Installation Guide

/ ForgeRock Identity Management 6.5

Latest update: 6.5.2.0

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

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

Abstract

Guide to installing, updating, and uninstalling ForgeRock® Identity Management 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 ForgeRock Identity Management 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 ForgeRock Identity Management 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 ForgeRock Identity Management 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 ForgeRock Identity Management software installed, see "*Compatibility*" in the *Release Notes* before you install this version.

2. 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.

3. 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 IDM.

Note

This documentation set 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 IDM.

1.1.1. Java Prerequisites

For details of the supported Java Environment, see "Preparing 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\jre8`.
7. Click OK.

On Linux systems, if `startup.sh` reports `JAVA_HOME not available, Java is needed to run IDM` and you've already installed Java, use the following steps to set `JAVA_HOME`:

1. Open the user shell configuration file found in your home directory.

2. Add the `JAVA_HOME` variable to the user shell configuration file, setting the value to `/usr`. In Bash, this would appear as `export JAVA_HOME="/usr"`.

1.1.2. Application Container

IDM 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 IDM. To set up the server on a read-only volume, read "*Installing on a Read-Only Volume*".

To Install IDM

Follow these steps to install IDM:

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

```
$ java -version
java version "1.8.0_121"
Java(TM) SE Runtime Environment (build 1.8.0_121-b13)
Java HotSpot(TM) 64-Bit Server VM (build 25.121-b13, mixed mode)
```

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

2. Download IDM from the ForgeRock BackStage download site. Releases on the ForgeRock BackStage download site are thoroughly validated for ForgeRock customers who run the software in production deployments, and for those who want to try or test a given release.

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

```
$ unzip ~/Downloads/IDM-6.5.2.0.zip
Archive:  IDM-6.5.2.0.zip
  inflating: openidm/.checksums.csv
   creating: openidm/bundle/
 extracting: openidm/bundle/openidm-audit-6.5.2.0
.jar
...
```

4. By default, IDM listens for HTTP and HTTPS connections on ports 8080 and 8443, respectively. To change these port numbers, edit the following settings in your `resolver/boot.properties` file:

   - `openidm.port.http`
   - `openidm.port.https`

When you deploy IDM in production, you *must* set `openidm.host` to the URL of your deployment, in the same `resolver/boot.properties` file. Otherwise, calls to the `/admin` endpoint are not properly redirected.

Deployment URLs will vary, depending on whether you're using a load balancer. While IDM documentation does not specify how you'd configure a load balancer, you'll need to configure IDM in a cluster as described in "Configuring an IDM Instance as Part of a Cluster" in the *Integrator's Guide*, and specifically in "Deploying Securely Behind a Load Balancer" in the *Integrator's Guide*.

5. Before running IDM in production, replace the default embedded DS repository with a supported repository.

   For more information, see "*Selecting a Repository*".

To Start IDM

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

Follow these steps to run IDM interactively:

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

   - Start IDM (UNIX):

     ```
     $ cd /path/to/openidm
     $ ./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
     -> OpenIDM version "6.5.2.0"
     OpenIDM ready
     ```

   - Start IDM (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"
     -> OpenIDM version "6.5.2.0"
     OpenIDM ready
     ->
     ```

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

```
-> scr list
BundleId Component Name Default State
    Component Id State      PIDs (Factory PID)
[   5]   org.forgerock.openidm.config.enhanced.starter  enabled
    [   1] [active      ] org.forgerock.openidm.config.enhanced.starter
[   5]   org.forgerock.openidm.config.manage  enabled
    [   0] [active      ] org.forgerock.openidm.config.manage
[  10]   org.forgerock.openidm.datasource.jdbc  enabled
[  10]   org.forgerock.openidm.repo.jdbc  enabled
[  11]   org.forgerock.openidm.repo.ds  enabled
    [  48] [active      ] org.forgerock.openidm.repo
.ds
..
.
->
```

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 IDM

You can stop IDM 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 IDM by using the **shutdown.sh** script, located in the `/path/to/openidm` directory:

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

1.3. Installing IDM as a Service

The following sections describe how to install and run IDM as a service, on Windows and Linux systems:

1.3.1. Installing as a Windows Service

You can install IDM 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 IDM as a service, but starting the service will fail.

Before you launch the `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-6.5.2.0.zip file, as described previously, and navigate to the *`install-directory`*`\bin` directory:

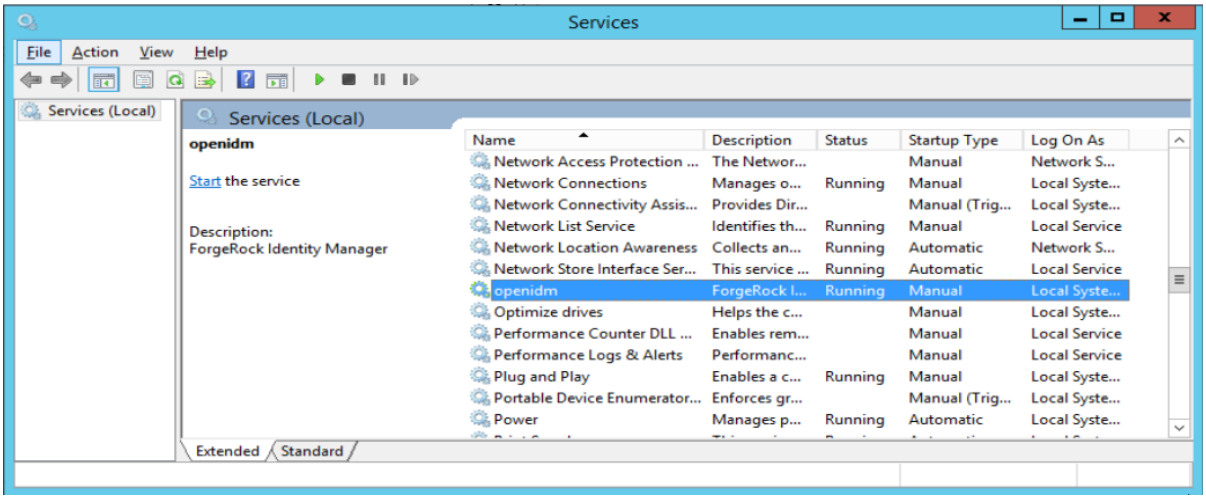
```
C:\>cd openidm\bin
C:\openidm\bin>
```

2. Run the `service.bat` command with the `/install` option, specifying the name that the service should run as:

```
C:\openidm\bin>service.bat /install openidm
ForgeRock Identity Management Server successfully installed as "openidm" service
```

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

Running as a Windows Service



4. By default, the IDM service is run by `Local System`, which is a system-level service account built in to Windows. Before deploying to production, it is recommended you switch to an account with fewer permissions. The account running the IDM service needs to be able to read, write, and execute only the directories related to IDM. For more information about service accounts, see Service Accounts in the Microsoft documentation.

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

6. If you want to uninstall the IDM service, first use the Windows Service Manager to stop IDM and then run the following command:

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

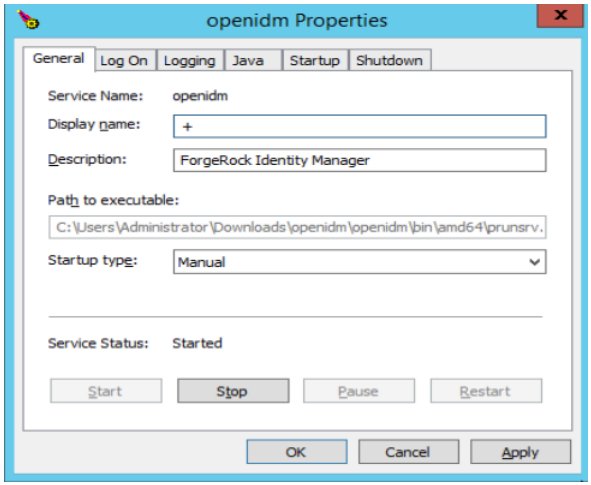
7. If desired, you can then set up IDM with a specific project directory:

```
C:\install-directory\openidm\bin>service.bat /install openidm -p C:\project-directory
ForgeRock Identity Management Server successfully installed as "openidm" service
```

You can also manage configuration details with the Procrun monitor application. IDM includes the associated `prunmgr.exe` executable in the `C:\install-directory\openidm\bin` directory.

For example, you can open the Windows service configuration application for IDM with the following command, where `ES` stands for *Edit Service Configuration*

```
C:\install-directory\openidm\bin>prunmgr.exe //ES/openidm
```



The `prunmgr.exe` executable also includes the monitor application functionality described in the following Apache Commons page on the: *Procrun monitor Application*. However, IDM does not include the Procrun service application.

For example, if you've configured IDM as a Windows service, you can start and stop it with the following commands:

```
C:\install-directory\openidm\bin>prunmgr.exe //MR/openidm
C:\install-directory\openidm\bin>prunmgr.exe //MQ/openidm
```

In these commands, `MR` is the option to *Monitor and Run* IDM, and `MQ` stands for *Monitor Quit*, which stops the IDM service.

1.3.2. Installing as a Linux Service

IDM provides a script that can generate `SysV` or `Systemd` service initialization scripts. You can start the script as the root user, or configure it to start during the boot process.

When IDM runs as a service, logs are written to the installation directory.

1. If you have not yet installed IDM, follow the steps in "To Install IDM".

2. Review the options by running the following script:

```
$ cd /path/to/openidm/bin
$ ./create-openidm-rc.sh
Usage: ./create-openidm-rc.sh --[systemd|chkconfig|lsb]
Outputs OpenIDM init file to stdout for the given system

--systemd    Generate Systemd init script. This is preferred for all modern distros
.
--chkconfig  Generate SysV init script with chkconfig headers (RedHat/
CentOS)
--lsb        Generate SysV init script with LSB headers (Debian/
Ubuntu)
...
```

The following sections describe how you can create each of these scripts:

1.3.2.1. Setting up a Systemd Service

If you're running relatively standard versions of Red Hat Enterprise Linux (CentOS Linux) version 7.x, or Ubuntu 16.04 and later, you'll want to set up a systemd service script. To set up such a script, navigate to the `/path/to/openidm/bin` directory, and run the following command:

```
$ ./create-openidm-rc.sh --systemd
```

As noted in the output, you can set up the IDM service on a standard systemd-based Linux distribution with the following commands:

```
$ ./create-openidm-rc.sh --systemd > openidm.service
$ sudo cp openidm.service /etc/systemd/system/
$ systemctl enable openidm
$ systemctl start openidm
```

To stop the IDM service, run the following command:

```
$ systemctl stop openidm
```

You can modify the `openidm.service` script. The following excerpt would run IDM with a startup script in the `/home/idm/project` directory:

```
[Unit]
Description=ForgeRock OpenIDM
After=network.target auditd.target

[Service]
Type=simple
SuccessExitStatus=143
Environment=JAVA_HOME=/usr
User=testuser
ExecStart=/root/openidm/startup.sh -p /home/idm/project
ExecStop=/root/openidm/shutdown.sh

[Install]
WantedBy=multi-user.target
```

Run the following commands to reload the configuration and then start the IDM service script:

```
$ systemctl daemon-reload
$ systemctl start openidm
```

1.3.2.2. Setting up a SysV Service (Red Hat)

If you're running relatively standard versions of Red Hat Enterprise Linux (CentOS Linux) version 6.x, you'll want to set up a SysV service script, with runlevels controlled through the **chkconfig** command. To set up such a script, navigate to the `/path/to/openidm/bin` directory, and run the following command:

```
$ ./create-openidm-rc.sh --chkconfig
```

You can then set up and start the IDM service on a Linux distribution that uses SysV init scripts, with the following commands:

```
$ ./create-openidm-rc.sh --chkconfig  > openidm
$ sudo cp openidm /etc/init.d/
$ sudo chmod u+x /etc/init.d/openidm
$ sudo chkconfig --add openidm
$ sudo chkconfig openidm on
$ sudo service openidm start
```

To stop the IDM service, run the following command:

```
$ sudo service openidm stop
```

You can modify the `/etc/init.d/openidm` script. The following excerpt would run IDM with the `startup.sh` script in the `/path/to/openidm` directory:

```
START_CMD="PATH=$JAVA_BIN_PATH:$PATH;nohup $OPENIDM_HOME/startup.sh >$OPENIDM_HOME/logs/server.out 2>&1 &"
```

You can modify this line to point to some `/path/to/production` directory:

```
START_CMD="PATH=$JAVA_BIN_PATH:$PATH;nohup $OPENIDM_HOME/startup.sh -p /path/to/production >$OPENIDM_HOME/
logs/server.out 2>&1 &"
```

Run the following commands to reload the configuration and then start the IDM service script:

```
$ sudo service openidm start
```

If you run Linux with SELinux enabled, change the file context of the newly copied script with the following command:

```
$ sudo restorecon /etc/init.d/openidm
```

Verify the change to SELinux contexts with the `ls -Z /etc/init.d` command. For consistency, change the user context to match other scripts in the same directory with the `sudo chcon -u system_u /etc/init.d/openidm` command.

1.3.2.3. Setting up a SysV Service (Ubuntu)

If you're running relatively standard older versions of Ubuntu Linux, versions which support SysV services, you'll want to set up a SysV service script, with runlevels controlled through the **update-rc.d**

command. To set up such a script, navigate to the `/path/to/openidm/bin` directory, and run the following command:

```
$ ./create-openidm-rc.sh --lsb
```

You can then set up and start the IDM service on a Linux distribution that uses SysV init scripts, with the following commands:

```
$ ./create-openidm-rc.sh --lsb  > openidm
      $ sudo cp openidm /etc/init.d/
      $ sudo chmod u+x /etc/init.d/openidm
      $ sudo update-rc.d openidm defaults
      $ sudo service openidm start
```

To stop the IDM service, run the following command:

```
$ sudo service openidm stop
```

You can modify the `/etc/init.d/openidm` script. The following excerpt would run IDM with the `startup.sh` script in the `/path/to/openidm` directory:

```
START_CMD="PATH=$JAVA_BIN_PATH:$PATH;nohup $OPENIDM_HOME/startup.sh >$OPENIDM_HOME/logs/server.out 2>&1 &"
```

You can modify this line to point to some `/path/to/production` directory:

```
START_CMD="PATH=$JAVA_BIN_PATH:$PATH;nohup $OPENIDM_HOME/startup.sh -p /path/to/production >$OPENIDM_HOME/
logs/server.out 2>&1 &"
```

You can then run the following commands to reload the configuration and then start the IDM service script:

```
$ sudo service openidm restart
```

1.4. Getting Started With the REST Interface

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

IDM is accessible over the regular and secure HTTP ports of the Jetty Servlet container, 8080, and 8443. Most of the command-line examples in this documentation set use the regular HTTP port, to avoid you having to use certificates just to test IDM. In a production deployment, install a CA-signed certificate and restrict REST access to a secure (HTTPS) port.

To run **curl** over the secure port, 8443, you must either include the **--insecure** option, or follow the instructions in "Restricting 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 IDM starts, or with a `*.crt` file provided by a certificate authority.

Note

Some of the examples in this documentation set use client-assigned IDs (such as `bjensen` and `scarter`) when creating objects because it makes the examples easier to read. If you create objects using the Admin UI, they are created with server-assigned IDs (such as `55ef0a75-f261-47e9-a72b-f5c61c32d339`). Generally, immutable server-assigned UUIDs are used in production environments.

1. Access the following URL to obtain the JSON representation of all users in the IDM 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 IDM with an empty repository, no users exist.

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

   The following **curl** commands create a managed user in the repository, and set the user's ID to `jdoe`:

   - Create `joe` (UNIX):

```
$ curl \
 --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"
 }' \
 http://localhost:8080/openidm/managed/user?_action=create
{
  "_id": "joe",
  "_rev": "00000000c03fd7aa",
  "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 ^
 --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\"
 }" ^
 http://localhost:8080/openidm/managed/user?_action=create
```

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

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 http://localhost:8080/openidm/managed/user/joe
{
  "_id": "joe",
  "_rev": "00000000c03fd7aa",
  "userName": "joe",
  "givenName": "joe",
  "sn": "smith",
  "mail": "joe@example.com",
  "telephoneNumber": "555-123-1234",
  "description": "My first user",
  "accountStatus": "active",
  "effectiveRoles": [],
  "effectiveAssignments": []
}
```

1.4.1. 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":["internal/role/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": "00000000c03fd7aa","lastPasswordSet":"","postalCode":"",
"_id":"joe3","description":"My first user","accountStatus":"active","telephoneNumber":
"555-123-1234","roles":["internal/role/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 \
"http://localhost:8080/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.

1.5. IDM User Interfaces

You can manage IDM using Web-based user interfaces, called the UI in this documentation set.

IDM provides UIs at two different endpoints, `/` and `/admin`. We refer to the administrative tools available at each endpoint as the End User UI and the Administrative UI (Admin UI), respectively.

The End User UI allows regular (non-administrative) users to manage their account data, consent, workflows, and shared resources. If self-service is enabled, regular users can also self-register and reset their own passwords. For more information, see "*Configuring User Self-Service*" in the *Integrator's Guide*.

In essence, the End User UI supports day-to-day administrative tasks for end users of an IDM system.

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

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

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

1.6. About the Repository

By default, IDM installs an embedded ForgeRock Directory Services (DS) instance for use as its repository. This makes it easy to get started. Before you use IDM in production, you must replace the embedded DS repository with a supported repository. For more information, see "*Selecting a Repository*".

You can query the internal repository directly by using the LDAP command-line utilities provided with DS. For example, the following command returns all the objects in the repository of a default IDM project:

```
$ ldapsearch \
 --hostname localhost \
 --port 31389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --baseDN "dc=openidm,dc=forgerock,dc=com" \
 "(objectclass=*)"

dn: dc=openidm,dc=forgerock,dc=com
objectClass: top
objectClass: domain
dc: openidm

dn: ou=links,dc=openidm,dc=forgerock,dc=com
objectClass: top
objectClass: organizationalUnit
ou: links

dn: ou=internal,dc=openidm,dc=forgerock,dc=com
objectClass: top
objectClass: organizationalUnit
ou: internal

dn: ou=users,ou=internal,dc=openidm,dc=forgerock,dc=com
objectClass: top
objectClass: organizationalUnit
ou: users
...
```

For more information about the DS command-line utilities, see the DS Tools Reference.

1.7. Starting a New Project

When you extract the IDM .zip file, you have a default project under `/path/to/openidm`. You can use this project to test customizations, but you should not run the default project in production.

Set up a new project as follows:

1. Create a directory for your new project:

   ```
   $ mkdir /path/to/my-project
   ```

   Note that the automated update process does not work for projects that are subdirectories of the default project. You should therefore create your new project directory somewhere outside of `/path/to/openidm/`.

2. Set up a minimal configuration:

   - If your project will be similar to any of the sample configurations (described in the Samples Guide) copy the contents of the sample to your new project.

     For example:

     ```
     $ cp -r /path/to/openidm/samples/sync-with-ldap/* /path/to/my-project/
     ```

     You can then customize the sample configuration according to your requirements.

   - If you do not want to start with one of the sample configurations, copy the `conf/` and `script/` directories from the default project to your new project directory:

     ```
     $ cd /path/to/openidm
     $ cp -pr conf /path/to/my-project/
     $ cp -pr script /path/to/my-project/
     ```

     You can then customize the basic configuration according to your requirements.

3. Start your new project as follows:

   ```
   $ cd /path/to/openidm
   $ ./startup.sh -p /path/to/my-project
   ```

Chapter 2
Selecting a Repository

By default, IDM uses an embedded ForgeRock Directory Services (DS) instance for its internal repository. This means that you do not need to install a database in order to evaluate the software. Before using IDM in production, however, you must replace the embedded DS repository with a supported repository.

In production environments, the following repositories are supported:

External DS instance

See "Using an External DS Repository".

> **Important**
>
> Both the default embedded and the external DS repositories do *not* support storage of audit or workflow data. Audit logging to the repository is disabled by default. Do not enable logging to the repository if you are using a DS repository.

MySQL

See "Setting Up a MySQL Repository".

MariaDB

The instructions in "Setting Up a MySQL Repository" work equally well for MariaDB.

Microsoft SQL

See "Setting Up a Microsoft SQL Repository".

PostgreSQL

See "Setting Up a PostgreSQL Repository".

Oracle Database (Oracle DB)

See "Setting Up an Oracle DB Repository".

IBM DB2 Database

See "Setting Up an IBM DB2 Repository".

For supported versions, see "Supported Repositories" in the *Release Notes*.

This chapter describes how to set up IDM to work with each of these supported repositories, and lists the minimum rights required for database installation and operation.

For information about the repository configuration, and how to map IDM objects to database tables or to DS LDAP objects, see "*Managing the Repository*" in the *Integrator's Guide*.

2.1. Using the Default DS Repository

By default, IDM uses the `conf/repo.ds.json` file to start an embedded DS instance. The embedded DS repository is not supported in production environments.

The embedded DS server has the following configuration by default:

- `hostname` - `localhost`
- `ldapPort` - `31389`
- `bindDN` - `cn=Directory Manager`
- `bindPassword` - `password`
- `adminPort` - `34444`

To change the administrative port of the embedded DS server, add an `adminPort` property to your project's `conf/repo.ds.json` file before you start IDM. To change any of the other default values, add an `ldapConnectionFactories` property, as shown in the following example.

This excerpt of a `repo.ds.json` sets the administrative port to `4444`. The example changes the bind password to `MyPassw0rd` but shows the structure of the entire `ldapConnectionFactories` property for reference:

```
{
  "embedded": true,
  "maxConnectionAttempts" : 5,
  "adminPort": 4444,
  "ldapConnectionFactories": {
    "bind": {
      "primaryLdapServers": [{ "hostname": "localhost", "port": 31389 }]
    },
    "root": {
      "authentication": {
        "simple": { "bindDn": "cn=Directory Manager", "bindPassword": "MyPassw0rd" }
      }
    }
  },
  "queries": {
...
```

It is not necessary to add the entire `ldapConnectionFactories` block to your configuration file but you must respect the JSON structure. For example, to change only the `hostname`, you would need to add at least the following:

```
{
  ...
  "ldapConnectionFactories": {
    "bind": {
      "primaryLdapServers": [{ "hostname": "my-hostname" }]
    }
  },
  "queries": {
...
```

If you do not specify a connection property here, IDM assumes the default.

You can also configure an external DS instance as a repository. For more information, see "Using an External DS Repository".

Note

If you are running Red Hat Enterprise Linux 6 or an AWS-based Ubuntu 16.04 system and do not have your own public key certificate, include the hostname of your system in your `/etc/hosts` file. Otherwise, an attempt to start IDM will fail with an `UnknownHostException` error.

2.2. Using an External DS Repository

IDM supports the use of a single external DS instance as a repository. You can use a replicated instance for backup purposes, but using multiple replicated instances (in a multimaster DS deployment) is not supported.

To configure a DS instance as an external IDM repository, follow these steps:

1. If you have not yet installed DS, download it from the ForgeRock BackStage download site and extract the zip archive.
2. Install DS with the `idm-repo` profile, as described in the DS Installation Guide.

   This step configures DS on the localhost, listening on ports `31389` and `34444` so that it does not conflict with the default ports used in the LDAP samples. You can use any hostname and available ports in the setup. If you use a different host and an LDAP port other than `31389`, change the `primaryLdapServers` property in your `repo.ds-external.json` file accordingly.
3. In your IDM installation, remove the default DS repository configuration file (`repo.ds.json`) from your project's `conf/` directory. For example:

   ```
   $ cd /path/to/openidm/my-project/conf/
   $ rm repo.ds.json
   ```
4. Copy the external DS repository configuration file (`repo.ds-external.json`) to your project's `conf` directory and rename it `repo.ds.json`:

   ```
   $ cd /path/to/openidm/
   $ cp db/ds/conf/repo.ds-external.json my-project/conf/repo.ds.json
   ```

5. If your DS instance is *not* running on the localhost and listening for LDAP connections on port `31389`, adjust the `primaryLdapServers` property in that file to match your DS setup.

6. Start IDM with the configuration for your project. For example:

```
$ cd /path/to/openidm
 $ ./startup.sh -p my-project
 Executing ./startup.sh...
 Using OPENIDM_HOME:   /path/to/openidm
 Using PROJECT_HOME:   /path/to/my-project
 Using OPENIDM_OPTS:   -Xmx1024m -Xms1024m
 Using LOGGING_CONFIG: -Djava.util.logging.config.file=/path/to/my-project/conf/logging.properties
 -> OpenIDM version "6.5.2.0"
 OpenIDM ready
```

2.3. Database Access Rights For a JDBC Repository

In general, IDM 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 your deployment.

The JDBC repository used by IDM 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 `db/database-type/scripts/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, by the scripts that create the tables specific to the Activiti workflow engine, and by the script that sets up the audit tables if you are using the repository audit event handler.

2.4. Configuring Case Insensitivity For a JDBC Repository

A DS repository is case-insensitive by default. The supported JDBC repositories are generally case-sensitive by default. Case-sensitivity can cause issues if queries expect results to be returned, regardless of case.

For example, with the default configuration of a MySQL database, a search for an email address of `scarter@example.com` might return a results, while a search for `scarter@EXAMPLE.COM` might return an `Unable to find account` error.

If you need to support case-insensitive queries, you must configure a case-insensitive collation in your JDBC repository, on the specific columns that require it.

For example, for a generic managed object mapping in MySQL or MariaDB, change the default collation of the `managedobjectproperties.propvalue` column to `utf8_general_ci`. Note that this changes case-

sensitivity for *all* managed object properties. To change case-sensitivity for all the properties of a specific object, specify a different table for the `propertiesTable` entry in your `repo.jdbc.json` for that object, and adjust the collation on that table. To change case-sensitivity only for certain properties of an object, use an explicit mapping.

For a PostgreSQL repository, use an explicit table structure if you require case-insensitivity. Managing case-insensitivity at scale with generic tables in PostgreSQL is not supported. For more information about generic and explicit object mappings, see "Generic and Explicit Mappings With a JDBC Repository" in the *Integrator's Guide*.

To set the collation for an Oracle DB repository, see the corresponding Oracle documentation.

To set the collation for a SQL Server repository, see the corresponding Windows documentation.

For a DB2 repository, see the corresponding DB2 documentation.

2.5. Setting Up a MySQL Repository

After you have installed MySQL on the local host and *before starting IDM for the first time*, configure the server 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 IDM is stopped:

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

3. Remove the default DS repository configuration file (`repo.ds.json`) from your project's `conf/` directory. For example:

   ```
   $ cd /path/to/openidm/my-project/conf/
   $ rm repo.ds.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. If you have previously set up a MySQL repository for IDM, you *must* drop the openidm database and users before you continue:

```
mysql> drop database openidm;
Query OK, 21 rows affected (0.63 sec)
mysql> drop user openidm;
Query OK, 0 rows affected (0.02 sec)
mysql> drop user openidm@localhost;
Query OK, 0 rows affected (0.00 sec)
```

6. Import the IDM data definition language script into MySQL:

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

Note

If you see errors like `Access denied for user 'root'@'localhost'`, and are deploying on a *new* installation of Ubuntu 16.04 and above, the UNIX_SOCKET plugin may be installed, which applies Linux `root` credentials to MySQL. In that case, substitute **sudo mysql -u root** for **mysql -u root -p** in the commands in this section.

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;

+---------------------------+
| Tables_in_openidm         |
+---------------------------+
| clusteredrecontargetids   |
| clusterobjectproperties   |
| clusterobjects            |
| configobjectproperties    |
| configobjects             |
| genericobjectproperties   |
| genericobjects            |
| ...                       |
| schedulerobjects          |
| schedulerobjectproperties |
| uinotification            |
| updateobjectproperties    |
| updateobjects             |
+---------------------------+
```

Exit the mysql console.

```
mysql> exit
Bye
```

7. Create the IDM database user.

   If you are running MySQL 5.7 or higher, run the following script:

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

   If you are running a MySQL version prior to 5.7, run the following script:

   ```
   $ cd /path/to/mysql
   $ mysql -u root -p < /path/to/openidm/db/mysql/scripts/createuser.mysql56.sql
   Enter password:
   ```

8. 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:
   ```

9. If you are planning to direct audit logs to this repository, run the script that sets up the audit tables:

   ```
   $ mysql -D openidm -u root -p < openidm/db/mysql/scripts/audit.sql
   Enter password:
   ```

10. 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 `resolver/boot.properties` or your project's `conf/system.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 -p /path/to/openidm/my-project
  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/resolver/boot
  .properties
  -> OpenIDM version "6.5.2.0"
  OpenIDM ready
  ```

Tip

In a production environment, configure a secure connection to the repository.

When you have set up MySQL for use as the internal repository, start the server to check that the setup has been successful. After startup, you should see that `repo.jdbc` is `active`, whereas `repo.ds` is `enabled` but not `active`:

```
$ cd /path/to/openidm
  $ ./startup.sh -p my-project
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/resolver/boot.properties
-> scr list
BundleId Component Name Default State
    Component Id State      PIDs (Factory PID)
 [   5]   org.forgerock.openidm.config.enhanced.starter  enabled
    [   1] [active      ] org.forgerock.openidm.config.enhanced.starter
 [   5]   org.forgerock.openidm.config.manage  enabled
    [   0] [active      ] org.forgerock.openidm.config.manage
 [  10]   org.forgerock.openidm.datasource.jdbc  enabled
 [  10]   org.forgerock.openidm.repo.jdbc  enabled
    [  48] [active      ] org.forgerock.openidm.repo.jdbc
 [  11]   org.forgerock.openidm.repo.ds
 enabled
...
```

2.6. Setting Up a Microsoft SQL Repository

These instructions are specific to Microsoft 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, pay attention to 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). IDM *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 Microsoft SQL).

5. Click OK.

6. Restart Microsoft 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 IDM to access Microsoft SQL.

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

1. Use SQL Management Studio to import the IDM data definition language script into Microsoft SQL:

   a. Navigate to SQL Server Management Studio.

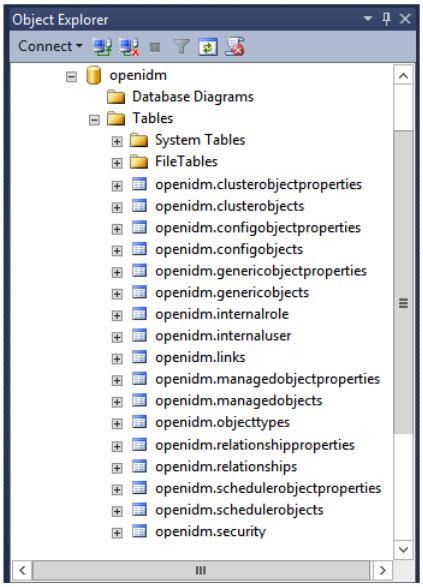
   b. On the Connect to Server panel, select Windows Authentication and click Connect.

   c. Select File > Open > File and navigate to the 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 IDM tables in the openidm database, as shown in the following example.



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 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. If you are going to direct audit logs to this repository, run the script that sets up the audit tables:

```
PS C:\Users\Administrator> sqlcmd -S localhost -d openidm ^
 -i C:\path\to\openidm\db\mssql\scripts\audit.sql
```

5. Download the Microsoft JDBC Drivers for SQL Server:

a. Download the JDBC Drivers from Microsoft's download site. IDM requires at least version 7.2 of the driver, which supports OSGi by default.

b. Extract the driver JAR files using 7-zip or an equivalent file management application.

c. Copy the JAR file that corresponds to your Java environment to the `\path\to\openidm\bundle` directory. For example:

```
copy mssql-jdbc-7.4.1.jre8.jar \path\to\openidm\bundle
```

6. Download the JDBC OSGi Service Package JAR and place it in the `\path\to\openidm\bundle` directory:

   IDM was tested with version 1.0.0 of the service package.

7. Remove the default DS repository configuration file (`repo.ds.json`) from your project's `conf/` directory. For example:

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

8. Copy the database connection configuration file for Microsoft 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\
```

9. Update the connection configuration to reflect your Microsoft 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 `resolver/boot.properties` or your project's `conf/system.properties` file, for example:

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

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

```
set:OPENIDM_OPTS="-Xmx1024m -Xms1024m -Dopenidm.repo.host=localhost -Dopenidm.repo.port=1433"
```

After you have set up Microsoft SQL Server as the repository, make sure that IDM starts without errors.

2.7. Setting Up an Oracle DB Repository

Before you set up Oracle DB as the IDM repository, confer with your Oracle DBA to create the database schema, tables, and users. This section assumes that you have configured an Oracle DB with *Local Naming Parameters (tnsnames.ora)* and a service user for IDM.

Important

IDM supports two connection pools for an Oracle DB:

- Hikari Connection Pool (HikariCP), described in the HikariCP GitHub Repository
- Oracle Universal Connection Pool (Oracle UCP), described in the Universal Connection Pool for JDBC Developer's Guide

Many steps in this procedure will depend on your connection pool type.

To Set Up Oracle as an IDM Repository

1. As the appropriate schema owner, import the IDM schema using the data definition language script (`/path/to/openidm/db/oracle/scripts/openidm.sql`).
2. Run the scripts that set up the tables required by the Activiti workflow engine.

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

   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. If you are planning to direct audit logs to this repository, run the script that sets up audit tables.

   Use the Oracle SQL Developer Data Modeler to run the following script:

   ```
   /path/to/openidm/db/oracle/scripts/audit.sql
   ```

4. Set the host and port of the Oracle DB instance, either in the `resolver/boot.properties` file or through the `OPENIDM_OPTS` environment variable.

- If you use the `resolver/boot.properties` file, set values for the following variables:
  - `openidm.repo.host = localhost`
  - `openidm.repo.port = 1521`
- If you use the `OPENIDM_OPTS` environment variable, include the JVM memory options when you set the repo host and port. For example:

```
export OPENIDM_OPTS="-Xmx1024m -Xms1024m -Dopenidm.repo.host=localhost -Dopenidm.repo.port=1521"
```

5. Remove the default DS repository configuration file (`repo.ds.json`) from your project's `conf/` directory. For example:

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

6. Copy the Oracle DB repository configuration file (`repo.jdbc.json`) to your project's configuration directory:

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

7. **For OracleUCP only**

   Edit the `repo.jdbc.json` file as follows:

```
{
    "dbType" : "ORACLE",
    "useDataSource" : "ucp-oracle",
    ...
}
```

8. Copy the connection configuration file to your project's configuration directory and edit the file for your Oracle DB deployment. The connection configuration file depends on the connection pool that you use:

   For Hikari CP

   Copy the following file:

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

   Edit the file to reflect your deployment. The default configuration for a HikariCP connection pool 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
    }
}
```

The `jdbcUrl` corresponds to the URL of the Oracle DB listener, including the service name, based on your configured Local Naming Parameters `tnsnames.ora`. Set this parameter according to your database environment.

The `DEFAULTCATALOG` refers to the SID (system identifier), for example, `orcl`.

The `username` and `password` correspond to the credentials of the service user that connects from IDM.

For Oracle UCP

Copy the following file:

```
cp /path/to/openidm/db/oracle/conf/datasource.jdbc-ucp-oracle.json my-project/conf/
```

Edit the file to reflect your deployment. The default connection configuration for an Oracle UCP connection pool is as follows:

```
{
    "databaseName" : "openidm",
    "jsonDataSource" : {
        "class" : "oracle.ucp.jdbc.PoolDataSourceImpl",
        "settings" : {
            "connectionFactoryClassName" : "oracle.jdbc.pool.OracleDataSource",
            "url" : "jdbc:oracle:thin:@&{openidm.repo.host}:&{openidm.repo.port}:SID",
            "user" : "openidm",
            "password" : "openidm",
            "connectionTimeout" : "30000",
            "minPoolSize" : 20,
            "maxPoolSize" : 50
        }
    }
}
```

The `url` corresponds to the URL of the Oracle DB listener, including the service ID (`SID`), based on your configured Local Naming Parameters `tnsnames.ora`. Set this property to the appropriate value for your environment, for example: `jdbc:oracle:thin:@localhost:1521:orcl`.

The `user` and `password` correspond to the credentials of the service user that connects from IDM.

9. Create an OSGi bundle for the Oracle DB driver, as follows:

   a. Download the JDBC drivers for your Oracle DB version.

      The files that you download depend on your Oracle DB version, and on whether you are using HikariCP or Oracle UCP. Because the version numbers change with minor updates, you must search for the precise corresponding files on `oracle.com`:

      - Download the `ojdbc*.jar` file that corresponds to your Oracle DB version.
      - Download bnd-2.4.0.jar. This file lets you create OSGi bundles. For more information about `bnd`, see http://bnd.bndtools.org/.
      - **For OracleUCP only**

        Download the following files:

        `ucp.jar`
        `ons.jar`

      Copy the downloaded files to the `/path/to/openidm/db/oracle/scripts` directory.

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

      You can use the sample `bnd` file located in `openidm/db/mssql/scripts`. Copy that file to the directory with the JDBC driver, and rename it `ojdbc8.bnd`:

      ```
      cd /path/to/openidm/db
      cp mssql/scripts/sqljdbc4.bnd oracle/scripts/ojdbc8.bnd
      ```

      Edit the file for your Oracle version. The resulting file should look similar to the following:

      ```
      version=12.2.0.1
      Export-Package: *;version=6.5.2.0
      Bundle-Name: Oracle Database 12.2.0.1 JDBC Driver
      Bundle-SymbolicName: oracle.jdbc.OracleDriver
      Bundle-Version: 6.5.2.0
      Import-Package: *;resolution:=optional
      ```

      Note

      Do not include trailing zeros in the version number. For example, for Oracle 12.2.0.1.0, set the version string to `version=12.2.0.1`.

   c. From the `/path/to/openidm/db/oracle/scripts` directory, run the following command to create the OSGi bundle, replacing the `*` with your Oracle DB driver version:

      ```
      java -jar bnd-2.4.0.jar wrap --properties ojdbc*.bnd --output ojdbc*-osgi.jar ojdbc*.jar
      ```

d. **For OracleUCP only**

Create `bnd` files for the `ucp.jar` and `ons.jar` files. The following examples assume version 12.2.0 Oracle JDBC drivers:

- `ucp.bnd`

```
version=12.2.0
Export-Package: oracle.ucp.*;version=${version}
Bundle-Name: Oracle Universal Connection Pool
Bundle-SymbolicName: oracle.ucp
Bundle-Version: ${version}
Import-Package: *;resolution:=optional
DynamicImport-Package: *
```

- `ons.bnd`

```
version=12.2.0
Export-Package: *;version=${version}
Bundle-Name: Oracle ONS
Bundle-SymbolicName: oracle.ons
Bundle-Version: ${version}
Import-Package: *;resolution:=optional
```

Save the `bnd` files in the `/path/to/openidm/db/oracle/scripts` directory, then run the following commands to create the corresponding OSGi bundles:

```
cd /path/to/openidm/db/oracle/scripts
java -jar bnd-2.4.0.jar wrap --properties ucp.bnd --output ucp-osgi.jar ucp.jar
java -jar bnd-2.4.0.jar wrap --properties ons.bnd --output ons-osgi.jar ons.jar
```

You can ignore any `private references` warnings that are logged when you build these bundles.

e. Move all the OSGi bundle files to the `openidm/bundle` directory.

10. When you have set up Oracle DB for use as the internal repository, make sure that the server starts without errors.

2.8. 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 IDM for the first time, configure the server to use a PostgreSQL repository, as described in the following procedure:

1. The `path/to/openidm/db/postgresql/scripts/createuser.pgsql` script 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. If you plan to direct audit logs to this repository, run the script that sets up the audit tables:

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

7. Remove the default DS repository configuration file (`repo.ds.json`) from your project's `conf/` directory. For example:

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

8. 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/postgresql/conf/datasource.jdbc-default.json my-project/conf/
$ cp db/postgresql/conf/repo.jdbc.json my-project/conf/
```

9. 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 `resolver/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"
$ cd /path/to/openidm
$ ./startup.sh -p my-project
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/resolver/boot
.properties
-> OpenIDM version "6.5.2.0"
OpenIDM ready
```

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

    Start IDM with the configuration for your project. Monitor the console for the success of your setup. After startup, run a `scr list` command. In the output, you should see that `repo.jdbc` is `active`, whereas `repo.ds` is `enabled` but not `active`:

```
$ cd /path/to/openidm
$ ./startup.sh -p my-project
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/resolver/boot.properties
-> scr list
BundleId Component Name Default State
    Component Id State      PIDs (Factory PID)
 [   5]   org.forgerock.openidm.config.enhanced.starter  enabled
    [   1] [active      ] org.forgerock.openidm.config.enhanced.starter
 [   5]   org.forgerock.openidm.config.manage  enabled
    [   0] [active      ] org.forgerock.openidm.config.manage
 [  10]   org.forgerock.openidm.datasource.jdbc  enabled
 [  10]   org.forgerock.openidm.repo.jdbc  enabled
    [  48] [active      ] org.forgerock.openidm.repo.jdbc
 [  11]   org.forgerock.openidm.repo.ds
 enabled
...
```

11. Set up indexes to tune the PostgreSQL repository according to your specific deployment.

Important

No indexes are set by default. If you do not tune the repository correctly by creating the required indexes, the performance of your service can be severely impacted. For example, setting too many indexes can have an adverse effect on performance during managed object creation. Conversely, not indexing fields that are searched will severely impact search performance.

IDM includes a `/path/to/openidm/db/postgresql/scripts/default_schema_optimization.pgsql` script that sets up a number of indexes. This script includes extensive comments on the indexes that are being created. Review the script before you run it to ensure that all the indexes are suitable for your deployment.

When you have refined the script for your deployment, execute the script as a user with superuser privileges, so that the required extensions can be created. By default, this is the `postgres` user:

```
$ 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
```

2.9. 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 Kerberos Authentication With a DB2 Repository".

Before you start, make sure that the server is stopped.

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

Configure IDM to use the DB2 repository, as described in the following steps:

1. Create an OSGi bundle for the DB2 JDBC driver, 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 `bnd` file and edit it to match the version information for your JDBC driver.

   You can use the sample `bnd` file located in `openidm/db/mssql/scripts`. Copy that file to the directory with 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 `bnd` 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` JAR file (bnd-2.4.0.jar) that lets you create OSGi bundles. For more information about `bnd`, see http://bnd.bndtools.org/.

Place the `bnd` JAR file in the same directory as the JDBC driver:

```
$ ls /path/to/openidm/db/db2/scripts
bnd-2.4.0.jar  db2jcc.jar
```

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-2.4.0.jar wrap --properties sqljdbc4.bnd --output db2jcc-osgi.jar db2jcc.jar
```

This command creates an OSGi bundle, as defined by the `--output` option: `db2jcc-osgi.jar`:

```
$ ls
bnd-2.4.0.jar  db2jcc-osgi.jar  db2jcc.jar
```

e. Move the OSGi bundle fle to the `openidm/bundle` directory:

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

2. Remove the default DS repository configuration file (`repo.ds.json`) from your project's `conf/` directory. For example:

```
$ cd /path/to/openidm/my-project/conf/
$ rm repo.ds.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 `resolver/boot.properties` or your project's `conf/system.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"
$ cd /path/to/openidm
$ ./startup.sh -p my-project
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/resolver/boot
.properties
-> OpenIDM version "6.5.2.0"
OpenIDM ready
```

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

```
$ db2 create database dopenidm
```

6. Import the IDM data definition language script 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
------------------------------- --------------- ----- --------------------------
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
...
```

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
```

9. If you plan to direct audit logs to this repository, run the script that sets up the audit tables:

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

When you have set up DB2 for use as the internal repository, start the server to check that the setup has been successful. After startup, you should see that `repo.jdbc` is `active`, whereas `repo.ds` is `enabled` but not `active`:

```
$ cd /path/to/openidm
$ ./startup.sh -p my-project
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/resolver/boot.properties
-> scr list
BundleId Component Name Default State
    Component Id State      PIDs (Factory PID)
 [   5]   org.forgerock.openidm.config.enhanced.starter  enabled
    [   1] [active      ] org.forgerock.openidm.config.enhanced.starter
 [   5]   org.forgerock.openidm.config.manage  enabled
    [   0] [active      ] org.forgerock.openidm.config.manage
 [  10]   org.forgerock.openidm.datasource.jdbc  enabled
 [  10]   org.forgerock.openidm.repo.jdbc  enabled
    [  48] [active      ] org.forgerock.openidm.repo.jdbc
 [  11]   org.forgerock.openidm.repo.ds
 enabled
...
```

2.9.1. Configuring Kerberos Authentication With a DB2 Repository

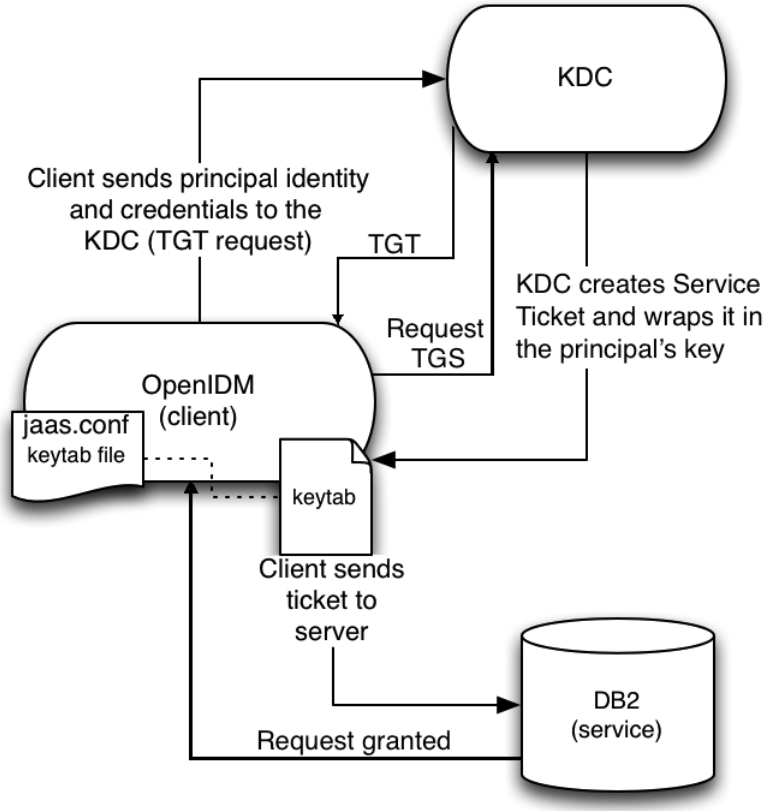
By default, IDM uses the username and password configured in the repository connection configuration file (`conf/datasource.jdbc-default.json`) to connect to the DB2 repository. You can configure IDM to use Kerberos authentication instead.

In this scenario, IDM 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 IDM's `jaas.conf` file.

Using Kerberos to Connect to a DB2 Repository



To configure IDM for Kerberos authentication:

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

   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 IDM. You must create a keytab file on the host that IDM runs on. The keytab file must be secured in the same way that you would secure any password file. Specifically, only the user running IDM 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 IDM `security` directory:

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

   By default, IDM 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 IDM.

Chapter 3

Removing and Moving Server Software

This chapter covers uninstallation of an IDM server.

To Remove IDM

1. (Optional) Stop the server if it is running, as described in "To Stop IDM".
2. Remove the directory where you installed the software:

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

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

Chapter 4

Updating Servers

This chapter describes how to update an existing deployment to IDM 6.5.2.0.

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

The automated update process available with previous IDM versions is no longer supported. This chapter describes the manual process required to update an existing IDM deployment. At a high level, the manual update process involves the following steps:

1. Create a new installation of IDM.
2. Migrate your existing configuration to the new installation. To migrate your configuration, copy your IDM 6.0 configuration files to the new IDM 6.5.2.0 installation, making the changes specified later in this chapter and in "Required Changes to IDM" in the *Release Notes*, then systematically enable each IDM 6.5.2.0 feature that you need.

   Your 6.0 configuration includes any customizations made to the Admin UI, generally with files in the `openidm/ui/admin/extension` directory.

   Note: you cannot copy configuration files directly from a version prior to IDM 6. You must either manually migrate your customizations to the new configuration files, or run the old update process for each major version successively, so from 4 to 4.5 then from 4.5 to 5, and so on. Instructions for updating to each major version are available in the *Installation Guide* for that version.
3. Migrate your existing data to the new installation. To migrate you can either:
   - Use your existing repository, making the changes specified later in this chapter and in "*Updating to IDM 6.5*" in the *Release Notes*.
   - Create a new repository and migrate existing data to it with the new data migration service. For more information about the data migration service, see "Migrating Data from Previous Versions of IDM".

4.1. Preparing Systems for An Update

Take the following steps before you start to update your deployment:

1. Update your Java environment.

IDM requires a supported Java environment, as described in "Preparing the Java Environment" in the *Release Notes*.

If your server uses an older version, install a newer Java version before you update and follow the instructions in "Java Prerequisites".

2. Download the new software.

   Download and extract `IDM-6.5.2.0.zip` from the ForgeRock BackStage download site.

3. Back up your existing deployment by archiving the `openidm` directory and the contents of your repository.

4. If you have encrypted or obfuscated any properties in configuration files, decrypt them into their plain text values before you start the update. When you have completed the update, you can encrypt or obfuscate the values again.

5. Save your audit data. If you want to preserve a record of the audit logs collected while IDM 6.0 was running, save these log files manually before you start the update process. You can find the log files in the `/path/to/openidm/audit/` directory. For more information on these files, see "Audit Event Topics" in the *Integrator's Guide*.

   Tip

   File rotation tip: When you delete files in the `/path/to/openidm/audit/` directory, IDM creates new files as needed in the same directory.

4.2. Migrating Your Existing Server Configuration

Before you start migrating your configuration, be sure to review "*Compatibility*" in the *Release Notes* and "*Updating to IDM 6.5*" in the *Release Notes*. These include important compatibility notes and instructions for updating your IDM installation.

In most cases, the configuration files from your IDM 6.0 installation will work without further modification, and can be migrated directly to your new 6.5.2.0 installation. There are, however, changes in some files that are required for your migration to work correctly. These changes are covered in "*Updating to IDM 6.5*" in the *Release Notes*, and noted where appropriate below.

4.2.1. Migrating Configuration Files

Because there is no automated way to migrate a customized configuration to IDM 6.5.2.0, you will need to migrate these files to your new installation manually. Before migrating your configuration, note the new files added in IDM 6.5.2.0:

- `endpoint-removeRepoPathFromRelationships.json`: Used to modify existing relationship references to no longer include `repo/` in the path. For more information, see "Changes to `repo/internal`" in the *Release Notes*.
- `endpoint-updateInternalUserAndInternalRoleEntries.json`: Used to update internal users and internal roles to accommodate new privileges functionality. For more information, see "Enabling Privileges" in the *Release Notes*.
- `internal.json`: Provides access to relationship resource collections for internal objects. This is primarily related to new privileges functionality. For more information, see "Enabling Privileges" in the *Release Notes*.
- `java.security`: Used by IDM to extend the security properties defined in your JDK's java.security file. For more information, see "Configuring IDM to Support an HSM Provider" in the *Integrator's Guide*.
- `notification-passwordUpdate.json`: Part of IDM's new notification service. For more information, see "Configuring Notifications" in the *Integrator's Guide*.
- `notification-profileUpdate.json`: Part of IDM's new notification service. For more information, see "Configuring Notifications" in the *Integrator's Guide*.
- `notificationFactory.json`: Part of IDM's new notification service. If you are planning to use the old notification system, you should either delete or disable this file, and add your existing `endpoint-usernotifications.json` file. For more information, see "Configuring Notifications" in the *Integrator's Guide*.
- `repo.init.json`: Used by IDM when starting, defining initial internal users and roles. For more information, see "Internal Users" in the *Integrator's Guide* and "Roles and Authentication" in the *Integrator's Guide*.
- `secrets.json`: Used by IDM for managing security and encryption related configuration. This information was previously stored in `boot.properties`. For more information, see "Changes for the New Secrets Service" in the *Release Notes*.
- `ui.context-enduser.json`: Used by the new IDM End User UI. If you intend to use the old End User UI, this file should be removed and replaced by your existing `ui.context-selfservice.json` file.

Also note that two files are no longer included by default in the new `conf/` directory:

- `endpoint-usernotifications.json`: This file was used by the old notification service (IDM versions 6.0 and prior). Do not migrate this file if you are planning to use the new notification service. Any customizations to this file should be adapted to use the new notification service instead.
- `ui.context-selfservice.json`: This file was used by the old End User UI. Do not migrate this file if you are planning to use the new End User UI. Any customizations to this file should be adapted to use the new UI instead.

For all other files found in `conf/`, double-check "*Compatibility*" in the *Release Notes* to ensure any customizations you've made are compatible with changes in IDM 6.5, then copy your old

configuration to your new installation. In some cases, you may find it easier to adapt your customizations to the new version of the file instead, in particular if you plan to use IDM's new features. For example, the following files changed significantly from the previous version of IDM:

- `managed.json`
- `policy.json`
- `repo.ds.json` and `repo.jdbc.json`
- `authorization.json`

While the older versions of these files should still work (as long as you perform any changes noted as required in "Required Changes to IDM" in the *Release Notes* and migrate the scripts noted in "Migrating Custom Scripts"), it may be easier to migrate your customizations to the new version of these files instead.

4.2.2. Migrating `boot.properties`

On the IDM 6.5 server, edit the `resolver/boot.properties` file to match any customizations that you made on your IDM 6 server. Specifically, check the following elements:

- The HTTP, HTTPS, and mutual authentication ports are specified in the `resolver/boot.properties` file. If you changed the default ports in your IDM 6 deployment, make sure that the corresponding ports are specified in this file.
- Security-related configurations (such as secrets and encryption) have been moved to the new `conf/secrets.json` file. Any customizations you've made to these fields should be migrated to use `secrets.json`. For more information about `secrets.json`, see "Accessing IDM Keys and Certificates" in the *Integrator's Guide*. Notes about further changes to your configuration related to the new secrets service can be found in "Changes for the New Secrets Service" in the *Release Notes*.

  Check that the keystore and truststore passwords match the current passwords for the keystore and truststore of your IDM 6 deployment.

Depending on the level of customization you have made in your current deployment, it might be simpler to start with your IDM 6 `boot.properties` file, and copy all customized settings from that file to the corresponding IDM 6.5 file. 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.

4.2.3. Migrating Security Settings

Copy the contents of your IDM 6 `security/` folder to the IDM 6.5 instance. By default, this contains three files:

- `keystore.jceks`

- `realm.properties`
- `truststore`

If you made no changes to `realm.properties`, use the newer version of the file. If you use the older file, adjust `openidm-authorized` to be `internal/role/openidm-authorized`.

Warning

If you do not copy your old truststore and keystore files to your new instance, you will be unable to decrypt anything that was encrypted by your old instance of IDM.

As noted in "Migrating `boot.properties`", security and encryption-related properties previously stored in `boot.properties` are now stored in `conf/secrets.json`. Further notes about changes to your configuration files related to the new secrets service can be found in "Changes for the New Secrets Service" in the *Release Notes*.

4.2.4. Migrating Custom Scripts

Migrate any custom scripts or default scripts *that you have modified* to the `script` directory of your IDM 6.5.2.0 instance. In general, custom and customized scripts should be located in the `script` directory of the existing IDM deployment.

For custom scripts, review "*Compatibility*" in the *Release Notes*. If you're confident that the scripts will work as intended on IDM 6.5.2.0, then copy these scripts to the new instance.

If you modified an existing script, compare the default versions of the IDM 6.0 and IDM 6.5.2.0 scripts. If nothing has changed between the default versions, review your customizations against "*Compatibility*" in the *Release Notes*. If you are confident that your changes will work as intended on the new version, copy the customized scripts to the new `script` directory.

If a default script has changed since the IDM 6 release, you will need to test that your customizations work with the new default script before porting your changes to that new script.

Note

Some scripts that were previously found in `defaults/` are no longer included with IDM. Be sure to check that any scripts you were previously using are still present in your new installation.

Two examples are `onDelete-user-cleanup.js`, which was referenced in the `managed/user` onDelete action; and `userNotifications.js`, which was called by `endpoint-usernotifications.json`.

Special attention should be paid to the changes made in `access.js`, which received a large number of changes in the new version of IDM. If you are planning to use the new features of IDM, you may find it easier to migrate your customizations to the new file, rather than adapt your old access file to the new features.

If you modify any shell scripts, such as `startup.sh`, you must migrate your changes manually to the new version of the script.

4.2.5. Migrating Provisioner Files

Modify any customized provisioner configurations in your existing deployment to point to the connectors that are provided with IDM 6.5.2.0. Specifically, make sure that the `connectorRef` properties reflect the new connector versions, where applicable. For example:

```
"connectorRef" : {
    "bundleName": "org.forgerock.openicf.connectors.ldap-connector",
    "bundleVersion": "[1.5.19.0,1.6.0.0)",
    "connectorName": "org.identityconnectors.ldap.LdapConnector"
},
```

Alternatively, copy the connector .jar files from your existing deployment into the `openidm/connectors` directory of the new installation.

4.2.6. Migrating Custom Workflows

Previously, Activiti workflow templates used JQuery and Handlebars. If your deployment includes existing workflows, you *must* rewrite these to use Vue JS if you want to view them in the new End User UI. The new UI does not support older workflow templates that use JQuery and Handlebars.

To rewrite existing workflows for the new UI, you must have a basic understanding of the Vue JS framework and how to create components. For more information, see the Vue documentation. For an example of a workflow template written for the new UI, see `/path/to/samples/provisioning-with-workflow/workflow/contactorOnboarding.bar`. This archive file includes the workflow definition (`contactorOnboarding.bpmn20.xml`) and the corresponding JavaScript template (`contractorForm.js`) to render the workflow in the new UI.

If you previously generated your workflows with a bpmn file (and never created custom JavaScript files), the new UI will just generate these as before and you will not have to convert them.

"In progress" workflows must be compatible with IDM 6.5, or must be rewritten before you upgrade to IDM 6.5. You will not be able to complete incompatible "in progress" workflows on an upgraded system.

4.3. Updating the IDM Repository

After migrating your configuration files to your new IDM installation, You will need to handle the data stored in your IDM repository. There are two options: upgrade your existing repository, or create a new repository.

Once the repository has been updated or created and populated, complete the IDM 6.5.2.0 installation, as described in "*Preparing to Install and Run Servers*". Your new IDM instance should be ready for testing to ensure all scripts and functionality are working as intended.

4.3.1. Upgrade Your Existing Repository

Copy or connect to your existing repository, making updates as needed for newer features and functionality. This has the benefit of not needing any data migration, but does require running a series of provided scripts to modify the repository, in order to make use of any of the new capabilities of IDM.

If you intend to use your existing repository with your new IDM instance, please review the steps found in "*Updating to IDM 6.5*" in the *Release Notes*, following all required instructions, as well as any recommended or feature-specific instructions appropriate for your deployment.

There are a few steps you will need to take when preparing your existing repository for use with the new version of IDM:

1. Clear all `configobjects` related tables. For example, in MySQL run:

```
DELETE FROM openidm.configobjects;
DELETE FROM openidm.configobjectproperties;
```

2. Run each of the schema update scripts and make any configuration modifications identified in "Required Changes to IDM" in the *Release Notes*.

Important

- When you run the `alter_uinotification.sql` update script, you might see an error similar to the following:

```
Column createDate to be modified to NOT NULL is already NOT NULL
```

  You can safely ignore this error.

- If you are using a managed relational database service such as Amazon RDS, be aware that some update scripts might require root level access to the system tables in the underlying database.

  Specifically, certain PostgreSQL update scripts require access to the `pg_attribute` table. Because the database service master user is not the same as the PostgreSQL root user, such scripts might fail with a permissions error. In this case, investigate the failing script, and use an `ALTER TABLE` command on the specific IDM table instead.

3. Run each of the schema update scripts and configuration modifications identified in "Enabling New Features in IDM" in the *Release Notes* for the new features you wish to enable.

4. Launch IDM and run the following Groovy script to clear the `reconprogressstate` data in your repository:

```
def result = openidm.query(
  "repo/reconprogressstate", [ "_queryFilter" : "true", "_fields" : "_id" ]).result;
for ( item in result ) {
  openidm.delete("repo/reconprogressstate/" + item["_id"], null);
}
return result.size() + " reconprogressstate records deleted";
```

This script will work regardless of the type of repository, and can be sent as a REST call. For example:

```
curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --header "Content-Type: application/json" \
 --request POST \
 --data '{
   "type":"groovy",
   "source":"def result = openidm.query(\"repo/reconprogressstate\", [ \"_queryFilter\" : \"true\",
 \"_fields\" : \"_id\" ]).result; for ( item in result ) { openidm.delete(\"repo/reconprogressstate/\"
 + item[\"_id\"], null); }; return result.size() + \" reconprogressstate records deleted\";"
 }' \
 "http://localhost:8080/openidm/script?_action=eval"
"1 reconprogressstate records deleted"
```

5. Verify all scripts and functions behave as expected.

Note

For particularly large repositories, there are additional optimizations recommended to improve performance during your upgrade. When running the `removeRepoPathFromRelationships` endpoint:

- If you are using PostgreSQL, creating an index of `_id` in the `openidm.relationships` table can help improve performance. For example:

  ```
  create unique index on openidm.relationships(json_extract_path_text(fullobject,
   VARIADIC ARRAY['_id'::text]));
  ```

  Once your migration is complete and you have performed any other updates necessary for your installation, this index can be safely removed.

- Increase `_pageSize` for the `removeRepoPathFromRelationships` endpoint. This value can be found in `bin/defaults/script/update/removeRepoPathFromRelationships.js`. The default is set to 1000 records per page, but can be increased depending on the resources for your installation.

4.3.2. Create a New Repository

Set up a new repository, following the steps found in "*Selecting a Repository*". This has the benefit of being already configured for all the new capabilities in IDM, but does require migrating your existing data to the new repository.

If you intend to use a new repository with your new IDM instance, please see "Migrating Data from Previous Versions of IDM" for more information. Please note, if you choose to create a new repository, you will still need to update your configuration files to effectively make use of the new features listed in "*Updating to IDM 6.5*" in the *Release Notes*.

4.4. Migrating Data from Previous Versions of IDM

IDM 6.5 includes a data migration service to help move information stored in IDM to a new deployment. This service is off by default. To enable it, copy `migration.json` from `samples/example-configurations/conf/` into your `conf/` directory.

Migration is run from your new installation, using your previous deployment as a data source. The data migration service supports importing information from IDM instances back to version 4. If you are migrating from a version of IDM earlier than that, you will need to follow previous update instructions to get your deployment into a state where it can be migrated using this service.

Note

Because the migration service migrates information that may be encrypted, such as passwords, it is important to make sure you have copied the `truststore` and `keystore` files from your previous deployment *prior* to starting the migration.

By default, the data migration service will import:

- Internal Roles
- Internal Users
- Internal User Metadata
- Managed Roles
- Managed Users
- Managed Assignments
- Links and Relationships
- Scheduler jobs

Note

If you are migrating scheduler jobs from IDM 4.0 or 4.5, you will need to modify the entry in `migration.json` to be:

```
{
    "source" : "scheduler",
    "target" : "scheduler/job"
}
```

If you have added additional object types (for example, managed devices), modify `migration.json` to include these objects.

4.4.1. Configuring the Migration service

The data migration service is configured through `migration.json`, and comes preconfigured for migrating the information that IDM stores in a default installation. If you have made additional customizations, you may need to modify this file to include your custom data. Several properties are available to help with migration:

connection

Connection provides configuration information for the source IDM instance you are migrating from. Available properties:

instanceUrl

The URI for the source IDM instance.

userName

Used for authenticating on the source IDM instance.

password

Used for authenticating on the source IDM instance.

socketTimeout

The TCP socket timeout, when waiting for HTTP responses. If you do not set a time duration, the default is 10 seconds.

Example valid time duration values:

- 4 days
- 59 minutes and 1 millisecond
- 1 minute and 10 seconds
- 42 millis
- unlimited
- none
- zero

mappings

A list of mappings you wish to migrate from your old IDM instance to your new deployment. Each mapping can contain:

source

This is the only property that is *required* for data migration. The source should be the path to the resource within the repo; for example, `repo/managed/user`.

target

The path to the resource within the target repository. By default, this will be the same as the source path.

runTargetPhase

Specifies whether the migration should run the target phase of reconciliation. By default, this is set to false.

reconSourceQueryPaging

Specifies whether the migration service should use paging when querying the source installation. By default, this is set to false. If you have a large data set and are concerned about memory usage, you may wish to turn paging on.

reconSourceQueryPageSize

Specifies the number of results to return per page, if paging is turned on. By default, 1000 results per page are returned.

allowEmptySourceSet

Specifies whether the migration service should continue if it encounters an empty source mapping. This is enabled by default.

properties

An array of properties you wish to perform additional actions on, such as modifying the contents of a property during the migration. (This follows the pattern you would find in a standard reconciliation. For more information about transforming data during a reconciliation, see"Transforming Attributes in a Mapping" in the *Integrator's Guide*.)

policies

An array of policies you wish to apply to the data being migrated.

onCreate

The script used by the migration service for creating the data that is being migrated in the new installation. By default, this points to a Groovy script: `update/mapLegacyObject.groovy`.

onUpdate

The script used by the migration service for updating the data that is being migrated in the new installation. By default, this points to a Groovy script: `update/mapLegacyObject.groovy`.

onDelete

If you wish to also delete data during your migration, you can specify an onDelete script with this property. By default, this property is empty.

correlationQuery

You can specify a custom correlation query. By default, this is:

```
"var map = {'_queryFilter': '_id eq \"' + source._id + '\"'}; map;"
```

For more information about writing correlation queries, see "Correlating Source Objects With Existing Target Objects" in the *Integrator's Guide*.

validSource

You can specify a script to validate the source object prior to migration. By default, this property is empty.

endpoint

By default, the migration service endpoint is `migration`. You can use the `endpoint` property to change this if needed.

Note

If your IDM repository is large, you may be able to improve migration performance by turning on paging (using `reconSourceQueryPaging`), and increasing the query page size using `reconSourceQueryPageSize` in your `migration.json` file. This defaults to 1000 records per page. The most effective page size will vary depending on the resources available to your installation.

Since the data migration service is performing a reconciliation between your old installation and your new installation, the types of optimizations used in other types of reconciliations should also be effective with the data migration service. For more information about reconciliation optimization, see "Optimizing Reconciliation Performance" in the *Integrator's Guide*.

4.4.2. Running Your Migration

Before you run your migration, ensure that you have done the following:

- Paused any scheduled jobs on the source deployment
- Configured your `conf/migration.json` and `update/mapLegacyObject.groovy` files on the new IDM installation
- Moved your config files from the old deployment to the new one

When you launch the new IDM installation, a new `migration` endpoint should be available. This endpoint supports the following actions:

- `migrate`: Triggers a migration of all legacy objects from the remote system. Optionally takes a mapping parameter in order to specify a specific mapping to migrate. For example:

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request POST \
 "http://localhost:8080/openidm/migration?_action=migrate&mapping=repoManagedUser_repoManagedUser"
```

- `status`: Returns the last status for all reconciliations triggered via the migration service.
- `mappingConfigurations`: Returns the full list of migration mapping configurations.
- `mappingNames`: Returns the list of migration mapping names.

The period of time a migration takes will depend on the amount of information being migrated. Migrated data will retain the same object IDs they had in the previous deployment.

4.5. Updating a Clustered Deployment

Follow these general steps when you are updating servers in a cluster:

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

4.6. Updating UI Customizations

Note

The End User UI has been completely rewritten, using a new framework. This greatly improves customization and maintainability, but does mean you will need to adapt your old UI customizations by hand to the new system.

If you have a custom Admin UI, save any custom files from the `openidm/ui/admin/extension` subdirectory.

1. Delete the existing `openidm/ui/admin/extension` subdirectory.
2. Copy the default UI files from the `openidm/ui/admin/default` subdirectory with the following command:

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

3. Review any custom UI files from your IDM 6.0 deployment and compare them against the IDM 6.5.2.0 version of these files.

4. Apply custom changes to each new IDM 6.5 UI file in the `openidm/ui/admin/extension` and subdirectory.

4.7. Updating to IDM 6.5.2.0

In the following sections, run these procedures:

- "Preparing Your 6.5.2.0 Update".
- "Updating to Version 6.5.2.0".

You can run an update to IDM 6.5.2.0 from IDM version 6.5.0 using the command-line interface script, `cli.sh`. If you are updating from an IDM deployment prior to 6.5.0, use the manual migration steps in "Migrating Your Existing Server Configuration".

Important

The automated update process is not supported on Windows platforms.

Preparing Your 6.5.2.0 Update

1. Download `IDM-6.5.2.0.zip` from ForgeRock's BackStage site.

2. Copy `IDM-6.5.2.0.zip` to `/path/to/openidm/bin/update`.

3. Back up your 6.5 deployment. Save any customized `*.json` configuration files, located in your project's `/conf` directory. 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 IDM with AM, disable the authentication modules that you used.

6. If you have a custom `resolver/boot.properties` file in your IDM 6.5 deployment, you will need to copy the file to your 6.5.2.0 deployment. For example, if you have an IDM 6.5 deployment, extract `IDM-6.5.2.0.zip` to a temporary directory. Then, copy the custom IDM 6.5 `resolver/boot.properties` file to your 6.5.2.0 `resolver` directory.

   If you have not customized the default `resolver/boot.properties` file in your deployment, you can simply overwrite it with version 6.5.2.0 of this file.

7. Start IDM. IDM must be running when you launch an update.

Note

You must use the CLI to update your system. As of IDM 6.5, the facility to update servers through the Admin UI has been removed.

8. Disable and re-enable connections to AM.

Updating to Version 6.5.2.0

1. Go to your IDM installation:

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

2. Run the **cli.sh** command with the **--skipRepoUpdatePreview** option. There are no repository updates during a patch release:

```
$ ./cli.sh update \
  --acceptLicense \
  --user openidm-admin:openidm-admin \
  --url http://localhost:8080/openidm \
  --skipRepoUpdatePreview \
  IDM-6.5.2.0.zip
```

3. The update process continues and completes with "Scheduler has been resumed."

```
Executing ./cli.sh...
Starting shell in /Users/namespace/Downloads/openidm
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-6.5.2.0.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...
Update procedure is still processing...
The update process is complete.
Exiting maintenance mode...
No longer in maintenance mode.
Resuming the job scheduler.
Scheduler has been resumed..
```

4. After update, check if you have multiple versions of the `bundle/openidm-repo-opendj-<version>` files, for example, `bundle/openidm-repo-opendj-6.5.0.x.jar`. Manually remove the oldest version, which should be the file generated from the release before the 6.5.2.0 update.

5. Manually update the `conf/ui-themeconfig.json`. Change the Bootstrap version to `css/bootstrap-3.4.1-custom.css` from the previous 3.3.7 version.

   The update does not touch this file, to avoid overwriting possible customization of the UI theme.

6. If you have a custom Jetty configuration, see the Release Notes for an important change in functionality.

7. If you have customized your version 6 deployment, you might find files with the following extensions: `.old` and `.new`.

   On Linux/UNIX systems, you can locate 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 your configuration before starting this update process. Files with the `.new-unix_time` extension are files from the IDM 6 configuration that have not been included in your updated installation.

   For example, if you find a `system.properties.new-unix_time` file in your project directory, IDM is still using the version of this file before the update (which would still be named `system.properties`).

   To make sure that you have a completely upgraded configuration, analyze the new features in any files with the `.new-unix_time` extension, and copy those changes into your existing configuration. If you have similar files with multiple `.new-unix_time` extensions, use the file with the latest *unix_time*.

8. Restart your server.

   Your update has successfully completed.

4.8. Placing a Server in Maintenance Mode

The Maintenance Service disables non-essential services of a running IDM 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 \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request POST \
 "http://localhost:8080/openidm/maintenance?_action=enable"
{
  "maintenanceEnabled": true
}
```

To disable maintenance mode, run the following command:

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request POST \
 "http://localhost:8080/openidm/maintenance?_action=disable"
{
  "maintenanceEnabled": false
}
```

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

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request POST \
 "http://localhost:8080/openidm/maintenance?_action=status"
{
  "maintenanceEnabled": false
}
```

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

4.9. Delete Orphaned Meta Entries

Due to an issue in IDM 6.0, if you used a PUT request to update an existing managed object that included metadata, the request would create an additional, orphaned meta object before the create failed. This might have resulted in a number of orphaned meta objects in your deployment.

After you have migrated your data, delete these orphaned meta objects as follows:

1. Update the following script with the credentials of your IDM administrative user and host system:

   ```
   /path/to/openidm/bin/update/scripts/remove-orphan-meta.sh
   ```

2. Start IDM, if it is not running, then launch the script.

4.10. Applying Patch Bundle Releases

ForgeRock issues periodic patch bundle releases containing bug fixes and improvements to IDM. You can view the available list of patch bundles and download them from the ForgeRock BackStage site. When a patch bundle is available, you can view the key fixes provided in that patch bundle release, in the Release Notes.

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 IDM 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 IDM binary in the `/idm` directory, and give user `idm` ownership of all files in that directory:

```
$ sudo unzip /idm/IDM-6.5.2.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 IDM 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 "*Directing Audit Information To a MySQL Database*" 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
   ```

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

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

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

```
{
   "auditServiceConfig" : {
      "handlerForQueries" : "json",
      "availableAuditEventHandlers" : [
         "org.forgerock.audit.handlers.csv.CsvAuditEventHandler",
         "org.forgerock.audit.handlers.elasticsearch.ElasticsearchAuditEventHandler",
         "org.forgerock.audit.handlers.jms.JmsAuditEventHandler",
         "org.forgerock.audit.handlers.json.JsonAuditEventHandler",
         "org.forgerock.audit.handlers.json.stdout.JsonStdoutAuditEventHandler",
         "org.forgerock.openidm.audit.impl.RepositoryAuditEventHandler",
         "org.forgerock.openidm.audit.impl.RouterAuditEventHandler",
         "org.forgerock.audit.handlers.splunk.SplunkAuditEventHandler",
         "org.forgerock.audit.handlers.syslog.SyslogAuditEventHandler"
      ]
     ...
   },
   "eventHandlers" : [
      {
         "class" : "org.forgerock.audit.handlers.json.JsonAuditEventHandler",
         "config" : {
            "name" : "json",
            "logDirectory" : "/var/log/openidm/audit",
            "buffering" : {
                "maxSize" : 100000,
                "writeInterval" : "100 millis"
            },
         "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 and redirect the `org.osgi.framework.storage` property as follows:

   ```
   # 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|&{user.dir}}/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 `idm.data.dir`

A.3. Additional Details

In a production environment, you must configure a supported repository, as described in "*Selecting a Repository*".

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 and Hardening Servers*" in the *Integrator's Guide*.

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

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

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

Stop IDM:

```
-> 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 IDM, configured on a secure read-only volume.

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

IDM 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.
JCE	Java Cryptographic Extension, which is part of the Java Cryptography Architecture, provides a framework for encryption, key generation, and digital signatures.
JSON	JavaScript Object Notation, a lightweight data interchange format based on a subset of JavaScript syntax. For more information, see the JSON site.

JSON Pointer	A JSON Pointer defines a string syntax for identifying a specific value within a JSON document. For information about JSON Pointer syntax, see the JSON Pointer RFC.
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 IDM, the JWT is associated with the `JWT_SESSION` authentication module.
managed object	An object that represents the identity-related data managed by IDM. Managed objects are configurable, JSON-based data structures that IDM 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 more information, see What is OSGi? Currently, only the Apache Felix container 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	IDM distinguishes between two distinct role types - 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 IDM scans before attempting to find a corresponding object on the target system. Depending on the defined mapping, IDM 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 IDM for the period during which IDM 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 IDM scans after locating its corresponding object on the source system. Depending on the defined mapping, IDM then adjusts the target object to match the corresponding source object.