FORGEROCK®

Installation Guide

/ ForgeRock Identity Management 5.5

Latest update: 5.5.1.3

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

Copyright © 2011-2018 ForgeRock AS.

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. Formatting Conventions

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

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

Command-line, terminal sessions are formatted as follows:

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

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

Program listings are formatted as follows:

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

3. Accessing Documentation Online

ForgeRock publishes comprehensive documentation online:

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

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

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

4. Using the ForgeRock.org Site

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

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

Chapter 1
Preparing to Install and Run Servers

This chapter covers the tasks required to prepare, install and start 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 site. The ForgeRock BackStage site provides access to ForgeRock releases. These releases 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 location:

```
$ cd /path/to
$ unzip ~/Downloads/IDM-5.5.1.3.zip
Archive:  IDM-5.5.1.3.zip
  inflating: openidm/.checksums.csv
    creating: openidm/bundle/
 extracting: openidm/bundle/openidm-audit-5.5.1.3
.jar
...
```

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

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

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

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

To Start IDM

To run IDM as a background process, see "*Starting and Stopping 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):

   ```
   $ ./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 "5.5.1.3"
   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 "5.5.1.3"
   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.opendj  enabled
    [  48] [active      ] org.forgerock.openidm.repo
.opendj
..
.
->
```

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)
```

To Install IDM 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 `install-service.bat` file, which registers the service within the Windows registry, make sure that your `JAVA_HOME` environment variable points to a valid 64-bit version of the JRE or JDK. If you have already installed the service with the `JAVA_HOME` environment variable pointing to a 32-bit JRE or JDK, delete the service first, then reinstall the service.

1. Unpack the IDM-5.5.1.3.zip file, as described previously, and change to the *install-location*\bin directory:

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

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

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

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

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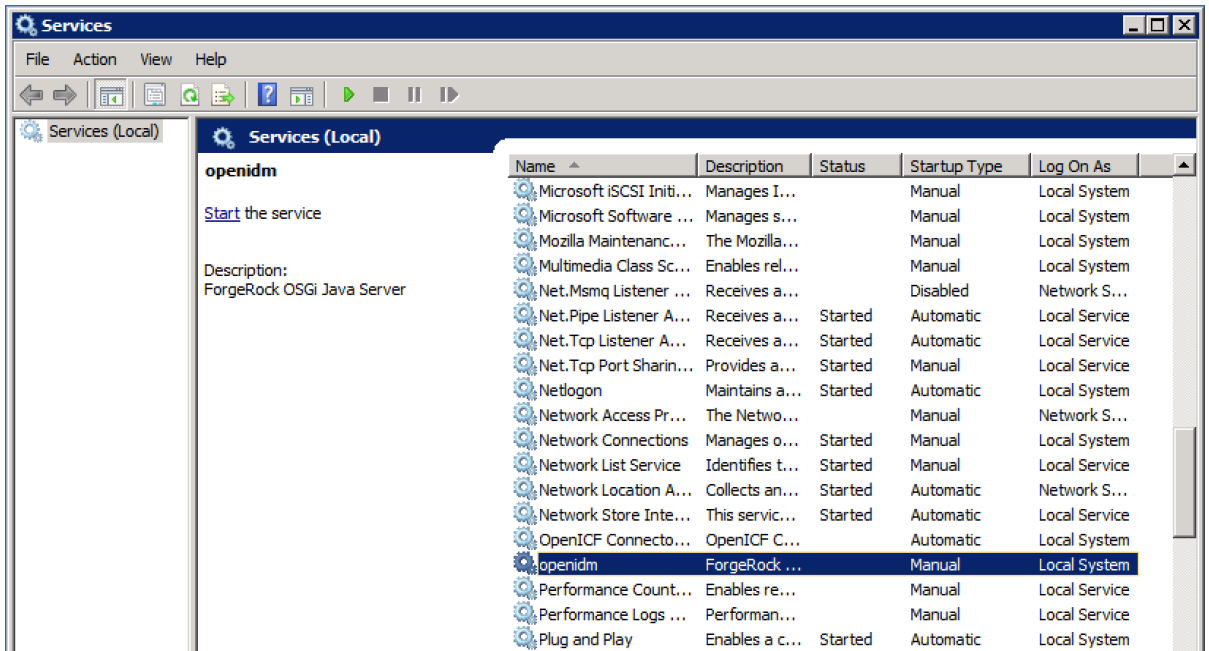
4. Change the user account for this service from the default (`local system`) account to an account with administrative privileges. The `local system` account has limited permissions and an IDM service that runs with this account will encounter problems during synchronization.

   To change the user account:

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

   b. Select the Log On tab.

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

   d. Enter the password for the administrative account.

Changing the Service User Account
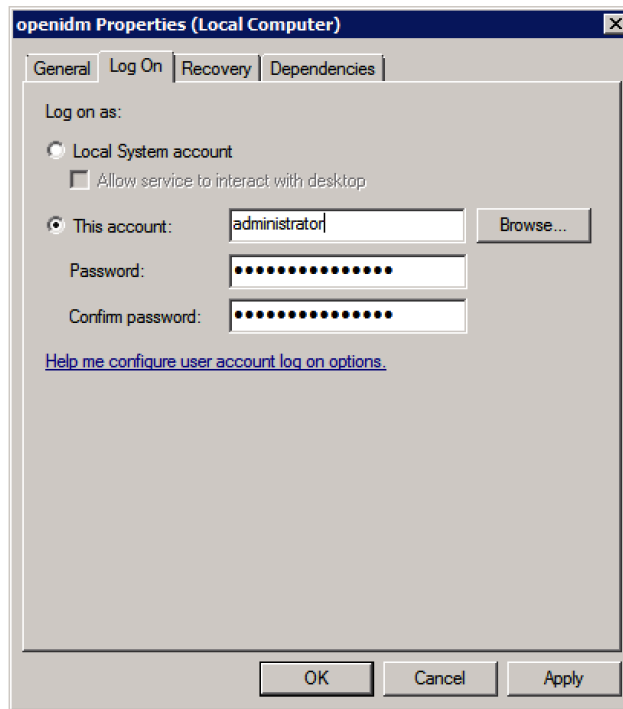
Click Apply to save the changes.

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

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

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

1.3. Getting Started With the REST Interface

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 **curl** command-line utility that is included with most operating systems.

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 when creating resources, as it makes the examples easier to read. In general, immutable server-assigned UUIDs should be used in production.

1. Access the following URL to obtain the JSON representation of all users in the 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 the user `joe` in the repository.

- 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": []
}
```

1.3.1. Format REST Output For Readability

By default, **curl**-based REST calls return the JSON object on one line.

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

```
{"mail":"joe@example.com","sn":"smith","passwordAttempts":"0",
"lastPasswordAttempt":"Mon Apr 14 2014 11:13:37 GMT-0800 (GMT-08:00)",
"address2":"","givenName":"joe","effectiveRoles":["openidm-authorized"],
"password":{"$crypto":{"type":"x-simple-encryption","value":{"data":
"OBFVL9cG8uaLoo1N+SMJ3g==","cipher":"AES/CBC/PKCS5Padding","iv":
"7rlV4EwkwdRHkt19F8g22A==","key":"openidm-sym-default"}}},"country":"",
"city":"","_rev": "00000000c03fd7aa","lastPasswordSet":"","postalCode":"",
"_id":"joe3","description":"My first user","accountStatus":"active","telephoneNumber":
"555-123-1234","roles":["openidm-authorized"],"effectiveAssignments":{},
"postalAddress":"","stateProvince":"","userName":"joe3"}
```

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

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

```
$ curl \
--header "X-OpenIDM-Username: openidm-admin"
 \
--header "X-OpenIDM-Password: openidm-admin"
 \
--request GET \
"http://localhost:8080/openidm/managed/user/joe" \
| jq .
```

The ForgeRock REST API includes an optional `_prettyPrint` request parameter. The default value is `false`. To use the ForgeRock REST API to format output, add a parameter such as `?_prettyPrint=true` or

`&_prettyPrint=true`, depending on whether it is added to the end of an existing request parameter. In this case, the following command would return formatted output:

```
$ curl \
--header "X-OpenIDM-Username: openidm-admin"
 \
--header "X-OpenIDM-Password: openidm-admin"
 \
--request GET \
"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.

1.4. 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 Self-Service UI and the Administrative UI (Admin UI), respectively.

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

In addition, administrative users can configure and manage workflows. For more information, see "Managing User Access to Workflows" in the *Integrator's Guide*.

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

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

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

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

1.5. About the Repository

By default, IDM comes with an internal ForgeRock Directory Services (DS) instance for use as its repository. This makes it easy to get started. DS is not supported as a repository in production, however, so use a supported JDBC database when moving to production.

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 61389 \
 --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*.

1.6. 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 location:

```
$ 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 DS with a supported JDBC repository.

In production environments, the following repositories are supported:

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 an MS 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 general JDBC repository configuration, and how to map IDM objects to JDBC database tables, see "*Managing the Repository*" in the *Integrator's Guide*.

For details about the supported repository versions, see "*Before You Install*" in the *Release Notes*.

2.1. Using the Default DS Repository

IDM uses the `conf/repo.opendj.json` file to start an embedded DS instance with the following default values:

- `hostname` - `localhost`

- `ldapPort` - `61389`

- `bindDN` - `cn=Directory Manager`

- `bindPassword` - `password`

- `adminPort` - `64444`

To change the administrative port of the embedded DS server, add an `adminPort` property to your project's `conf/repo.opendj.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.opendj.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,
  "adminPort": 4444,
  "ldapConnectionFactories": {
    "bind": {
      "primaryLdapServers": [{ "hostname": "localhost", "port": 61389 }]
    },
    "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.

> **Important**
>
> The DS repository does not support storage of audit data. Audit logging to the repository is disabled by default. Do not enable logging to the repository while you are using the default repository.

> **Note**
>
> If you are running Red Hat Enterprise Linux 6 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.

2.1.1. Using the Default Repository With a PKCS #11 (HSM) Device

By default, the embedded DS repository uses the IDM keystore and truststore. If IDM is using a JKS or JCEKS keystore, the embedded DS server generates its own self-signed certificate and stores it in the IDM keystore the first time you start IDM.

Currently, you cannot use the default repository with an external PKCS #11 (HSM) device (see OPENIDM-7947).

2.2. Minimum Database Access Rights

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 `database-type/conf/openidm.sql` script. The `openidm` user account must have SELECT, UPDATE, INSERT, and DELETE permissions on all the openidm tables that are created by this script, 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.

> **Important**
>
> The scripts that set up the audit tables are new in IDM 5.5.0. Prior to this version, the audit tables were set up by default for all JDBC repositories. Now, you need run these scripts *only* if you plan to direct audit information to your JDBC repository.

2.3. 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.opendj.json`) from your project's `conf/` directory. For example:

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

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

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

5. Import the 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:
   $
   ```

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

   ```
   $ mysql -u root -p
   Enter password:
   Welcome to the MySQL monitor.  Commands end with ; or \g.
   Your MySQL connection id is 18
   ```

```
Server version: 5.5.19 MySQL Community Server
 (GPL)
...
mysql> use openidm;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;

mysql> show tables;
+--------------------------+
| Tables_in_openidm        |
+--------------------------+
| clusteredrecontargetids  |
| clusterobjectproperties  |
| clusterobjects           |
| configobjectproperties   |
| configobjects            |
| genericobjectproperties  |
| genericobjects           |
| ...                      |
| schedulerobjects         |
| securitykeys             |
| uinotification           |
| updateobjectproperties   |
| updateobjects            |
+--------------------------+
21 rows in set (0.00 sec)
```

Exit the mysql console.

```
mysql> exit
Bye
```

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

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

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

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

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

7. 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:
```

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

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

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

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

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

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

```
$ export OPENIDM_OPTS="-Xmx1024m -Xms1024m -Dopenidm.repo.host=localhost -Dopenidm.repo.port=3306"
$ ./startup.sh -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/conf/boot/boot
.properties
-> OpenIDM version "5.5.1.3"
OpenIDM ready
```

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.opendj` 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/conf/boot/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.opendj
 enabled
...
```

2.4. Setting Up an MS SQL Repository

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

When you install Microsoft SQL Server, 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 MS SQL).

  5. Click OK.

  6. Restart MS SQL Server for the configuration changes to take effect.

     To restart the server, select SQL Server Services in the left pane, double click SQL Server (MSSQLSERVER) and click Restart.

  7. If you have a firewall enabled, ensure that the port you configured in the previous step is open for IDM to access MS SQL.

After you have installed MS 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 MS SQL repository, as described in the following steps:

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

   a. Navigate to SQL Server Management Studio.

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

   c. Select File > Open > File and navigate to the 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.
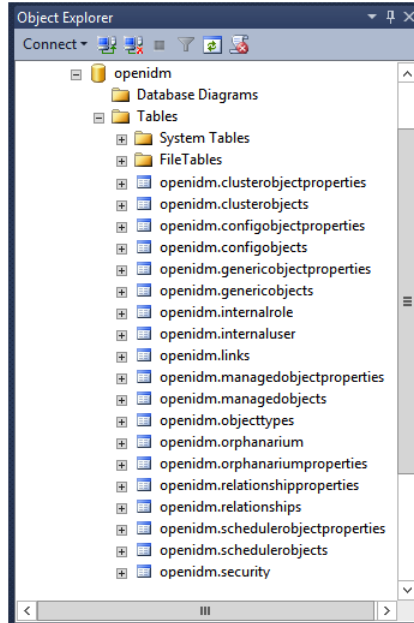
3. Execute the three scripts that set up the tables required by the Activiti workflow engine:

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

   a. Download the JDBC Driver for SQL Server from Microsoft's download site. IDM requires at least version 4.2 of the driver (`sqljdbc_4.2.8112.100_enu.tar.gz`). The precise download URL might vary, depending on your location.

      Extract the JAR file (`sqljdbc42.jar`), using 7-zip or an equivalent file management application and copy the file to the `openidm\db\mssql\scripts` directory.

   b. Download the `bnd` JAR file (`bnd-2.4.0.jar`) that enables you to create OSGi bundles. For more information about `bnd`, see http://bnd.bndtools.org/.

      Copy this JAR file to `openidm\db\mssql\scripts`.

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

      ```
      bnd-2.4.0.jar  sqljdbc4.bnd  sqljdbc42.jar
      ```

   d. Create an OSGi bundle from the driver JAR with the following command:

      ```
      C:\> cd \path\to\openidm\db\mssql\scripts
      ./> java -jar bnd-2.4.0.jar wrap --properties sqljdbc4.bnd --output sqljdbc42-osgi.jar sqljdbc42
      .jar
      ```

      This step creates the OSGi bundle named `sqljdbc42-osgi.jar`.

   e. Copy the `sqljdbc42-osgi.jar` file to the `openidm\bundle` directory:

      ```
      ./> copy sqljdbc42-osgi.jar \path\to\openidm\bundle
      ```

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

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

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

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

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

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

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

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

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

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

```
$ export OPENIDM_OPTS="-Xmx1024m -Xms1024m -Dopenidm.repo.host=localhost -Dopenidm.repo.port=1433"
$ 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=1433
Using LOGGING_CONFIG: -Djava.util.logging.config.file=/path/to/openidm/conf/logging.properties
Using boot properties at /path/to/openidm/conf/boot/boot
.properties
-> OpenIDM version "5.5.1.3"
OpenIDM ready
```

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.opendj` is `enabled` but not `active`:

```
C:> cd \path\to\openidm
./> startup.bat -p my-project
"Using OPENIDM_HOME:    \path\to\openidm"
"Using OPENIDM_OPTS:   -Xmx1024m -Xms1024m -Dfile.encoding=UTF-8"
"Using LOGGING_CONFIG:
-Djava.util.logging.config.file=\path\to\openidm\conf\logging.properties"
 Using boot properties at \path\to\openidm\conf\boot\boot.properties
-> scr list
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.opendj
 enabled
...
```

2.5. Setting Up an Oracle DB Repository

When you set up an Oracle database as the repository, confer with an Oracle DBA when creating 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.

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

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

```
SQL> select * from internaluser;

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

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

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

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

   You must run the following scripts:

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

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

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

   You must run the following script:

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

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

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

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

   b. Create a `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 the `bnd` file to the same location as the JDBC driver:

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

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

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

Edit the `bnd` file to match the JDBC driver version:

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

c. Download the `bnd` JAR file (bnd-2.4.0.jar) that enables you to 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, and the `bnd` file:

```
$ ls /path/to/openidm/db/oracle/scripts
bnd-2.4.0.jar    ojdbc7_g.jar    openidm.sql    sqljdbc4.bnd
```

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

```
$ cd /path/to/openidm/db/oracle/scripts
$ java -jar bnd-2.4.0.jar wrap -properties sqljdbc4.bnd ojdbc7_g.jar
Dec 10, 2013 9:53:28 AM java.util.prefs.FileSystemPreferences$1 run
INFO: Created user preferences directory.
ojdbc7_g.jar 984 0
```

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

```
$ ls
bnd-2.4.0.jar  ojdbc7_g.bar  ojdbc7_g.jar  openidm.sql  sqljdbc4.bnd
```

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

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

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

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

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

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

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

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

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

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

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

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

```
$ export OPENIDM_OPTS="-Xmx1024m -Xms1024m -Dopenidm.repo.host=localhost -Dopenidm.repo.port=1525"
$ 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=1525
Using LOGGING_CONFIG: -Djava.util.logging.config.file=/path/to/openidm/conf/logging.properties
Using boot properties at /path/to/openidm/conf/boot/boot
.properties
-> OpenIDM version "5.5.1.3"
OpenIDM ready
```

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

The `DEFAULTCATALOG` 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.

When you have set up an Oracle database 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` `.opendj` 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/conf/boot/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.opendj
 enabled
...
```

2.6. 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 are going 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.opendj.json`) from your project's `conf/` directory. For example:

```
$ cd /path/to/openidm/my-project/conf/
$ rm repo.opendj.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 project's `conf/boot/boot.properties` file:

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

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

```
$ export OPENIDM_OPTS="-Xmx1024m -Xms1024m -Dopenidm.repo.host=localhost -Dopenidm.repo.port=5432"
$ 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/conf/boot/boot
.properties
-> OpenIDM version "5.5.1.3"
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.opendj` 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/conf/boot/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.opendj
 enabled
...
```

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

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

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

2.7. 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 the `bnd` file to the same location as the JDBC driver.

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

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

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

      Edit the `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 enables you to 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, and the `bnd` file.

```
$ ls /path/to/openidm/db/db2/scripts
bnd-2.4.0.jar  db2jcc.jar  openidm.sql  sqljdbc4.bnd
```

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

```
$ cd /path/to/openidm/db/db2/scripts
```

```
$ java -jar bnd-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.bar  db2jcc.jar  openidm.sql  sqljdbc4.bnd
```

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.opendj.json`) from your project's `conf/`
    directory. For example:

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

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

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

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

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

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

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

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

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

```
$ export OPENIDM_OPTS="-Xmx1024m -Xms1024m -Dopenidm.repo.host=localhost -Dopenidm.repo.port=50000"
$ 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/conf/boot/boot
.properties
-> OpenIDM version "5.5.1.3"
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.opendj` 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/conf/boot/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.opendj
 enabled
...
```

2.7.1. 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
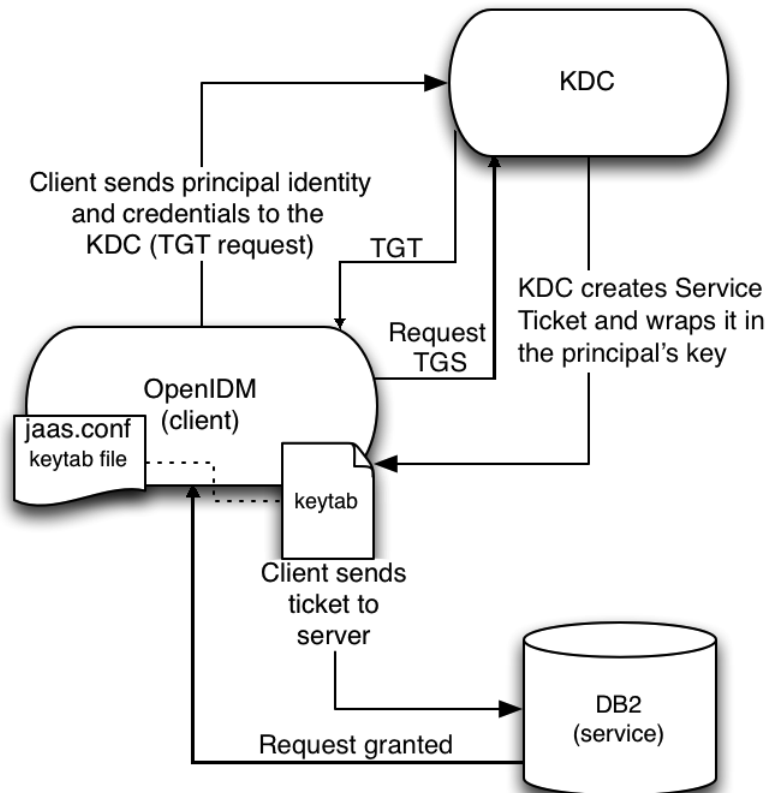
To configure IDM for Kerberos authentication:

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 and describes how to move an existing installation to a different location.

To Remove IDM

1. (Optional) Stop the server if it is running, as described in "To Stop IDM".

2. Remove the file system directory where you installed the software:
   ```
   $ rm -rf /path/to/openidm
   ```

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

To Move IDM

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

1. Stop the server, as described in "To Stop IDM".

2. Remove the `felix-cache` directory:
   ```
   $ cd path/to/openidm
   $ rm -rf felix-cache
   ```

3. Move the files:
   ```
   $ mv path/to/openidm path/to/new-openidm
   ```

4. Start the server in the new location:
   ```
   $ cd path/to/new-openidm
   $ ./startup.sh
   ```

Chapter 4

Updating the Server

This chapter describes how to update an existing deployment, for example, IDM 5.0 to IDM 5.5.0 and then from version 5.5.x to 5.5.1.3.

If you are on IDM 5.0.0 version and plan to update to IDM 5.5.1.3, you must first update to version 5.5.0, then update to 5.5.1.3. For information on updating to 5.5.1.3, see "Updating the Server to 5.5.1.3".

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

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

4.1. Limitations of the Automated Update Process

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

Phased Update From Previous Versions

You cannot update directly from IDM 4.0 or IDM 4.5 to IDM 5.5.1.3. You must update to each major version successively, so from 4.0 to 4.5, from 4.5 to 5, from 5 to 5.5.0, and so on.

- To update from version 4.0 to version 4.5, follow the instructions shown in *Updating From OpenIDM 4.0 to OpenIDM 4.5* in the *OpenIDM 4.5 Installation Guide*.

- To update from version 4.5 to version 5, follow the instructions shown in *Updating To OpenIDM 5* in the *IDM 5 Installation Guide*.

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

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

- To migrate a deployment from 3 to 3.1, see *Migrating to OpenIDM 3.1* in the *OpenIDM 3.1 Installation Guide*.

- To migrate a deployment from 2.1 to 3, see *Migrating to OpenIDM 3* in the *OpenIDM 3 Installation Guide*.

Update on Windows Systems

Automated update is not supported on Microsoft Windows systems. If you are updating on that operating system, follow the manual update process:

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

- To migrate a Windows deployment from version 4.5 to 5, see *Migrating From OpenIDM 4.5 to OpenIDM 5 on Windows* in the *OpenIDM 5 Installation Guide*.

  To migrate a Windows deployment from version 5 to 5.5.0, see "Migrating to IDM 5.5.0 on Windows".

Updating a Clustered Deployment

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

- Redirect client traffic to a different IDM system or cluster.

- Shut down every node in the cluster.

- Update one node.

- Clone the first node to the other nodes in that cluster.

Supported Project Directories

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

- The default project directory, `/path/to/openidm`

- Outside of the IDM directory tree, such as `/home/project` or `/other/hard_drive/idm`

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

This documentation represents the project directory as `project-dir`.

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

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

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

Supported Repositories Only

> Update is supported *only* with the supported JDBC repositories. As such, repository update scripts are provided only for those JDBC repositories.
>
> For instructions on installing a supported repository, see "*Selecting a Repository*".

Custom Configuration Files

> If you customized configuration files for your IDM 5 deployment, such as JSON files, or files related to a custom UI, you'll need to reapply what you customized after the update is complete. For more information, see "Manage Custom Changes After Updating".

4.2. An Overview of the Update Process

This section provides background information on the update process. If you want to jump to the steps required to update to version 5.5.0, read "Updating to Version 5.5.0".

4.2.1. Review the Supported Update Paths

Before you plan out your update process, take the time and review "Supported Update Paths" in the *Release Notes*. Your update may involve multiple intermediate steps depending on your IDM version.

4.2.2. The Update Process in Labels

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

- `PREVIEW_ARCHIVE`: Preview the archive using appropriate checksums, to define the files to be updated.

- `ACCEPT_LICENSE`: Present the appropriate license for the update. If the license is not accepted, the update stops, and you retain all files in your current IDM installation.

- `LIST_REPO_UPDATES`: Identify and save scripts to update the current repository.

- `PAUSING_SCHEDULER`: Pause the scheduler, to prevent new jobs from starting during the update process.

- `WAIT_FOR_JOBS_TO_COMPLETE`: Wait for any currently running jobs to complete.

- `ENTER_MAINTENANCE_MODE`: Set the server in maintenance mode, which disables non-essential services.

- `INSTALL_ARCHIVE`: Install the update archive.

- `WAIT_FOR_INSTALL_DONE`: Wait until the installation of the update archive is complete.

- `MARK_REPO_UPDATES_COMPLETE`: Note status when repository updates are complete.

- `EXIT_MAINTENANCE_MODE`: Exit from maintenance mode.

- `ENABLE_SCHEDULER`: Enable the scheduler. Any schedules that were previously in place are re-established.

- `FORCE_RESTART` Restart the server to implement all changes that require it.

4.2.3. Files Changed During the Update Process

As of IDM 4, the server includes MD5 checksums for each file. These checksums allow the update process to verify file changes.

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

- The checksum of the file as originally installed.

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

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

Files in `openidm/bundle`

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

> New files are written to the target directory.

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

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

> New files are written to the target directory.

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

project-dir`/script/access.js`

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

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

> *Project configuration files (JSON)*: Files related to your project are patched with the content of the corresponding version 5.5.0 configuration file, regardless of whether they have been customized.

> **Warning**
>
> If you have customized standard IDM JSON configuration files, be careful. Analyze the corresponding version 5.5.0 file. It may include new features that you want. As a best practice, apply each customization to the corresponding version 5.5.0 file and test the change before you put it into production. For more information, see "Manage Custom Changes After Updating".

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

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

Project configuration files (JSON): Files related to your project are overwritten with the content of the corresponding version 5.5.0 configuration file, regardless of whether they have been customized.

> **Warning**
>
> If you have customized any standard IDM JSON configuration files, be careful. Analyze the corresponding version 5.5.0 file. It may include new features that you want. As a best practice, apply and test each custom change to the version 5.5.0 configuration file. For more information, see "Manage Custom Changes After Updating".

System configuration files: (`boot.properties`, `system.properties`, `config.properties`, `logging.properties`, and `jetty.xml`) are not patched if they have been customized. Instead, the update process creates configuration files with a `.new-unix_time` extension in the same directory. You *must* merge any changes from these `.new-` files into your customized configuration files. For more information, see "Manage Custom Changes After Updating". If you have not customized these files, the update process replaces the existing configuration file with the corresponding version 5.5.0 file.

Files in any other directory

Existing files are overwritten and no backup files are created.

Configuration in the repository

Configuration information is stored in your repository. The update process overwrites information in that data store.

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

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

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

> **Warning**
>
> If you've changed defaults in the `boot.properties` file, be careful, as this can affect the boot process.

4.2.3.1. Updating `logging.properties`

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

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

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

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

4.2.4. Files Added During the Update Process

Several new files are added to `project-dir/conf` during the update process.

Files added to `project-dir/conf` during the update process

File	Description
`emailTemplate-resetPassword.json`	Configures password reset emails. For more information, see "User Password Reset" in the *Integrator's Guide*.
`emailTemplate-welcome.json`	Configures account creation welcome emails. For more information, see "Configuring Self-Service Email Validation / Username" in the *Integrator's Guide*.
`info-uiconfig.json`	Specifies options for the `info/uiconfig` endpoint in the "API Explorer" in the *Integrator's Guide*.
`jsonstore.json`	Configures how IDM stores JSON values.
`metrics.json`	An internal/undocumented feature, per "ForgeRock Product Interface Stability" in the *Integrator's Guide*.

File	Description
`repo.opendj.json`	Configures an embedded DS instance, so you can evaluate the software without needing to install a separate database. **Warning** You must remove this file before you start the updated IDM instance. The presence of two repository configuration files in your *project-dir/conf* directory will cause errors.
`ui.context-oauth.json`	Configures an OAuth-based redirect endpoint for linked identity providers such as a social identity provider or AM. For more information, see "*Configuring Social Identity Providers*" in the *Integrator's Guide*.
`ui-profile.json`	Configures tabs in user profiles. For more information, see "Privacy: My Account Information in the Self-Service UI" in the *Integrator's Guide*.

4.2.5. The `update.json` File

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

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

The version numbers may be either specific, such as `5.5.0` or a range, such as `[5.0.0.1,5.0.0.3)`. That range supports version numbers `5.0.0.1` and `5.0.0.2` (but not `5.0.0.3`). For a full explanation of how you can specify a range of versions, see "Setting the Connector Reference Properties" in the *Integrator's Guide*.

If you want to delete files during the update process, include them in the `removeFile` code block. You can include files in the following formats:

- Individual files.

- Directories: If you specify a directory, the update process will remove that directory, all files within that directory, and all subdirectories, recursively.

- Files and directories with wildcards: If you specify a UNIX-style wildcard, `*` or `?`, the update process will remove all files that meet the requirements of the wildcard.

To remove configurations during the update, include the corresponding `.json` files in the `removeConfig` array. Do not include the path to these files. During the update, IDM recognizes these as configuration files and attempts to remove the corresponding configuration from the repository.

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

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

4.2.5.1. Setting up a Custom Update

You can create your own custom update binary. To do so, you'll need to modify two files:

- `.checksum.csv`

  Add an MD5 checksum for each file in your binary to a new `.checksum.csv` file. For guidance, see the same hidden file in the `/path/to/openidm` directory.

- `update.json`

  If you want to try your own update, make sure that the `origin` and `destination` version numbers match your current and planned update versions of IDM. For more information, see "The `update.json` File".

> **Warning**
>
> If you use the IDM update process with your own update binary, this may affect your ability to update IDM to 5.5.0 and beyond.

4.2.6. Command-Line Update Options

On UNIX/Linux systems, you can update to IDM 5.5.0 by using the **cli.sh update** command. For general information on **cli.sh** and **cli.bat**, see "*Command-Line Interface*" in the *Integrator's Guide*.

The following command updates the local system with the `IDM-5.5.0.zip` binary:

```
$ cd /path/to/openidm
$ ./cli.sh update
 \
--acceptLicense
 \
--user openidm-admin:openidm-admin
 \
--url http://localhost:8080/openidm \
IDM-5.5.0.zip
```

The `update` subcommand takes the following options:

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

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

--url URL

The URL of the REST service. The default URL is `http://localhost:8080/openidm/`.

-P or --port PORT

The port number associated with the REST service. If specified, this option overrides any port number specified with the **--url** option. The default port is 8080.

--acceptLicense

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

--skipRepoUpdatePreview

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

--maxJobsFinishWaitTimeMs TIME

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

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

--maxUpdateWaitTimeMs TIME

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

Default: `30000` ms

-l or --log LOG_FILE

Path to the log file.

Default: `logs/update.log`

-Q or --quiet

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

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

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

```
Executing ./cli.sh...
Starting shell in /path/to/openidm
Using boot properties at /path/to/openidm/conf/boot/boot.properties
License was accepted via command line argument.
Pausing the Scheduler
Scheduler has been paused.
Waiting for running jobs to finish.
All running jobs have finished.
Entering into maintenance mode...
Now in maintenance mode.
Installing the update archive IDM-5.5.0.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...
The update process is complete with a status of COMPLETE
Restarting OpenIDM.
Restart request completed.
```

4.2.7. Update File States

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

- "Preview of File Updates"

- "Update Status Message"

- "Updated Files: What Happened"

Preview of File Updates

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

Update Status Message

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

Status	Description
COMPLETE	Update is complete
FAILED	Update failed

Updated Files: What Happened

Status	Description
REPLACED	Original file replaced; if the original file was changed by a user, it is saved with a `.old` extension.
PRESERVED	Original file saved with changes made by a user. New file saved with a `.new` extension.
APPLIED	The update changed the file.
REMOVED	The update removed the file.
FAILED	The attempt to update the file, via a patch, did not work.

4.3. Updating to Version 5.5.0

If you are on IDM version 5.0, you must update first to 5.5.0, then update again to version 5.5.1.3. For more information on upgrading to IDM 5.5.1.3, see "Updating the Server to 5.5.1.3".

In the following sections, you'll:

• "Prepare Your Update".

• "Update to IDM 5.5.0".

• "Manage Custom Changes After Updating".

4.3.1. Prepare Your Update

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

Update Your Java Environment

IDM 5.5.0 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".

Download the Software

For update to IDM 5.5.0: Download the following IDM binaries from ForgeRock's BackStage site:

- `IDM-5.5.0-pre_update.zip`

- `IDM-5.5.0.zip`

Back up your deployment

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

Save any customized `*.json` configuration files, typically located in your project's `conf/` subdirectory. You'll need to make judgements on how to apply these customizations to your version 5.5.0 deployment after the update is complete.

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

Save your audit data. If you want to preserve a record of your audit logs collected while you've run IDM 5.0, you'll have to save these log files manually before starting the update process. You can find these 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.

Change read-only deployments

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

Disable authentication modules used with AM

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

Review Repository Update Scripts

IDM 5.5.0 includes a number of scripts that you must run to update your repository schema. The update process detects which scripts are required for your particular repository, extracts them from the IDM 5.5.0 binary, and prompts you to save them in the directory of your choice.

The scripts vary by database. For a list of scripts needed for the update process, see "Repository Update Scripts".

You might need to get Database Administrator (DBA) help and approval to run these update scripts.

Start the server

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

> **Warning**
>
> If you use custom versions of IDM Admin and Self-Service UIs, the update process may affect you in different ways; for more information, see "Files Changed During the Update Process". If you followed the procedure described in "Customizing the Admin UI" in the *Integrator's Guide*, you'll have custom files in the `openidm/ui/admin/extension` and `openidm/ui/selfservice/extension` directories.
>
> IDM 5.5.0 includes significant UI improvements. The update process does not copy those improvements to the noted `extension/` subdirectories.

When you're ready to run the update process, see "Update to IDM 5.5.0".

> **Note**
>
> Some of you may have already updated IDM with the update patch, known as *Social Providers Add-ons*, available from ForgeRock's BackStage site. If you've done so, the process to update to IDM 5.5.0 is the same.
>
> The update to IDM 5.5.0 includes the same additional social identity providers and more.

4.3.1.1. Disabling and Re-enabling Connections to AM

Before updating, you'll need to disable connections to AM. If you set up this connection as described in "*Integrating IDM With the ForgeRock Identity Platform*" in the *Samples Guide*, you can use the following procedures to disable and re-enable the connection to AM:

Disabling a Connection to AM 13.x

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

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

1. Select Configure > System Preferences. In the default Authentication tab, edit the `OpenAM Session` module and set `Module Enabled` to false.

2. You should now be able to log into IDM normally, without authenticating through AM.

3. You can now proceed with the update process described in this chapter.

Disabling a Connection to AM 5

If you've followed the procedure in the version 5 *Integrating IDM With the ForgeRock Identity software*, you've set up a ForgeRock Identity Provider.

In that sample, you've configured the `OPENID_CONNECT` module, which you can disable as follows from the IDM 5 Admin UI:

1. Select Configure > Authentication. If you've configured the connection with AM, the authentication provider should be set to the `ForgeRock Identity Provider`. Change it to `Local`.

2. Follow the prompts. You should now be able to log into IDM normally, without authenticating through AM.

3. You can now proceed with the update process described in this chapter.

4.3.1.1.1. Re-enabling a Connection to AM

If you need integration with AM, you should also update to AM 5.5.0. For more information, see the *AM Upgrade Guide*.

When the update process is complete, you can then re-enable the connection to AM. To do so, follow the steps described in "*Integrating IDM With the ForgeRock Identity Platform*" in the *Samples Guide*. The process has changed significantly for this release.

4.3.1.1.2. Repository Update Scripts

IDM keeps a record of all repository update scripts in the `openidm/db/repo/scripts/updates` directory. The following table includes the repository update scripts that apply to the update from IDM 5.0 to IDM 5.5.0.

Preview of File Updates

Database	Scripts
• DB2	• `v6_add_primary_key_to_properties_tables.sql`
• MSSQL	• `v7_create_clusteredrecon_targetids_table.sql`
• MySQL	• `v8_add_uinotification_index.sql`
	• `v9_add_provider_method_context_fields.sql`
	• `v10_increase_uinotification_id_sizes.sql`
• Oracle	• `v7_add_primary_key_to_properties_tables.sql`
	• `v8_create_clusteredrecon_targetids_table.sql`
	• `v9_add_uinotification_index.sql`

Database	Scripts
	• v10_add_provider_method_context_fields.sql
	• v11_add_foreign_key_to_schedulerobjects_table.sql
	• v12_increase_uinotification_id_sizes.sql
	• v13_add_misc_foreign_keys.sql
• PostgreSQL	• v9_add_primary_key_to_properties_tables.sql
	• v10_create_clusteredrecon_targetids_table.sql
	• v11_add_uinotification_index.sql
	• v12_add_provider_method_context_fields.sql
	• v13_increase_uinotification_id_sizes.sql

You may ignore the other scripts that appear in the applicable `openidm/db/`*`repo`*`/scripts/updates` directory. They're present as an archive. When you update IDM in the future, they can help track scripts that have already been installed.

4.3.2. Update to IDM 5.5.0

When you have completed the steps in "Prepare Your Update", your IDM instance is ready for an update to version 5.5.0.

Copy the following two new update binaries to the `/path/to/openidm/bin/update` directory. That directory is created the first time you start IDM.

• `IDM-5.5.0-pre_update.zip`

• `IDM-5.5.0.zip`

Start the update process, using the CLI or the Admin UI. This section includes separate procedures for those options.

4.3.2.1. Updating from the CLI

1. Start the update process from the command-line, with the first binary, IDM-5.5.0-pre_update.zip:

```
$ cd /path/to/openidm
$ ./cli.sh update
 \
--acceptLicense
 \
--user openidm-admin:openidm-admin
 \
--url http://localhost:8080/openidm \
IDM-5.5.0-pre_update.zip
```

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

> **Tip**
>
> The first time you run the update process with IDM-5.5.0-pre_update.zip, you may see an error message, before you're taken back to the command line:
>
> ```
> ERROR: Error during execution. The state of OpenIDM is now
> unknown. Last Attempted step was PREVIEW_ARCHIVE. Now attempting recovery
> ```
>
> You may ignore the message, and rerun the same update command.

The first update process should take a relatively short time with the following messages:

```
Executing ./cli.sh...
Starting shell in /path/to/openidm
Using boot properties at /path/to/openidm/conf/boot/boot.properties
License was accepted via command line argument.
No repository updates included in the archive.
```

2. To ensure that IDM reads new bundles in the /path/to/openidm/bundle directory, navigate to the console. Shutdown and restart IDM.

```
-> shutdown
$ ./startup.sh
```

3. Now continue with the update process to 5.5.0:

```
$ cd /path/to/openidm
$ ./cli.sh update
 \
--acceptLicense
 \
--user openidm-admin:openidm-admin
 \
--url http://localhost:8080/openidm \
IDM-5.5.0.zip
```

4. During the update process, you are prompted to save applicable repository update scripts. For a full list, see "Repository Update Scripts". You might see a message similar to the following:

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

Enter a directory name, such as update-scripts. The update process creates this directory, relative to your IDM installation directory, and saves the required scripts to that directory.

The update then displays the following message:

```
Repository scripts are in the following directory: update-scripts.
Please apply the update scripts now. When complete, run the update again with the
--skipRepoUpdatePreview option.
```

5. Apply the database repository scripts in numeric order. How you apply these scripts depends on your repository. The following example applies a hypothetical update script required for a MySQL repository:

```
$ mysql -u root -p < openidm/update-scripts/v999_something_added.sql
```

6. After you've run the database respository scripts, run the same **cli.sh update** command again; add the following option: **--skipRepoUpdatePreview**:

```
$ cd /path/to/openidm
$ ./cli.sh update
 \
--acceptLicense
 \
--skipRepoUpdatePreview
 \
--user openidm-admin: openidm-admin
 \
--url http://localhost:8080/openidm \
IDM-5.5.0.zip
```

The update continues. The process may take a number of minutes. When the update is complete, you'll see the following messages:

```
Repo Updates status: COMPLETE
Exiting maintenance mode...
No longer in maintenance mode.
Resuming the job scheduler.
Scheduler has been resumed.
```

7. Go to the IDM console. You may ignore the following errors:

```
SEVERE: Bundle: org.forgerock.openidm.infoservice [20]
[org.forgerock.openidm.info(68)] The activate method has thrown an exception
org.apache.felix.log.LogException: org.osgi.service.component.ComponentException:
Failed to take script: uiconfig
```

```
org.forgerock.openidm.maintenance.upgrade.UpdateManagerImpl$UpdateThread applyConfigPatch
WARNING: Failed to apply patch selfservice-registration.json.patch, error:
Patch request failed
```

8. When you see the `OpenIDM ready` message at the console, shut down IDM:

```
OpenIDM ready

-> shutdown
```

9. The update creates a `repo.opendj.json` file in your *project-dir*/conf directory. This file configures an embedded DS instance, so you can evaluate the software without needing to install a separate database.

You must delete this file before you continue. The presence of two repository configuration files in your `conf/` directory will cause errors.

10. If you are updating from IDM 5.0.0.2, 5.0.0.3, or 5.0.1.0 to IDM 5.5.0, delete the `forgerock-jaspi-openid-connect-module-21.7.0.jar` file from the `/path/to/openidm/bundle` directory:

```
$ rm /path/to/openidm/bundle/forgerock-jaspi-openid-connect-module*.jar
```

If you are updating from 5.0.0 or 5.0.0.1 to 5.5.0, this file is deleted automatically.

11. Proceed to "Manage Custom Changes After Updating".

4.3.2.2. Updating From the Admin UI

Updating From the Admin UI, Part One

1. Start IDM.

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

3. Select Configure > System Preferences, and choose the Update tab. You should see `IDM-5.5.0-pre_update.zip` and `IDM-5.5.0.zip` in the list of Available Updates.

   > **Tip**
   >
   > If nothing appears under the update tab, refresh your browser.

4. Next to `IDM-5.5.0-pre_update.zip`, select Install Update to start the initial update process.

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

   > **Note**
   >
   > For this first `IDM-5.5.0-pre_update.zip` update, the UI will show no files to be replaced. That is an error. The following is a list of files replaced with the contents of `IDM-5.5.0-pre_update.zip`:

```
$ tree
.
├── bin
│   └── openidm.jar
├── legal-notices
│   ├── CDDLv1.0.txt
│   ├── copyrights.txt
│   └── Forgerock_License.txt
├── ui
│   └── admin
│       └── default
│           ├── locales
│           │   └── en
│           │       └── translation.json
│           └── org
│               └── forgerock
│                   └── openidm
│                       └── ui
│                           └── admin
│                               └── settings
│                                   └── update
│                                       ├── InstallationPreviewView.js
│                                       ├── InstallView.js
│                                       └── RepoUpdateView.js
└── update.json

14 directories, 9 files
```

You'll see these files in the installation report.

6. Return to the IDM console. Shut down IDM:

```
-> shutdown
```

7. Proceed to "Updating From the Admin UI, Part Two".

Updating From the Admin UI, Part Two

1. Start IDM:

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

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

3. Select Configure > System Preferences, and choose the Update tab. You should see `IDM-5.5.0-pre_update.zip` and `IDM-5.5.0.zip` in the list of Available Updates.

> **Tip**
>
> If nothing appears under the update tab, refresh your browser.

4. Next to `IDM-5.5.0.zip`, select Install to start the update process, then Confirm and Install to enable Maintenance Mode.

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

6. The Repository Update Script Preview shows the scripts that you must run to update your repository. Run these scripts before continuing the update process.

   Download all of the scripts and move them to a local directory (for example `openidm/update-scripts`). Then, *before* you click Continue, open a command line interface and apply the specified scripts, which vary by repository (as shown in "Repository Update Scripts".

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

   ```
   $ mysql -u root -p < openidm/update-scripts/v999_something_added.sql
   ```

   When you have applied all update scripts shown, select Continue.

7. The update restarts and processes needed files.

8. When you have applied the scripts, select Mark Complete to continue the update process.

9. When the update is complete, go to the console and shut down IDM.

10. The update creates a `repo.opendj.json` file in your *project-dir*/conf directory. This file configures an embedded DS instance, so you can evaluate the software without needing to install a separate database.

    You must delete this file before you continue. The presence of two repository configuration files in your `conf/` directory will cause errors.

11. If you are updating from IDM 5.0.0.2, 5.0.0.3, or 5.0.1.0 to IDM 5.5.0, delete the `forgerock-jaspi-openid-connect-module-21.7.0.jar` file from the `path/to/openidm/bundle` directory:

    ```
    $ rm /path/to/openidm/bundle/forgerock-jaspi-openid-connect-module-21.7.0.jar
    ```

    If you are updating from 5.0.0 or 5.0.0.1 to 5.5.0, this file is deleted automatically.

12. Proceed to "Manage Custom Changes After Updating".

4.3.3. Manage Custom Changes After Updating

If you customized your existing deployment, you may find files with the following extensions: `.old` and `.new`. For more information, see "An Overview of the Update Process".

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

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

- Files with the `.old-unix_time` extension are saved from the configuration you had when starting this update process.

- Files with the `.new-unix_time` extension are files from IDM 5.5.0 that have not been incorporated into your updated installation. For example, if you find a `system.properties.new-unix_time` file in your *project-dir* directory, IDM is still using your pre-update version of this file, which would still be named `system.properties`.

  To take full advantage of IDM 5.5.0, analyze the new features shown in files with the `.new-unix_time` extension. If you have files with multiple `.new-unix_time` extensions, use the file with the latest *unix_time*.

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

4.3.3.1. Migrating BPMN Workflows After an Update

If your deployment includes BPMN workflows, you must migrate the workflow definitions after updating to IDM 5.5.0 or after applying Security Advisory #201705.

For information on updating your workflows, see the corresponding Knowledge Base Article.

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

The update process does not account for any changes that you made to existing standard JSON files such as `sync.json` and `managed.json`. In fact, the update process overwrites these files with the standard IDM 5.5.0 versions of those files.

Do not overwrite the new version 5.5.0 files. Instead, analyze the custom settings from your original JSON files. Review "*What's New*" in the *Release Notes*. Apply each custom setting to the files in your updated deployment and test the results to make sure they still work as intended.

> **Note**
>
> If you've followed the recommendations in "Configuring the Server for Production" in the *Integrator's Guide*, changes you've written directly to JSON files may not be loaded into the repository. This includes JSON files that may have been removed per the `removeFile` code block in "The `update.json` File".
>
> To address these issues, you have two options:
>
> - Write changes directly to the repository, using an HTTP PATCH request.
>
> - Temporarily change production settings that may block configuration updates from JSON files, such as those described in "Disabling Automatic Configuration Updates" in the *Integrator's Guide*.

4.3.3.3. What You Should Do With Your UI After Updating

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

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

> **Warning**
>
> Make sure you've saved any custom files from the `openidm/ui/admin/extension` and `openidm/ui/selfservice/extension` subdirectories, as described in the introduction to "Updating to Version 5.5.0", and then follow these steps:

1. Delete the existing `openidm/ui/admin/extension` and `openidm/ui/selfservice/extension` subdirectories.

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

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

3. Review your UI custom files. Compare them against the IDM 5.5.0 version of these files.

4. Apply your custom changes to each new IDM 5.5.0 UI file in the `openidm/ui/admin/extension` and `openidm/ui/selfservice/extension` subdirectories.

4.3.4. Ready to Start IDM 5.5.0

Now that you've updated IDM and applied customizations from your previous installation, you're ready to restart IDM:

```
$ ./startup.sh -p /path/to/project
```

After you restart IDM, you'll see the following message

```
-> OpenIDM version "5.5.0" (revision: <someUUIDsubset>) jenkins-OpenIDM - Release-num origin/5.5.0
OpenIDM ready
```

4.3.5. Migrating to IDM 5.5.0 on Windows

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

> **Note**
>
> Updates to IDM 5.5.0 on Microsoft Windows remain a manual process.

To perform a migration from IDM 5 to IDM 5.5.0 on Windows, follow these steps. For the purposes of this procedure, the path to the existing instance of IDM 5 is defined as `\path\to\idm-5`. In contrast, the path to the IDM 5.5.0 deployment is defined as `\path\to\idm-5.5.0`:

1. Download and extract IDM 5.5.0 (`IDM-5.5.0.zip`).

2. Stop your existing IDM 5 server, if it is running. Access the Java console where it is running and enter the **shutdown** command at the OSGi console:

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

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

4. Boot properties: On the IDM 5.5.0 server, edit the `conf\boot\boot.properties` file to match any customizations that you made on your IDM 5 server. Specifically, check the following elements:

   - The HTTP, HTTPS, and mutual authentication ports are specified in the `conf\boot\boot.properties` file. If you changed the default ports in your IDM 5 deployment, make sure that the corresponding ports are specified in this file.

   - Check that the keystore and truststore passwords match the current passwords for the keystore and truststore of your IDM 5 deployment.

   Depending on the level of customization you have made in your current deployment, it might be simpler to start with your IDM 5 `boot.properties` file, and copy all customized settings from that file to the corresponding IDM 5.5.0 file. However, as a best practice, you should keep all configuration customizations (including new properties and changed settings) in a single location. You can then copy and paste these changes as appropriate.

5. Security files: Copy the contents of your IDM 5 `security\` folder to the IDM 5.5.0 instance.

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

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

6. Scripts: Migrate any custom scripts or default scripts that you have modified to the scripts directory of your IDM 5.5.0 instance. In general, custom and customized scripts should be located in the `idm-5\script` directory on the IDM 5 deployment:

   • For custom scripts, review "*Compatibility*" in the *Release Notes*. If you're confident that the scripts will work as intended on IDM 5.5.0, then copy these scripts to the new instance. For example:

   ```
   PS C:\> cd \path\to\openidm-5.5.0
   PS C:\> cp \path\to\idm-5\script\my-custom-script.js .\script
   ```

   • If you modified an existing IDM 5 script, compare the default versions of the IDM 5 and IDM 5.5.0 scripts. If you verify that nothing has changed, review what you've customized against "*Compatibility*" in the *Release Notes*. If you're confident that your changes will work as intended, then copy the customized scripts to the new `idm-5.5.0\script` directory. For example:

   ```
   PS C:\> cd \path\to\idm-5.5.0
   PS C:\> cp \path\to\idm-5\script\policy.js .\script\
   ```

   • If a default script has changed since the IDM 5 release, copy the modified script to the `idm-5.5.0\script` directory. For example:

   ```
   PS C:\> cd \path\to\idm-5.5.0
   PS C:\> cp bin\default\script\linkedView.js .\script
   ```

   Check that your customizations work as expected, then port the changes to the new script in the `idm-5.5.0\script` directory.

7. Provisioner files: Modify any customized provisioner configurations in your existing project to point to the connectors that are provided with IDM 5.5.0. Specifically, make sure that the `connectorRef` properties reflect the new connectors, where applicable. For example:

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

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

8. Complete the IDM 5.5.0 installation, as described in "*Preparing to Install and Run Servers*".

9. As there is no automated way to migrate a customized configuration to IDM 5.5.0, we recommend the following strategy:

   - Start with the default IDM 5 configuration.

   - For each configuration file that you have customized, use a file comparison tool such as the Windows **fc.exe** utility to assess the differences between your customized file and the IDM 5.5.0 file.

   - Based on the results of the **fc.exe** review, use either your existing file as a base and port the IDM 5.5.0 changes to that file, or vice versa. Ultimately, you want to preserve your customizations but ensure that you are up to date with the latest default configuration. All files should end up in the `idm-5.5.0/conf` directory.

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

11. Start IDM 5.5.0:

    ```
    PS C:\> cd \path\to\idm-5.5.0
    PS C:\> .\startup.bat
    ```

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

4.4. Updating the Server to 5.5.1.3

This chapter describes how to update IDM 5.5.0 to IDM 5.5.1.3.

Before you begin, review the following sections for IDM 5.5.0, which also apply to version 5.5.1.3:

- "Limitations of the Automated Update Process"

- "An Overview of the Update Process"

- "Prepare Your Update". The only difference is that you need to download one file only: `IDM-5.5.1.3.zip`.

When you have completed the steps in "Prepare Your Update", your instance of IDM is ready for an update to version 5.5.1.3.

Copy the following binary to the `/path/to/openidm/bin/update` directory. That directory is created the first time you start IDM.

- `IDM-5.5.1.3.zip`

Start the update process, using the CLI or the Admin UI. This section includes separate procedures for those options.

4.4.1. Updating to 5.5.1.3 Using the CLI

1. Begin the update process to 5.5.1.3.

```
$ cd /path/to/openidm
$ ./cli.sh update \
  --acceptLicense \
  --skipRepoUpdatePreview
  --user openidm-admin:openidm-admin \
  --url http://localhost:8080/openidm \
  IDM-5.5.1.3.zip
```

The update continues. The process may take a number of minutes. When the update is complete, you'll see the following messages:

```
Repo Updates status: COMPLETE
Exiting maintenance mode...
No longer in maintenance mode.
Resuming the job scheduler.
Scheduler has been resumed.
```

2. Go to the IDM console.

3. When you see the `OpenIDM ready` message at the console, shut down IDM:

```
OpenIDM ready

-> shutdown
```

4. Proceed to "Manage Custom Changes After Updating".

4.4.2. Updating to 5.5.1.3 Using the Admin UI

1. Start IDM:

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

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

3. Select Configure > System Preferences, and choose the Update tab. You should see `IDM-5.5.1.3.zip` in the list of Available Updates.

> **Tip**
>
> If nothing appears under the update tab, refresh your browser.

4. Next to `IDM-5.5.1.3.zip`, select Install to start the update process, then Confirm and Install to enable Maintenance Mode.

5. When the update is complete, go to the console and shut down IDM.

6. Follow the instructions in "Manage Custom Changes After Updating". The process is the same for version 5.5.1.3.

4.4.3. Ready to Start IDM 5.5.1.3

Now that you've updated IDM and applied customizations from your previous installation, you're ready to restart IDM:

```
$ ./startup.sh -p /path/to/project
```

After you restart IDM, you'll see the following message

```
-> OpenIDM version "5.5.1.3" (revision: <someUUIDsubset>) jenkins-OpenIDM - Release-num origin/5.5.1.3
    OpenIDM ready
```

4.4.4. Migrating to IDM 5.5.1.3 on Windows

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

> **Note**
>
> Updates to IDM 5.5.1.3 on Microsoft Windows remain a manual process.

To perform a migration from IDM 5 to IDM 5.5.1.3 on Windows, follow these steps. For the purposes of this procedure, the path to the existing instance of IDM 5 is defined as `\path\to\idm-5`. In contrast, the path to the IDM 5.5.1.3 deployment is defined as `\path\to\idm-5.5.1.3`:

1. Download and extract IDM 5.5.1.3 (`IDM-5.5.1.3.zip`).

2. Stop your existing IDM 5 server, if it is running. Type **shutdown** in the Java console where IDM is running:

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

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

4. Boot properties: On the IDM 5.5.1.3 server, edit the `conf\boot\boot.properties` file to match any customizations that you made on your IDM 5 server. Specifically, check the following elements:

   • The HTTP, HTTPS, and mutual authentication ports are specified in the `conf\boot\boot.properties` file. If you changed the default ports in your IDM 5 deployment, make sure that the corresponding ports are specified in this file.

   • Check that the keystore and truststore passwords match the current passwords for the keystore and truststore of your IDM 5 deployment.

Depending on the level of customization you have made in your current deployment, it might be simpler to start with your IDM 5 `boot.properties` file, and copy all customized settings from that file to the corresponding IDM 5.5.1.3 file. However, as a best practice, you should keep all configuration customizations (including new properties and changed settings) in a single location. You can then copy and paste these changes as appropriate.

5. Security files: Copy the contents of your IDM 5 `security\` folder to the IDM 5.5.1.3 instance.

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

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

6. Scripts: Migrate any custom scripts or default scripts that you have modified to the scripts directory of your IDM 5.5.1.3 instance. In general, custom and customized scripts should be located in the `idm-5\script` directory on the IDM 5 deployment:

   - For custom scripts, review "*Compatibility*" in the *Release Notes*. If you're confident that the scripts will work as intended on IDM 5.5.1.3, then copy these scripts to the new instance. For example:

     ```
     PS C:\> cd \path\to\idm-5.5.1.3
     PS C:\> cp \path\to\idm-5\script\my-custom-script.js .\script
     ```

   - If you modified an existing IDM 5 script, compare the default versions of the IDM 5 and IDM 5.5.1.3 scripts. If you verify that nothing has changed, review what you've customized against "*Compatibility*" in the *Release Notes*. If you're confident that your changes will work as intended, then copy the customized scripts to the new `idm-5.5.1.3\script` directory. For example:

     ```
     PS C:\> cd \path\to\idm-5.5.1.3
     PS C:\> cp \path\to\idm-5\script\policy.js .\script\
     ```

   - If a default script has changed since the IDM 5 release, copy the modified script to the `idm-5.5.1.3\script` directory. For example:

     ```
     PS C:\> cd \path\to\idm-5.5.1.3
     PS C:\> cp bin\default\script\linkedView.js .\script
     ```

   Check that your customizations work as expected, then port the changes to the new script in the `idm-5.5.1.3\script` directory.

7. Provisioner files: Modify any customized provisioner configurations in your existing project to point to the connectors that are provided with IDM 5.5.1.3. Specifically, make sure that the `connectorRef` properties reflect the new connectors, where applicable. For example:

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

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

8. Complete the IDM 5.5.1.3 installation, as described in "*Preparing to Install and Run Servers*".

9. As there is no automated way to migrate a customized configuration to IDM 5.5.1.3, we recommend the following strategy:

   - Start with the default IDM 5 configuration.

   - For each configuration file that you have customized, use a file comparison tool such as the Windows **fc.exe** utility to assess the differences between your customized file and the IDM 5.5.1.3 file.

   - Based on the results of the **fc.exe** review, use either your existing file as a base and port the IDM 5.5.1.3 changes to that file, or vice versa. Ultimately, you want to preserve your customizations but ensure that you are up to date with the latest default configuration. All files should end up in the `idm-5.5.1.3/conf` directory.

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

11. Start IDM 5.5.1.3:

    ```
    PS C:\> cd \path\to\idm-5.5.1.3
    PS C:\> .\startup.bat
    ```

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

4.4.5. Post-Processing Tasks

At this stage, you can read the following sections for IDM 5.5.0, which is similar for version 5.5.1.3:

- "Placing a Server in Maintenance Mode"

- "Update Messages"

You have completed the IDM 5.5.1.3 installation.

4.5. 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
}
```

When the update process starts, maintenance mode is enabled automatically. Before the update process is complete, maintenance mode is disabled. You can disable maintenance mode over REST with the following command:

```
$ curl \
 --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`.

4.6. Update Messages

You can review the events of the update process in the IDM log files. During the update process, you'll find these events in the `openidm0.log.0` file in the `/path/to/openidm/logs` directory.

After the IDM update is complete, it moves that information to a `openidm0.log.1` file, and then creates a new `openidm0.log.0` file. You may find error messages in either file.

When you look through these files, you'll see straightforward INFO messages such as:

- `Upgrading to: 5.5.0`

- `INFO: Updating configuration file`

If there are problems, you may also see self-explanatory messages such as:

- `Encrypted zip files are not supported`

- `Update is not currently running`

Error Messages During the Update Process

Error	Explanation (if needed)
Cannot instantiate service without OSGiFrameworkService	Problem with the OSGi container; the default is Apache Felix, as described in "*Architectural Overview*" in the *Integrator's Guide*
Cannot create temporary directory to unzip archive	Suggests a permissions issue with creating directories, possibly due to a read-only installation of IDM, such as described in "*Installing on a Read-Only Volume*".
Messages related to checksums	Without checksums, IDM can't verify files to be updated. For more information, see "Files Changed During the Update Process"
Messages related to file state	Review "Update File States" for status messages
Messages related to the license file	License file are available in the following directory: `/path/to/openidm/legal-notices`

4.7. Applying Patch Releases

ForgeRock issues periodic patch releases containing bug fixes and improvements to IDM. You can view the available list of patch releases and download them from the ForgeRock BackStage site. To view the key fixes in each patch release, see "Key Fixes" in the *Release Notes*.

If you are satisfied with the changes, you can apply them to your deployment. The following procedure shows how to update your system with a patch release. If your system is Windows-based, see "To Apply a Patch Release on Windows Systems".

To Apply a Patch Release

1. Download the patch release from the ForgeRock BackStage site. For example: download `IDM-<patch-version>-patch.zip`.

   > **Note**
   >
   > Patch releases are cumulative. There is no need to install a previous patch release. For example, you only need to download and install `IDM-<patch-version>-patch.zip`.

2. Unzip the patch file to a temporary folder and compare the contents to `/path/to/openidm-<patch-version>`. If you are comfortable with the changes, continue with the process.

3. Make sure that your server instance is started.

4. Copy the update binary to the `/path/to/openidm/bin/update` directory:

   ```
   $ cp IDM-<patch-version>-patch.zip path/to/openidm/bin/update
   ```

5. Run the update process, using the CLI or the Admin UI. The following command runs the process over the command line. See "Updating From the Admin UI" for a similar process using the Admin UI:

   ```
   $ cd /path/to/openidm

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

6. When the update is complete, restart IDM.

To Apply a Patch Release on Windows Systems

If you are on a Windows-based system, you must apply the patch release manually.

This procedure outlines how to apply the patch release `IDM-<patch-version>-patch.zip` to an existing `IDM <patch-version>` instance located at `\path\to\idm-<patch-version>`.

1. Download the patch release from the ForgeRock BackStage site. For example: download `IDM-<patch-version>-patch.zip`.

2. Unzip the patch file to a temporary folder and compare the contents to `\path\to\idm-<patch-version>`. If you are comfortable with the changes, continue with the process.

3. Shut down the `IDM <patch-version>` instance.

4. Back up your current `IDM <patch-version>` instance located at `\path\to\idm-<patch-version>`.

5. Unzip the `IDM-<patch-version>-patch.zip` to `\path\to\idm-<patch-version>`.

```
cd \path\to\idm-<patch-version>
unzip IDM-<patch-version>-patch.zip
```

6. Inspect the contents of the `\path\to\idm-<patch-version>\bundle*` directory. For example, remove any duplicates that were applied in the patch. Find each instance of `*-IDM-<patch-version>.jar` and remove the corresponding `IDM-<patch-version>.jar`.

   For example, for patch version IDM 5.5.0.3, remove the following affected jar files:

   openidm-authnfilter-5.5.0.3.jar
   openidm-provisioner-openicf-5.5.0.3.jar
   openidm-security-jetty-5.5.0.3.jar
   openidm-ui-admin-5.5.0.3.jar
   script-javascript-5.5.0.3.jar
   openidm-core-5.5.0.3.jar
   openidm-external-rest-5.5.0.3.jar
   openidm-security-5.5.0.3.jar
   openidm-system-5.5.0.3.jar
   script-groovy-5.5.0.3.jar
   openidm-quartz-fragment-5.5.0.3.jar
   openidm-servlet-registrator-5.5.0.3.jar
   script-common-5.5.0.3.jar

7. Remove the `felix-cache` directory.

```
rm -rf felix-cache
```

8. Restart IDM.

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-5.5.1.3.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.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 the org.osgi.framework.storage property. Activate and redirect the felix.cache.rootdir
    property and change them as shown:

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

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

Note

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

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

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

A.3. Additional Details

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

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 a good introduction, see the OSGi site. 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.

Index

A
Application container
    Requirements, 2

R
Repository database
    Table names, 35

U
Update
    Errors, 71
    File Changes, 51
    File Preview, 50, 54
    File Status, 50