep the update process from overwriting other files, add them to the same code block.

4.2.4. Command-Line Update Options

On UNIX/Linux systems, you can update OpenIDM 4.5 to OpenIDM 5 by using the **cli.sh update** command. For general information on **cli.sh** and **cli.bat**, see "*Command-Line Interface*" in the *Integrator's Guide*.

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

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

The `update` subcommand takes the following options:

`-u` or `--user` USER[:PASSWORD]

Allows you to specify the server user and password. Specifying a username is mandatory. If you do not specify a username, the following error is shown in the OSGi console: `Remote operation failed: Unauthorized`. If you do not specify a password, you are prompted for one. This option is used by all three subcommands.

`--url` URL

The URL of the REST service. The default URL is `http://localhost:8080/openidm/`. This can be used to import configuration files from a remote running instance. This option is used by all three subcommands.

`-P` or `--port` PORT

The port number associated with the REST service. If specified, this option overrides any port number specified with the **--url** option. The default port is 8080. This option is used by all three subcommands.

`--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.

`--skipRepoUpdatePreview`

Bypasses a preview of repository updates. Suitable if you have already downloaded and approved changes to your repository.

`--maxJobsFinishWaitTimeMs` TIME

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` TIME

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

Default: `30000` ms

`-l` or `--log` LOG_FILE

Path to the log file.

Default: `logs/update.log`

`-Q` or `--quiet`

Use quiet mode to minimize messages at the console; messages are still available in the log file defined by `--log`.

> **Note**
>
> If you use `--quiet` mode for updates, include the `--acceptLicense` option.

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
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 idm-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.
```

4.2.5. Update File States

During the update process, you may see status information for each file, during three stages of an update:

- "Preview of File Updates"
- "Update Status Message"
- "Updated Files: What Happened"

Preview of File Updates

Status	Description
UNEXPECTED	Existing file is not in the list of known files for the original distribution.

Status	Description
NONEXISTENT	A file in the new installation that does not exist in the original distribution. This is always the status for *versioned* files, such as the `openidm-*.jar` files in the `openidm/bundle/` directory.
DELETED	The file should exist in the current installation but does not; The file is installed during the update.
DIFFERS	The file, located in a read-only directory, has changed, since the original deployment.
UNCHANGED	The file is not changed from the original distribution.

Update Status Message

Status	Description
IN_PROGRESS	Update has started, not yet complete
PENDING_REPO_UPDATES	The update is complete; however, repository updates are pending
COMPLETE	Update is complete
FAILED	Update failed

Updated Files: What Happened

Status	Description
REPLACED	Original file replaced; if the original file was changed by a user, it is saved with a `.old` extension.
PRESERVED	Original file saved with changes made by a user. New file saved with a `.new` extension.
APPLIED	The update changed the file.
REMOVED	The update removed the file.

4.3. Updating to OpenIDM 5

In the following sections, you'll:

- "Preparing for Updates".
- "Updating to OpenIDM 5".
- "Managing Custom Changes After an Update".

4.3.1. Preparing for Updates

The following section lists the steps you *must* take before you start an automated update:

Download the Software

Download OpenIDM (`IDM-5.0.0.zip`) from ForgeRock's BackStage site.

Back up your deployment

Before starting the update process, back up your existing deployment by archiving the `openidm` directory, as well as the contents of your repository. As there is no "undo" option available during the update process, this backup can be used to restore your deployment or restart the update process if something goes wrong.

Save any customized `*.json` configuration files, typically located in your project's `conf/` subdirectory. You'll need to make judgements on how to apply these customizations to your OpenIDM 5 deployment after the update is complete.

Identify files in custom directories. Save them, and apply them to your updated deployment after all stages of the update process are complete. For more information, see "Managing Custom Changes After an Update".

Change read-only deployments

If your project directory is located on a read-only volume, mount that directory in read-write mode before starting the update process.

Disable authentication modules used with OpenAM

If you have integrated OpenIDM with OpenAM, first disable the authentication module you used (`OPENID_CONNECT`, `OAUTH`, or `OPENAM_SESSION`), as described in "Disabling and Re-enabling Connections to OpenAM". You can re-enable this module when the update is complete.

Update `boot.properties`

OpenIDM 5 is not compatible with the 4.5 version of the `boot.properties` file. If you start the update process with version 4.5 of this file, the update will fail on restart. The `boot.properties` file must include the options available with version 5 of the file. New defaults are shown in bold.

```
openidm.port.http=8080
openidm.port.https=8443
openidm.port.mutualauth=8444

openidm.auth.clientauthonlyports=8444

openidm.https.keystore.cert.alias=openidm-localhost

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

# Keystore password, adjust to match your keystore and protect this file
openidm.keystore.password=changeit
openidm.truststore.password=changeit
```

```
# Optionally use the crypto bundle to obfuscate the password and set one of these:
#openidm.keystore.password=OBF:
#openidm.keystore.password=CRYPT:

# PKCS#11 configuration file
# openidm.security.pkcs11.config=

# key in keystore to handle config encryption
openidm.config.crypto.alias=openidm-sym-default
#openidm.script.javascript.debug=transport=socket,suspend=y,address=9888,trace=true
#openidm.script.javascript.sources=/Eclipse/workspace/External JavaScript Source/

# key in keystore to handle selfservice sharedkey
openidm.config.crypto.selfservice.sharedkey.alias=openidm-selfservice-key

# key in keystore to handle jwtsession hmac signing key
openidm.config.crypto.jwtsession.hmackey.alias=openidm-jwtsessionhmac-key

# optionally map a hostname to a specific client key alias
openidm.ssl.host.aliases=localhost=

# policy enforcement enable/disable
openidm.policy.enforcement.enabled=true

# node id if clustered; each node in a cluster must have a unique node id
openidm.node.id=node1

# enables the execution of persistent schedulers
openidm.scheduler.execute.persistent.schedules=true

# substitute proper values for the datasource json files by specifying
# the repo url and port (MSSQL = 1433, MYSQL = 3306, POSTGRES = 5432) ;
# those can also be passed via java properties
#
# openidm.repo.host =
# openidm.repo.port =

# enables the statistics MBean for BoneCP. Enabling this will have a performance impact on BoneCP.
openidm.bonecp.statistics.enabled=false

# determines whether javascript exceptions will include debug information - e.g. file name, line
 number
javascript.exception.debug.info=false

# determines the TLS version used by the http client in the external rest service to connect to TLS-
protected resources
# valid values: SSLv3, TLSv1, TLSv1.1, TLSv1.2
# defaults to TLSv1.2 if not specified
#openidm.external.rest.tls.version=TLSv1.1

# enables API Descriptor generation, which if set to 'false', renders the API Explorer non-functional
openidm.apidescriptor.enabled=true
```

Follow this process before you start the update:

- If you have customized the default `conf/boot/boot.properties` file in your OpenIDM 4.5 instance, rename that file to something like `conf/boot/boot.properties.bak`.

- Extract `IDM-5.0.0.zip`.
- Copy the default `conf/boot/boot.properties` file from the version 5 delivery to your 4.5 `conf/boot` directory.

  If you have not customized the default 4.5 `conf/boot/boot.properties` file, you can overwrite it with version 5 of this file.
- Compare the 4.5 file (`conf/boot/boot.properties.bak`) with the version 5 `boot.properties` file and copy any customized properties from the 4.5 file to the new version 5 file.

Note that the version 5 file does not include the following lines from the previous version:

```
# valid instance types for node include standalone, clustered-first, and clustered-additional
openidm.instance.type=standalone
```

The `openidm.instance.type` property was used to propagate a keystore to different nodes in a cluster. That responsibility is now up to you. For more information, see "*Clustering, Failover, and Availability*" in the *Integrator's Guide*.

After your changes to `boot.properties` are complete, the update process will add a `boot.properties.new-unix-time` file to your project's `conf/boot/` subdirectory.

Review Repository Update Scripts

OpenIDM 5 includes a number of scripts that you must run to update your repository schema. The update process detects which scripts are required for your repository, extracts them from the OpenIDM 5 binary, and prompts you to save them in a directory relative to the 4.5 installation directory.

The scripts must be run at different stages during the update process. The following section describes which scripts you must run at which time. You might need to get Database Administrator (DBA) help and approval to run these update scripts.

You can review what the scripts do by extracting the OpenIDM 5 binary and locating the scripts in the `/path/to/openidm/db/repo/script/update` directory, where *repo* refers to your JDBC repository type. The OpenIDM 5 binary includes repository update scripts for the update from version 4.0 to version 4.5. You can ignore these scripts in your update from version 4.5 to version 5. The update will extract only those scripts required for this specific update. For example, the following scripts are required for a MySQL repository:

```
v3_add_audit_access_response_detail.sql
v4_drop_security_table.sql
v5_modify_indices_for_audit.sql
```

Start OpenIDM

OpenIDM must be running when you launch an update (using the UI or the CLI).

> **Warning**
>
> If you use anything but the standard Admin and Self-Service UIs, the update process may affect you in different ways; see "Files Changed During the Update Process". If you followed the procedure described in "Customizing

the UI" in the *Integrator's Guide*, you'll have custom files in the `openidm/ui/admin/extension` and `openidm/ui/selfservice/extension` directories.

OpenIDM 5 includes significant UI improvements. The update process does not copy those improvements to the noted `extension/` subdirectories.

When you're ready to run the update process, "Updating to OpenIDM 5".

4.3.1.1. Disabling and Re-enabling Connections to OpenAM

Before you update, you must disable connections to OpenAM. If you set up this connection as described in "*Integrating OpenIDM With the ForgeRock Identity Platform*" in the *Samples Guide*, you can use the following procedures to disable and re-enable the connection to OpenAM:

Disabling a Connection to OpenAM

If you've followed the procedure in the OpenIDM 4.5 *Full Stack Sample - Using OpenIDM in the ForgeRock Identity Platform*, you've set up a ForgeRock Identity Provider.

In that sample, you've configured a `OPENAM_SESSION` module, which you can disable as follows from the OpenIDM 4.5 Admin UI:

1. Select Configure > System Preferences. In the default OpenIDM 4.5 Authentication tab, edit the `OpenAM Session` module and set `Module Enabled` to false.
2. You should now be able to log into OpenIDM normally, without authenticating through OpenAM.
3. You can now proceed with the update process described in this chapter.

4.3.1.1.1. Re-enabling a Connection to OpenAM

If you need integration with OpenAM, we recommend that you also upgrade to AM 5. For more information, see the *OpenAM Upgrade Guide*.

You can then configure integration with OpenAM based on the scenario described in "*Integrating OpenIDM With the ForgeRock Identity Platform*" in the *Samples Guide*.

4.3.2. Updating to OpenIDM 5

When you have completed the steps in "Preparing for Updates", your OpenIDM 4.5 instance is ready for an update to version 5.

Copy the new update binary, `IDM-5.0.0.zip`, to the `/path/to/openidm/bin/update` directory. OpenIDM creates that directory the first time you start OpenIDM 4.5.

Now you can start the update process, using the CLI or the Admin UI. This section includes separate procedures for those options.

Note

During the update, you will see warnings similar to the following in the OpenIDM console:

```
WARNING: Unable to create dummy audit event handler for...
```

These warnings have no effect on the successful completion of the update and can be safely ignored.

Updating From the CLI

1. Start the update process from the command-line:

   ```
   $ cd /path/to/openidm
   $ ./cli.sh update \
    --acceptLicense \
    --user openidm-admin:openidm-admin \
    --url http://localhost:8080/openidm \
    IDM-5.0.0.zip
   ```

   Note

   If you are using a port other than `8080`, specify the port number. Include **--port *number*** in the **./cli.sh update** command.

2. During the update process, you are prompted to save the repository update scripts for your particular repository. For example, the following message is displayed for a MySQL repository:

   ```
   Database repo update files present in archive were found:
   v3_add_audit_access_response_detail.sql
   v4_drop_security_table.sql
   v5_modify_indices_for_audit.sql
   Please enter the directory to save repository update files:
   ```

   Enter a directory name, for example, `update-scripts`. The update process creates this directory, relative to your OpenIDM 4.5 installation directory, and saves the required scripts to that directory.

   The update then displays the following message and restarts the server:

   ```

   Repository scripts are in the following directory: update-scripts.
   Please come back and run the update again with --skipRepoUpdatePreview after updates have been
    reviewed.
   Restarting OpenIDM.
   Restart request completed.
   ```

3. Apply the following update script, depending on your repository:

   - For MySQL, MSSQL, and DB2, apply the `v3_add_audit_access_response_detail.sql` script.
   - For Oracle Database, apply the `v3_add_fk_relationships.sql` and `v4_add_audit_access_response_detail.sql` scripts, in numeric order.

- For PostgreSQL, apply the `v5_add_audit_access_response_detail.sql` script.

How you apply these scripts will depend on your repository. The following example applies the first update script required for a MySQL repository:

```
$ mysql -u root -p < openidm/update-scripts/v3_add_audit_access_response_detail.sql
```

4. Launch the update command again. This time, include the **--skipRepoUpdatePreview** option because you have already obtained the repository update scripts:

```
$ ./cli.sh update \
 --skipRepoUpdatePreview \
 --acceptLicense \
 --user openidm-admin:openidm-admin \
 --url http://localhost:8080/openidm \
 IDM-5.0.0.zip
```

The update can take some time. During the update, you will see the following messages in the terminal where you launched the update:

```
Update procedure is still processing...
The update process is complete with a status of PENDING_REPO_UPDATES
Run repository update scripts now, and then enter 'yes' to complete the OpenIDM update process.
```

5. At this point you can apply the remaining repository update scripts.

These scripts are numbered differently for the different repositories, but are essentially the same scripts: `v*_drop_security_table.sql` and `v*_modify_indices_for_audit.sql`.

Apply the scripts in numeric order. The following example applies the remaining update scripts for a MySQL repository:

```
$ mysql -u root -p < openidm/update-scripts/v4_drop_security_table.sql
$ mysql -u root -p < openidm/update-scripts/v5_modify_indices_for_audit.sql
```

6. When you have applied the scripts, type `yes` in the terminal where you launched the update, to continue the update process.

The update completes and restarts the server.

You should see the following message in the console:

```
-> OpenIDM version "5.0.0"
OpenIDM ready
```

7. If you are using the Admin UI, you *must* refresh your browser *after* the update is complete.

Updating From the Admin UI

1. Log into the Admin UI at `http://localhost:8080/admin`.

2. Select Configure > System Preferences, and choose the Update tab. You should see `IDM-5.0.0.zip` in the list of Available Updates.

3. Click Install to start the process, then Confirm and Install to enable Maintenance Mode.

4. The instructions in the UI are intuitive.

   During the update, an Installation Preview screen will show you a list of the files that will be affected by the update, in the categories described in "Preview of File Updates". You'll be asked to confirm the Update and accept the license agreement.

5. The Repository Update Script Preview shows the scripts that you must run to update your repository. The scripts must be run *at specific times* during the update process, so follow these steps exactly.

   Download all of the scripts and move them to a local directory (for example `openidm/update-scripts`). Then, *before* you click Continue, apply the following scripts, depending on your repository:

   - For MySQL, MSSQL, and DB2, apply the `v3_add_audit_access_response_detail.sql` script.

   - For Oracle Database, apply the `v3_add_fk_relationships.sql` and `v4_add_audit_access_response_detail.sql` scripts, in numeric order.

   - For PostgreSQL, apply the `v5_add_audit_access_response_detail.sql` script.

   How you apply these scripts will depend on your repository. The following example applies the first update script required for a MySQL repository:

   ```
   $ mysql -u root -p < openidm/update-scripts/v3_add_audit_access_response_detail.sql
   ```

   When you have applied the first script/scripts, click Continue.

6. The update restarts and processes most of the files that need to be updated. It then pauses with the following message:

   ```
   Update paused. Process all repository updates manually before resuming.
   ```

   At this point you can apply the remaining repository update scripts.

   These scripts are numbered differently for the different repositories, but are essentially the same scripts: `v*_drop_security_table.sql` and `v*_modify_indices_for_audit.sql`.

   Apply the scripts in numeric order. The following example applies the remaining update scripts for a MySQL repository:

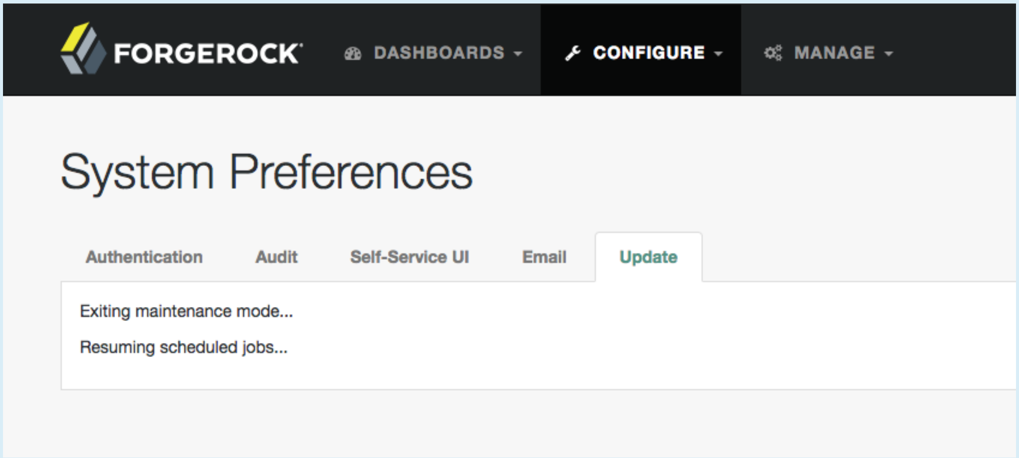
   ```
   $ mysql -u root -p < openidm/update-scripts/v4_drop_security_table.sql
   $ mysql -u root -p < openidm/update-scripts/v5_modify_indices_for_audit.sql
   ```

7. When you have applied the scripts, click Mark Complete to continue the update process.

   The update completes and restarts the server.

Note

For updates from OpenIDM 4.5 to OpenIDM 5, the update process appears to "get stuck" just before completion. This issue has been fixed in version 5, and should not appear when you update *from* OpenIDM 5 to a later version.



Exiting maintenance mode...

Resuming scheduled jobs...

You should already see the following messages in the console:

```
-> OpenIDM version "5.0.0" (revision: <someUUIDsubset>) jenkins-OpenIDM - postcommit-num origin/master
OpenIDM ready
```

8. When the update is complete, refresh the browser. Scroll to the bottom of the Admin UI. You should see the updated version number in the footer of the web page. You can also see the updated version by navigating to Configure > System Preferences > Update.

   Refresh your browser *after* this update is complete.

Important

Regardless of the method you have used to run the update, you will have additional work to do before putting your system back into production. For customizations to the server, start with files that include `.new-unix_time` extensions. For more information, see "Managing Custom Changes After an Update".

In addition, if you are updating from OpenIDM 4.5.1 to IDM 5, remove the following file manually when the update has completed:

```
/path/to/openidm/bundle/json-patch-20.1.5.jar
```

4.3.3. Managing Custom Changes After an Update

If you've customized OpenIDM 4.5, you may find files with the following extensions: `.old` and `.new`. For more information, see "An Overview of the Update Process".

On Linux/UNIX systems, you can find *some* of these files with the following commands:

```
$ cd /path/to/openidm
$ find . -type f -name "*.old*"
$ find . -type f -name "*.new*"
```

- Files with the `.old-unix_time` extension are saved from the configuration you had when starting this update process.
- Files with the `.new-unix_time` extension are files from OpenIDM 5 that have not been incorporated into your updated installation. For example, if you find a `system.properties.new-unix_time` file in your `project-dir` directory, OpenIDM is still using your pre-update version of this file, which would still be named `system.properties`.

  To take full advantage of OpenIDM 5, analyze the new features shown in files with the `.new-unix_time` extension. If you have files with multiple `.new-unix_time` extensions, use the file with the latest *unix_time*.

  Pay particular attention to your connector configuration files (`provisioner.openicf-connector-name.json`). The update removes outdated connector versions so you *must* make sure that the `bundleVersion` in your connector configuration matches the version of the connector in `/path/to/openidm/connectors`, or specifies a range that includes the connector version, for example `[1.1.0.0,1.4.0.0]`. For more information, see "Setting the Connector Reference Properties" in the *Integrator's Guide*.

4.3.3.1. What You Should Do With Your JSON Files After Updating

The update process does not account for any changes that you made to existing standard JSON files such as `sync.json` and `managed.json`. In fact, the update process overwrites these files with the standard OpenIDM 5 versions of those files.

Do not overwrite the new version 5 files. Instead, analyze the custom settings from your original JSON files. Review "*What's New*" in the *Release Notes*. Apply each custom setting to the files in your updated deployment and test the results to make sure they still work as intended.

> **Note**
>
> If you've followed the recommendations in "Configuring the Server for Production" in the *Integrator's Guide*, changes you've written directly to JSON files may not be loaded into the repository. This includes JSON files that may have been removed per the `removeFile` code block in "The `update.json` File".

To address these issues, you have two options:

- Write changes directly to the repository, using an HTTP PATCH request.
- Temporarily change production settings that may block configuration updates from JSON files, such as those described in "Disabling Automatic Configuration Updates" in the *Integrator's Guide*.

4.3.3.2. What You Should Do With Your UI After Updating

If you have a custom Admin UI or Self-Service UI, you need to take a few extra steps.

You will need the updated UI files from the `openidm/ui/admin/default` and `openidm/ui/selfservice/default` directories. So you'll have to delete some files first.

Warning

Make sure you've saved any custom files from the `openidm/ui/admin/extension` and `openidm/ui/selfservice/extension` subdirectories, as described in the introduction to "Updating to OpenIDM 5", and then follow these steps:

1. Delete the existing `openidm/ui/admin/extension` and `openidm/ui/selfservice/extension` subdirectories.
2. Copy files from the `openidm/ui/admin/default` and `openidm/ui/selfservice/default` subdirectories with the following commands:

```
$ cd /path/to/openidm/ui
$ cp -r selfservice/default/. selfservice/extension
$ cp -r admin/default/. admin/extension
```

3. Review your UI custom files. Compare them against the OpenIDM 5 version of these files.
4. Apply your custom changes to each new OpenIDM 5 UI file in the `openidm/ui/admin/extension` and `openidm/ui/selfservice/extension` subdirectories.

4.3.4. Migrating to OpenIDM 5 on Windows

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

Note

Updates to OpenIDM 5 on Microsoft Windows remain a manual process.

To perform a migration from OpenIDM 4.5 to OpenIDM 5 on Windows, follow these steps. For the purposes of this procedure, the path to the existing instance of OpenIDM 4.5 is defined as `\path\to\openidm-4.5`. In contrast, the path to the OpenIDM 5 deployment is defined as `\path\to\openidm-5`:

1. Download and extract the OpenIDM 5 (`IDM-5.0.0.zip`).

2. Stop your existing OpenIDM 4.5 server, if it is running. Access the Java console where it is running and enter the **shutdown** command at the OSGi console:

   ```
   -> OpenIDM ready
   -> shutdown
   ```

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

4. Boot properties: On the OpenIDM 5 server, edit the `conf\boot\boot.properties` file to match any customizations that you made on your OpenIDM 4.5 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 4.5 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 4.5 deployment.

   Depending on the level of customization you have made in your current deployment, it might be simpler to start with your OpenIDM 5 `boot.properties` file, and copy all new settings from that file to the version associated with OpenIDM 5. 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. Security files: Copy the contents of your OpenIDM 4.5 `security\` folder to the OpenIDM 5 instance.

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

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

6. Scripts: Migrate any custom scripts or default scripts that you have modified to the scripts directory of your OpenIDM 5 instance. In general, custom and customized scripts should be located in the `openidm-4.5\script` directory on the OpenIDM 4.5 deployment:

   - For custom scripts, review "*Compatibility*" in the *Release Notes*. If you're confident that the scripts will work as intended on OpenIDM 5, then copy these scripts to the new instance. For example:

     ```
     PS C:\> cd \path\to\openidm-5
     PS C:\> cp \path\to\openidm-4.5\script\my-custom-script.js .\script
     ```

- If you modified an existing OpenIDM 4.5 script, compare the default versions of the OpenIDM 4.5 and OpenIDM 5 scripts. If you verify that nothing has changed, review what you've customized against "*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-5\script` directory. For example:

  ```
  PS C:\> cd \path\to\openidm-5
  PS C:\> cp \path\to\openidm-4.0\script\policy.js .\script\
  ```

- If a default script has changed since the 4.5 release, copy the modified script to the `openidm-5\script` directory. For example:

  ```
  PS C:\> cd \path\to\openidm-5
  PS C:\> cp bin\default\script\linkedView.js .\script
  ```

  Check that your customizations work as expected, then port the changes to the new script in the `openidm-5\script` directory.

7. Provisioner files: Modify any customized provisioner configurations in your existing project to point to the connectors that are provided with OpenIDM 5. Specifically, make sure that the `connectorRef` properties reflect the new connectors, where applicable. For example:

   ```
   "connectorRef" : {
         "bundleName": "org.forgerock.openicf.connectors.ldap-connector",
         "bundleVersion": "[1.4.0.0,1.5.0.0)",
         "connectorName": "org.identityconnectors.ldap.LdapConnector"
   },
   ```

   Alternatively, copy the connector .jar files from your existing installation into the `openidm\connectors\` folder of the new installation.

8. Complete the OpenIDM 5 installation, as described in "*Preparing to Install and Run Servers*".

9. As there is no automated way to migrate a customized configuration to OpenIDM 5, we recommend the following strategy:

   - Start with the default 4.5 configuration.

   - For each configuration file that you have customized, use a file comparison tool such as the Windows **fc.exe** utility to assess the differences between your customized file and the OpenIDM 5 file.

   - Based on the results of the **fc.exe** review, either use your existing file as a base and port the OpenIDM 5 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-5/conf` directory.

10. If you are using the UI, clear your browser cache after the migration. The browser cache contains files from the previous release, that might not be refreshed when you log in to the new UI.

11. Start OpenIDM 5:

```
PS C:\> cd \path\to\openidm-5
PS C:\> .\startup.bat
```

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

4.4. Updating to OpenIDM 5.0.1.1

You can only run an update to OpenIDM 5.0.1.1 from versions 5.0.0, 5.0.0.1, 5.0.0.2, 5.0.0.3 and 5.0.1.0. If you are on a version prior to 5.0.0, you must first update to 5.0.0. For more information, see "Updating to OpenIDM 5".

You can run the update process in one of two ways: using the command-line interface script `cli.sh` or the Admin UI.

Preparing Your 5.0.1.1 Update

The following procedure is similar to the steps outlined in "Updating to OpenIDM 5". The steps are summarized here for your convenience:

1. Download OpenIDM (`openidm-5.0.1.1.zip`) from ForgeRock's BackStage site.

2. Copy openidm-5.0.1.1.zip to `/path/to/openidm/bin/update`.

3. Back up your 5.0.0 deployment. Save any customized `*.json` configuration files, located in a `/path/to/openidm/conf` subdirectory. Save your custom directories.

4. If you have a read-only deployment, mount the directory in read-write mode before starting the update.

5. If you integrated OpenIDM with OpenAM, disable the authentication modules that you enabled for the integration.

6. If you have a custom `conf/boot/boot.properties` file, rename the file to something like `conf/boot/boot.properties.bak`.

7. Extract openidm-5.0.1.1.zip to a temporary directory. Copy the custom `conf/boot/boot.properties` file to your 5.0.1.1 `conf/boot` directory if you have customized the `boot/boot.properties` file.

   If you have not customized the default 5.0.0 `conf/boot/boot.properties` file, you can simply overwrite it with version 5.0.1.1 of this file.

8. Start OpenIDM. OpenIDM must be running when you launch an update using the Admin UI or the CLI.

9. Disable and re-enable connections to OpenAM. See "Disabling and Re-enabling Connections to OpenAM" for more information.

Updating to Version 5.0.1.1 Using the CLI

The update steps using the CLI (`cli.sh`) are the same as the steps in "Updating From the CLI", except for the zip file.

1. Go to your OpenIDM installation:

   ```
   $ cd /path/to/openidm
   ```

2. Create a directory to store any update scripts:

   ```
   $ mkdir update-scripts
   ```

3. Start the update process from the command-line:

   ```
   $ cd /path/to/openidm
   $ ./cli.sh update \
     --acceptLicense \
     --user openidm-admin:openidm-admin \
     --url http://localhost:8080/openidm \
     openidm-5.0.1.1.zip
   ```

   > **Note**
   >
   > If you are using a port other than `8080`, specify the port number. Include **--port *number*** in the **./cli.sh update** command.

4. The update process continues.

   ```
   Executing ./cli.sh...
   Starting shell in /idm
   Using boot properties at /idm/conf/boot/boot.properties
   License was accepted via command line argument.
   Database repo update files present in archive were found:
   v6_add_primary_key_to_properties_tables.sql
   ```

   If the script discovers a database repository update file, it will ask you to specify a directory to save the file. Enter a directory and then press return. The update will continue then stop and ask you to run the **cli.sh** command again.

   ```
   Please enter the directory to save the repository update files:
   <userinput>update-scripts</userinput>
   Repository scripts are in the following directory: update-scripts.
   Please come back and run the update again with --skipRepoUpdatePreview after updates have been
    reviewed.
   ```

5. Apply the update script, depending on your repository. The following example applies the update script required for a MySQL repository:

   ```
   $ mysql -u root -p < openidm/update-scripts/v6_add_primary_key_to_properties_tables.sql
   ```

6. Run the **cli.sh** command again with the **--skipRepoUpdatePreview** option:

```
$ cd /path/to/openidm
$ ./cli.sh update \
  --acceptLicense \
  --user openidm-admin:openidm-admin \
  --url http://localhost:8080/openidm \
  --skipRepoUpdatePreview \
  openidm-5.0.1.1.zip
```

7. The update process continues and completes with a `PENDING_REPO_UPDATES` status. Enter `yes` to complete the process.

   ```
   Executing ./cli.sh...
   Starting shell in /idm
   Using boot properties at /idm/conf/boot/boot.properties
   License was accepted via command line argument.
   Repository update preview was skipped.
   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 IDM-5.0.1.1.zip
   Update procedure is still processing...
   The update process is complete with a status of PENDING_REPO_UPDATES
   Run repository update scripts now, and then enter 'yes' to complete the IDM update process.
   <userinput>yes</userinput>
   Repo Updates status: COMPLETE
   Exiting maintenance mode...
   No longer in maintenance mode.
   Resuming the job scheduler.
   Scheduler has been resumed.
   ```

8. Restart your server.

   Your update has successfully completed.

Updating to Version 5.0.1.1 Using the Admin UI

1. Log into the Admin UI at `http://localhost:8080/admin`.

2. Select Configure > System Preferences, and choose the Update tab. You should see `openidm-5.0.1.1.zip` in the list of Available Updates.

3. Click Install to start the process, then click Confirm and Install to enable Maintenance Mode.

4. Next, review the Installation Preview screen. The screen shows a list of the files that will be affected by the update in the categories described in "Preview of File Updates".

   After reviewing the files, click Install Update to continue.

5. Review the License Agreement, and if you agree, click Accept Licence Agreement. At this point, you should see an Installing Update page.

6. Near the end of the update process, review the Repository Updates screen. The Repository Updates screen shows the script that you must apply to update your repository.

   Click Download to save the script, and move it to a local directory (for example `openidm/update-scripts`). Depending on your repository, download the following files:

   - For MySQL, MSSQL, and DB2, apply the `v6_add_primary_key_to_properties_tables.sql` script.
   - For Oracle databases: use `v7_add_primary_key_to_properties_tables.sql`.
   - For PostgreSQL databases: use `v9_add_primary_key_to_properties_tables.sql`.

   Important

   OpenIDM 5.0.1.1 has an existing bug with the Download button when downloading the `v6_add_primary_key_to_properties_tables.sql` file (for MySQL, MSSQL, or DB2 databases), which results in a file with an undefined value.

   If you have a MySQL, MSSQL, or DB2 database, manually copy the `v6_add_primary_key_to_properties_tables.sql` file from `/path/to/openidm/db/`*`repo_type`*`/scripts/updates` to the `openidm/update-scripts` directory, then apply the script as shown in the next step.

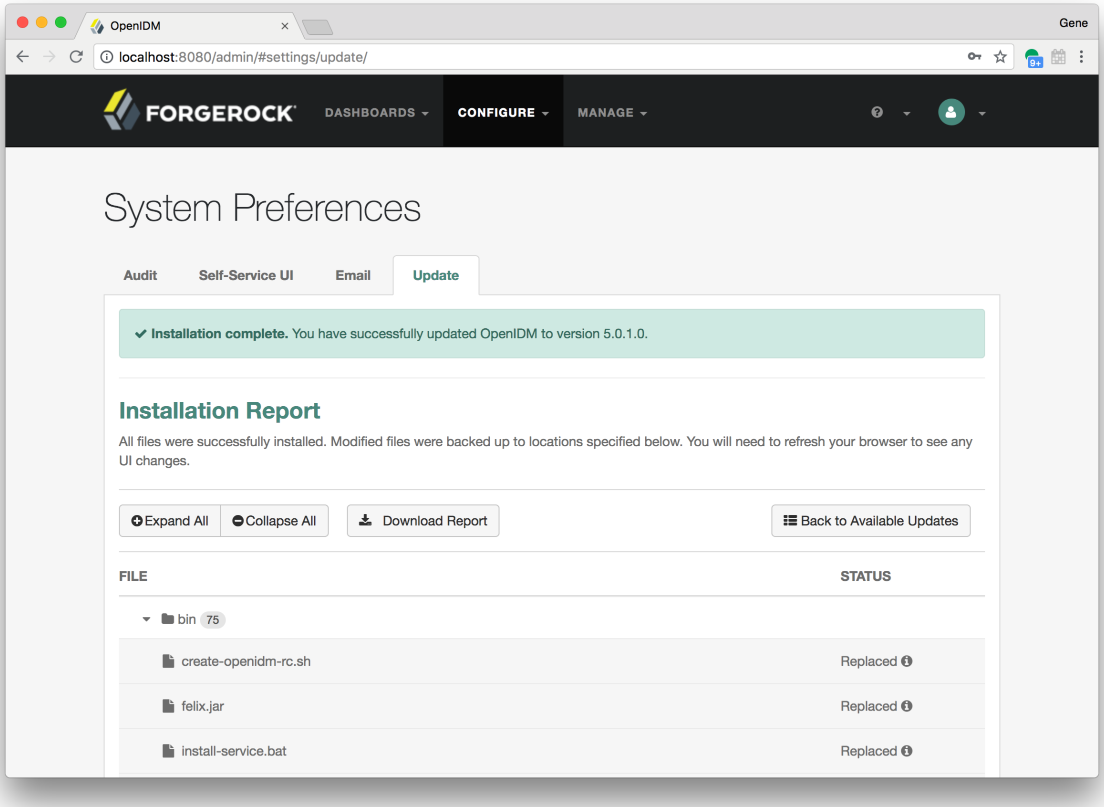
7. Apply the script depending on your repository.

   For example, the following example applies the update script required for a MySQL repository:

   ```
   $ mysql -u root -p < /path/to/openidm/update-scripts/v6_add_primary_key_to_properties_tables.sql
   ```

8. When you have applied the scripts, click Mark Complete.

   When the update completes, refresh your browser. You should see an Installation Complete message that indicates that you have successfully updated OpenIDM to version 5.0.1.1. The page also displays an installation report of all the files that were modified and backed up to specific locations.



✓ **Installation complete.** You have successfully updated OpenIDM to version 5.0.1.0.

Installation Report

All files were successfully installed. Modified files were backed up to locations specified below. You will need to refresh your browser to see any UI changes.

FILE	STATUS
bin 75	
create-openidm-rc.sh	Replaced
felix.jar	Replaced
install-service.bat	Replaced

9. Restart your server.

   Your update has successfully completed.

4.5. Placing a Server in Maintenance Mode

The Maintenance Service disables non-essential services of a running OpenIDM instance, in preparation for an update 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 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 Jobs" 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 the update process starts, maintenance mode is enabled automatically. Before the update process is complete, maintenance mode is disabled. You can disable maintenance mode over REST with 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=disable"
```

To check whether a server 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 server is in maintenance mode, the command returns `"maintenanceEnabled": true`, otherwise it returns `"maintenanceEnabled": false`.

Appendix A. Installing 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. This appendix describes 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 "*Preparing to Install and Run Servers*", as well as the prerequisites described in "*Before You Install*" 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-5.0.0.zip
$ sudo chown -R idm.idm /idm
```

A.2. Redirect Output Through Configuration Files

In this section, you will modify appropriate configuration files 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

   You can also route audit data to a remote data store. For an example of how to send audit data to a MySQL repository, see "*Audit Samples*" 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"
      },
      "topics" : [ "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

In a production environment, you must configure a supported repository, 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.

When you log in again as user `idm`, your `OPENIDM_PID_FILE` variable should redirect the 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 Servers*" 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
```

Glossary

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 Source Objects With 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