



# Installation Guide

OpenIDM 2.0.3

Mark Craig  
Paul Bryan  
Andi Egloff  
Laszlo Hordos  
Matthias Tristl

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

## Abstract

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



This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License.

To view a copy of this license, visit <https://creativecommons.org/licenses/by-nc-nd/3.0/> or send a letter to Creative Commons, 444 Castro Street, Suite 900, Mountain View, California, 94041, USA.

ForgeRock® and ForgeRock Identity Platform™ are trademarks of ForgeRock Inc. or its subsidiaries in the U.S. and in other countries. Trademarks are the property of their respective owners.

UNLESS OTHERWISE MUTUALLY AGREED BY THE PARTIES IN WRITING, LICENSOR OFFERS THE WORK AS-IS AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE WORK, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF TITLE, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR THE ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO SUCH EXCLUSION MAY NOT APPLY TO YOU.

EXCEPT TO THE EXTENT REQUIRED BY APPLICABLE LAW, IN NO EVENT WILL LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES ARISING OUT OF THIS LICENSE OR THE USE OF THE WORK, EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

DejaVu Fonts

Bitstream Vera Fonts Copyright

Copyright (c) 2003 by Bitstream, Inc. All Rights Reserved. Bitstream Vera is a trademark of Bitstream, Inc.

Permission is hereby granted, free of charge, to any person obtaining a copy of the fonts accompanying this license ("Fonts") and associated documentation files (the "Font Software"), to reproduce and distribute the Font Software, including without limitation the rights to use, copy, merge, publish, distribute, and/or sell copies of the Font Software, and to permit persons to whom the Font Software is furnished to do so, subject to the following conditions:

The above copyright and trademark notices and this permission notice shall be included in all copies of one or more of the Font Software typefaces.

The Font Software may be modified, altered, or added to, and in particular the designs of glyphs or characters in the Fonts may be modified and additional glyphs or characters may be added to the Fonts, only if the fonts are renamed to names not containing either the words "Bitstream" or the word "Vera".

This License becomes null and void to the extent applicable to Fonts or Font Software that has been modified and is distributed under the "Bitstream Vera" names.

The Font Software may be sold as part of a larger software package but no copy of one or more of the Font Software typefaces may be sold by itself.

THE FONT SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF COPYRIGHT, PATENT, TRADEMARK, OR OTHER RIGHT. IN NO EVENT SHALL BITSTREAM OR THE GNOME FOUNDATION BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, INCLUDING ANY GENERAL, SPECIAL, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF THE USE OR INABILITY TO USE THE FONT SOFTWARE OR FROM OTHER DEALINGS IN THE FONT SOFTWARE.

Except as contained in this notice, the names of Gnome, the Gnome Foundation, and Bitstream Inc., shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Font Software without prior written authorization from the Gnome Foundation or Bitstream Inc., respectively. For further information, contact: fonts at gnome dot org.

Arev Fonts Copyright

Copyright (c) 2006 by Tavmjong Bah. All Rights Reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of the fonts accompanying this license ("Fonts") and associated documentation files (the "Font Software"), to reproduce and distribute the modifications to the Bitstream Vera Font Software, including without limitation the rights to use, copy, merge, publish, distribute, and/or sell copies of the Font Software, and to permit persons to whom the Font Software is furnished to do so, subject to the following conditions:

The above copyright and trademark notices and this permission notice shall be included in all copies of one or more of the Font Software typefaces.

The Font Software may be modified, altered, or added to, and in particular the designs of glyphs or characters in the Fonts may be modified and additional glyphs or characters may be added to the Fonts, only if the fonts are renamed to names not containing either the words "Tavmjong Bah" or the word "Arev".

This License becomes null and void to the extent applicable to Fonts or Font Software that has been modified and is distributed under the "Tavmjong Bah Arev" names.

The Font Software may be sold as part of a larger software package but no copy of one or more of the Font Software typefaces may be sold by itself.

THE FONT SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF COPYRIGHT, PATENT, TRADEMARK, OR OTHER RIGHT. IN NO EVENT SHALL TAVMJONG BAH BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, INCLUDING ANY GENERAL, SPECIAL, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF THE USE OR INABILITY TO USE THE FONT SOFTWARE OR FROM OTHER DEALINGS IN THE FONT SOFTWARE.

Except as contained in this notice, the name of Tavmjong Bah shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Font Software without prior written authorization from Tavmjong Bah. For further information, contact: tavmjong @ free . fr.

FontAwesome Copyright

Copyright (c) 2017 by Dave Gandy, <http://fontawesome.io>.

This Font Software is licensed under the SIL Open Font License, Version 1.1. This license is available with a FAQ at: <http://scripts.sil.org/OFL>

---

# Table of Contents

Preface .....	iv
1. Who Should Use this Guide .....	iv
2. Formatting Conventions .....	iv
3. Accessing Documentation Online .....	v
4. Using the ForgeRock.org Site .....	v
1. Installing OpenIDM Services .....	1
1.1. Before You Run OpenIDM .....	1
1.2. Installing & Running OpenIDM .....	1
2. First OpenIDM Sample .....	6
2.1. Before You Begin .....	6
2.2. About the Sample .....	7
2.3. Running Reconciliation .....	10
2.4. Viewing Users & Logs .....	10
2.5. Adding Users in a Resource .....	12
2.6. Adding Users Through REST .....	14
3. More OpenIDM Samples .....	15
3.1. Before You Begin .....	15
3.2. Sample 1 - XML File .....	16
3.3. Sample 2 - LDAP One Way .....	16
3.4. Sample 2b - LDAP Two Way .....	18
3.5. Sample 3 - Scripted SQL .....	20
3.6. Sample 4 - CSV File .....	22
3.7. Sample 5 - Synchronization of Two Resources .....	23
3.8. Sample 6 - LiveSync Between Two LDAP Servers .....	24
4. Installing a Repository For Production .....	31
5. Removing OpenIDM Software .....	34
Index .....	35

# 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, upgrade, 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.

## 2. Formatting Conventions

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

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

Command-line, terminal sessions are formatted as follows:

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

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

Program listings are formatted as follows:

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

## 3. Accessing Documentation Online

ForgeRock publishes comprehensive documentation online:

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

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

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

## 4. Using the ForgeRock.org Site

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

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

## Chapter 1

# Installing OpenIDM Services

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

## 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 JDK 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 2.0.3, the only supported configuration is running the services in Apache Felix and Jetty.

## 1.2. Installing & 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* in the *Release Notes*.

2. Download OpenIDM from the download page.
3. Unpack the contents of the .zip file into the install location.

```
$ cd /path/to  
$ unzip ~/Downloads/openidm-2.0.3.zip  
...  
  inflating: openidm/connectors/openicf-scriptedsql-connector-1.1.0.0.jar  
  inflating: openidm/bin/felix.jar  
  inflating: openidm/bin/openidm.jar  
$
```

4. (Optional) 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 the chapter on *Starting & Stopping OpenIDM* in the *Integrator's Guide* 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  
->
```

- Start OpenIDM (Windows).

```
$ cd \path\to\openidm  
$ startup.bat  
Start in debug mode [1] default  
Start in normal mode [2]  
Show help [3]  
Chose a number (1-3):  
  
Listening for transport dt_socket at address: 5005  
...  
->
```

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
[ 16] [active] org.forgerock.openidm.config.starter
[  7] [active] org.forgerock.openidm.external.rest
[ 11] [active]
org.forgerock.openidm.provisioner.openicf.connectorinfoprovider
[  1] [active] org.forgerock.openidm.router
[ 18] [active] org.forgerock.openidm.scheduler
[ 13] [active] org.forgerock.openidm.restlet
[  6] [unsatisfied] org.forgerock.openidm.external.email
[ 15] [active] org.forgerock.openidm.repo.orientdb
[  5] [active] org.forgerock.openidm.sync
[  3] [active] org.forgerock.openidm.script
[  2] [active] org.forgerock.openidm.scope
[  9] [active] org.forgerock.openidm.http.contextregistrator
[ 17] [active] org.forgerock.openidm.config
[  0] [active] org.forgerock.openidm.audit
[ 14] [unsatisfied] org.forgerock.openidm.repo.jdbc
[  4] [active] org.forgerock.openidm.managed
[ 12] [active] org.forgerock.openidm.provisioner.openicf
[  8] [active] org.forgerock.openidm.authentication
[ 10] [active] org.forgerock.openidm.provisioner
->
```

If startup was successful, all states except `email` and `repo.jdbc` are active. If any other services remain `unsatisfied`, check `openidm/logs` for errors, and refer to the chapter on *Troubleshooting* in the *Integrator's Guide* in the *Integrator's Guide*.

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.

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/?_query-id=query-all-ids
```

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

The **curl** command line tool is included with most operating systems.<sup>1</sup>

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"],  
"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\"],  
\"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.

<sup>1</sup>For more information on **curl**, see <http://curl.haxx.se/>.

```
$ curl  
--header "X-OpenIDM-Username: openidm-admin"  
--header "X-OpenIDM-Password: openidm-admin"  
http://localhost:8080/openidm/managed/user/joe  
  
{  
    "familyName": "smith",  
    "userName": "joe",  
    "givenName": "joe",  
    "_id": "joe",  
    "_rev": "0",  
    "email": [ "joe@example.com" ],  
    "description": "My first user"  
}
```

The JSON output shown above is formatted for easier reading. OpenIDM returns the JSON object all on one line.

#### *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

In OpenIDM 2.0.3, the sample in `openidm/samples/sample1` is configured and enabled by default. This chapter provides an overview of the 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 and start 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, download and unzip OrientDB 1.0. Once you have downloaded and unzipped OrientDB, you can 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-1.0rc6/bin/console.sh
>
> connect remote:localhost/openidm admin admin
Connecting to database [remote:localhost/openidm:localhost/openidm] with user 'admin'...OK
>
```

Once 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.

## 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, the section on *Types of Synchronization* in the *Integrator's Guide* in the *Integrator's Guide*.

This sample connects to an XML file holding 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 samples, keep the following in mind.

1. OpenIDM regularly scans for any scheduler configuration files in the `openidm/conf` directory.
2. OpenIDM's reconciliation service reads the mappings and actions for the source and target users from `openidm/conf-sync.json`.
3. 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.
4. 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.

### openidm/conf/provisioner.openicf-xml.json

This connector configuration file serves for the XML file resource. In this sample, this 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 users in XML format.

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

### openidm/conf/scheduler-reconcile\_systemXmlAccounts\_managedUser.json

The sample scheduler 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": "0 0/1 * * * ?",  
    "invokeService": "org.forgerock.openidm.sync",  
    "invokeContext": {  
        "action": "reconcile",  
        "mapping": "systemXmlfileAccounts_managedUser"  
    }  
}
```

You can also start reconciliation 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/sync?_action=recon&mapping=systemXmlfileAccounts_managedUser"
```

For details on the scheduler configuration see the *Scheduling Synchronization* in the *Integrator's Guide* chapter in the *Integrator's Guide*.

### openidm/conf/sync.json

This sample configuration file defines the configuration for reconciliation and synchronization. The `systemXmlAccounts_managedUser` is the mapping for the reconciliation in the scheduler configuration. 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 = {'_query-id' : 'for-userName',  
                    'userName' : source.name};query;"  
            }  
        }  
    ]  
}
```

```
        },
        "properties": [
            {
                "source": "description",
                "target": "description"
            },
            {
                "source": "firstname",
                "target": "givenName"
            },
            {
                "source": "email",
                "target": "email"
            },
            {
                "source": "lastname",
                "target": "familyName"
            },
            {
                "source": "name",
                "target": "userName"
            },
            {
                "source": "name",
                "target": "_id"
            }
        ],
        "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 starting with `managed`, such as `managed/user`, always refer to objects in the local OpenIDM repository, whereas paths starting 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 only users matching 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* in the *Integrator's Guide* appendix in the *Integrator's Guide*.

For details on synchronization, reconciliation, and `sync.json`, see the *Configuring Synchronization* in the *Integrator's Guide* chapter in the *Integrator's Guide*.

## 2.3. Running Reconciliation

If OpenIDM is not running, start it as described in the procedure *To Start OpenIDM Services*.

Reconcile the objects in the resources either by editing `conf-sync.json` to set `"enabled" : true` and then waiting until the scheduled reconciliation happens, or by using the REST interface.

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

To see what happened, review CSV format log, `openidm/audit/recon.csv`.

## 2.4. Viewing Users & Logs

After reconciliation runs, you can use the REST interface to display all users in the local repository, by performing an HTTP GET on the following URL: `http://localhost:8080/openidm/managed/user/?_query-id=query-all-ids`.

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/?_query-id=query-all-ids

{
  "query-time-ms":1,
  "result":[
    {
      "_id":"joe",
      "_rev":"0"
    },
    {
      "_id":"DDOE1",
      "_rev":"0"
    }
  ],
  "conversion-time-ms":0
}
```

If you created user `joe` as described previously in this guide, you see IDs for two users, the second user `DDOE1` created during reconciliation. Now try a RESTful GET of user `DDOE1` 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"
http://localhost:8080/openidm/managed/user/DDOE1

{
  "familyName": "Doe1",
  "userName": "DDOE1",
  "givenName": "Darth1",
  "_id": "DDOE1",
  "_rev": "0",
  "email": [ "mail1@example.com" ],
  "description": "Created By XML1"
}
```

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.

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

> select * from managed_user

+-----+-----+-----+
#| RID      |familyName      |...|email          |description
+-----+-----+-----+
0| #6:0|smith        |...|[1]           |My first user
1| #6:1|Doe1         |...|[1]           |Created By XML1
+-----+-----+-----+

2 item(s) found. Query executed in 0.011 sec(s).

> select * from audit_activity
```

```

+-----+-----+-----+-----+...
#| RID |rev |status |timestamp |...
+-----+-----+-----+-----+...
0| #11:0|0 |SUCCESS |2011-12-02T07:34:19 |...
1| #11:1|0 |SUCCESS |2011-12-02T07:34:46 |...
+-----+-----+-----+-----+...

2 item(s) found. Query executed in 0.013 sec(s).

> select * from audit_recon

+-----+-----+-----+-----+...
#| RID |timestamp |sourceObjectId |_openidm_id |...
+-----+-----+-----+-----+...
0| #12:0|2011-12-02T07:34:46 |system/xmlfile/account/1|02f5c8fd-0cc4-4a5...
1| #12:1|2011-12-02T07:34:46 |null |4707745d-6b10-4c75-9b...
+-----+-----+-----+-----+...

2 item(s) found. Query executed in 0.01 sec(s).

```

Again, 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_>12345</icf:_UID_>
  <icf:_NAME_>Daffy Duck</icf:_NAME_>
  <icf:_PASSWORD_>123456789</icf:_PASSWORD_>
  <ri:email>daffy.duck@forgerock.com</ri:email>
  <ri:lastname>Duck</ri:lastname>
  <ri:firstname>Daffy</ri:firstname>
</ri:_ACCOUNT_>
```

Run reconciliation 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/?_query-id=query-all-ids`.

```
$ curl  
--header "X-OpenIDM-Username: openidm-admin"  
--header "X-OpenIDM-Password: openidm-admin"  
--request GET  
http://localhost:8080/openidm/managed/user/?_query-id=query-all-ids  
  
{  
  "query-time-ms":1,  
  "result": [  
    {  
      "_id":"DD0E1",  
      "_rev":"0"  
    },  
    {  
      "_id":"joe",  
      "_rev":"0"  
    },  
    {  
      "_id":"Daffy Duck",  
      "_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 two reconciliation runs.

```
"_id", "action",..., "reconId", "situation", "sourceObjectId",...,  
"targetObjectId", "timestamp";  
"02...", "CREATE",..., "cc0...", "ABSENT", "system/xmlfile/account/1",..., "managed/user/DD0E1",...;  
"47...", "IGNORE",..., "cc0...", "UNQUALIFIED", "", ..., "managed/user/joe",...;  
"79...", "UPDATE",..., "d15...", "CONFIRMED", "system/xmlfile/account/1",..., "managed/user/DD0E1",...;  
"af...", "CREATE",..., "d15...", "ABSENT", "system/xmlfile/account/12345",..., "managed/user/Daffy  
Duck",...;  
"23...", "IGNORE",..., "d15...", "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 `cc0...`), 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 `d15...`) after you added a user to the source XML, OpenIDM performs an UPDATE on the user object `DD0E1` that already exists in the target, in this case changing the internal ID. OpenIDM performs a CREATE on the target for the new user.

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* in the *Integrator's Guide* chapter in the *Integrator's Guide*.

For details on auditing, see the *Using Audit Logs* in the *Integrator's Guide* 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 '{
  "name": "james",
  "lastname": "Berg",
  "firstname": "James",
  "email": "james2@examplerock.com",
  "fullname": "hallo2",
  "description": "Created by OpenIDM REST.",
  "userPassword": "asdfkj23"
}'
http://localhost:8080/openidm/managed/user/james

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

Create `james` (Windows).

```
C:\>curl
--header "X-OpenIDM-Username: openidm-admin"
--header "X-OpenIDM-Password: openidm-admin"
--request PUT
--data {
  \"name\": \"james\",
  \"lastname\": \"Berg\",
  \"firstname\": \"James\",
  \"email\": \"james2@examplerock.com\",
  \"fullname\": \"hallo2\",
  \"description\": \"Created by OpenIDM REST.\",
  \"userPassword\": \"asdfkj23\"
}
http://localhost:8080/openidm/managed/user/james

{"_id": "james", "_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, OpenAM next 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/](#). The first, [openidm/samples/sample1](#), is installed by default, and described in the *First OpenIDM Sample* chapter.

## 3.1. 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.

### 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 copy all files in the sample folder into the appropriate folder under [openidm/](#). Some, but not all samples require additional software, such as an external LDAP server or database.

### 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.

Remove the pre-installed `sample1` files before starting with other samples.

```
$ cd /path/to/openidm
$ rm conf/provisioner.openicf-xml.json
  conf/sync.json
  conf/scheduler-reconcile_systemXmlAccounts_managedUser.json
```

After removing the `sample1` files, copy the relevant files from the sample you want to try. For example, if you want to configure OpenIDM to use `sample2` then copy the configuration files from that sample

```
$ cp -r samples/sample2/conf .
```

## 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 sample 1, but in sample 2 OpenIDM is connected to a local LDAP server. The sample has been tested with OpenDJ directory server, 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. Install the Sample

Prepare OpenIDM as described in Section 3.1.2, "Preparing OpenIDM", copying the configuration for sample 2.

```
$ cd /path/to/openidm
$ cp -r samples/sample2/conf .
```

### 3.3.2. 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
```

### 3.3.3. Running the Sample

First start OpenIDM. 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/sync?_action=recon&mapping=systemLdapAccounts_managedUser"
```

Successful reconciliation returns a "reconId" 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/?_query-id=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/conf/sync.json](#) and [openidm/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. Install the Sample

Prepare OpenIDM as described in Section 3.1.2, "Preparing OpenIDM". Copy the sample configuration, and copy the script to the `script` folder.

```
$ cd /path/to/openidm
$ cp -r samples/sample2b/conf samples/sample2b/script .
```

If you already installed sample 2, then simply add the second mapping from sample 2b's `sync.json` file to the current `sync.json` file, and copy the sample script to the `script` folder. The script is referenced in the second mapping.

### 3.4.2. External LDAP Configuration

Configure the LDAP server as for sample 2, Section 3.3.2, "LDAP Server Configuration". The LDAP user must have write access to create users from OpenIDM on the LDAP server.

### 3.4.3. Running the Sample

First start OpenIDM. 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/sync?_action=recon&mapping=systemLdapAccounts_managedUser"
```

Successful reconciliation returns a "reconId" 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/?_query-id=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.

```
$ 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/repoUser1"
```

Run reconciliation again to create the new user in the LDAP server as well.

## 3.5. 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.

Scripts are located in the `openidm/samples/sample3/tools/` folder. The `openidm/samples/sample3/data/` folder contains a data definition language script for setting up the MySQL database schema to hold the external user objects.

Prepare a fresh installation of OpenIDM before trying this sample.

### 3.5.1. Install the Sample

Prepare OpenIDM as described in Section 3.1.2, "Preparing OpenIDM". Copy the `conf` and `tools` folders.

```
$ cd /path/to/openidm  
$ cp -r samples/sample3/conf samples/sample3/tools .
```

In `conf/provisioner.openicf-scriptedsql.json`, edit the paths starting with `/opt/111` to match your installation.

You do not need to copy the `sample3/data/` folder, as its content is needed only to set up the external user database.

In this example OpenIDM communicates with MySQL database server. OpenIDM requires a MySQL driver, the MySQL Connector/J. Download MySQL Connector/J, unpack the delivery, and copy the .jar into the `openidm/bundle` directory.

```
$ cp mysql-connector-java-5.1.18-bin.jar /path/to/openidm/bundle/
```

### 3.5.2. External Configuration

For this sample, OpenIDM connects to an external MySQL database server. Sample 3 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, and then restart OpenIDM to make sure the Connector/J bundle is picked up.

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

If the configuration of the external database is correct, then OpenIDM should show five users during startup. The check method, executed for each connected resource, executes a `select * from Users` statement.

### 3.5.3. 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/sync?_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/?_query-id=query-all-ids"
```

The result should resemble the following JSON object.

```
{"query-time-ms":2,"result":[{"_id":"dc870bff-c2e9-4378-8ac1-45ee085b09bf","_rev":"0"}, {"_id":"9046dea4-1dea-4ae8-8335-070839b12b9c","_rev":"0"}, {"_id":"17fc954f-828a-454c-b05e-f8e9934c6e64","_rev":"0"}, {"_id":"371855d1-44c9-4854-a576-3397275211e4","_rev":"0"}, {"_id":"97066201-e0de-48d9-8c9e-bdf7f0f0c7e5","_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: openidm-admin"  
--header "X-OpenIDM-Password: openidm-admin"  
--request GET  
"http://localhost:8080/openidm/managed/user/dc870bff-c2e9-4378-8ac1-45ee085b09bf"
```

## 3.6. 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.

### 3.6.1. Install the Sample

Prepare OpenIDM as described in Section 3.1.2, "Preparing OpenIDM". Copy the configuration, and copy the `data` folder holding the `.csv` file.

```
$ cd /path/to/openidm  
$ cp -r samples/sample4/conf samples/sample4/data .
```

### 3.6.2. External Configuration

The only external resource you need is the `data/hr.csv` file you copied.

### 3.6.3. Run the Sample

Start up OpenIDM before running the sample if you have not already done so.

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

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/sync?_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/?_query-id=query-all-ids"
```

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

```
{  
    "query-time-ms": 1,  
    "result": [  
        {  
            "_id": "a8c6a158-3e82-450e-8e28-6ebf5a01ala6",  
            "_rev": "0"  
        },  
        {  
            "_id": "750a2375-6983-4e5f-bdbd-7e121e94cf74",  
            "_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/a8c6a158-3e82-450e-8e28-6ebf5a01ala6"  
  
{  
    "userNames": "Doe",  
    "givenName": "Darth",  
    "employeeNumber": "123456",  
    "_id": "a8c6a158-3e82-450e-8e28-6ebf5a01ala6",  
    "_rev": "0",  
    "email": "doe@forgerock.org"  
}
```

## 3.7. 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.

### 3.7.1. Install the Sample

Prepare OpenIDM as described in Section 3.1.2, "Preparing OpenIDM". Copy the configuration, and copy the `script` folder that holds sample JavaScript files.

```
$ cd /path/to/openidm
$ cp -r samples/sample5/conf samples/sample5/script .
```

### 3.7.2. External Configuration

No extra external resource needs configuration for this example. The XML files 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.

### 3.7.3. Run the Sample

Start up OpenIDM before running the sample if you have not already done so.

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

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/sync?_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.

## 3.8. Sample 6 - LiveSync Between Two LDAP Servers

Sample 6 resembles sample 5, but sample 6 uses two real LDAP connections. To simplify setup, both provisioners 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 directory 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.

### 3.8.1. Install the Sample

Prepare OpenIDM as described in Section 3.1.2, "Preparing OpenIDM". Copy the configuration for sample 6.

```
$ cd /path/to/openidm
$ cp -r samples/sample6/conf .
```

### 3.8.2. External Configuration

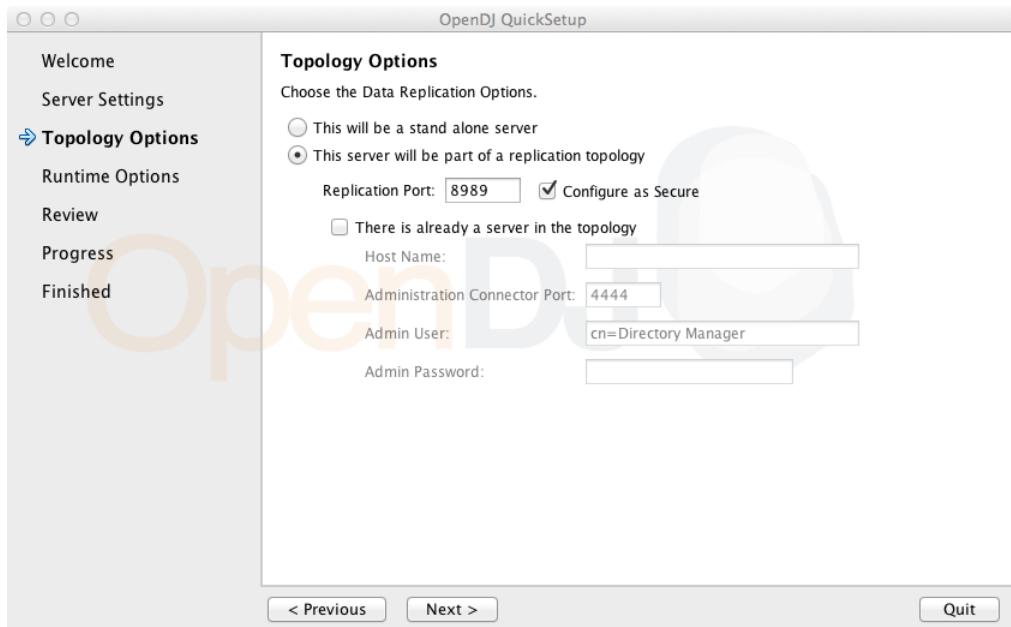
Out of the box, the sample provisioners are configured to use two independent LDAP servers. Change the sample to connect to a single LDAP server representing both external resources by using the same port numbers in both provisioner.json files. For example, change `conf/provisioner.openicf-ad.json` so the port number line reads `"port" : 1389`.

#### 3.8.2.1. 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 three days).

You turn on the 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 (though it can log, in this case, harmless error messages because it is not connected to another replica).

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



### 3.8.2.2. LDAP Configuration

Sample 6 is configuration for an external LDAP server set up as follows.

- The LDAP server runs on the local host.
- The LDAP server "LDAP" listens on port 1389.
- The LDAP server "AD" listens on port 4389. Change this to 1389 to use a single LDAP server.
- The LDAP server both 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 server by creating two base DNs, `dc=example,dc=com` and `o=AD`, 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
```

### 3.8.3. Running the Sample

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

#### 3.8.3.1. Using Reconciliation

Start up OpenIDM, and then run reconciliation.

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

The result of a successful reconciliation is a `reconId` object.

```
{"reconId":"7da56fc0-54de-4c7b-bae3-de7c7a999387"}
```

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

After reconciliation, list all users.

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

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

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

You can also view users created in the AD suffix with the following `ldapsearch` command, assuming you changed the port number to 1389.

```
$ /path/to/OpenDJ/bin/ldapsearch
--bindDN "cn=Directory Manager"
--bindPassword password
--hostname `hostname`
--port 1389
--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
```

### 3.8.3.2. Using LiveSync

In contrast to reconciliation, which you can start by using a scheduler configuration or by using the REST interface directly, you must start LiveSync using a scheduler. The sample comes with the following scheduler configuration file for LiveSync in [conf/scheduler-activeSynchroniser\\_systemLdapAccount.json](#).

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

Activated LiveSync by editing the file, [conf/scheduler-activeSynchroniser\\_systemLdapAccount.json](#), to change the "enabled" property value to [true](#). With LiveSync enabled, you can change LDAP users and see them show up in AD as OpenIDM flows the data between resources dynamically.

## 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, you must however replace OrientDB.

### Procedure 4.1. To Set Up OpenIDM With MySQL

OpenIDM 2.0.3 supports use of MySQL as an internal repository. After installing MySQL on the local host and *before starting OpenIDM for the first time*, perform the following steps.

1. Download MySQL Connector/J, unpack the delivery, and copy the .jar into the `openidm/bundle` directory.

```
$ cp mysql-connector-java-5.1.18-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.

```
$ 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
+-----+
12 rows in set (0.01 sec)
```

The table names are similar to those used with OrientDB.

6. (Optional) Optionally protect access to the `openidm` database by creating a specific user and granting that user all privileges to update the database.
7. Update `openidm/conf/repo.jdbc.json` as necessary, especially the login and password parameters if you have created a specific user for OpenIDM.

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

After setting up MySQL for use as the OpenIDM internal repository, you can start OpenIDM. 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
[ 16] [active]  org.forgerock.openidm.config.starter
[  7] [active]  org.forgerock.openidm.external.rest
[ 11] [active]
org.forgerock.openidm.provisioner.openicf.connectorinfoprovider
[  1] [active]  org.forgerock.openidm.router
[ 18] [active]  org.forgerock.openidm.scheduler
[ 13] [active]  org.forgerock.openidm.restlet
[  6] [unsatisfied] org.forgerock.openidm.external.email
[ 15] [unsatisfied] org.forgerock.openidm.repo.orientdb
[  5] [active]  org.forgerock.openidm.sync
[  3] [active]  org.forgerock.openidm.script
[  2] [active]  org.forgerock.openidm.scope
[  9] [active]  org.forgerock.openidm.http.contextregistrator
[ 17] [active]  org.forgerock.openidm.config
[  0] [active]  org.forgerock.openidm.audit
[ 14] [active]  org.forgerock.openidm.repo.jdbc
[  4] [active]  org.forgerock.openidm.managed
[ 12] [active]  org.forgerock.openidm.provisioner.openicf
[  8] [active]  org.forgerock.openidm.authentication
[ 10] [active]  org.forgerock.openidm.provisioner
```

## Chapter 5

# Removing OpenIDM Software

This chapter shows you how to uninstall OpenIDM software.

### *Procedure 5.1. To Remove OpenIDM Software*

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

# Index

## A

Application container  
    Requirements, 1

## D

Downloading, 2

## G

Getting started, 3, 6

## I

Installing, 1  
    Samples, 15

## J

Java  
    Requirements, 1

## R

Repository database  
    Evaluation version, 6  
    Production ready, 31  
    Requirements, 2  
    Table names, 31

## S

Samples  
    Sample 1 - XML file, 6  
    Sample 2 - LDAP one way, 16  
    Sample 2b - LDAP two way, 18  
    Sample 3 - Scripted SQL, 20  
    Sample 4 - CSV file, 22  
    Sample 5 - Synchronization of two resources, 23  
    Sample 6 - LiveSync between two LDAP  
        servers, 24  
Starting OpenIDM, 2  
Stopping OpenIDM, 5

## U

Uninstalling, 34