OpenIDM 2.1.0 Installation Guide

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

Publication date: March 28, 2013

Abstract

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

Table of Contents

Preface

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

1. Who Should Use this Guide

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

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

This guide also takes you through all of the samples provided with OpenIDM.

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

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

2. Formatting Conventions

Some items are formatted differently from other text, like `filenames`, **commands**, and `literal values`.

```
$ echo Command line sessions are formatted with lines folded for easier reading.
 In HTML documents click the [-] image for a flat, copy-paste version. Click
 the [+] image for an expanded, line-wrapped version. > /dev/null
```

In many cases, sections pertaining to UNIX, GNU/Linux, Mac OS X, BSD, and so forth are marked (UNIX). Sections pertaining to Microsoft Windows might be marked (Windows). To avoid repetition, however, file system directory names are often given only in UNIX format as in `/path/to/openidm`, even if the text applies to `C:\path\to\openidm` as well.

Absolute path names usually begin with the placeholder `/path/to/`, which might translate to `/opt/`, `C:\Program Files\`, or somewhere else on your system. Unless you install from native packages, you create this location before you install.

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

3. Accessing OpenIDM Documentation Online

Core documentation, such as what you are now reading, aims to be technically accurate and complete with respect to the software documented. Core documentation therefore follows a three-phase review process designed to eliminate errors. The review process should slow authors down enough that documentation you get with a stable release has had time to bake fully.

Fully baked core documentation is available at docs.forgerock.org.

The OpenIDM Wiki regularly brings you more, fresh content. In addition, you are welcome to sign up and then edit the Wiki if you notice an error, or if you have something to share.

4. Joining the OpenIDM Community

After you sign up at ForgeRock, you can also login to the Wiki and the issue database to follow what is happening with the project.

If you have questions regarding OpenIDM which are not answered by the documentation, there is a mailing list which can be found at https://lists.forgerock.org/mailman/listinfo/openidm where you are likely to find an answer. You can also make suggestions regarding updates at the documentation mailing list (https://lists.forgerock.org/mailman/listinfo/docs).

The Wiki has information on how to check out OpenIDM source code. There is also a mailing list for OpenIDM development which can be found at https://lists.forgerock.org/mailman/listinfo/openidm-dev. Should you want to contribute a patch, test, or feature, or want to author part of the core documentation, first have a look on the ForgeRock site at how to get involved.

Chapter 1. Installing OpenIDM Services

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

1.1. Before You Run OpenIDM

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

1.1.1. Java Environment

OpenIDM requires Oracle Java SE 6 update 24 or later.

The equivalent version of OpenJDK should work for evaluation, too.

1.1.2. Application Container

OpenIDM services run in an OSGi container with an embedded Servlet container, and an embedded noSQL database. By default the OSGi container is Apache Felix. The default Servlet container is Jetty. For OpenIDM 2.1.0, the only supported configuration is running the services in Apache Felix and Jetty.

1.2. Installing and Running OpenIDM

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

Procedure 1.1. To Install OpenIDM Services

Follow these steps to install OpenIDM.

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

   ```
   $ java -version
   java version "1.6.0_24"
   Java(TM) SE Runtime Environment (build 1.6.0_24-b07-334)
   Java HotSpot(TM) 64-Bit Server VM (build 19.1-b02-334, mixed mode)
   ```

   Check the release notes for Java requirements in the chapter, *Before You Install OpenIDM Software*.

2. Download OpenIDM from one of the following locations:

   - Enterprise Downloads has the latest stable, supported release of OpenIDM and the other products in the ForgeRock identity stack.
   - Builds includes the nightly build, the nightly experimental build, and the OpenIDM agents. Note that this is the working version of the trunk and should not be used in a production environment.

- Archives includes the stable builds for all previous releases of OpenIDM.

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

```
$ cd /path/to
$ unzip ~/Downloads/openidm-2.1.0.zip
...
  inflating: openidm/connectors/scriptedsql-connector-1.1.1.0.jar
  inflating: openidm/bin/felix.jar
  inflating: openidm/bin/openidm.jar
$
```

4. By default, OpenIDM listens for HTTP connections on port 8080. To change the default port, edit `openidm/conf/jetty.xml`.

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

   See the chapter on *Installing a Repository For Production* for details.

Procedure 1.2. To Start OpenIDM Services

Follow these steps to run OpenIDM interactively.

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

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

   - Start OpenIDM (UNIX).

```
$ ./startup.sh
Using OPENIDM_HOME:   /path/to/openidm
Using OPENIDM_OPTS:   -Xmx1024m
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 "2.1.0" (revision: XXXX)
-> OpenIDM ready
```

   - Start OpenIDM (Windows).

```
< cd \path\to\openidm
< startup.bat
"Using OPENIDM_HOME:   \path\to\openidm"
"Using OPENIDM_OPTS:   -Xmx1024m -Dfile.encoding=UTF-8"
"Using LOGGING_CONFIG:
 -Djava.util.logging.config.file=\path\to\openidm\conf\logging.properties"
Using boot properties at \path\to\openidm\conf\boot\boot.properties
OpenIDM version "2.1.0" (revision: XXXX)
```

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

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

```
-> scr list
   Id   State          Name
scr list
   Id   State          Name
   Id   State          Name
[  27] [active       ] org.forgerock.openidm.endpoint
...
[  18] [unsatisfied  ] org.forgerock.openidm.info
[  22] [active       ] org.forgerock.openidm.provisioner.openicf.connectorinfoprovider
[  31] [active       ] org.forgerock.openidm.ui.simple
[  29] [active       ] org.forgerock.openidm.restlet
[   3] [active       ] org.forgerock.openidm.repo.orientdb
[   7] [active       ] org.forgerock.openidm.scope
[   5] [active       ] org.forgerock.openidm.audit
[  33] [unsatisfied  ] org.forgerock.openidm.schedule
[   2] [unsatisfied  ] org.forgerock.openidm.repo.jdbc
[  32] [active       ] org.forgerock.openidm.workflow
[   9] [active       ] org.forgerock.openidm.managed
[  28] [unsatisfied  ] org.forgerock.openidm.provisioner.openicf
[  17] [active       ] org.forgerock.openidm.health
[  21] [active       ] org.forgerock.openidm.provisioner
[   0] [active       ] org.forgerock.openidm.config.starter
[  35] [active       ] org.forgerock.openidm.taskscanner
[  15] [active       ] org.forgerock.openidm.external.rest
[   6] [active       ] org.forgerock.openidm.router
[  34] [active       ] org.forgerock.openidm.scheduler
[  14] [unsatisfied  ] org.forgerock.openidm.external.email
[  11] [unsatisfied  ] org.forgerock.openidm.sync
[  20] [active       ] org.forgerock.openidm.policy
[   8] [active       ] org.forgerock.openidm.script
[  10] [active       ] org.forgerock.openidm.recon
[   4] [active       ] org.forgerock.openidm.http.contextregistrator
[   1] [active       ] org.forgerock.openidm.config
[  13] [active       ] org.forgerock.openidm.endpointservice
[  30] [unsatisfied  ] org.forgerock.openidm.servletfilter
[  19] [active       ] org.forgerock.openidm.infoservice
[  16] [active       ] org.forgerock.openidm.authentication
->
```

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.

2. Alternatively, you can manage the container and services from the Felix administration console.

   Use these hints to connect to the console.

   - Default Console URL: http://localhost:8080/system/console

- Default user name: `admin`
- Default password: `admin`

Some basic hints on using the Felix administration console follow.

- Select the Components tab to see OpenIDM core services and their respective states.
- Select the Shell tab to access the `->` prompt.
- Select the System Information tab to stop or restart the container.

Procedure 1.3. To Get Started With the OpenIDM REST Interface

OpenIDM provides RESTful access to users in the OpenIDM repository. To access the OpenIDM repository over REST, you can use a browser-based REST client, such as the Simple REST Client for Chrome, or RESTClient for Firefox. Alternatively you can use the **curl** command-line utility that is included with most operating systems. For more information on **curl**, see http://curl.haxx.se/. If you cannot locate the **curl** command on your system, you can download it from http://curl.haxx.se/download.html.

1. Access the following URL to get a JSON file including all users in the OpenIDM repository.

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

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

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

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

   - Create `joe` (UNIX).

```
$ curl
 --header "X-OpenIDM-Username: openidm-admin"
 --header "X-OpenIDM-Password: openidm-admin"
 --request PUT
 --data '{
 "userName":"joe",
 "givenName":"joe",
 "familyName":"smith",
 "email":"joe@example.com",
 "phoneNumber":"555-123-1234",
 "password":"TestPassw0rd",
 "description":"My first user"
```

```
 }'
 http://localhost:8080/openidm/managed/user/joe

{"_id":"joe","_rev":"0"}
```

- Create `joe` (Windows).

```
C:\>curl
 --header "X-OpenIDM-Username: openidm-admin"
 --header "X-OpenIDM-Password: openidm-admin"
 --request PUT
 --data "{
 \"userName\":\"joe\",
 \"givenName\":\"joe\",
 \"familyName\":\"smith\",
 \"email\":\"joe@example.com\",
 \"phoneNumber\":\"555-123-1234\",
 \"password\":\"TestPassw0rd\",
 \"description\":\"My first user\"
 }"
 http://localhost:8080/openidm/managed/user/joe

{"_id":"joe","_rev":"0"}
```

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"
 http://localhost:8080/openidm/managed/user/joe

{
  "stateProvince": "",
  "userName": "joe",
  "roles": "openidm-authorized",
  "givenName": "joe",
  "address2": "",
  "lastPasswordAttempt": "Wed Nov 28 2012 22:19:35 GMT+0200 (SAST)",
  "address1": "",
  "familyName": "smith",
  "passwordAttempts": "0",
  "_rev": "0",
  "_id": "joe",
  "country": "",
  "city": "",
  "lastPasswordSet": "",
  "postalCode": "",
  "phoneNumber": "555-123-1234",
  "email": "joe@example.com",
  "description": "My first user",
  "accountStatus": "active"
}
```

OpenIDM returns the JSON object all on one line. To format the JSON for legibility, use a JSON parser, such as jq.

Notice that more attributes are returned for user `joe` than the attributes you added in the previous step. The additional attributes are added by

a script named `onCreate-user-set-default-fields.js` that is triggered whan a new user is created. For more information, see *Managed Object Configuration* in the *Integrator's Guide*.

Procedure 1.4. To Stop the OpenIDM Services

- You can stop OpenIDM Services from the `->` prompt, or through the Felix console.
  - Either enter the **shutdown** command at the `->` prompt.

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

  - Or click Stop on the System Information tab of the Felix console, by default http://localhost:8080/system/console.

    This stops the Servlet container as well, and the console is no longer accessible.

Chapter 2. First OpenIDM Sample

This chapter provides an overview of the first sample and how it is configured. To see a listing and an overview of the rest of the samples provided, see the README found in `openidm/samples` and in the chapter *More OpenIDM Samples*.

2.1. Before You Begin

Install OpenIDM as described in the chapter on *Installing OpenIDM Services*.

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

If you want to query the internal noSQL database, you can download OrientDB (version 1.3.0) from http://code.google.com/p/orient/downloads/ list. You will find the shell console in the `bin` directory. Start OrientDB console using either **console.sh** or **console.bat**, and then connect to the running OpenIDM with the **connect** command.

```
$ /path/to/orientdb-1.3.0/bin/console.sh
>
> connect remote:localhost/openidm admin admin

Connecting to database [remote:localhost/openidm] with user 'admin'...OK

>
```

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

info
Shows classes and records

select * from managed_user
Shows all users in the OpenIDM repository

select * from audit_activity
Shows all activity audit records

This table is created when there is some activity.

select * from audit_recon
Shows all reconciliation audit records

This table is created when you run reconciliation.

You can also use OrientDB Studio to query the default OrientDB repository. After you have installed and started OpenIDM, point your browser to http://

localhost:2480/studio/. The default database is `openidm` and the default user and password are `admin` and `admin`. Click Connect to connect to the repository. For more information about OrientDB Studio, see the OrientDB Studio documentation.

2.2. About the Sample

OpenIDM connects identity data objects held in external resources by mapping one object to another. To connect to external resources, OpenIDM uses OpenICF connectors, configured for use with the external resources.

When objects in one external resource change, OpenIDM determines how the changes affect other objects, and can make the changes as necessary. This sample demonstrates how OpenIDM does this by using *reconciliation* and *synchronization*. OpenIDM reconciliation compares objects in one object set to mapped objects in another object set. Reconciliation can work in write mode, where OpenIDM writes changes to affected objects, or in report mode, where OpenIDM reports on what changes would be written without making the changes. OpenIDM synchronization reflects changes in objects to any mapped objects, making changes as necessary to create or remove mapped objects and links to associate them. For a more thorough explanation of reconciliation and synchronization, see the section on *Types of Synchronization* in the *Integrator's Guide*.

This sample connects to an XML file that holds sample user data. The XML file is configured as the authoritative source. In this sample, users are created in the local repository to show you how you can manage local users through the REST APIs. You can also use OpenIDM without storing managed objects for users in the local repository, instead reconciling and synchronizing objects directly through connectors to external resources.

Furthermore, this sample involves only one external resource. In practice, you can connect as many resources as needed for your deployment.

Sample Configuration Files

You can find configuration files for the sample under the `openidm/samples/sample1/conf` directory. As you review the sample, keep the following in mind.

1. You must start OpenIDM with the sample configuration (**$ ./startup.sh -p samples/sample1**). For more information, see Section 2.3, “Running Reconciliation”.

2. OpenIDM regularly scans for any scheduler configuration files in the `conf` directory.

3. OpenIDM's reconciliation service reads the mappings and actions for the source and target users from `conf/sync.json`.

4. Reconciliation runs, querying all users in the source, and then creating, deleting, or modifying users in the local OpenIDM repository according to the synchronization mappings.

5. OpenIDM writes all operations to the audit logs in both the internal database and also the flat files in the `openidm/audit` directory.

The following configuration files play important roles in this sample.

`samples/sample1/conf/provisioner.openicf-xml.json`
: This connector configuration file serves as the XML file resource. In this sample, the connector instance acts as the authoritative source for users. In the configuration file you can see that the `xmlFilePath` is set to `samples/sample1/data/xmlConnectorData.xml`, which contains two users, in XML format.

  For details on the OpenICF connector configuration files see *Connecting to External Resources* in the *Integrator's Guide*.

`samples/sample1/conf/schedule-reconcile_systemXmlAccounts_managedUser.json`
: The sample schedule configuration file defines a reconciliation job that, if enabled by setting `"enabled" : true`, starts a reconciliation each minute for the mapping named `systemXmlAccounts_managedUser`. The mapping is defined in the configuration file, `conf/sync.json`.

```
{
    "enabled" : false,
    "type": "cron",
    "schedule": "30 0/1 * * * ?",
    "persisted" : true,
    "misfirePolicy" : "fireAndProceed",
    "invokeService": "org.forgerock.openidm.sync",
    "invokeContext": {
        "action": "reconcile",
        "mapping": "systemXmlfileAccounts_managedUser"
    }
}
```

  For information about the schedule configuration see *Scheduling Tasks and Events* in the *Integrator's Guide*.

  Apart from the scheduled reconciliation run, you can also start the reconciliation run through the REST interface. The call to the REST interface is an HTTP POST such as the following.

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

`samples/sample1/conf/sync.json`

This sample configuration file defines the configuration for reconciliation and synchronization. The `systemXmlAccounts_managedUser` is the mapping for the reconciliation. This entry in `conf/sync.json` defines the synchronization mappings between the XML file connector (source) and the local repository (target).

```
{
    "mappings": [
        {
            "name": "systemXmlfileAccounts_managedUser",
            "source": "system/xmlfile/account",
            "target": "managed/user",
            "correlationQuery": {
                "type": "text/javascript",
                "source": "var query = {'_queryId' : 'for-userName',
                    'uid' :  source.name};query;"
            },
            "properties": [
                {
                    "source": "_id",
                    "target": "_id"
                },
                {
                    "source": "description",
                    "target": "description"
                },
                {
                    "source": "firstname",
                    "target": "givenName"
                },
                {
                    "source": "email",
                    "target": "email"
                },
                {
                    "source": "lastname",
                    "target": "familyName"
                },
                {
                    "source": "name",
                    "target": "userName"
                },
                {
                    "source": "password",
                    "target": "password"
                }
                {
                    "source" : "mobileTelephoneNumber",
                    "target" : "phoneNumber"
                },
                {
                    "source" : "securityQuestion",
                    "target" : "securityQuestion"
                },
                {
                    "source" : "securityAnswer",
                    "target" : "securityAnswer"
                },
```

```
                {
                    "source" : "passPhrase",
                    "target" : "passPhrase"
                },
                {
                    "source" : "roles",
                    "target" : "roles"
                }
            ],
            "policies": [
                {
                    "situation": "CONFIRMED",
                    "action": "UPDATE"
                },
                {
                    "situation": "FOUND",
                    "action": "IGNORE"
                },
                {
                    "situation": "ABSENT",
                    "action": "CREATE"
                },
                {
                    "situation": "AMBIGUOUS",
                    "action": "IGNORE"
                },
                {
                    "situation": "MISSING",
                    "action": "IGNORE"
                },
                {
                    "situation": "SOURCE_MISSING",
                    "action": "IGNORE"
                },
                {
                    "situation": "UNQUALIFIED",
                    "action": "IGNORE"
                },
                {
                    "situation": "UNASSIGNED",
                    "action": "IGNORE"
                }
            ]
        }
    ]
}
```

Source and target paths that start with `managed`, such as `managed/user`, always refer to objects in the local OpenIDM repository. Paths that start with `system`, such as `system/xmlfile/account`, refer to connector objects, in this case the XML file connector.

To filter objects from the resource for a particular target, you can use the `validTarget` script in the mapping to ensure that only users who match specified criteria are considered part of the reconciliation. You can use an `onCreate` script in a mapping to set default values for a user created in the target resource. For details on scripting see the *Scripting Reference* appendix in the *Integrator's Guide*.

For more information about synchronization, reconciliation, and `sync.json`, see *Configuring Synchronization* in the *Integrator's Guide*.

2.3. Running Reconciliation

Start OpenIDM with the configuration for sample 1.

```
$ cd /path/to/openidm
$ ./startup.sh -p samples/sample1
```

Reconcile the objects in the resources, either by setting `"enabled" : true` in the schedule configuration file (`conf/schedule-reconcile_systemXmlAccounts_managedUser.json.json`) and then waiting until the scheduled reconciliation happens, or by using the REST interface, as follows:

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

Successful reconciliation returns a reconciliation run ID, similar to the following:

```
{"_id":"2d87c817-3d00-4776-a705-7de2c65937d8"}
```

To see what happened, look at the CSV format log file, `openidm/audit/recon.csv`.

2.4. Viewing Users and Logs

After reconciliation runs, you can use the REST interface to display all users in the local repository. Use a REST client to perform an HTTP GET on the following URL: `http://localhost:8080/openidm/managed/user/?_queryId=query-all-ids` with the headers `"X-OpenIDM-Username: openidm-admin"` and `"X-OpenIDM-Password: openidm-admin"`.

OpenIDM returns a JSON file. Depending on your browser, it can display the JSON or download it as a file. Alternatively, you can use the following **curl** command to get the JSON file.

```
$ 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"
```

```
{
 "query-time-ms":1,
 "result":[
  {
   "_id":"joe",
   "_rev":"0"
  },{
   "_id":"bjensen",
   "_rev":"0"
  },{
   "_id":"scarter",
   "_rev":"0"
  }
  ],
 "conversion-time-ms":0
 }
```

If you created user `joe` as described previously in this guide, you see IDs for three users. The second and third users, `bjensen` and `scarter`, were created during the reconcilillation. Now try a RESTful GET of user `bjensen` by appending the user ID to the managed user URL (`http://localhost:8080/openidm/managed/user/`).

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

{
  "stateProvince": "",
  "userName": "bjensen@example.com",
  "roles": "openidm-authorized",
  "description": "Created By XML1",
  "givenName": "Barbara",
  "address2": "",
  "lastPasswordAttempt": "Mon Dec 17 2012 11:56:56 GMT+0200 (SAST)",
  "address1": "",
  "familyName": "Jensen",
  "passwordAttempts": "0",
  "_rev": "0",
  "_id": "bjensen",
  "securityQuestion": "1",
  "country": "",
  "city": "",
  "lastPasswordSet": "",
  "postalCode": "",
  "phoneNumber": "1234567",
  "email": "bjensen@example.com",
  "accountStatus": "active"
}
```

In the OrientDB console, connect to the database, and then query the users and audit logs. The following shows edited excerpts from a console session querying OrientDB. To make it easier to view the records, the first query only requests three specific fields.

```
> connect remote:localhost/openidm admin admin
```

```
Connecting to database [remote:localhost/openidm] with user 'admin'...OK

> select familyName,email,description from managed_user

---+---------+--------------------+--------------------+--------------------
  #| RID     |familyName          |email               |description
---+---------+--------------------+--------------------+--------------------
  0|    #-2:0|smith               |[1]                 |My first user
  1|    #-2:1|Jensen              |bjensen@example.com |Created By XML1
  2|    #-2:2|Carter              |scarter@example.com |Created By XML1
---+---------+--------------------+--------------------+--------------------

3 item(s) found. Query executed in 0.0040 sec(s).

> select * from audit_activity

---+---------+--------------+------------------+-------------------------+...
  #| RID     |rev           |status            |timestamp                |...
---+---------+--------------+------------------+-------------------------+...
  0|    #-2:0|0             |SUCCESS           |2012-10-26T12:05:50.923Z |...
  1|    #-2:1|0             |SUCCESS           |2012-10-26T12:05:50.966Z |...
  2|    #-2:2|0             |SUCCESS           |2012-10-26T12:05:51.530Z |...
  3|    #-2:3|0             |SUCCESS           |2012-10-26T12:05:51.605Z |...
  ...
 18 item(s) found. Query executed in 0.0090 sec(s).

> select * from audit_recon

---+---------+--------------------+-----------+-----------------------+---------...
  #| RID     |reconId             |status     |timestamp              |message  ...
---+---------+--------------------+-----------+-----------------------+---------...
  0|    #22:0|48650107-66ef-48f...|SUCCESS    |2012-10-26T12:05:50.701Z|Reconcili...
  1|    #22:1|48650107-66ef-48f...|SUCCESS    |2012-10-26T12:05:52.160Z|null     ...
  2|    #22:2|48650107-66ef-48f...|SUCCESS    |2012-10-26T12:05:52.856Z|null     ...
  3|    #22:3|48650107-66ef-48f...|SUCCESS    |2012-10-26T12:05:52.861Z|SOURCE_IG...
---+---------+--------------------+-----------+-----------------------+---------...

4 item(s) found. Query executed in 0.0070 sec(s).
```

This information is also available in the CSV format audit logs located in the `openidm/audit` directory.

```
$ ls /path/to/openidm/audit/
access.csv activity.csv recon.csv
```

2.5. Adding Users in a Resource

Add a user to the source connector XML data file to see reconciliation in action. During the next reconciliation, OpenIDM finds the new user in the source connector, and creates the user in the local repository. To add the user, copy the following XML into `openidm/samples/sample1/data/xmlConnectorData.xml`.

```
<ri:__ACCOUNT__>
    <icf:__UID__>tmorris</icf:__UID__>
    <icf:__NAME__>tmorris@example.com</icf:__NAME__>
    <ri:password>TestPassw0rd#</ri:password>
```

```
    <ri:firstname>Toni</ri:firstname>
    <ri:lastname>Morris</ri:lastname>
    <ri:email>tmorris@example.com</ri:email>
    <ri:mobileTelephoneNumber>1234567</ri:mobileTelephoneNumber>
    <ri:securityQuestion>1</ri:securityQuestion>
    <ri:securityAnswer>Some security answer</ri:securityAnswer>
    <ri:roles>openidm-authorized</ri:roles>
    <icf:__DESCRIPTION__>Created By XML1</icf:__DESCRIPTION__>
 </ri:__ACCOUNT__>
```

Run reconciliation again, as described in the section on *Running Reconciliation*. After reconciliation has run, query the local repository to see the new user appear in the list of all users under `http://localhost:8080/openidm/managed/user/?_queryId=query-all-ids`.

```
$ 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

{
 "query-time-ms":1,
 "result":[{
    "_id":"joe",
    "_rev":"0"
    },{
    "_id":"bjensen",
    "_rev":"0"
    },{
    "_id":"scarter",
    "_rev":"0"
    },{
    "_id":"tmorris",
    "_rev":"0"
    }],
 "conversion-time-ms":0
 }
```

Also look at the reconciliation audit log, `openidm/audit/recon.csv` to see what took place during reconciliation. This formatted excerpt from the log covers the two reconciliation runs done in this sample.

```
"_id",  "action",...,"reconId","situation","sourceObjectId",               "targetObjectId","timestamp";
"7e...","CREATE",...,"486...", "ABSENT",   "system/xmlfile/account/bjensen","managed/user/bjensen",...;
"1a...","CREATE",...,"486...", "ABSENT",   "system/xmlfile/account/scarter","managed/user/scarter",...;
"47...","IGNORE",...,"486...", "UNQUALIFIED",""            ,...,           "managed/user/joe",...;
"33...","UPDATE",...,"aa9...", "CONFIRMED","system/xmlfile/account/bjensen","managed/user/bjensen",...;
"1d...","UPDATE",...,"aa9...", "CONFIRMED","system/xmlfile/account/scarter","managed/user/scarter",...;
"0e...","CREATE",...,"aa9...", "ABSENT",   "system/xmlfile/account/tmorris","managed/user/tmorris",...;
"23...","IGNORE",...,"aa9...", "UNQUALIFIED","",...,                      "managed/user/joe",...;
```

The important fields in the audit log are the action, the situation, the source `sourceObjectId`, and the target `targetObjectId`. For each object in the source, reconciliation results in a situation that leads to an action on the target.

In the first reconciliation run (the abbreviated `reconID` is shown as `486...`), the source object does not exist in the target, resulting in an ABSENT situation and an action to CREATE the object in the target. The object created earlier in the target does not exist in the source, and so is IGNORED.

In the second reconciliation run (the abbreviated `reconID` is shown as `aa9...`), after you added a user to the source XML, OpenIDM performs an UPDATE on the user objects `bjensen` and `scarter` that already exist in the target, in this case changing the internal ID. OpenIDM performs a CREATE on the target for the new user (`tmorris`).

You configure the action that OpenIDM takes based on an object's situation in the configuration file, `conf/sync.json`. For the list of all possible situations and actions, see the *Configuring Synchronization* chapter in the *Integrator's Guide*.

For details on auditing, see the *Using Audit Logs* chapter in the *Integrator's Guide*.

2.6. Adding Users Through REST

You can also add users directly to the local repository through the REST interface. The following example adds a user named James Berg.

Create `james` (UNIX).

```
$ curl
 --header "X-OpenIDM-Username: openidm-admin"
 --header "X-OpenIDM-Password: openidm-admin"
 --request PUT
 --data '{
  "userName":"jberg",
  "familyName":"Berg",
  "givenName":"James",
  "email":"jberg@example.com",
  "phoneNumber":"5556787",
  "description":"Created by OpenIDM REST.",
  "password":"MyPassw0rd"
 }'
 "http://localhost:8080/openidm/managed/user/jberg"

{"_id":"jberg","_rev":"0"}
```

Create `james` (Windows).

```
C:\>curl
 --header "X-OpenIDM-Username: openidm-admin"
 --header "X-OpenIDM-Password: openidm-admin"
 --request PUT
 --data "{
 \"userName\":\"jberg\",
 \"familyName\":\"Berg\",
 \"givenName\":\"James\",
```

```
 \"email\":\"jberg@example.com\",
 \"phoneNumber\":\"5556787\",
 \"description\":\"Created by OpenIDM REST.\",
 \"password\":\"MyPassw0rd\"
 }"
 "http://localhost:8080/openidm/managed/user/jberg"

{"_id":"jberg","_rev":"0"}
```

OpenIDM creates the new user in the repository. If you configure a mapping to apply changes from the local repository to the XML file connector as a target, OpenIDM then updates the XML file to add the new user.

Chapter 3. More OpenIDM Samples

The current distribution of OpenIDM comes with a variety of samples in `openidm/samples/`. Sample 1 is described in *First OpenIDM Sample*. This chapter describes the remaining OpenIDM samples.

3.1. Before You Begin

Install OpenIDM, as described in *Installing OpenIDM Services*.

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

3.1.1. Installing the Samples

Each sample folder in `openidm/samples/` contains a list of sub folders, such as `conf/` and `script/`, depending on which files you need to run the sample. The easiest way to configure a new installation for one of the samples is to use the `-p` option of the `startup` command to point to the directory whose configuration you want to use. Some, but not all samples require additional software, such as an external LDAP server or database.

When you move from one sample to the next, bear in mind that you are changing the OpenIDM configuration. For information on how configuration changes work, see *Changing the Configuration* in the *Integrator's Guide*.

3.1.2. Preparing OpenIDM

Install an instance of OpenIDM specifically to try the samples. That way you can experiment as much as you like, and discard the result if you are not satisfied.

Shut down OpenIDM, and delete the `openidm/felix-cache` directory before you try a new sample.

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

3.2. Sample 1 - XML File

Sample 1 is described in the chapter, *First OpenIDM Sample*.

3.3. Sample 2 - LDAP One Way

Sample 2 resembles the first sample, but in sample 2 OpenIDM is connected to a local LDAP server. The sample has been tested with OpenDJ , but it should work with any LDAPv3 compliant server.

Sample 2 demonstrates how OpenIDM can pick up new or changed objects from an external resource. The sample contains only one mapping, from the external LDAP server resource to the OpenIDM repository. The sample therefore does not push any changes made to OpenIDM managed user objects out to the LDAP server.

3.3.1. LDAP Server Configuration

Sample 2 expects the following configuration for the external LDAP server:

- The LDAP server runs on the local host.
- The LDAP server listens on port 1389.
- A user with DN `cn=Directory Manager` and password `password` has read access to the LDAP server.
- User objects are stored on the LDAP server under base DN `ou=People,dc=example,dc=com`.
- User objects have the object class `inetOrgPerson`.
- User objects have the following attributes:
  - `uid`
  - `sn`
  - `cn`
  - `givenName`
  - `mail`
  - `description`

  An example user object follows.

```
dn: uid=jdoe,ou=People,dc=example,dc=com
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: top
givenName: John
uid: jdoe
cn: John Doe
telephoneNumber: 12345
sn: Doe
mail: jdoe@example.com
description: Created by OpenIDM
```

Prepare the LDAP server by creating a base suffix of `dc=example,dc=com`, and importing these objects from `samples/sample2/data/Example.ldif`.

```
dn: dc=com
objectClass: domain
objectClass: top
dc: com

dn: dc=example,dc=com
objectClass: domain
objectClass: top
dc: example

dn: ou=People,dc=example,dc=com
ou: people
description: people
objectclass: organizationalunit

dn: uid=jdoe,ou=People,dc=example,dc=com
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: top
givenName: John
uid: jdoe
cn: John Doe
telephoneNumber: 12345
sn: Doe
mail: jdoe@example.com
description: Created for OpenIDM
```

3.3.2. Install the Sample

Prepare OpenIDM as described in Section 3.1.2, “Preparing OpenIDM”, then start OpenIDM with the configuration for sample 2.

```
$ cd /path/to/openidm
$ ./startup.sh -p samples/sample2
```

3.3.3. Running the Sample

Run reconciliation over the REST interface.

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

Successful reconciliation returns an "_id" object.

With the configuration of sample 2, OpenIDM creates user objects from LDAP in OpenIDM, assigning the new objects random unique IDs. To list user objects by ID, run a query over the REST interface.

```
$ 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"
```

The resulting JSON object should look something like this, but all on one line.

```
{
    "query-time-ms": 1,
    "result": [
        {
            "_id": "56f0fb7e-3837-464d-b9ec-9d3b6af665c3",
            "_rev": "0"
        }
    ],
    "conversion-time-ms": 0
 }
```

To retrieve the user, get the object by ID.

```
$ curl
 --header "X-OpenIDM-Username: openidm-admin"
 --header "X-OpenIDM-Password: openidm-admin"
 --request GET
 "http://localhost:8080/openidm/managed/user/56f0fb7e-3837-464d-b9ec-9d3b6af665c3"
```

Read `openidm/samples/sample2/conf/sync.json` and `openidm/samples/sample2/conf/provisioner.openicf-ldap.json` to understand the layout of the user object in the repository.

3.4. Sample 2b - LDAP Two Way

Like sample 2, sample 2b also connects to an external LDAP server.

Unlike sample 2, however, sample 2b has two mappings configured, one from the LDAP server to the OpenIDM repository, and the other from the OpenIDM repository to the LDAP server.

3.4.1. External LDAP Configuration

Configure the LDAP server as for sample 2, Section 3.3.1, “LDAP Server Configuration”. The LDAP user must have write access to create users from OpenIDM on the LDAP server.

3.4.2. Install the Sample

Prepare OpenIDM as described in Section 3.1.2, “Preparing OpenIDM”, then start OpenIDM with the configuration for sample 2b.

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

```
$ ./startup.sh -p samples/sample2b
```

3.4.3. Running the Sample

Run reconciliation over the REST interface.

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

Successful reconciliation returns an "_id" object.

With the configuration of sample 2b, OpenIDM creates user objects from LDAP in OpenIDM, assigning the new objects random unique IDs. To list user objects by ID, run a query over the REST interface.

```
$ 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"
```

The resulting JSON object should look something like this, but all on one line.

```
{
    "query-time-ms": 1,
    "result": [
        {
            "_id": "56f0fb7e-3837-464d-b9ec-9d3b6af665c3",
            "_rev": "0"
        }
    ],
    "conversion-time-ms": 0
 }
```

To retrieve the user, get the object by ID.

```
$ curl
 --header "X-OpenIDM-Username: openidm-admin"
 --header "X-OpenIDM-Password: openidm-admin"
 --request GET
 "http://localhost:8080/openidm/managed/user/56f0fb7e-3837-464d-b9ec-9d3b6af665c3"
```

Test the second mapping by creating a user in the OpenIDM repository. On UNIX:

```
$ curl
 --header "X-OpenIDM-Username: openidm-admin"
 --header "X-OpenIDM-Password: openidm-admin"
 --data '{"email":"fdoe@example.com","familyName":"Doe","userName":"fdoe",
 "givenName":"Felicitas","displayName":"Felicitas Doe"}'
```

```
 --request PUT
 "http://localhost:8080/openidm/managed/user/fdoe"
```

On Windows:

```
$ curl
 --header "X-OpenIDM-Username: openidm-admin"
 --header "X-OpenIDM-Password: openidm-admin"
 --request PUT
 --data "{\"email\":\"fdoe@example.com\",\"familyName\":\"Doe\", \"userName\":\"fdoe\",
 \"givenName\":\"Felicitas\",\"displayName\":\"Felicitas Doe\"}"
 "http://localhost:8080/openidm/managed/user/fdoe"
```

Run reconciliation again to create the new user in the LDAP directory.

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

Test that the reconciliation has been successful by locating the new user in the LDAP directory.

```
$ /path/to/OpenDJ/bin/ldapsearch
 --bindDN "cn=Directory Manager"
 --bindPassword password
 --hostname localhost
 --port 1389
 --baseDN "dc=example,dc=com"
 "uid=fdoe"
dn: uid=fdoe,ou=People,dc=example,dc=com
mail: fdoe@example.com
givenName: Felicitas
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: top
uid: fdoe
cn: Felicitas Doe
sn: Doe
```

3.5. Sample 2c - Synchronizing LDAP Group Membership

Like sample 2b, sample 2c also connects to an external LDAP server. The only difference is that in sample 2c, LDAP Group Memberships are synchronized.

3.5.1. External LDAP Configuration

Configure the LDAP server as for sample 2, Section 3.3.1, “LDAP Server Configuration”. The LDAP user must have write access to create users from OpenIDM on the LDAP server.

In addition, two LDAP Groups should be created, which can be found in the LDIF file: `openidm/samples/sample2c/data/Example.ldif`:

```
dn: ou=Groups,dc=example,dc=com
ou: Groups
objectClass: organizationalUnit
objectClass: top

dn: cn=openidm,ou=Groups,dc=example,dc=com
uniqueMember: uid=jdoe,ou=People,dc=example,dc=com
cn: openidm
objectClass: groupOfUniqueNames
objectClass: top

dn: cn=openidm2,ou=Groups,dc=example,dc=com
cn: openidm2
objectClass: groupOfUniqueNames
objectClass: top
```

The user with dn `uid=jdoe,ou=People,dc=example,dc=com` is also imported with the `Example.ldif` file.

3.5.2. Install the Sample

Prepare OpenIDM as described in Section 3.1.2, “Preparing OpenIDM”, then start OpenIDM with the configuration for sample 2c.

```
$ cd /path/to/openidm
$ ./startup.sh -p samples/sample2c
```

3.5.3. Running the Sample

Run reconciliation over the REST interface.

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

Successful reconciliation returns an "_id" object.

With the configuration of sample 2c, OpenIDM creates user objects from LDAP in OpenIDM, assigning the new objects random unique IDs. To list user objects by ID, run a query over the REST interface.

```
$ 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"
```

The resulting JSON object should look something like this, but all on one line.

```
{
    "query-time-ms": 1,
    "result": [
        {
            "_id": "56f0fb7e-3837-464d-b9ec-9d3b6af665c3",
            "_rev": "0"
        }
    ],
    "conversion-time-ms": 0
 }
```

To retrieve the user, get the object by ID.

```
$ curl
 --header "X-OpenIDM-Username: openidm-admin"
 --header "X-OpenIDM-Password: openidm-admin"
 --request GET
 "http://localhost:8080/openidm/managed/user/56f0fb7e-3837-464d-b9ec-9d3b6af665c3"
```

Your user's object should contain a property like:

```
"ldapGroups":["cn=openidm,ou=Groups,dc=example,dc=com"]
```

Now change the user on the OpdenIDM side with the following REST call (on UNIX):

```
$ curl
 --header "X-OpenIDM-Username: openidm-admin"
 --header "X-OpenIDM-Password: openidm-admin"
 --request POST
 -d '[{"replace":"ldapGroups","value": ["cn=openidm2,ou=Groups,dc=example,dc=com"]}]'
 "http://localhost:8080/openidm/managed/user?_action=patch&_queryId=for-userName&uid=jdoe"
```

On Windows, you might need to escape certain characters, so your REST call will look like this:

```
$ curl
 --header "X-OpenIDM-Username: openidm-admin"
 --header "X-OpenIDM-Password: openidm-admin"
 --request POST
 -d "[{\"replace\":\"ldapGroups\",\"value\": [\"cn=openidm2,ou=Groups,dc=example,dc=com\"]}]"
 "http://localhost:8080/openidm/managed/user?_action=patch&_queryId=for-userName&uid=jdoe"
```

This will change the user's `ldapGroups` property in OpenIDM from `"cn=openidm,ou=Groups,dc=example,dc=com"` to `"cn=openidm2,ou=Groups,dc=example,dc=com"` and, as a result, the user will be removed from the one LDAP group and added to the other LDAP group on OpenDJ.

By default, automatic synchronization is enabled. This means that when you update a managed object, any mappings defined in the `sync.json` file are automatically executed to update the target system. For more information, see *Synchronization Mappings File* in the *Integrator's Guide*.

3.6. Sample 2d - Synchronizing LDAP Groups

Sample 2d also connects to an external LDAP server. This sample focuses on LDAP Group synchronization.

3.6.1. External LDAP Configuration

Configure the LDAP server as for sample 2, Section 3.3.1, “LDAP Server Configuration”. The LDAP user must have write access to create users from OpenIDM on the LDAP server.

In addition, two LDAP Groups should be created, which can be found in the LDIF file: `openidm/samples/sample2d/data/Example.ldif` (if they have not already been added through sample 2c):

```
dn: ou=Groups,dc=example,dc=com
ou: Groups
objectClass: organizationalUnit
objectClass: top

dn: cn=openidm,ou=Groups,dc=example,dc=com
uniqueMember: uid=jdoe,ou=People,dc=example,dc=com
cn: openidm
objectClass: groupOfUniqueNames
objectClass: top

dn: cn=openidm2,ou=Groups,dc=example,dc=com
uniqueMember: uid=jdoe,ou=People,dc=example,dc=com
cn: openidm2
objectClass: groupOfUniqueNames
objectClass: top
```

The user with dn `uid=jdoe,ou=People,dc=example,dc=com` is also imported with the `Example.ldif` file.

3.6.2. Install the Sample

Prepare OpenIDM as described in Section 3.1.2, “Preparing OpenIDM”, then start OpenIDM with the configuration for sample 2d.

```
$ cd /path/to/openidm
$ ./startup.sh -p samples/sample2d
```

3.6.3. Running the Sample

Run reconciliation *for the groups mapping* over the REST interface.

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

Successful reconciliation returns an "_id" object.

With the configuration of sample 2d, OpenIDM creates group objects from LDAP in OpenIDM. To list group objects by ID, run a query over the REST interface.

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

The resulting JSON object should look something like this, but all on one line.

```
{
   "query-time-ms":1,
   "result":[
      {
         "_id":"3c704429-aacd-4995-816a-fac33451c642"
      },
      {
         "_id":"b0982152-5099-4358-bdd1-45a39ebe0d77"
      }
   ]
}
```

To retrieve a group, get the object by ID.

```
$ curl
 --header "X-OpenIDM-Username: openidm-admin"
 --header "X-OpenIDM-Password: openidm-admin"
 --request GET
 "http://localhost:8080/openidm/managed/group/b0982152-5099-4358-bdd1-45a39ebe0d77"
```

Your group's object should be similar to the following:

```
{
   "_rev":"0",
   "dn":"cn=openidm,ou=Groups,dc=example,dc=com",
   "_id":"b0982152-5099-4358-bdd1-45a39ebe0d77",
   "description":[],
   "uniqueMember":["uid=jdoe,ou=People,dc=example,dc=com",],
   "name":["openidm"]
}
```

3.7. Sample 3 - Scripted SQL

Sample 3 shows an example configuration for the Scripted SQL connector. The Scripted SQL connector communicates with the database through configurable SQL scripts. Each operation, like create or delete, is represented by its own script.

Prepare a fresh installation of OpenIDM before trying this sample.

3.7.1. External Configuration

In this example OpenIDM communicates with an external MySQL database server.

The sample expects the following configuration for MySQL:

- The database is available on the local host.
- The database listens on port 3306.
- You can connect over the network to the database with user `root` and password `password`.
- MySQL serves a database called `HRDB` with a table called `Users`.
- The database schema is as described in the data definition language file, `openidm/samples/sample3/data/sample_HR_DB.mysql`. Import the file into MySQL before running the sample.

```
$ ./bin/mysql -u root -p < /path/to/openidm/samples/sample3/data/sample_HR_DB.mysql
Enter password:
$
```

Make sure MySQL is running.

3.7.2. Install the Sample

- Prepare OpenIDM as described in Section 3.1.2, “Preparing OpenIDM”.
- OpenIDM requires a MySQL driver, the MySQL Connector/J. Download MySQL Connector/J, version 5.1 or later. Unpack the delivery and copy the .jar into the `openidm/bundle` directory.

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

- In `openidm/samples/sample3/conf/provisioner.openicf-scriptedsql.json`, edit the paths to the scripts (starting with `/opt/openidm/`) to match your installation.
- Start OpenIDM with the configuration for sample 3.

```
$ cd /path/to/openidm
$ ./startup.sh -p samples/sample3
```

If the configuration of the external database is correct, then OpenIDM should show five users during startup, for example:

```
./startup.sh -p samples/sample3
Using OPENIDM_HOME:   /path/to/openidm
Using OPENIDM_OPTS:   -Xmx1024m
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 "2.1.0" (revision: XXXX)
bob
rowley
louis
john
jdoe
```

The check method, executed for each connected resource, executes a `select * from Users` statement.

3.7.3. Run the Sample

The sample 3 `sync.json` configuration file contains a mapping to reconcile OpenIDM and the external database. Run the reconciliation with the following command.

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

Reconciliation creates the five users from the database in the OpenIDM repository. Check the result with the following command.

```
$ 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"
```

The result should resemble the following JSON object.

```
{
  "conversion-time-ms": 0,
  "result": [
    {
      "_rev": "0",
      "_id": "8366a23d-f6cf-46df-9746-469bf45aafcd"
    },
    {
      "_rev": "0",
```

```
      "_id": "3f90933b-9397-4897-84d0-03ed8d99f61e"
    },
    {
      "_rev": "0",
      "_id": "8fbf759d-bebc-42ed-b321-b69487b4470f"
    },
    {
      "_rev": "0",
      "_id": "9592de42-a8ef-4db3-9c6c-7d191e39b084"
    },
    {
      "_rev": "0",
      "_id": "fd962b71-752a-444b-8492-35bff57bec69"
    }
  ],
  "query-time-ms": 1
}
```


To view the JSON for one of the users, get the user by the value of the `_id`.

```
$ curl
 --header "X-OpenIDM-Username: openidm-admin"
 --header "X-OpenIDM-Password: openidm-admin"
 --request GET
 "http://localhost:8080/openidm/managed/user/8366a23d-f6cf-46df-9746-469bf45aafcd"
```

3.8. Sample 4 - CSV File

Sample 4 deals with a comma-separated value file as the external resource. The file name is part of the sample configuration. Therefore you do not need to manage any other external resources.

3.8.1. Install the Sample

No external configuration is required for this sample. Prepare OpenIDM as described in Section 3.1.2, “Preparing OpenIDM”, then start up OpenIDM with the configuration of sample 4.

```
$ cd /path/to/openidm
$ ./startup.sh -p samples/sample4
```

3.8.2. Run the Sample

The `sample4/data/hr.csv` file contains two example users. The first line of the file sets the attribute names. Running reconciliation creates two users in the OpenIDM repository

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

Check the results of reconciliation with the following command.

```
$ 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"
```

The result should resemble the following JSON object, but all on one line.

```
{
    "query-time-ms": 1,
    "result": [
        {
            "_id": "VDART",
            "_rev": "0"
        },
        {
            "_id": "DDOE",
            "_rev": "0"
        }
    ],
    "conversion-time-ms": 0
 }
```

To view the JSON for one of the users, get the user by the value of the `_id`.

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

{
  "password": "Z29vZA==",
  "employeeNumber": "123456",
  "givenName": "Darth",
  "userName": "DDOE",
  "familyName": "Doe",
  "email": "doe@forgerock.org",
  "_rev": "0",
  "_id": "DDOE"
}
```

3.9. Sample 5 - Synchronization of Two Resources

Sample 5 demonstrates the flow of data from one external resource to another. The resources are called LDAP and AD, but in the sample both directory-like resources are simulated with XML files.

3.9.1. Install the Sample

No external configuration is required for this sample. Prepare OpenIDM as described in Section 3.1.2, “Preparing OpenIDM”, then start up OpenIDM with the configuration of sample 5.

```
$ cd /path/to/openidm
$ ./startup.sh -p samples/sample5
```

The XML files that are used are located in the `openidm/samples/sample5/data/` folder. When you start OpenIDM with the sample 5 configuration, it creates `xml_AD_Data.xml`, which does not contain users until you run reconciliation.

3.9.2. Run the Sample

Run reconciliation between OpenIDM and the pseudo-LDAP resource.

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

This command creates a user in the repository and also in the pseudo AD resource, represented by the `samples/sample5/data/xml_AD_Data.xml` file.

3.10. Sample 6 - LiveSync Between Two LDAP Servers

Sample 6 resembles sample 5, but sample 6 uses two real LDAP connections. The default sample configuration assumes two LDAP server instances running on the local host, but listening on different LDAP ports.

To simplify setup, you can configure both provisioners to point to the same LDAP server, and only use different base DNs, so you can simulate use of two directory servers with a single OpenDJ server, for example.

Sample 6 picks up new and changed users from the LDAP suffix, `ou=people,dc=example,dc=com`, and sends updates to the "Active Directory" suffix `ou=people,o=ad`. To keep the example relatively simple, no configuration is provided for the flow from AD to LDAP.

3.10.1. External Configuration

Out of the box, the sample provisioners are configured to use two independent LDAP servers, one listening on port 1389 and one on port 4389.

To simplify your setup, you can change the sample to connect to a single LDAP server representing both external resources by using the same port numbers in both provisioner .json files. For example, change `conf/provisioner.openicf-ad.json` so that the port number line reads `"port" : 1389`.

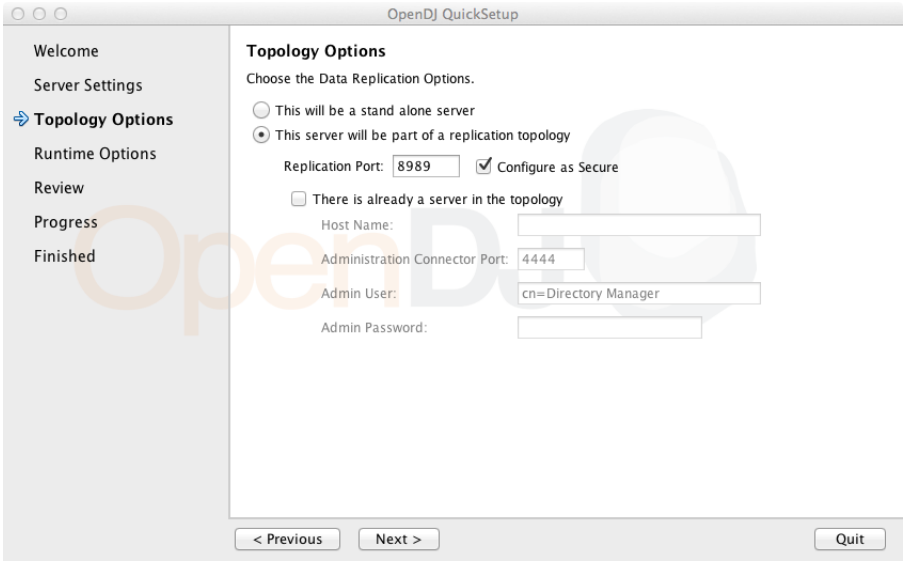
3.10.1.1. Prepare OpenDJ For LiveSync

With LiveSync, OpenIDM detects changes in an external resource as they happen. OpenIDM detects changes in OpenDJ by reading the External

Change Log (ECL). The ECL is presented as an LDAP subtree with base DN `cn=changelog`. Each change is represented as an entry in the subtree. Each change entry remains in the subtree until the log is purged (by default every three days).

You turn on the external change log in OpenDJ by enabling replication. OpenDJ provides the change log even if it does not, in fact, replicate data to another OpenDJ server. Note that OpenDJ will log, in this case, harmless error messages if replication is enabled without a connection to another replica.

To enable replication without another server, set up replication when you install OpenDJ.



3.10.1.2. LDAP Configuration

Sample 6 provides the configuration for two external LDAP servers, set up as follows.

- Both LDAP servers run on the local host.
- The LDAP server "LDAP" listens on port 1389.
- The LDAP server "AD" listens on port 4389. (If you want to use a single LDAP server instance, change this to 1389.)
- Both LDAP servers have a user with DN `cn=Directory Manager` and password `password` who can read and write to the data and read the change log.
- User objects are stored under:

  - Base DN `ou=people,o=ad` for the connector called "AD".
  - Base DN `ou=people,dc=example,dc=com` for the connector called "LDAP".
- User objects have the object class `inetOrgPerson`.
- User objects have the following attributes:
  - `uid`
  - `sn`
  - `cn`
  - `givenName`
  - `mail`
  - `description`

  The LDIF representation of an example user is as follows.

```
dn: uid=jdoe,ou=People,dc=example,dc=com
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: top
givenName: John
uid: jdoe
cn: John Doe
telephoneNumber: 12345
sn: Doe
mail: ddoe@example.com
description: Created by OpenIDM
```

Prepare the LDAP servers by creating the base DN `dc=example,dc=com` on the first server and `o=AD` on the second server, and then importing the following objects.

For the "LDAP" directory, import `samples/sample6/data/Example.ldif`.

```
dn: dc=com
objectClass: domain
objectClass: top
dc: com

dn: dc=example,dc=com
objectClass: domain
objectClass: top
dc: example

dn: ou=People,dc=example,dc=com
```

```
ou: people
description: people
objectclass: organizationalunit

dn: uid=jdoe,ou=People,dc=example,dc=com
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: top
givenName: John
uid: jdoe
cn: John Doe
telephoneNumber: 12345
sn: Doe
mail: jdoe@example.com
description: Created for OpenIDM
```

For the "AD" directory import `samples/sample6/data/AD.ldif`.

```
dn: o=AD
objectClass: domain
objectClass: top
dc: organization

dn: ou=People,o=AD
ou: people
description: people
objectclass: organizationalunit
```

3.10.2. Install the Sample

Prepare OpenIDM as described in Section 3.1.2, “Preparing OpenIDM”, then start OpenIDM with the configuration for sample 6.

```
$ cd /path/to/openidm
$ ./startup.sh -p samples/sample6
```

3.10.3. Running the Sample

The following sections show how to run the sample once off, with reconciliation, and continuously with LiveSync.

3.10.3.1. Using Reconciliation

Start up OpenIDM, and then run reconciliation over the REST interface.

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

The result of a successful reconciliation is an `_id` object.

```
{"_id":"9ece3807-08c3-4ec6-87fb-a6a2d0c71cee"}
```

With the configuration for sample 6, OpenIDM creates user objects from LDAP in the internal repository, and also in the target AD suffix.

After reconciliation, list all users in the 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"
```

The result should resemble the following JSON object.

```
{
  "conversion-time-ms": 0,
  "result": [
    {
      "_rev": "0",
      "_id": "b6b76e9c-d534-4d0a-ac81-87153169a223"
    }
  ],
  "query-time-ms": 0
}
```

To read the user object, use the `_id` value.

```
$ curl
 --header "X-OpenIDM-Username: openidm-admin"
 --header "X-OpenIDM-Password: openidm-admin"
 --request GET
 "http://localhost:8080/openidm/managed/user/b6b76e9c-d534-4d0a-ac81-87153169a223"
```

The result should resemble the following JSON object, though all on one line.

```
{
  "displayName": "John Doe",
  "givenName": "John",
  "userName": "jdoe",
  "familyName": "Doe",
  "description": "Created for OpenIDM",
  "email": "jdoe@example.com",
  "_rev": "0",
  "_id": "b6b76e9c-d534-4d0a-ac81-87153169a223"
}
```

You can also view users created in the AD suffix with the following `ldapsearch` command:

```
$ /path/to/OpenDJ/bin/ldapsearch
 --bindDN "cn=Directory Manager"
 --bindPassword password
 --hostname `hostname`
 --port 4389
 --baseDN o=AD
```

```
 "(uid=*)"

dn: uid=jdoe,ou=people,o=ad
objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: top
givenName: John
description: Created for OpenIDM
uid: jdoe
cn: John Doe
sn: Doe
mail: jdoe@example.com
```

3.10.3.2. Using LiveSync

You can start reconciliation by using a schedule configuration or by using the REST interface directly. However, you must start LiveSync by using a schedule. The sample comes with the following schedule configuration file for LiveSync in `samples/sample6/conf/schedule-activeSynchroniser_systemLdapAccount.json`.

```
{
    "enabled" : false,
    "type" : "cron",
    "schedule" : "0/15 * * * * ?",
    "invokeService" : "provisioner",
    "invokeContext" : {
        "action" : "liveSync",
        "source" : "system/ldap/account"
    },
    "invokeLogLevel" : "debug"
}
```

LiveSync is disabled by default. Activate LiveSync by editing the file, `schedule-activeSynchroniser_systemLdapAccount.json`, to change the "enabled" property value to `true`. With LiveSync enabled, you can add and change LDAP users, and see the changes in AD as OpenIDM flows the data between resources dynamically.

3.11. Sample 7 - Scripting a SCIM-like Schema

Sample 7 demonstrates how you can use OpenIDM to expose user data with a SCIM-like schema. The sample uses the XML file connector to read in attributes from external accounts and construct a JSON object for users stored in the OpenIDM repository. For more information about SCIM schema, see System for Cross-Domain Identity Management: Core Schema 1.1.

3.11.1. Install the Sample

Prepare OpenIDM as described in Section 3.1.2, “Preparing OpenIDM”, then start OpenIDM with the configuration for sample 7.

```
$ cd /path/to/openidm
$ ./startup.sh -p samples/sample7
```

3.11.2. Running the Sample

Run a reconciliation to pull the user from `samples/sample7/data/xmlConnectorData.xml` into the OpenIDM internal repository.

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

Reconciliation creates a user object in the repository. Retrieve the user from the repository.

```
$ curl
 --header "X-OpenIDM-Username: openidm-admin"
 --header "X-OpenIDM-Password: openidm-admin"
 --request GET
 "http://localhost:8080/openidm/managed/user/DDOE1"
```

The user object has the following JSON representation.

```
{
  "groups": [
    {
      "display": "US Employees",
      "value": "usemploys"
    },
    {
      "display": "EU Employees",
      "value": "euemploys"
    }
  ],
  "addresses": [
    {
      "country": "USA",
      "type": "work",
      "locality": "Hollywood",
      "primary": "true",
      "postalCode": "91608",
      "formatted": "100 Universal City Plaza\nHollywood, CA 91608 USA",
      "region": "CA",
      "streetAddress": "100 Universal City Plaza"
    },
    {
      "country": "USA",
      "type": "home",
      "locality": "Hollywood",
      "primary": "false",
      "postalCode": "91622",
      "formatted": "222 Universal City Plaza\nHollywood, CA 91622 USA",
      "region": "CA",
      "streetAddress": "222 Universal City Plaza"
    }
  ],
```

```
"displayName": "John Doe",
"userName": "DDOE1",
"name": {
  "honorificPrefix": "Dr.",
  "honorificSuffix": "III",
  "givenName": "John",
  "formatted": "Dr. John H Doe III",
  "middleName": "Hias",
  "familyName": "Doe"
},
"externalId": "DDOE1",
"emails": [
  {
    "primary": true,
    "value": "hallo@example.com",
    "type": "work"
  },
  {
    "type": "home",
    "value": "jdoe@forgerock.com"
  }
],
"phoneNumbers": [
  {
    "type": "work",
    "value": "1234567"
  },
  {
    "type": "home",
    "value": "1234568"
  }
],
"locale": null,
"ims": [
  {
    "type": "aim",
    "value": "jonyOnAim"
  },
  {
    "type": "skype",
    "value": "skyperHiasl"
  }
],
"schemas": "['urn:scim:schemas:core:1.0']",
"_rev": "0",
"_id": "DDOE1",
"preferredLanguage": "en_US",
"meta": {
  "lastModified": "Tue Dec 04 2012 17:22:56 GMT+0200 (SAST)"
},
"userType": "permanent",
"photos": [
  {
    "type": "photo",
    "value": "https://photos.example.com/profilephoto/72930000000Ccne/F"
  },
  {
    "type": "thumbnail",
    "value": "https://photos.example.com/profilephoto/72930000000Ccne/T"
  }
],
"title": "Mr.Univers",
```

```
  "timezone": "America/Denver",
  "profileUrl": "https://login.example.com/DDOE1",
  "nickName": "Jonny"
}
```

The sample scripts (`samples/sample7/script/setScim*.js`) transform the user data from the resource into the JSON object layout required by SCIM schema.

3.12. Sample 8 - Logging in Scripts

OpenIDM provides a `logger` object with `debug()`, `error()`, `info()`, `trace()`, and `warn()` functions that you can use to log messages to the OpenIDM console from your scripts.

3.12.1. Install the Sample

Prepare OpenIDM as described in Section 3.1.2, “Preparing OpenIDM”, then start OpenIDM with the configuration for sample 8.

```
$ cd /path/to/openidm
$ ./startup.sh -p samples/sample8
```

The scripts under `samples/sample8/script/` show brief log message examples.

3.12.2. Running the Sample

Run reconciliation over the REST interface.

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

Successful reconciliation returns an "_id" object.

Notice the log messages displayed on the OpenIDM (Felix) console. The following example omits timestamps and so forth to show only the message strings.

```
->
...Case no Source: the source object contains: = null
...Case emptySource: the source object contains: = {__UID__=1, email=[mail1@...
...Case sourceDescription: the source object contains: = Created By XML1
...Case onCreate: the source object contains: = {__UID__=1, email=[mail1@...
...Case result: the source object contains: = {UNQUALIFIED={count=0, ids=[]},...
```

3.13. Sample 9 - Asynchronous Reconciliation Using Workflows

Sample 9 demonstrates asynchronous reconciliation using workflows. Reconciliation generates an approval request for each ABSENT user. The configuration for this action is defined in the `conf/sync.json` file, which specifies that an `ABSENT` condition should launch the `managedUserApproval` workflow:

```
...
                {
                    "situation" : "ABSENT",
                    "action" : {
                        "workflowName" : "managedUserApproval",
                        "type" : "text/javascript",
                        "file" : "bin/defaults/script/workflow/workflow.js"
                    }
                },
 ...
```

When the request is approved by an administrator, the absent users are created by an asynchronous reconciliation process.

Prepare a fresh installation of OpenIDM before trying this sample.

3.13.1. Install the Sample

Prepare OpenIDM as described in Section 3.1.2, “Preparing OpenIDM”, then start OpenIDM with the configuration for sample 9.

```
$ cd /path/to/openidm
$ ./startup.sh -p samples/sample9
```

3.13.2. Running the Sample

1. Run reconciliation over the REST interface.

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

   Successful reconciliation returns an "_id" object.

   The reconciliation starts an approval workflow for each ABSENT user. These approval workflows (named `managedUserApproval`) wait for the request to be approved by an administrator.

2. Query the invoked workflow task instances over REST.

```
$ curl
```

```
 --header "X-OpenIDM-Username: openidm-admin"
 --header "X-OpenIDM-Password: openidm-admin"
 --request GET
 "http://localhost:8080/openidm/workflow/taskinstance?_queryId=query-all-ids"
```

The request returns a workflow process ID.

```
{
  "result": [
    {
      "name": "Evaluate request",
      "_id": "13"
    }
  ]
}
```

3. Approve the requests over REST, by setting the `"requestApproved"` parameter for the specified task instance to `"true"`.

   On UNIX:

   ```
   $ curl
    --header "X-OpenIDM-Username: openidm-admin"
    --header "X-OpenIDM-Password: openidm-admin"
    --request POST
    --data '{"requestApproved": "true"}'
    "http://localhost:8080/openidm/workflow/taskinstance/13?_action=complete"
   ```

   On Windows:

   ```
   $ curl
    --header "X-OpenIDM-Username: openidm-admin"
    --header "X-OpenIDM-Password: openidm-admin"
    --request POST
    --data "{\"requestApproved\": \"true\"}"
    "http://localhost:8080/openidm/workflow/taskinstance/13?_action=complete"
   ```

   A successful call returns the following:

   ```
   {"Task action performed":"complete"}
   ```

4. Once the request has been approved, an asynchronous reconciliation operation runs, which creates the users whose accounts were approved in the previous step.

   List the users that were created by the asynchronous reconciliation.

   ```
   $ 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"
```

One user is returned.

```
 {
  "conversion-time-ms": 0,
  "result": [
    {
      "_rev": "0",
      "_id": "1"
    }
  ],
  "query-time-ms": 1
}
```

Chapter 4. Installing a Repository For Production

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

OpenIDM 2.1.0 supports the use of MySQL and MS SQL as an internal repository. For details of the supported versions, see *Before You Install OpenIDM Software* in the *Release Notes*.

Procedure 4.1. To Set Up OpenIDM With MySQL

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

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

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

2. Make sure that OpenIDM is stopped.

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

3. Remove `openidm/conf/repo.orientdb.json`.

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

4. Copy `openidm/samples/misc/repo.jdbc.json` to the `openidm/conf` directory.

   ```
   $ cd /path/to/openidm/conf
   $ cp ../samples/misc/repo.jdbc.json .
   ```

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

   ```
   $ ./bin/mysql -u root -p < /path/to/openidm/db/scripts/mysql/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.

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

Database changed
mysql> show tables;
+---------------------------+
| Tables_in_openidm         |
+---------------------------+
| auditaccess               |
| auditactivity             |
| auditrecon                |
| configobjectproperties    |
| configobjects             |
| genericobjectproperties   |
| genericobjects            |
| internaluser              |
| links                     |
| managedobjectproperties   |
| managedobjects            |
| objecttypes               |
| schedulerobjectproperties |
| schedulerobjects          |
| uinotification            |
+---------------------------+
17 rows in set (0.00 sec)
```

The table names are similar to those used with OrientDB.

6. Update `openidm/conf/repo.jdbc.json` as necessary, to reflect your MySQL deployment.

```
"connection" : {
    "dbType" : "MYSQL",
    "jndiName" : "",
    "driverClass" : "com.mysql.jdbc.Driver",
    "jdbcUrl" : "jdbc:mysql://localhost:3306/openidm",
    "username" : "openidm",
    "password" : "openidm",
    "defaultCatalog" : "openidm",
    "maxBatchSize" : 100,
    "maxTxRetry" : 5,
    "enableConnectionPool" : true
},
```

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

```
$ cd /path/to/openidm
        $ ./startup.sh
        Using OPENIDM_HOME:   /path/to/openidm
        Using OPENIDM_OPTS:   -Xmx1024m
        Using LOGGING_CONFIG:
        -Djava.util.logging.config.file=/path/to/openidm/conf/logging.properties
        Using boot properties at /path/to/openidm/conf/boot/boot.properties
        -> scr list
        Id   State          Name
        [  19] [active       ] org.forgerock.openidm.config.starter
        [  23] [active       ] org.forgerock.openidm.taskscanner
        [   8] [active       ] org.forgerock.openidm.external.rest
        [  12] [active       ] org.forgerock.openidm.provisioner.openicf.connectorinfoprovider
        [  15] [active       ] org.forgerock.openidm.ui.simple
        [   1] [active       ] org.forgerock.openidm.router
        [  22] [active       ] org.forgerock.openidm.scheduler
        [  14] [active       ] org.forgerock.openidm.restlet
        [   7] [unsatisfied  ] org.forgerock.openidm.external.email
        [  18] [unsatisfied  ] org.forgerock.openidm.repo.orientdb
        [   6] [active       ] org.forgerock.openidm.sync
        [   3] [active       ] org.forgerock.openidm.script
        [   5] [active       ] org.forgerock.openidm.recon
        [   2] [active       ] org.forgerock.openidm.scope
        [  10] [active       ] org.forgerock.openidm.http.contextregistrator
        [  20] [active       ] org.forgerock.openidm.config
        [   0] [active       ] org.forgerock.openidm.audit
        [  21] [active       ] org.forgerock.openidm.schedule
        [  17] [active       ] org.forgerock.openidm.repo.jdbc
        [  16] [active       ] org.forgerock.openidm.workflow
        [  13] [active       ] org.forgerock.openidm.provisioner.openicf
        [   4] [active       ] org.forgerock.openidm.managed
        [   9] [active       ] org.forgerock.openidm.authentication
        [  11] [active       ] org.forgerock.openidm.provisioner
```

Procedure 4.2. To Set Up OpenIDM With MS SQL

These instructions are specific to MS SQL Server 2008 R2 Express running on a local Windows XP system. Adapt the instructions for your environment.

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

- OpenIDM requires SQL Server authentication. During the MS SQL Server installation, make sure that you select SQL Server authentication and not just Windows authentication.
- 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.

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

  1. Click Start > All Programs > MS SQL Server 2008 R2 > Configuration Tools > SQL Server Configuration Manager

  2. Expand the SQL Server Network Configuration item and select "Protocols for SQLEXPRESS"

  3. Double click TCP/IP and select Enabled > Yes

  4. Select the IP Adresses 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 Apply, then 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 (SQLEXPRESS) and click Restart.

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

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

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

   a. Click Start > All Programs > MS SQL Server 2008 R2 > SQL Server Management Studio

   b. On the Connect to Server panel, select SQL Server Authentication from the Authentication drop down list and log in as the current user (for example, Administrator).

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

   d. Click Execute to run the script.

2. This step creates an `openidm` database for use as the internal repository, and a user `openidm` with password `Passw0rd` who has all the required

privileges to update the database. You might need to refresh the view in SQL Server Management Studio to see the `openidm` database in the Object Explorer.

Expand Databases > openidm > Tables. You should see the following tables in the openidm database:

The table names are similar to those used with OrientDB.

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

   a. Download the JDBC Driver 4.0 for SQL Server (`sqljdbc_4.0.2206.100_enu.tar.gz`) from Microsoft's download site. The precise URL may vary, depending on your location.

      Extract the executable Java archive file (`sqljdbc4.jar`) from the zip file, using 7-zip or an equivalent file management application.

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

   b. Download the `bnd` Java archive file (`biz.aQute.bnd.jar`) that enables you to create OSGi bundles. The file can be downloaded from http://dl.dropbox.com/u/2590603/bnd/biz.aQute.bnd.jar. For more information about `bnd`, see http://www.aqute.biz/Bnd/Bnd.

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

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

      ```
      .\> ls \path\to\openidm\db\scripts\mssql
       biz.aQute.bnd.jar  openidm.sql  sqljdbc4.bnd  sqljdbc4.jar
      ```

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

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

      This step creates a single `.bar` file, named `sqljdbc4.bar`.

   e. Rename the `sqljdbc4.bar` file to `sqljdbc4-osgi.jar` and copy it to the `openidm\bundle` directory.

      ```
      ./> mv sqljdbc4.bar sqljdbc4-osgi.jar
      ./> cp sqljdbc4-osgi.jar \path\to\openidm\bundle
      ```

4. Remove the default OrientDB repository configuration file (`openidm\conf\repo.orientdb.json`) from the configuration directory.

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

5. Copy the repository configuration file for MS SQL (`openidm\samples\misc\repo.jdbc.json`) to the configuration directory.

```
C:\> cd \path\to\openidm\conf\
.\> cp ..\samples\misc\repo.jdbc-mssql.json .
```

6. Rename the MS SQL repository configuration file to `repo.jdbc.json`.

```
.\> mv repo.jdbc-mssql.json repo.jdbc.json
```

7. Update `openidm\conf\repo.jdbc.json` as necessary, to reflect your MS SQL deployment.

```
{
    "connection" : {
        "dbType" : "SQLSERVER",
        "jndiName" : "",
        "driverClass" : "com.microsoft.sqlserver.jdbc.SQLServerDriver",
        "jdbcUrl" : "jdbc:sqlserver://localhost:1433;instanceName=default;databaseName=openidm;applicat
        "username" : "openidm",
        "password" : "Passw0rd",
        "defaultCatalog" : "openidm",
        "maxBatchSize" : 100,
        "maxTxRetry" : 5,
        "enableConnectionPool" : true
    },
...
```

   Specifically, check that the port matches what you have configured in MS SQL.

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

```
C:> cd \path\to\openidm
        ./> startup.bat
        "Using OPENIDM_HOME:   \path\to\openidm"
        "Using OPENIDM_OPTS:   -Xmx1024m"
        "Using LOGGING_CONFIG:
        -Djava.util.logging.config.file=\path\to\openidm\conf\logging.properties"
        Using boot properties at \path\to\openidm\conf\boot\boot.properties
        -> scr list
        Id   State          Name
        [  19] [active       ] org.forgerock.openidm.config.starter
        [  23] [active       ] org.forgerock.openidm.taskscanner
        [   8] [active       ] org.forgerock.openidm.external.rest
```

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

Chapter 5. Removing and Moving OpenIDM Software

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

Procedure 5.1. To Remove OpenIDM Software

1. Stop OpenIDM services if they are running, by entering `shutdown` at the `->` prompt either on the command line, or on the System Information tab of the Felix console.

```
-> shutdown
```

2. Remove the file system directory where you installed OpenIDM software.

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

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

Procedure 5.2. To Move OpenIDM Software

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

1. Shutdown OpenIDM, as described in *To Stop the OpenIDM Services*.

2. Remove the `felix-cache` directory.

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

3. Move the files.

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

4. Start OpenIDM in the new location.

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

OpenIDM Glossary

JSON	JavaScript Object Notation, a lightweight data interchange format based on a subset of JavaScript syntax. For more information, see the JSON site.
managed object	An object that represents the identity-related data managed by OpenIDM. Managed objects are configurable, JSON-based data structures that OpenIDM stores in its pluggable repository. The default configuration of a managed object is that of a user, but you can define any kind of managed object, for example, groups or roles.
mapping	A policy that is defined between a source object and a target object during reconciliation or synchronization. A mapping can also define a trigger for validation, customization, filtering, and transformation of source and target objects.
OSGi	A module system and service platform for the Java programming language that implements a complete and dynamic component model. For a good introduction, see the OSGi site. OpenIDM services are designed to run in any OSGi container, but OpenIDM currently runs in Apache Felix.
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.
source object	In the context of reconciliation, a source object is a data object on the source system, that OpenIDM scans before attempting to find a corresponding object on the target system. Depending on the defined mapping, OpenIDM

	then adjusts the object on the target system (target object).
synchronization	The synchronization process creates, updates, or deletes objects on a target system, based on the defined mappings from the source system. Synchronization can be scheduled or on demand.
system object	A pluggable representation of an object on an external system. For example, a user entry that is stored in an external LDAP directory is represented as a system object in OpenIDM for the period during which OpenIDM requires access to that entry.System objects follow the same RESTful resource-based design principles as managed objects.
target object	In the context of reconciliation, a target object is a data object on the target system, that OpenIDM scans after locating its corresponding object on the source system. Depending on the defined mapping, OpenIDM then adjusts the target object to match the corresponding source object.

Index