OpenIDM Samples Guide

Version 4.5

Lana Frost
Mike Jang

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

Copyright © 2011-2016 ForgeRock AS.

Abstract

This guide provides a number of "sample deployments" that walk you through the essential OpenIDM features, as they would be implemented.

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 v
1. Who Should Use This Guide v
2. Formatting Conventions v
3. Accessing Documentation Online vi
4. Joining the ForgeRock Community vi
1. Overview of the OpenIDM Samples 1
1.1. Overview of the Samples Provided With OpenIDM 1
1.2. Installing the Samples 5
1.3. Preparing OpenIDM 5
2. XML Samples - Reconciling Data Between OpenIDM and an XML File 6
2.1. First OpenIDM Sample - Reconciling an XML File Resource 6
2.2. Logging Sample - Using Scripts to Generate Log Messages 17
2.3. Workflow Sample - Demonstrating Asynchronous Reconciling Using a Workflow 18
3. LDAP Samples - Reconciling Data Between OpenIDM and One or More LDAP Directories 22
3.1. Sample 2 - LDAP One Way 22
3.2. Sample 2b - LDAP Two Way 27
3.3. Sample 2c - Synchronizing LDAP Group Membership 31
3.4. Sample 2d - Synchronizing LDAP Groups 37
3.5. Sample 5 - Synchronization of Two LDAP Resources 41
3.6. Sample 5b - Failure Compensation With Multiple Resources 44
3.7. Sample 6 - LiveSync With an AD Server 47
3.8. Linking Historical Accounts 54
3.9. Storing Multiple Passwords For Managed Users 63
4. Samples That Use the Groovy Connector Toolkit to Create Scripted Connectors 77
4.1. Sample 3 - Using the Custom Scripted Connector Bundler to Build a ScriptedSQL Connector 77
4.2. Sample - Using the Groovy Connector Toolkit to Connect to OpenDJ With ScriptedREST 88
4.3. Using the Groovy Connector Toolkit to Connect to OpenDJ With ScriptedCREST 96
5. Samples That Use the PowerShell Connector Toolkit to Create Scripted Connectors 105
5.1. Connect to Active Directory 105
5.2. Connect to Azure AD 115
6. Audit Samples 133
6.1. Directing Audit Information To a MySQL Database 133
6.2. Show Audit Events Published on a JMS Topic 136
7. Roles Samples - Demonstrating the OpenIDM Roles Implementation 140
7.1. CRUDOPS Role Sample - Working With Managed Roles 141
7.2. Provisioning Role Sample - Provisioning to an LDAP Server 152
7.3. Temporal Constraints Sample - Applying Time-Based Constraints to Roles .. 173
8. The Multi-Account Linking Sample 181
8.1. External LDAP Configuration 181

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

8.2. Install the Sample ... 181
8.3. Create New Identities for the Sample ... 181
8.4. Create New Roles for the Sample ... 183
8.5. Assign Roles to Appropriate Users ... 184
8.6. Background: Link Qualifiers, Agents, and Insured Customers ... 186
8.7. Update Roles With Desired LDAP Attributes ... 189
8.8. Reconciling Managed Users to the External LDAP Server ... 191
8.9. Reviewing the Result ... 192
9. The Trusted Servlet Filter Sample ... 193
9.1. Before You Start ... 193
9.2. The Sample Servlet Filter ... 193
9.3. Run the Sample ... 194
9.4. Create a Trusted User ... 194
9.5. Customizing the Sample for an External System ... 195
10. Full Stack Sample - Using OpenIDM in the ForgeRock Identity Platform ... 197
10.1. External OpenDJ Configuration ... 197
10.2. OpenAM Server Configuration ... 198
10.3. Install the Sample ... 198
10.4. Configuring OpenIDM for the Full Stack Sample ... 198
10.5. Run the Sample ... 205
10.6. Verify the Sample on OpenAM ... 205
11. Workflow Samples ... 207
11.1. Sample Workflow - Provisioning User Accounts ... 207
11.2. Workflow Use Cases ... 212
12. Google Sample - Connecting to Google With the Google Apps Connector ... 230
12.1. Before You Start ... 230
12.2. Configuring the Google Apps Connector ... 231
12.3. Running the Google Apps Sample ... 234
13. Salesforce Sample - Salesforce With the Salesforce Connector ... 241
13.1. Before you Start ... 241
13.2. Install the Sample ... 244
13.3. Running the Sample by Using the Admin UI ... 244
13.4. Running the Sample by Using the Command Line ... 249
14. Scripted Kerberos Connector Sample ... 255
14.1. Editing the Kerberos Connector Configuration ... 255
14.2. Running the Kerberos Sample ... 256
15. Custom Endpoint Sample ... 264
OpenIDM Glossary ... 268
Index ... 271

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

Preface

This guide describes a number of sample deployments that demonstrate the core functionality of OpenIDM. The samples correspond to the configurations provided in the `openidm/samples` directory.

1. Who Should Use This Guide

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

The guide covers a number OpenIDM features, often including multiple features in a single sample.

You do not need to be an OpenIDM wizard to learn something from this guide, although 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.");
    }
}
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

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. Joining the ForgeRock Community

Visit the Community resource center where you can find information about each project, browse the resource catalog, ask and answer questions on the forums, find community events near you, and find the source code for open source software.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

Chapter 1

Overview of the OpenIDM Samples

This chapter lists all the samples provided with OpenIDM and gives a high-level overview of the purpose of each sample. This chapter also provides information that is required for all of the samples. Read this chapter, specifically Section 1.2, "Installing the Samples" and Section 1.3, "Preparing OpenIDM" before you try any of the samples provided with OpenIDM.

1.1. Overview of the Samples Provided With OpenIDM

OpenIDM provides a number of samples in the `openidm/samples` directory. This section describes the purpose of each sample:

Chapter 2, "*XML Samples - Reconciling Data Between OpenIDM and an XML File*"

The XML samples all use the XML file connector to interact with an XML file resource. The samples demonstrate the following OpenIDM functionality:

- Section 2.1, "First OpenIDM Sample - Reconciling an XML File Resource"

  The basic XML sample demonstrates a connection to an XML file that holds user data. The sample includes one mapping and shows reconciliation from an XML file to the OpenIDM repository.

- Section 2.2, "Logging Sample - Using Scripts to Generate Log Messages"

  The logging sample demonstrates the logging capabilities available to OpenIDM scripts, and provides an alternative method for debugging scripts.

- Section 2.3, "Workflow Sample - Demonstrating Asynchronous Reconciling Using a Workflow"

  The workflow sample uses the XML connector to demonstrate asynchronous reconciliation using the workflow mechanism.

Chapter 3, "*LDAP Samples - Reconciling Data Between OpenIDM and One or More LDAP Directories*"

The LDAP samples all assume a connection to an LDAP directory, usually OpenDJ, or Active Directory. Samples 5 and 5b simulate an LDAP directory with an XML file, and use the XML connector. These samples demonstrate a wide variety of OpenIDM functionality and are broken down as follows:

- Section 3.1, "Sample 2 - LDAP One Way"

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

This sample uses the generic LDAP connector to connect to an LDAP directory. The sample includes one mapping from the LDAP directory to the managed user repository, and demonstrates reconciliation from the external resource to the repository.

- Section 3.2, "Sample 2b - LDAP Two Way"

  This sample uses the generic LDAP connector to connect to an LDAP directory. The sample includes two mappings, one from the LDAP directory to the managed user repository, and one from the repository to the LDAP directory. The sample demonstrates reconciliation in both directions.

- Section 3.3, "Sample 2c - Synchronizing LDAP Group Membership"

  This sample uses the generic LDAP connector to connect to an LDAP directory. The sample includes two mappings, one from the LDAP directory to the managed user repository, and one from the repository to the LDAP directory. The sample demonstrates synchronization of group membership, that is, how the value of the `ldapGroups` property in a managed user object is mapped to the corresponding user object in LDAP.

- Section 3.4, "Sample 2d - Synchronizing LDAP Groups"

  This sample uses the generic LDAP connector to connect to an LDAP directory. The sample builds on the previous sample by providing an additional mapping, from the LDAP groups object, to the managed groups object. The sample illustrates a new managed object type (groups) and shows how this object type is synchronized with group containers in LDAP.

- Section 3.5, "Sample 5 - Synchronization of Two LDAP Resources"

  Although this sample is grouped with the LDAP samples, it actually *simulates* two LDAP directories with XML files, and uses the XML file connector to connect the two. The purpose of this sample is to demonstrate reconciliation directly between two external resources, without the data passing through the OpenIDM repository. The sample also demonstrates the configuration of an outbound email service to send reconciliation summaries by mail.

- Section 3.6, "Sample 5b - Failure Compensation With Multiple Resources"

  This sample builds on the previous sample to demonstrate a failure compensation mechanism that relies on script event hooks. The failure compensation mechanism ensures that reconciliation changes are propagated throughout a multiple-resource deployment, or rolled back in the case of error. The purpose of this mechanism is to keep the data consistent across multiple resources.

- Section 3.7, "Sample 6 - LiveSync With an AD Server"

  This sample illustrates the LiveSync mechanism that pushes changes from an external resource to the OpenIDM repository. The sample uses an LDAP connector to connect to an LDAP directory, either OpenDJ or Active Directory.

- Section 3.8, "Linking Historical Accounts"

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

This sample demonstrates the retention of inactive (historical) LDAP accounts that have been linked to a corresponding managed user account. The sample builds on sample 2b and uses the LDAP connector to connect to an OpenDJ instance. You can use any LDAP-v3 compliant directory server.

- Section 3.9, "Storing Multiple Passwords For Managed Users"

  This sample demonstrates how to set up multiple passwords for managed users and how to synchronize separate passwords to different external resources. The sample includes two target LDAP servers, each with different password policy and encryption requirements. The sample also shows how to extend the password history policy to apply to multiple password fields.

Chapter 4, "*Samples That Use the Groovy Connector Toolkit to Create Scripted Connectors*"

The samples in this section use the Groovy Connector Toolkit to create a scripted connector. Because you can use scripted Groovy connectors to connect to a large variety of systems, the samples in this section show connections to several different external resources. The samples are broken down as follows:

- Section 4.1, "Sample 3 - Using the Custom Scripted Connector Bundler to Build a ScriptedSQL Connector"

  This sample uses the *custom scripted connector bundler* to create a new scripted connector. The connector bundler generates a scripted connector, the connector configuration and the Groovy scripts required to communicate with an external MySQL database (HRDB).

- Section 4.2, "Sample - Using the Groovy Connector Toolkit to Connect to OpenDJ With ScriptedREST"

  This sample uses the Groovy Connector Toolkit to implement a ScriptedREST connector, which interacts with the OpenDJ REST API.

- Section 4.3, "Using the Groovy Connector Toolkit to Connect to OpenDJ With ScriptedCREST"

  This sample uses the Groovy Connector Toolkit to create a ScriptedCREST connector, which connects to an OpenDJ instance. The main difference between a CREST-based API and a generic REST API is that the CREST API is inherently recognizable by all ForgeRock products. As such, the sample can leverage CREST resources in the groovy scripts, to create CREST requests.

Chapter 5, "*Samples That Use the PowerShell Connector Toolkit to Create Scripted Connectors*"

This sample uses the PowerShell Connector Toolkit to create a PowerShell connector, and provides a number of PowerShell scripts that enable you to perform basic CRUD (create, read, update, delete) operations on an Active Directory server. The samples use the MS Active Directory PowerShell module.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

Chapter 14, "*Scripted Kerberos Connector Sample*"

This sample demonstrates how to use the scripted Kerberos connector to manage Kerberos user principals and to reconcile user principals with OpenIDM managed user objects.

Chapter 6, "*Audit Samples*"

This sample uses a ScriptedSQL implementation of the Groovy Connector Toolkit to direct audit information to a MySQL database.

Chapter 7, "*Roles Samples - Demonstrating the OpenIDM Roles Implementation*"

This sample builds on Section 3.1, "Sample 2 - LDAP One Way", and extends that sample to demonstrate how roles are implemented in OpenIDM.

Chapter 8, "*The Multi-Account Linking Sample*"

This sample illustrates how OpenIDM addresses links from multiple accounts to one identity.

Chapter 9, "*The Trusted Servlet Filter Sample*"

This sample demonstrates how to use a custom servlet filter and the Trusted Request Attribute Authentication Module in OpenIDM. Once configured, OpenIDM can use the servlet filter to authenticate through another service.

Chapter 10, "*Full Stack Sample - Using OpenIDM in the ForgeRock Identity Platform*"

This sample demonstrates the integration of three ForgeRock products: OpenIDM, OpenDJ, and OpenAM. With this sample, you can see how you can use OpenAM for authentication, for user identities that are maintained with OpenIDM, based on a data store of users in OpenDJ.

Chapter 11, "*Workflow Samples*"

The workflow sample and use cases demonstrate how OpenIDM uses workflows to provision user accounts. The samples demonstrate the use of the Self-Service UI to enable user self-registration,

- Section 11.1, "Sample Workflow - Provisioning User Accounts"

  The provisioning workflow sample demonstrates a typical use case of a workflow — provisioning new users. The sample demonstrates the use of the Admin UI, to configure user self-service and the Self-Service UI that enables users to complete their registration process.

- Section 11.2, "Workflow Use Cases"

  The workflow use cases work together to provide a complete business story, with the same set of sample data. Each of the use cases is integrated with the Self-Service UI.

Chapter 12, "*Google Sample - Connecting to Google With the Google Apps Connector*"

This sample uses the Google Apps Connector to manage the creation of users and groups on an external Google system, using OpenIDM's REST interface.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

Chapter 13, "*Salesforce Sample - Salesforce With the Salesforce Connector*"

This sample uses the Salesforce Connector demonstrate reconciliation of user accounts from the OpenIDM repository to Salesforce, and from Salesforce to the OpenIDM repository.

Chapter 15, "*Custom Endpoint Sample*"

OpenIDM supports scriptable custom endpoints that enable you to launch arbitrary scripts through an OpenIDM REST URI. This sample shows how custom endpoints are configured and returns a list of variables available to each method used in a custom endpoint script.

1.2. Installing the Samples

Each sample directory in `openidm/samples/` contains a number of subdirectories, such as `conf/` and `script/`. To start OpenIDM with a sample configuration, navigate to the `/path/to/openidm` directory and use the `-p` option of the **startup** command to point to the sample whose configuration you want to use. Some, but not all samples require additional software, such as an external LDAP server or database.

Many of the procedures in this guide refer to paths such as `samplex/...`. In each of these cases, the complete path is assumed to be `/path/to/openidm/samples/samplex/...`.

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 Section 6.2, "Changing the Default Configuration" in the *Integrator's Guide*.

The command-line examples in this chapter (and throughout the OpenIDM documentation) assume a UNIX shell. If you are running these samples on Windows, adjust the command-line examples accordingly. For an indication of what the corresponding Windows command would look like, see the examples in Section 2.1, "First OpenIDM Sample - Reconciling an XML File Resource".

1.3. 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.

If you are using the same instance of OpenIDM for multiple samples, it is helpful to clear out the repository created for an earlier sample. To do so, shut down OpenIDM and delete the `openidm/db/openidm` directory.

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

OpenIDM should then be ready to start with a new sample. For a number of the samples in this guide, users are created either with the UI or directly with a commons REST call. Users that have been created in the repository (managed users) should be able to log into the Self-Service UI.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

Chapter 2

XML Samples - Reconciling Data Between OpenIDM and an XML File

This chapter walks you through the XML samples (those samples labeled Sample 1, Sample 8, and Sample 9 in the `openidm/samples` directory. For a complete list of the samples provided with OpenIDM, and an overview of each sample, see Chapter 1, "*Overview of the OpenIDM Samples*".

2.1. First OpenIDM Sample - Reconciling an XML File Resource

This chapter provides an overview of the first sample and how it is configured. For a complete list of the samples provided with OpenIDM, and an overview of each sample, see Chapter 1, "*Overview of the OpenIDM Samples*" or the README in the `openidm/samples` directory.

2.1.1. About the XML Sample

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

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*. OpenIDM reconciliation compares the objects in one object set to mapped objects in another object set. For a complete explanation of reconciliation and synchronization, see Section 12.1, "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 as well as through the OpenIDM UI.

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.

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

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

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:

- Start OpenIDM with the configuration associated with Sample 1:

  ```
  $ ./startup.sh -p samples/sample1
  ```

  For more information, see Section 2.1.2, "Install the Sample".

- OpenIDM regularly scans for any scheduler configuration files in the `conf` directory.
- OpenIDM's reconciliation service reads the mappings and actions for the source and target users from `conf/sync.json`.
- When you initiate a reconciliation, OpenIDM queries all users in the source, and then creates, deletes, or modifies users in the local OpenIDM repository as mapped in `conf/sync.json`.
- OpenIDM writes all operations to the audit logs in both the internal database and also the flat files in the `openidm/audit` directory.
- The default Sample 1 version of the `conf/authentication.json` file includes several authentication modules: `STATIC_USER`, `MANAGED_USER`, `INTERNAL_USER`, and `CLIENT_CERT`. For more information, see Section 16.1.2.2, "Supported Authentication Modules" in the *Integrator's Guide*.

When you start OpenIDM with the `-p` project variable (**./startup.sh -p samples/sample1**), the `&{launcher.project.location}` is set to a value of `samples/sample1`. The configuration files use this location, as shown in the following sections.

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. It is a copy of the file of the same name found in the `samples/provisioners` directory.

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 `&{launcher.project.location}/data/xmlConnectorData.xml`.

The `&{launcher.project.location}`, in this case, is `sample/sample1`.

For details on the OpenICF connector configuration files, see Chapter 11, "*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`:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

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

For information about the schedule configuration, see Chapter 14, "*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&waitForCompletion=true"
```

The `waitForCompletion=true` parameter specifies that the operation should return only when it has completed.

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": "email",
                    "target": "mail"
                },
                {
                    "source": "firstname",
                    "target": "givenName"
                },
                {
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
                "source": "lastname",
                "target": "sn"
            },
            }
                "source": "description",
                "target": "description"
            },
            {
                "source": "_id",
                "target": "_id"
            },
            {
                "source": "name",
                "target": "userName"
            },
            {
                "source": "password",
                "target": "password"
            },
            {
                "source" : "mobileTelephoneNumber",
                "target" : "telephoneNumber"
            },
            {
                "source" : "roles",
                "transform" : {
                    "type" : "text/javascript",
                    "source" : "var _ = require('lib/lodash'); _.map(source.split(','),
                     function(role) { return {'_ref': 'repo/internal/role/' + role} });"
                },
                "target" : "authzRoles"
            }
        ],
        "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"
            },
            {
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
                    "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 external objects, in this case, objects in the XML file.

For more information about synchronization, reconciliation, and `sync.json`, see Chapter 12, "*Synchronizing Data Between Resources*" in the *Integrator's Guide*.

For additional examples related to scripting, see the Appendix F, "*Scripting Reference*" in the *Integrator's Guide*.

2.1.2. Install the Sample

Start OpenIDM with the configuration for Sample 1:

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

2.1.3. Review the Sample in the Administrative User Interface

OpenIDM includes a web-based Administrative User Interface, known as the Admin UI. For details, see Section 4.1, "Configuring OpenIDM from the Admin UI" in the *Integrator's Guide*.

After starting OpenIDM, you can access the Admin UI by navigating to `https://localhost:8443/admin`. The first time you log in, use the default administrative credentials, (Login: openidm-admin, Password: openidm-admin).

Warning

To protect your deployment in production, change the default administrative password. To do so, navigate to the Self-Service UI at `https://localhost:8443/` and click Change Password.

You should now see the Dashboard screen, with quick start cards for common administrative tasks. with the connectors and managed objects associated with that configuration.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

2.1.4. Running Reconciliation

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

```
$ 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&waitForCompletion=true"
```

Successful reconciliation returns a reconciliation run ID, and the status of the reconciliation operation, as follows:

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

Alternatively, you can run the same reconciliation in the Admin UI:

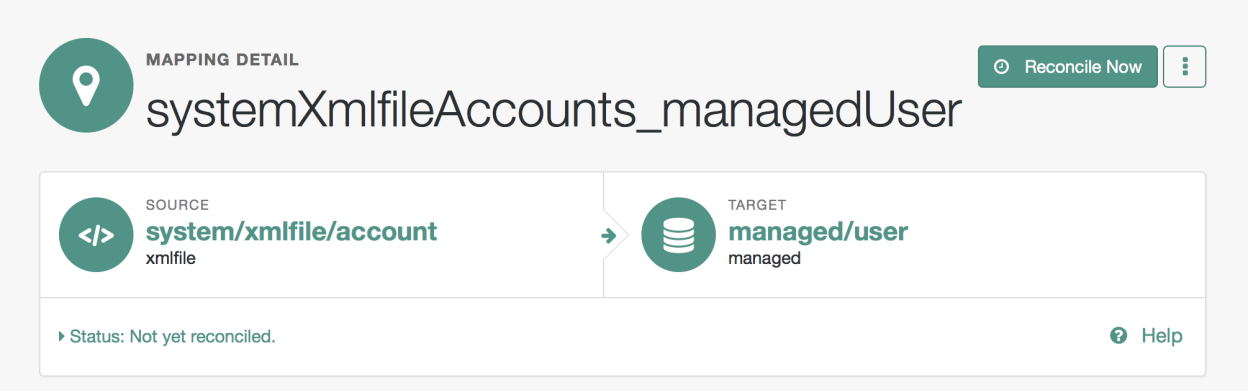
1. Click Configure > Mappings.

   For Sample 1, you should see one mapping, `systemXmlfileAccounts_managedUser`.

2. Select Edit to access the configuration options associated with reconciliation.

3. To run the reconciliation, click Reconcile Now.

Figure 2.1. OpenIDM Admin UI Mappings with Sample 1



Copyright © 2011-2016 ForgeRock AS. All rights reserved.

2.1.5. Viewing Users and Logs

After reconciliation, you can use the Admin UI to display user records in both the source and target resources:

1. Navigate to the URL where OpenIDM is installed.

   If it is local, navigate to `https://localhost:8443/admin`.

2. Click Configure > Mappings, then select the only available mapping (`systemXmlfileAccounts_managedUser`)

3. On the Association tab, you should see the result of the reconciliation, from source to target, at the bottom of the screen.

You can also 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 JSON data. Depending on the browser, you can use a REST client to display the JSON or download it as a file. Alternatively, you can use the following **curl** command to get the JSON response:

```
$ 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"

   {
  "result": [
    {
      "_id": "scarter",
      "_rev": "1"
    },
    {
      "_id": "bjensen",
      "_rev": "1"
    }
  ]
,
...
}
```

In addition to querying the users by their ID, you can set up arbitrary queries. For more information about using query expressions in a REST call, see Section 7.3, "Defining and Calling Queries" in the *Integrator's Guide*.

Now try a RESTful GET of user `bjensen` by appending the user ID to the managed user URL (`http://localhost:8080/openidm/managed/user/`):

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/managed/user/bjensen"
   {
  "_id": "bjensen",
  "_rev": "1",
  "mail": "bjensen@example.com",
  "givenName": "Barbara",
  "sn": "Jensen",
  "description": "Created By XML1",
  "userName": "bjensen@example.com",
  "telephoneNumber": "1234567",
  "accountStatus": "active",
  "effectiveRoles": [],
  "effectiveAssignments": []
}
```

The complete user record is returned. If you need this level of information for all users, substitute `query-all` for `query-all-ids`.

You can filter the output with the query expressions described in Section 7.3, "Defining and Calling Queries" in the *Integrator's Guide*.

As defined in the mapping file `conf/sync.json`, the `sn` and `mail` parameters correspond to surname (sn) and email address, respectively.

For example, the following RESTful GET filters output by surname (sn):

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/managed/user?_queryFilter=true&_fields=sn"

   {
  "result": [
    {
      "_id": "scarter",
      "_rev": "1",
      "sn": "Carter"
    },
    {
      "_id": "bjensen",
      "_rev": "1",
      "sn": "Jensen"
    }
  ]
,
...
}
```

Now that you have a list of users, you can add more fields to your query:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
   --header "X-OpenIDM-Username: openidm-admin" \
   --header "X-OpenIDM-Password: openidm-admin" \
   --request GET \
   "http://localhost:8080/openidm/managed/user?_queryFilter=true&_fields=sn,mail,description"

   {
  "result": [
    {
      "_id": "scarter",
      "_rev": "1",
      "sn": "Carter",
      "mail": "scarter@example.com",
      "description": "Created By XML1"
    },
    {
      "_id": "bjensen",
      "_rev": "1",
      "sn": "Jensen",
      "mail": "bjensen@example.com",
      "description": "Created By XML1"
    }
  ]
,
...
}
```

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
```

For more information about the contents of each of these files, see Section 19.3, "Audit Log Event Topics" in the *Integrator's Guide*.

You can get a similar level of information for each user. For example, after running reconciliation, follow the instructions in Section 2.1.5, "Viewing Users and Logs", and review information from the reconciled linked resource.

2.1.6. 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.

1. To add the user copy the following XML into `openidm/samples/sample1/data/xmlConnectorData.xml`:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
<ri:__ACCOUNT__>
    <icf:__UID__>tmorris</icf:__UID__>
    <icf:__NAME__>tmorris@example.com</icf:__NAME__>
    <ri:firstname>Toni</ri:firstname>
    <ri:lastname>Morris</ri:lastname>
    <ri:email>tmorris@example.com</ri:email>
    <ri:mobileTelephoneNumber>1234567</ri:mobileTelephoneNumber>
    <ri:roles>openidm-authorized</ri:roles>
    <icf:__DESCRIPTION__>Created By XML1</icf:__DESCRIPTION__>
 </ri:__ACCOUNT__>
```

2. Run reconciliation again, as described in Section 2.1.4, "Running Reconciliation".

3. After reconciliation has run, query the local repository to see the new user appear in the list of all managed users:

```
$ 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"
{
  "result": [
    {
      "_id": "scarter",
      "_rev": "2"
    },
    {
      "_id": "bjensen",
      "_rev": "2"
    },
    {
      "_id": "tmorris",
      "_rev": "1"
    }
  ]
,
...
}
```

To see what happened during the reconciliation operation, look at the reconciliation audit log, `openidm/audit/recon.csv`. 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/acc.../bjensen","managed/user/bjensen",...;
"1a...","CREATE",...,"486...", "ABSENT",   "system/xmlfile/acc.../scarter","managed/user/scarter",...;
"33...","UPDATE",...,"aa9...", "CONFIRMED","system/xmlfile/acc.../bjensen","managed/user/bjensen",...;
"1d...","UPDATE",...,"aa9...", "CONFIRMED","system/xmlfile/acc.../scarter","managed/user/scarter",...;
"0e...","CREATE",...,"aa9...", "ABSENT",   "system/xmlfile/acc.../tmorris","managed/user/tmorris",...;
```

The relevant audit log fields in this example are: action, situation, `sourceObjectId`, and `targetObjectId`. For each object in the source, reconciliation leads to an action on the target.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

In the first reconciliation run (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 (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. 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 Chapter 12, "*Synchronizing Data Between Resources*" in the *Integrator's Guide*.

For details about auditing, see Chapter 19, "*Using Audit Logs*" in the *Integrator's Guide*.

2.1.7. Adding Users Over REST

You can add users to the local repository over 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" \
 --header "Content-Type: application/json" \
 --request POST \
 --data '{
  "_id":"jberg",
  "userName":"jberg",
  "sn":"Berg",
  "givenName":"James",
  "mail":"jberg@example.com",
  "telephoneNumber":"5556787",
  "description":"Created by OpenIDM REST.",
  "password":"MyPassw0rd"
 }' \
 "http://localhost:8080/openidm/managed/user?_action=create"
   {
  "_id": "jberg",
  "_rev": "1",
  "userName": "jberg",
  "sn": "Berg",
  "givenName": "James",
  "mail": "jberg@example.com",
  "telephoneNumber": "5556787",
  "description": "Created by OpenIDM REST.",
  "accountStatus": "active",
  "effectiveRoles": [],
  "effectiveAssignments": []
}
```

Create `james` (Windows):

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
C:\> curl ^
 --header "X-OpenIDM-Username: openidm-admin" ^
 --header "X-OpenIDM-Password: openidm-admin" ^
 --header "Content-Type: application/json" ^
 --request POST ^
 --data "{\"_id\":\"jberg\",\"userName\":\"jberg\",\"sn\":\"Berg\",\"givenName\":\"James\",\"email\":
\"jberg@example.com\",\"telephoneNumber\":\"5556787\",\"description\":\"Created by OpenIDM REST.\",
\"password\":\"MyPassw0rd\"}" ^
 "http://localhost:8080/openidm/managed/user?_action=create"
```

The output is essentially the same as the UNIX command output.

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.

You can also add users through the UI, which uses the OpenIDM REST API. When you have logged into the UI as the OpenIDM administrator, click Manage > User > New User. The process is straightforward.

2.2. Logging Sample - Using Scripts to Generate Log Messages

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

2.2.1. Install the Sample

Prepare OpenIDM as described in Section 1.3, "Preparing OpenIDM", then start OpenIDM with the configuration for sample 8.

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

The `sync.json` file in the `sample8/conf` directory includes brief examples of log messages.

2.2.2. 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&waitForCompletion=true"
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

The reconciliation operation returns a reconciliation run ID, and the status of the operation.

Note the log messages displayed in the OSGi console. The following example omits timestamps and so forth to show only the message strings.

```
->
...Case no Source: the source object contains: = null [5235432-...
...Case emptySource: the source object contains: = {lastname=Carter, mobile...
...Case sourceDescription: the source object contains: = Created By XML1
...Case onCreate: the source object contains: = {lastname=Carter, mobile...
...Case result: the source object contains: = {SOURCE_IGNORED={count=0, ids=[]},...
```

2.3. Workflow Sample - Demonstrating Asynchronous Reconciling Using a Workflow

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" : "workflow/triggerWorkflowFromSync.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.

2.3.1. Install the Sample

Prepare OpenIDM as described in Section 1.3, "Preparing OpenIDM", then start OpenIDM with the configuration for sample 9.

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

2.3.2. Running the Sample

1. Run reconciliation over the REST interface.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ 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&waitForCompletion=true"
```

The reconciliation operation returns a reconciliation run ID, and the status of the operation.

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"
```

In this case, the request returns two workflow results, each with a process ID (`_id`) as well as a process definition ID. You will use the value of the `_id` shortly.

```
{
 "result" : [ {
   "tenantId" : "",
   "createTime" : "2014-05-01T13:48:42.980-08:00",
   "executionId" : "101",
   "delegationStateString" : null,
   "processVariables" : { },
   "_id" : "123",
   "processInstanceId" : "101",
   "description" : null,
   "priority" : 50,
   "name" : "Evaluate request",
   "dueDate" : null,
   "parentTaskId" : null,
   "processDefinitionId" : "managedUserApproval:1:3",
   "taskLocalVariables" : { },
   "suspensionState" : 1,
   "assignee" : "openidm-admin",
   "cachedElContext" : null,
   "queryVariables" : null,
   "activityInstanceVariables" : { },
   "deleted" : false,
   "suspended" : false,
   "_rev" : 1,
   "revisionNext" : 2,
   "category" : null,
   "taskDefinitionKey" : "evaluateRequest",
   "owner" : null,
   "eventName" : null,
   "delegationState" : null
 }, {
   "tenantId" : "",
   "createTime" : "2014-05-01T13:48:42.980-08:00",
   "executionId" : "102",
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
    "delegationStateString" : null,
    "processVariables" : { },
    "_id" : "124",
    "processInstanceId" : "102",
    "description" : null,
    "priority" : 50,
    "name" : "Evaluate request",
    "dueDate" : null,
    "parentTaskId" : null,
    "processDefinitionId" : "managedUserApproval:1:3",
    "taskLocalVariables" : { },
    "suspensionState" : 1,
    "assignee" : "openidm-admin",
    "cachedElContext" : null,
    "queryVariables" : null,
    "activityInstanceVariables" : { },
    "deleted" : false,
    "suspended" : false,
    "_rev" : 1,
    "revisionNext" : 2,
    "category" : null,
    "taskDefinitionKey" : "evaluateRequest",
    "owner" : null,
    "eventName" : null,
    "delegationState" : null
  } ],
  "resultCount" : 2,
  "pagedResultsCookie" : null,
  "remainingPagedResults" : -1
}
```

3. Approve the requests over REST, by setting the `"requestApproved"` parameter for the specified task instance to `"true"`. Note the use of one of the values of `_id` in the REST call, in this case, `124`.

   On UNIX:

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --header "Content-Type: application/json" \
 --request POST \
 --data '{"requestApproved": "true"}' \
 "http://localhost:8080/openidm/workflow/taskinstance/124?_action=complete"
```

   On Windows:

```
$ curl ^
 --header "X-OpenIDM-Username: openidm-admin" ^
 --header "X-OpenIDM-Password: openidm-admin" ^
 --header "Content-Type: application/json" ^
 --request POST ^
 --data "{\"requestApproved\": \"true\"}" ^
 "http://localhost:8080/openidm/workflow/taskinstance/124?_action=complete"
```

   A successful call returns the following:

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


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

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.

```
{
 "result": [ {
   "_rev": "0",
   "_id": "1"
 } ],
...
}
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

Chapter 3

LDAP Samples - Reconciling Data Between OpenIDM and One or More LDAP Directories

This chapter walks you through the LDAP samples (those samples labeled 2, 2b, 2c, 2d, 5, 5b and 6 in the `openidm/samples` directory). For a complete list of the samples provided with OpenIDM, and an overview of each sample, see Chapter 1, "*Overview of the OpenIDM Samples*".

3.1. Sample 2 - LDAP One Way

Sample 2 resembles Section 2.1, "First OpenIDM Sample - Reconciling an XML File Resource", but in sample 2 OpenIDM is connected to a local LDAP server. The sample has been tested with OpenDJ, but 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.

3.1.1. 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.
- Directory data for that server is stored under base DN `dc=example,dc=com`.
- User objects for that server are stored under base DN `ou=People,dc=example,dc=com`.
- User objects have the object class `inetOrgPerson`.
- User objects have the following attributes:
  - `cn`
  - `description`

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

- `givenName`
- `mail`
- `sn`
- `telephoneNumber`
- `uid`
- `userPassword`

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: 1-415-523-0772
sn: Doe
mail: jdoe@example.com
description: Created by OpenIDM
userPassword: password
```

The following steps provide setup instructions for an OpenDJ server. Adjust these instructions if you are using an alternative LDAP server.

1. Download OpenDJ from ForgeRock's download site and extract the zip archive.

   The LDIF data for this sample is provided in the file `openidm/samples/sample2/data/Example.ldif`. You will need to import this data during your OpenDJ setup.

2. Install OpenDJ using the command-line setup.

   Substitute the `--ldifFile` argument with the path to the `Example.ldif` file in your OpenIDM installation:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ cd /path/to/opendj
$ ./setup --cli
 \
--hostname localhost
 \
--ldapPort 1389
 \
--rootUserDN "cn=Directory Manager"
 \
--rootUserPassword password
 \
--adminConnectorPort 4444
 \
--baseDN dc=com
 \
--ldifFile /path/to/openidm/samples/sample2/data/Example.ldif
 \
--acceptLicense
 \
--no-prompt
...
Configuring Directory Server ..... Done.
Importing LDIF file /path/to/openidm/samples/sample2/data/Example.ldif ...... Done.
Starting Directory Server ...... Done.
.
...
```

3.1.2. Install the Sample

Prepare OpenIDM as described in Section 1.3, "Preparing OpenIDM", then start OpenIDM with the configuration for sample 2.

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

3.1.3. Reconcile the Repository

The mapping configuration file (`sync.json`) for this sample includes the mapping `systemLdapAccounts_managedUser`, which synchronize users from the source LDAP server with the target OpenIDM repository.

You can run this part of the sample by using the `curl` command-line utility, or by using the OpenIDM Administration UI. This section provides instructions for both methods.

Procedure 3.1. Run the Sample Using the Command Line

1. Reconcile the repository by running the following command:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ 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&waitForCompletion=true"
{
  "_id": "b1394d10-29b0-4ccf-81d8-c88948ea121c-4",
  "state": "SUCCESS"
}
```

The reconciliation operation creates the two users from the LDAP server in the OpenIDM repository, assigning the new objects random unique IDs.

2. To retrieve the users from the repository, query their IDs as follows:

```
$ 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"
{
  "result": [
    {
      "_id": "f52df646-7108-45e1-9342-1a17f257b497",
      "_rev": "1"
    },
    {
      "_id": "f7fccf54-e76a-404c-93f0-7486d30f1dc3",
      "_rev": "1"
    }
  ]
,
...
}
```

3. To retrieve individual user objects, include the ID in the URL, for example:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/managed/user/0a5546d6-149b-4f8b-b3be-4afa8a267d45"
{
  "_id": "f7fccf54-e76a-404c-93f0-7486d30f1dc3",
  "_rev": "1",
  "displayName": "Barbara Jensen",
  "description": "Created for OpenIDM",
  "givenName": "Barbara",
  "mail": "bjensen@example.com",
  "sn": "Jensen",
  "telephoneNumber": "1-360-229-7105",
  "userName": "bjensen",
  "accountStatus": "active",
  "effectiveRoles": [],
  "effectiveAssignments": []
}
```

Procedure 3.2. Run the Sample Using the Admin UI

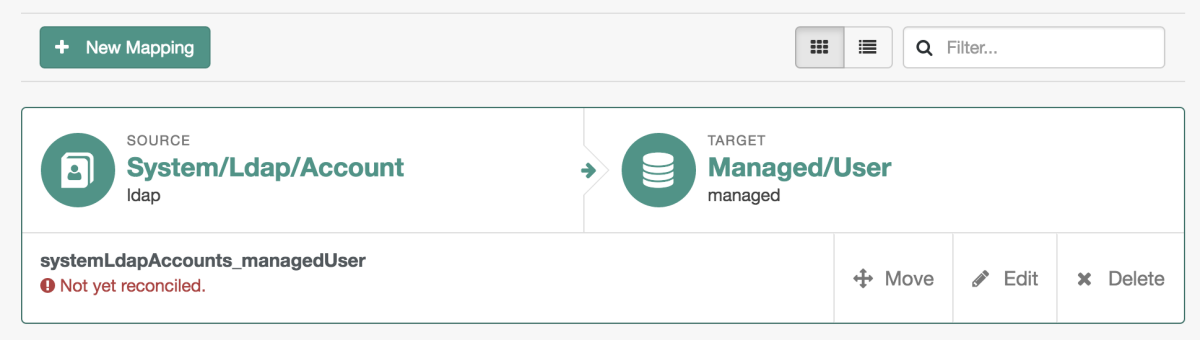
1. Log in to the Admin UI at the URL `https://localhost:8443/admin` as the default administrative user (`openidm-admin`) with password `openidm-admin`.

   Warning

   To protect your deployment in production, change the default administrative password. To do so, navigate to the Self-Service UI at `https://localhost:8443/` and click Change Password.

2. Click Configure > Mappings.

   This page shows one configured mapping, from the `ldap` server to the OpenIDM repository (`managed/user`).



3. Click anywhere on the mapping and click Reconcile Now.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

The reconciliation operation creates the two users from the LDAP server in the OpenIDM repository.

4. Retrieve the users in the repository. Click Manage > User.

   You should now see two users from the LDAP server, reconciled to the OpenIDM repository.

5. When you click a username, you can view the details of that user account.

3.2. Sample 2b - LDAP Two Way

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

3.2.1. External LDAP Configuration

As demonstrated for sample 2, you can use OpenDJ as an LDAP server. The LDIF data for this sample is provided in the file `openidm/samples/sample2b/data/Example.ldif`. You will need to import this data during your OpenDJ setup.

Configure the LDAP server as for sample 2, Section 3.1.1, "LDAP Server Configuration", but import the LDIF file that is specific to Sample 2b during the setup. The LDAP user must have write access to create users from OpenIDM on the LDAP server.

3.2.2. Install the Sample

Prepare OpenIDM as described in Section 1.3, "Preparing OpenIDM", then start OpenIDM with the configuration for sample 2b.

```
$ cd /path/to/openidm
$ ./startup.sh -p samples/sample2b
```

3.2.3. Run the Sample

The mapping configuration file (`sync.json`) for this sample includes two mappings, `systemLdapAccounts_managedUser`, which synchronizes users from the source LDAP server with the target OpenIDM repository, and `managedUser_systemLdapAccounts`, which synchronizes changes from the OpenIDM repository to the LDAP server.

You can run this part of the sample by using the `curl` command-line utility, or by using the OpenIDM Administration UI. This section provides instructions for both methods.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

Procedure 3.3. Run the Sample Using the Command Line

1. Reconcile the repository over the REST interface by running 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=systemLdapAccounts_managedUser&waitForCompletion=true"
{
  "state": "SUCCESS",
  "_id": "027e25e3-7a33-4858-9080-161c2b40a6bf-2"
}
```

   The reconciliation operation returns a reconciliation run ID and the status of the operation. Reconciliation creates user objects from LDAP in the OpenIDM repository, assigning the new objects random unique IDs.

2. To retrieve the users from the repository, query their IDs as follows:

```
$ 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"
{
  "result": [
    {
      "_id": "d460ed00-74f9-48fb-8cc1-7829be60ddac",
      "_rev": "1"
    },
    {
      "_id": "74fe2d25-4eb1-4148-a3ae-ff80f194b3a6",
      "_rev": "1"
    }
  ]
,
...
}
```

3. To retrieve individual user objects, include the ID in the URL, for example:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/managed/user/d460ed00-74f9-48fb-8cc1-7829be60ddac"
{
  "_id": "d460ed00-74f9-48fb-8cc1-7829be60ddac",
  "_rev": "1",
  "displayName": "Barbara Jensen",
  "description": "Created for OpenIDM",
  "givenName": "Barbara",
  "mail": "bjensen@example.com",
  "telephoneNumber": "1-360-229-7105",
  "sn": "Jensen",
  "userName": "bjensen",
  "accountStatus": "active",
  "effectiveRoles": [],
  "effectiveAssignments": []
}
```

4. 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" \
 --header "Content-Type: application/json" \
 --request POST \
 --data '{
    "mail":"fdoe@example.com",
    "sn":"Doe",
    "telephoneNumber":"555-1234",
    "userName":"fdoe",
    "givenName":"Felicitas",
    "description":"Felicitas Doe",
    "displayName":"fdoe"}' \
 "http://localhost:8080/openidm/managed/user?_action=create"
{
  "_id": "90d1f388-d8c3-4438-893c-be4e498e7a1c",
  "_rev": "1",
  "mail": "fdoe@example.com",
  "sn": "Doe",
  "telephoneNumber": "555-1234",
  "userName": "fdoe",
  "givenName": "Felicitas",
  "description": "Felicitas Doe",
  "displayName": "fdoe",
  "accountStatus": "active",
  "effectiveRoles": [],
  "effectiveAssignments": []
}
```

5. By default, *implicit synchronization* is enabled for mappings *from* the `managed/user` repository *to* any external resource. This means that when you update a managed object, any mappings defined in the `sync.json` file that have the managed object as the source are automatically executed to update the target system. For more information, see Section 12.3.2, "Mapping Source Objects to Target Objects" in the *Integrator's Guide*.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

Test that the implicit synchronization has been successful by querying the users in the LDAP directory over REST, as follows:

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/system/ldap/account?_queryId=query-all-ids"
{
  "result": [
    {
      "_id": "uid=jdoe,ou=People,dc=example,dc=com",
      "dn": "uid=jdoe,ou=People,dc=example,dc=com"
    },
    {
      "_id": "uid=bjensen,ou=People,dc=example,dc=com",
      "dn": "uid=bjensen,ou=People,dc=example,dc=com"
    },
    {
      "_id": "uid=fdoe,ou=People,dc=example,dc=com",
      "dn": "uid=fdoe,ou=People,dc=example,dc=com"
    }
  ]
,
...
}
```

Note the new entry for user `fdoe`.

6. Query the complete entry by including `fdoe`'s ID in the URL.

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/system/ldap/account/uid=fdoe,ou=People,dc=example,dc=com"
{
  "_id": "uid=fdoe,ou=People,dc=example,dc=com",
  "mail": "fdoe@example.com",
  "employeeType": null,
  "ldapGroups": [],
  "telephoneNumber": "555-1234",
  "givenName": "Felicitas",
  "cn": "fdoe",
  "dn": "uid=fdoe,ou=People,dc=example,dc=com",
  "uid": "fdoe",
  "sn": "Doe",
  "description": "Felicitas Doe"
}
```

Procedure 3.4. Run the Sample Using the Admin UI

1. Log in to the Admin UI at the URL `https://localhost:8443/admin` as the default administrative user (`openidm-admin`) with password `openidm-admin`.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

Warning

To protect your deployment in production, change the default administrative password. To do so, navigate to the Self-Service UI at `https://localhost:8443/` and click Change Password.

2. Click Configure > Mappings.

   This tab shows two configured mappings, one from the `ldap` server to the OpenIDM repository (`managed/user`) and one from the OpenIDM repository to the `ldap` server.

3. Click anywhere on the first mapping and click Reconcile Now.

   The reconciliation operation creates the two users from the LDAP server in the OpenIDM repository.

4. Retrieve the users in the repository. Click Manage > User.

5. You should see two users from the LDAP server, reconciled to the OpenIDM repository.

6. To retrieve the details of a specific user, click that username in the User List page.

7. Add a new user in the OpenIDM repository by clicking New User in the User List page.

   Complete the user details and click Save.

8. By default, *implicit synchronization* is enabled for mappings *from* the `managed/user` repository *to* any external resource. This means that when you update a managed object, any mappings defined in the `sync.json` file that have the managed object as the source are automatically executed to update the target system. For more information, see Section 12.3.2, "Mapping Source Objects to Target Objects" in the *Integrator's Guide*.

   To test that the implicit synchronization has been successful, look at `fdoe`'s record, and click the Linked Systems tab. The information under this tab includes the external resource to which this user entry is mapped.

3.3. Sample 2c - Synchronizing LDAP Group Membership

Like sample 2b, sample 2c connects to an external LDAP server and has mappings from the LDAP server to the OpenIDM repository, and from the OpenIDM repository to the LDAP server. However, in sample 2c, LDAP group memberships are synchronized, in addition to user entries.

As demonstrated for sample 2, you can use OpenDJ as an LDAP server. The LDIF data for this sample is provided in the file `openidm/samples/sample2c/data/Example.ldif`.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

3.3.1. External LDAP Configuration

Configure the LDAP server as for sample 2, Section 3.1.1, "LDAP Server Configuration". The LDAP user must have write access to create users from OpenIDM on the LDAP server. When you configure the LDAP server, import the LDIF file customized for this sample, `openidm/samples/sample2c/data/Example.ldif`. This file includes two LDAP groups:

```
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=bjensen,ou=People,dc=example,dc=com
cn: openidm2
objectClass: groupOfUniqueNames
objectClass: top
```

The users with DNs `uid=jdoe,ou=People,dc=example,dc=com` and `uid=bjensen,ou=People,dc=example,dc=com` are also imported with the `Example.ldif` file.

3.3.2. Install the Sample

Prepare OpenIDM as described in Section 1.3, "Preparing OpenIDM", then start OpenIDM with the configuration for sample 2c.

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

3.3.3. Run the Sample

The mapping configuration file (`sync.json`) for this sample includes two mappings, `systemLdapAccounts_managedUser`, which synchronizes users from the source LDAP server with the target OpenIDM repository, and `managedUser_systemLdapAccounts`, which synchronizes changes from the OpenIDM repository to the LDAP server.

You can run this part of the sample by using the `curl` command-line utility, or by using the OpenIDM Administration UI. This section provides instructions for both methods.

Procedure 3.5. Run the Sample Using the Command Line

1. Reconcile the repository over the REST interface by running the following command:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ 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&waitForCompletion=true"
{
  "_id": "6652c292-5309-40e5-b272-b74d67dd95c9-4",
  "state": "SUCCESS"
}
```

The reconciliation operation returns a reconciliation run ID and the status of the operation. Reconciliation creates user objects from LDAP in the OpenIDM repository, assigning the new objects random unique IDs.

2. To retrieve the users from the repository, query their IDs as follows:

```
$ 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"
{
  "result": [
    {
      "_id": "b63fb9a7-99bc-4eb4-8bfd-15f14a756e5b",
      "_rev": "1"
    },
    {
      "_id": "8462fe0c-2ab2-459a-a25e-474474889c9e",
      "_rev": "1"
    }
  ]
,
...
}
```

3. To retrieve individual user objects, include the ID in the URL. The following request retrieves the user object for John Doe:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/managed/user/8462fe0c-2ab2-459a-a25e-474474889c9e"
{
  "_id": "8462fe0c-2ab2-459a-a25e-474474889c9e",
  "_rev": "1",
  "displayName": "John Doe",
  "description": "Created for OpenIDM",
  "givenName": "John",
  "mail": "jdoe@example.com",
  "telephoneNumber": "1-415-599-1100",
  "sn": "Doe",
  "userName": "jdoe",
  "ldapGroups": [
    "cn=openidm,ou=Groups,dc=example,dc=com"
  ],
  "accountStatus": "active",
  "effectiveRoles": [],
  "effectiveAssignments": []
}
```

Note that John Doe's user object contains an `ldapGroups` property, the value of which indicates his groups on the LDAP server:

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

4. Update John Doe's `ldapGroups` property, to change his membership from the `openidm` group to the `openidm2` group.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --header "Content-Type: application/json" \
 --request POST \
 --data '[
    {
      "operation":"replace",
      "field":"/ldapGroups",
      "value": ["cn=openidm2,ou=Groups,dc=example,dc=com"]
    }
 ]' \
 "http://localhost:8080/openidm/managed/user?_action=patch&_queryId=for-userName&uid=jdoe"
{
  "displayName": "John Doe",
  "description": "Created for OpenIDM",
  "givenName": "John",
  "mail": "jdoe@example.com",
  "telephoneNumber": "1-415-599-1100",
  "sn": "Doe",
  "userName": "jdoe",
  "accountStatus": "active",
  "effectiveRoles": [],
  "effectiveAssignments": [],
  "_id": "8462fe0c-2ab2-459a-a25e-474474889c9e",
  "_rev": "2",
  "ldapGroups": [
    "cn=openidm2,ou=Groups,dc=example,dc=com"
  ]
}
```

This command changes John Doe's `ldapGroups` property in the OpenIDM repository, from `"cn=openidm,ou=Groups,dc=example,dc=com"` to `"cn=openidm2,ou=Groups,dc=example,dc=com"`. As a result of implicit synchronization, the change is propagated to the LDAP server. John Doe is removed from the first LDAP group and added to the second LDAP group in OpenDJ.

5. You can verify this change by querying John Doe's record on the LDAP server, as follows:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/system/ldap/account/uid=jdoe,ou=People,dc=example,dc=com"
{
  "_id": "uid=jdoe,ou=People,dc=example,dc=com",
  "telephoneNumber": "1-415-599-1100",
  "description": "Created for OpenIDM",
  "sn": "Doe",
  "dn": "uid=jdoe,ou=People,dc=example,dc=com",
  "ldapGroups": [
    "cn=openidm2,ou=Groups,dc=example,dc=com"
  ],
  "uid": "jdoe",
  "cn": "John Doe",
  "givenName": "John",
  "mail": "jdoe@example.com"
}
```

Procedure 3.6. Run the Sample Using the Admin UI

1. Log in to the Admin UI at the URL `https://localhost:8443/admin` as the default administrative user (`openidm-admin`) with password `openidm-admin`.

   Warning

   To protect your deployment in production, change the default administrative password. To do so, navigate to the Self-Service UI at `https://localhost:8443/` and click Change Password.

2. Click Configure > Mappings.

   This window shows two configured mappings, one from the `ldap` server to the OpenIDM repository (`managed/user`) and one from the OpenIDM repository to the `ldap` server.

3. Click anywhere on the first mapping and click Reconcile Now.

   The reconciliation operation creates the two users from the LDAP server in the OpenIDM repository.

4. Click Manage > User. Examine the users reconciled from the LDAP server to the internal repository.

5. Examine the two users from the LDAP server that have been reconciled to the OpenIDM repository.

6. To retrieve the details of a specific user, click that username. In this case, click on user `jdoe`.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

Examine the information stored for user `jdoe`. Click the Linked Systems tab. The Linked Resource item indicates the external resource on which John Doe's managed object is mapped, in this case, `ldap account`.

In this linked resource, John Doe's `ldapGroups` are displayed. Currently, John Doe is a member of `cn=openidm,ou=Groups,dc=example,dc=com`.

7. Update John Doe's `ldapGroups` property to change his membership from the `openidm` group to the `openidm2` group. Currently, you can only do this over the REST interface, as follows:

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --header "Content-Type: application/json" \
 --request POST \
 --data '[
    {
      "operation":"replace",
      "field":"/ldapGroups",
      "value": ["cn=openidm2,ou=Groups,dc=example,dc=com"]
    }
 ]' \
 "http://localhost:8080/openidm/managed/user?_action=patch&_queryId=for-userName&uid=jdoe"
{
  "displayName": "John Doe",
  "description": "Created for OpenIDM",
  "givenName": "John",
  "mail": "jdoe@example.com",
  "telephoneNumber": "1-415-599-1100",
  "sn": "Doe",
  "userName": "jdoe",
  "accountStatus": "active",
  "effectiveRoles": [],
  "effectiveAssignments": [],
  "_id": "8462fe0c-2ab2-459a-a25e-474474889c9e",
  "_rev": "2",
  "ldapGroups": [
    "cn=openidm2,ou=Groups,dc=example,dc=com"
  ]
}
```

This command changes John Doe's `ldapGroups` property in the OpenIDM repository, from `"cn=openidm,ou=Groups,dc=example,dc=com"` to `"cn=openidm2,ou=Groups,dc=example,dc=com"`. As a result of implicit synchronization, the change is propagated to the LDAP server. John Doe is removed from the first LDAP group and added to the second LDAP group in OpenDJ.

8. You can verify this change by reloading John Doe's user information, clicking Linked Systems, and examining the value of his `ldapGroups` property.

3.4. Sample 2d - Synchronizing LDAP Groups

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

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

As demonstrated for sample 2, you can use OpenDJ as an LDAP server. Before installing OpenDJ, you may need an LDIF file. The OpenIDM installation includes the following LDIF file, customized for this sample: `openidm/samples/sample2d/data/Example.ldif`. If you need a copy of this file, download and install OpenIDM as described in Procedure 1.1, "To Install OpenIDM Services" in the *Installation Guide*.

3.4.1. External LDAP Configuration

Configure the LDAP server as for sample 2, Section 3.1.1, "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, as found in the following 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=bjensen,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.

There is an additional user, `bjensen` in the sample LDIF file. This user is essentially a "dummy" user, provided for compliance with RFC 4519, which stipulates that every `groupOfUniqueNames` object must contain at least one `uniqueMember`. `bjensen` is not actually used in this sample.

3.4.2. Install the Sample

Prepare OpenIDM as described in Section 1.3, "Preparing OpenIDM", then start OpenIDM with the configuration for sample 2d.

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

3.4.3. Running the Sample

The mapping configuration file (`sync.json`) for this sample includes three mappings:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

- `systemLdapAccounts_managedUser`

  Synchronizes users from the source LDAP server with the target OpenIDM repository,

- `managedUser_systemLdapAccounts`

  Synchronizes changes from the OpenIDM repository to the LDAP server.

- `systemLdapGroups_managedGroup`

  Synchronizes groups from the source LDAP server with the target OpenIDM repository.

Due to the similarity with other OpenIDM samples, especially samples 2b and 2c, the focus of this sample is on the mapping unique to this sample, `systemLdapGroups_managedGroup`.

You can run this part of the sample by using the `curl` command-line utility, or by using the OpenIDM Administration UI. This section provides instructions for both methods.

Procedure 3.7. Run the Sample Using the Command Line

1. Reconcile the repository over the REST interface for the group mapping, `systemLdapGroups_managedGroup` by running 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=systemLdapGroups_managedGroup&waitForCompletion=true"
   ```

   The reconciliation operation returns a reconciliation run ID, and the status of the operation.

2. 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?_queryFilter=true"
   ```

   The resulting JSON object should include content similar to the following.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
{
  "result" : [ {
     "dn" : "cn=openidm,ou=Groups,dc=example,dc=com",
     "_id" : "837df489-35d6-48d1-81a5-23792b49838a",
     "_rev" : "1",
     "description" : [ ],
     "uniqueMember" : [ "uid=jdoe,ou=People,dc=example,dc=com" ],
     "name" : [ "openidm" ]
  }, {
     "dn" : "cn=openidm2,ou=Groups,dc=example,dc=com",
     "_id" : "7575c1c7-86cf-43bc-bf1d-5c9cfc539124",
     "_rev" : "1",
     "description" : [ ],
     "uniqueMember" : [ "uid=bjensen,ou=People,dc=example,dc=com" ],
     "name" : [ "openidm2" ]
  } ],
...
 }
```

Procedure 3.8. Run the Sample Using the Admin UI

1. Log in to the Admin UI at the URL `https://localhost:8443/admin` as the default administrative user (`openidm-admin`) with password `openidm-admin`.

   Warning

   To protect your deployment in production, change the default administrative password. To do so, navigate to the Self-Service UI at `https://localhost:8443/` and click Change Password.

2. Click Configure > Mappings.

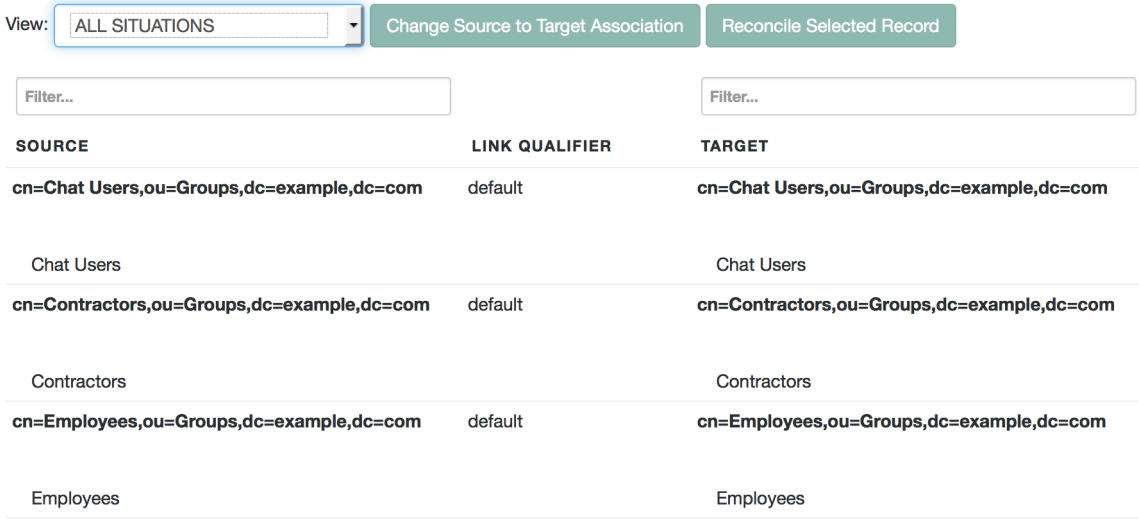
   This page shows three configured mappings, from the `ldap` server accounts repository to the OpenIDM repository (`managed/user`), from the OpenIDM repository back to the `ldap` server, and from the `ldap` server group accounts repository to the OpenIDM `managed/group` repository.

3. Click anywhere on the third mapping and click Reconcile Now.

   The reconciliation operation creates the two groups from the LDAP server in the OpenIDM repository.

4. Retrieve the groups in the repository by clicking the Association tab below the mapping. Scroll down to Data Association Management.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.



The three groups from the LDAP server (source) have been reconciled to the OpenIDM repository (target).

3.5. Sample 5 - Synchronization of Two LDAP Resources

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

You can optionally configure an outbound email service, if you want to receive emailed reconciliation summaries, as described in the following section.

3.5.1. Configure Email for the Sample

If you do not configure the email service, the functionality of the sample does not change. However, you might see the following message in the OSGi console when you run a reconciliation operation:

```
Email service not configured; report not generated.
```

To configure OpenIDM to send a reconciliation summary by email, follow these steps:

1. Copy the template `external.email.json` file from the `samples/misc` directory to the `conf` directory of Sample 5:

   ```
   $ cd /path/to/openidm
   $ cp samples/misc/external.email.json samples/sample5/conf
   ```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

2. Edit the `external.email.json` file for outbound email, as described in Chapter 21, "*Sending Email*" in the *Integrator's Guide*.

3. Edit the `reconStats.js` file from the `sample5/script` directory. Near the start of the file, configure the OpenIDM email service to send statistics to the email addresses of your choice:

```
var email = {
      //UPDATE THESE VALUES
      from : "openidm@example.com",
      to : "youremail@example.com",
      cc : "idmadmin2@example.com,idmadmin3@example.com",
      subject : "Recon stats for " + global.mappingName,
      type : "text/html"
},
template,
...
```

3.5.2. Install the Sample

No external configuration is required for this sample. Prepare OpenIDM as described in Section 1.3, "Preparing OpenIDM", then start OpenIDM with the configuration of sample 5.

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

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

3.5.3. Run the Sample

Run a reconciliation operation, to synchronize the contents of the simulated LDAP resource to 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=systemLdapAccounts_managedUser&waitForCompletion=true"
```

This command creates a user in the repository. It is not necessary to run a second reconciliation operation to synchronize the AD resource. Automatic synchronization propagates any change made to managed users in the OpenIDM repository to the simulated AD resource.

Review the contents of `xml_AD_Data.xml`. It should now contain information for the same user that was present in the startup version of the `xml_LDAP_Data.xml` file.

Alternatively, you can list users in the AD resource with the following command:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

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

   {
  "result" : [ {
    "name" : "DDOE1",
    "__UID__" : "8dad9df3-820d-41ea-a3ab-a80c241bbc98",
    "_id" : "8dad9df3-820d-41ea-a3ab-a80c241bbc98"
  } ]
,
...
}
```

You can use the `_id` of the user to read the user information from the AD resource, for example:

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/system/ad/account/8dad9df3-820d-41ea-a3ab-a80c241bbc98"
{
  "email" : [ "mail1@example.com" ],
  "name" : "DDOE1",
  "__UID__" : "8dad9df3-820d-41ea-a3ab-a80c241bbc98",
  "firstname" : "Darth",
  "lastname" : "Doe",
  "_id" : "8dad9df3-820d-41ea-a3ab-a80c241bbc98"
}[
```

To verify that the sample is working, repeat the process. Set up a second user in the `xml_LDAP_Data.xml` file. An example of how that file might appear with a second user (`GDOE1`) is shown here:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
<?xml version="1.0" encoding="UTF-8"?>
<icf:OpenICFContainer
    xmlns:icf="http://openidm.forgerock.com/xml/ns/public/resource/openicf/resource-schema-1.xsd"
    xmlns:ri="http://openidm.forgerock.com/xml/ns/public/resource/instances/resource-schema-extension"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://openidm.forgerock.com/xml/ns/public/resource/instances/resource-schema-
extension
    /path/to/openidm/samples/sample5/data/resource-schema-extension.xsd
    http://openidm.forgerock.com/xml/ns/public/resource/openicf/resource-schema-1.xsd
    /path/to/openidm/samples/sample5/data/resource-schema-1.xsd">
    <ri:__ACCOUNT__>
       <icf:__UID__>1</icf:__UID__>
       <icf:__PASSWORD__>TestPassw0rd2</icf:__PASSWORD__>
       <ri:firstname>Darth</ri:firstname>
       <icf:__DESCRIPTION__>Created By XML1</icf:__DESCRIPTION__>
       <icf:__NAME__>DDOE1</icf:__NAME__>
       <ri:email>mail1@example.com</ri:email>
       <ri:lastname>Doe</ri:lastname>
    </ri:__ACCOUNT__>
    <ri:__ACCOUNT__>
       <icf:__UID__>2</icf:__UID__>
       <icf:__PASSWORD__>TestPassw0rd2</icf:__PASSWORD__>
       <ri:firstname>Garth</ri:firstname>
       <icf:__DESCRIPTION__>Created By XML1</icf:__DESCRIPTION__>
       <icf:__NAME__>GDOE1</icf:__NAME__>
       <ri:email>mail2@example.com</ri:email>
       <ri:lastname>Doe</ri:lastname>
    </ri:__ACCOUNT__>
</icf:OpenICFContainer>
```

Rerun the reconciliation and query REST commands shown previously. The reconciliation operation creates the new user from the simulated LDAP resource in the OpenIDM repository. An implicit synchronization operation then creates that user in the AD resource.

3.6. Sample 5b - Failure Compensation With Multiple Resources

The compensated synchronization mechanism depicted in this sample can help manage the risks associated with synchronizing data across multiple resources.

Typically, when a managed/user object is changed, implicit synchronization replays that change to all configured external resources. If synchronization fails for one target resource (for example, due to a policy validation failure on the target, or the target being unavailable), the synchronization operation stops at that point. The effect is that a record might be changed in the repository, and in the targets on which synchronization was successful, but not on the failed target, or any targets that would have been synchronized after the failure. This situation can result in disparate data sets across resources. While a reconciliation operation would eventually bring all targets back in sync, reconciliation can be an expensive operation with large data sets.

The compensated synchronization mechanism ensures that either all resources are synchronized successfully, or that the original change is rolled back. This mechanism uses an `onSync` script hook

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

configured with a `compensate.js` script that can be used to "revert" the partial change to managed/user and to the corresponding external resources.

Sample 5b is similar to sample 5 in that it simulates two external resources with XML files (located in the `sample5b/data` directory). The `xml_LDAP_Data.xml` file simulates an LDAP data source. OpenIDM creates the `xml_AD_Data.xml` file when you start OpenIDM with the sample. Sample 5b adds the `onSync` script hook to the process, configured in the `sample5b/conf/managed.json` file.

The following excerpt of the `managed.json` file shows the `onSync` hook, which calls the `compensate.js` script, provided in the `/path/to/openidm/bin/defaults/script` directory.

```
...
},
"onSync" : {
   "type" : "text/javascript",
   "file" : "compensate.js"
},
```

You can use the `onSync` script hook to ensure that changes made in the repository are synchronized to all external resources, or that no changes are made. For more information about how implicit synchronization uses the `onSync` script hook, see Section 12.12, "Configuring Synchronization Failure Compensation" in the *Integrator's Guide*.

You can optionally configure an outbound email service for this sample, if you want to receive emailed reconciliation summaries. The email service configuration is identical to that of sample 5 (Section 3.5.1, "Configure Email for the Sample").

3.6.1. Install the Sample

No external configuration is required for this sample. Prepare OpenIDM as described in Section 1.3, "Preparing OpenIDM", then start OpenIDM with the configuration of sample 5b.

```
$ cd /path/to/openidm
$ ./startup.sh -p samples/sample5b
```

The XML files that simulate an external LDAP and AD resource are now located in the `openidm/samples/sample5b/data/` directory. The simulated AD data store file, `xml_AD_Data.xml`, does not contain users until you run reconciliation.

Run the sample in exactly the same way that you did for Sample 5, following the steps in Section 3.5.3, "Run the Sample". Those steps will reconcile a user to your internal managed user repository.

Unless you run the steps in Section 3.5.3, "Run the Sample", you will not be able to run the steps in the next section.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

3.6.2. Demonstrate onSync

To demonstrate integration of the samples with the OpenIDM UI, this sample uses the UI to view and make changes to user objects in the repository. However, you can also use the REST interface to make these changes, as shown in the previous section.

Log into the OpenIDM UI as the administrative user. On a local system, navigate to `https://localhost:8443/admin`. The default administrative account and password are both `openidm-admin`.

Select Manage > User. Make a change to the data of an existing user (`DDOE1`). As a result of the implicit synchronization from the managed object repository, that change is reflected almost immediately on the external resources. For sample 5b, you should see the changes in both XML files in the `sample5b/data` directory. Alternatively, you can query the external resources over REST, as described previously.

The synchronization is successful, across all configured external resources, so the updated user record can be seen in both the `xml_LDAP_Data.xml` and `xml_AD_Data.xml` files.

The next step is to simulate a problem connecting to the LDAP resource. One way to do so on the local system is to rename the LDAP data file so that it is unreadable. On a Linux system, the following command, as an administrative user, would serve that purpose:

```
$ cd /path/to/openidm/samples/sample5b/data
$ sudo mv xml_LDAP_Data.xml xml_LDAP_Data.xml.bak
```

In the UI, now try another update to user `DDOE1`. With the modified filename of the simulated LDAP resource, implicit synchronization cannot write to this resource. An error similar to the following is displayed in the log file, `openidm0.log.0`:

```
Data file does not exist:
/path/to/openidm/samples/sample5b/data/xml_LDAP_Data.xml
```

Although the AD resource is available, implicit synchronization will not reach this resource, because the mapping is specified *after* the managed/user to LDAP mapping in the `sync.json` file.

When the implicit synchronization operation fails for the LDAP resource, the `onSync` hook invokes the `compensate.js` script. This script attempts to revert the original change by performing another update to DDOE1 in the repository (managed/user). This change, in turn, triggers another automatic synchronization to the AD and LDAP resources.

Because the LDAP resource is still unreadable, the synchronization to LDAP fails again, which triggers the `compensate.js` script again. This time, however, the script recognizes that the change was originally called as a result of a compensation and aborts.

The original synchronization error from the first update is thrown from the script and the UI should display that error. If you refresh the UI, and view that user entry again, you will notice that the change to the entry has been reverted.

Note that if you change the name of the AD resource file (to make it unavailable), a change to a managed/user entry will be synchronized successfully with the LDAP resource (because that mapping

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

appears first in `sync.json`). The synchronization will fail for the AD resource. In this case, the change will be reverted on both the managed/user entry, and the LDAP resource.

3.7. Sample 6 - LiveSync With an AD Server

Sample 6 resembles sample 5, but demonstrates LiveSync from an external resource. Sample 6 includes configuration files for two scenarios, depending on whether you have a live Active Directory (AD) service, or whether you need to simulate an AD service with an OpenDJ server. Each scenario is associated with a file in the `sample6/alternatives` directory. Depending on your scenario, copy the corresponding file to the `sample6/conf` directory.

Active AD Deployment

If you have an actual AD deployment available, copy the `provisioner.openicf-realad.json` file to the `conf/` subdirectory. You can then configure synchronization between an OpenDJ Directory Server and an active AD deployment.

As this sample demonstrates synchronization *from* the AD server *to* OpenDJ, data on the AD server is not changed.

Simulated AD Deployment

If you need to simulate an AD deployment, copy the `provisioner.openicf-fakead.json` file to the `conf/` subdirectory. You can then configure synchronization between an OpenDJ Directory server and a simulated AD server.

This sample simulates an AD server on the same instance of OpenDJ, using a different base DN.

The options shown in the associated configuration files can be easily modified to work with any standard LDAP server.

3.7.1. Sample 6 with an Active AD Deployment

If you have an existing, active instance of AD, set up OpenDJ, as described in the *OpenDJ Installation Guide*.

During installation, populate OpenDJ with the data in the `Example.ldif` file, available in the `sample6/data` directory.

The actions run in this sample should not change any data on the AD server.

3.7.2. Sample 6 with a Simulated AD Deployment

In this sample, an AD deployment is simulated with a different baseDN (`dc=fakead,dc=com`) on the same OpenDJ server instance. You can also simulate the AD server with a separate OpenDJ instance,

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

running on the same host, as long as the two instances communicate on different ports. The data for the simulated AD instance is contained in the file `AD.ldif`. The data for the OpenDJ instance is contained in the file `Example.ldif`.

3.7.3. External Configuration

3.7.3.1. Prepare OpenDJ For LiveSync

This sample assumes a replicated OpenDJ server on the localhost system. When configured, OpenDJ replication includes an External Change Log (ECL), required to support LiveSync. LiveSync detects changes in OpenDJ by reading the ECL.

Follow these steps to install and configure an OpenDJ instance.

1. Download and extract the OpenDJ zip archive from https://forgerock.org/downloads/.
2. Install OpenDJ using the command-line setup, and import the data file for this sample, as follows:

```
$ cd /path/to/opendj
$ ./setup --cli
 \
--hostname localhost
 \
--ldapPort 1389
 \
--rootUserDN "cn=Directory Manager"
 \
--rootUserPassword password
 \
--adminConnectorPort 4444
 \
--baseDN dc=com
 \
--ldifFile /path/to/openidm/samples/sample6/data/Example.ldif
 \
--acceptLicense
 \
--no-prompt
...
Configuring Directory Server ..... Done.
Creating Base Entry dc=com ..... Done.
Starting Directory Server ....... Done
.
...
```

The sample assumes the following configuration:

- The OpenDJ server is installed on the localhost.
- The server listens for LDAP connections on port 1389.
- The administration connector port is 4444.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

- The root user DN is `cn=Directory Manager`.
- The root user password is `password`.

3. Configure the OpenDJ server for replication.

   To enable liveSync, this server must be configured for replication, even if it does not actually participate in a replication topology. The following commands configure the server for replication.

```
$ ./dsconfig create-replication-server \
 --hostname localhost \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --provider-name "Multimaster Synchronization" \
 --set replication-port:8989 \
 --set replication-server-id:2 \
 --type generic \
 --trustAll \
 --no-prompt

 $ ./dsconfig create-replication-domain \
 --hostname localhost \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --provider-name "Multimaster Synchronization" \
 --domain-name fakead_com \
 --set base-dn:dc=fakead,dc=com \
 --set replication-server:localhost:8989 \
 --set server-id:3 \
 --type generic \
 --trustAll \
 --no-prompt
```

Once OpenDJ is configured, you can proceed with either an active or simulated AD deployment.

3.7.3.2. External Configuration for an Active AD Deployment

To configure an active AD deployment for sample 6, open the `provisioner.openicf-realad.json` file in a text editor. Update it as needed. At minimum, you should check and if needed update the following parameters in that file, as shown in the following table:

Table 3.1. Key Parameters for an Active AD Configuration

Option	Description
host	The hostname/IP address of the AD server
port	The LDAP port; the default is 389.
ssl	By default, SSL is not used.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

Option	Description
principal	The full DN of the account to bind with, such as "CN=Administrator,CN=Users,DC=example,DC=com"
credentials	If a password is used, replace null with that password. When OpenIDM starts, it encrypts that password in the `provisioner.openicf-realad.conf` file.
baseContexts	The DNs for account containers, such as ["CN=Users,DC=Example,DC=com"]
baseContextsToSynchronize	Set to the same value as `baseContexts`
accountSearchFilter	Default searches for active user (not computer) accounts
accountSynchronizationFilter	Default synchronizes with active user (not computer) accounts

If you do not want to filter out computer and disabled user accounts, set the `accountSearchFilter` and `accountSynchronizationFilter` to `null`.

3.7.3.3. External Configuration for a Simulated AD Deployment

Not everyone has a testable instance of AD readily available. For such administrators, you can use the `AD.ldif` file from the `data/` subdirectory to simulate an AD deployment.

If you have not already done so, copy the `provisioner.openicf-fakead.json` file to the `conf` subdirectory.

As previously mentioned, you can use a separate OpenDJ instance to simulate the AD server. However, the following instructions assume that the simulated AD server runs on the same OpenDJ instance.

Open the `provisioner.openicf-fakead.json` file and note the following:

- OpenDJ directory server uses port 1389 by default for users who cannot use privileged ports, so this is the port that is specified in the provisioner file. Adjust the port if your OpenDJ server is listening on a different port.

- The simulated AD server uses the base DN `dc=fakead,dc=com`.

To load the data for the simulated AD instance, launch the OpenDJ control panel, add the simulated AD baseDN (`dc=fakead,dc=com`), and then import the `sample6/data/AD.ldif` file. When you import the `AD.ldif` file, select "Append to Existing Data", not "Overwrite Existing Data". Otherwise, the data in the `dc=example,dc=com` baseDN will be overwritten.

Alternatively, you could run the following command:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$  cd /path/to/opendj/bin
$ ./ldapmodify
 \
--defaultAdd
 \
--bindDN "cn=Directory Manager"
 \
--bindPassword password
 \
--hostname localhost
 \
--port 1389
 \
--filename /path/to/openidm/samples/sample6/data/AD.ldif
```

3.7.4. Start OpenIDM with Sample 6

Now that OpenDJ and a real or simulated AD database is configured, prepare OpenIDM as described in Section 1.3, "Preparing OpenIDM". You can then start OpenIDM with the configuration for sample 6.

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

3.7.5. Run the Sample

The following sections show how to run the sample with command-based reconciliation with a REST call, and to configure scheduled reconciliation with LiveSync.

3.7.5.1. Using Reconciliation

Now that OpenIDM is in operation, review the entries in the OpenDJ data store. When you run reconciliation, any entries that share the same `uid` with the AD data store will be updated with the contents from AD.

If you have set up the simulated AD data store as described in Section 3.7.3.3, "External Configuration for a Simulated AD Deployment", compare the entries for `uid=jdoe` as shown in the `AD.ldif` and `Example.ldif` files. Note the different values of `givenName` for `uid=jdoe`.

Run reconciliation over the REST interface. If you have followed the instructions for the simulated AD data store, the following command takes the information for user `jdoe` imported from the `AD.ldif` file, with a `givenName` of Johnny, and synchronizes that information to the LDAP database, overwriting the `givenName` of John for that same user `jdoe`.

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


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

The reconciliation operation returns a reconciliation run ID, and the status of the operation.

```
{
  "state": "SUCCESS",
  "_id": "985ee939-fbe1-4607-a757-00b404b4ef77"
}
```

The reconciliation operation synchronizes the data in the AD server with the OpenIDM repository (managed/user). That information is then automatically synchronized to the OpenDJ server, as described in Section 12.13, "Synchronization Situations and Actions" in the *Integrator's Guide*.

After reconciliation, list all users in the OpenDJ server data store.

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

The result should resemble the following JSON object.

```
{
  "result": [ {
      "dn" : "uid=jdoe,ou=People,dc=example,dc=com",
      "_id" : "uid=jdoe,ou=People,dc=example,dc=com"
      }, {
      "dn" : "uid=bjensen,ou=People,dc=example,dc=com",
      "_id" : "uid=bjensen,ou=People,dc=example,dc=com"
    } ],
...
}
```

You see only two entries, as the `uid=jdoe` entry from `dc=fakead,dc=com` overwrites the original LDAP entry for `uid=jdoe` in the reconciled LDAP data store.

To read the user object in the OpenDJ server, run the **ldapsearch** command. The following example returns the entry for user `uid=jdoe`:

```
$ ./ldapsearch \
 --port 1389 \
 --baseDN dc=example,dc=com \
 "(uid=jdoe)"
```

3.7.5.2. Using LiveSync

You can trigger a reconciliation operation by configuring a schedule, or by launching the operation directly over the REST interface. You can also launch a LiveSync operation over REST, but LiveSync requires a configured schedule to poll for changes. When this sample's default LiveSync schedule (`schedule-activeSynchroniser_systemAdAccount.json`) is enabled, a LiveSync operation is launched every 15 seconds.

LiveSync pushes changes made in the AD data store to the OpenIDM repository, automatically.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

The LiveSync schedule is disabled by default. To activate LiveSync, change the value of the `"enabled"` property from `false` to `true`.

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

Procedure 3.9. Testing LiveSync

Now you can test LiveSync. This procedure assumes that you have configured OpenDJ using the parameters and commands described in this section.

1. Create an LDIF file with a new user entry (`uid=bsmith`) that will be added to the simulated AD data store.

2. The following is the contents of a sample `bsmith.ldif` file for demonstration purposes:

```
dn: uid=bsmith,ou=People,dc=fakead,dc=com
objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: top
givenName: Barry
description: Created to see LiveSync work
uid: bsmith
cn: Barry
sn: Smith
mail: bsmith@example.com
telephoneNumber: 1-415-523-0772
userPassword: passw0rd
```

3. Navigate to the `/path/to/opendj/bin` directory.

4. Use the **ldapmodify** command to add the `bsmith.ldif` file to the directory.

```
$ ./ldapmodify \
 --port 1389 \
 --defaultAdd \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --filename /path/to/bsmith.ldif
```

5. Now you can test synchronization by viewing the new user in the OpenIDM repository. The easiest way to do this, is through OpenIDM UI. You should be able to log into the UI with any of the accounts in the AD data store. For this example, log into the UI as user `bsmith`, with password `passw0rd`. The fact that you can log into the UI as this new user indicates that LiveSync has synchronized the user from the AD data store to the managed/user repository.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

6. Implicit synchronization pushes this change out to the OpenDJ data store. To test this synchronization operation, search the OpenDJ baseDN for the new user entry.

```
$ ./ldapsearch \
 --port 1389 \
 --baseDN ou=people,dc=example,dc=com \
 "(uid=bsmith)"
```

3.8. Linking Historical Accounts

This sample demonstrates the retention of inactive (historical) LDAP accounts that have been linked to a corresponding managed user account. The sample builds on sample 2b and uses the LDAP connector to connect to an OpenDJ instance. You can use any LDAP-v3 compliant directory server.

In this sample, OpenIDM is the source resource. Managed users in the OpenIDM repository maintain a list of the accounts that they have been linked to on the local LDAP server. This list is stored in the `historicalAccounts` field of the managed user entry. The list contains a reference to all past and current LDAP accounts. Each LDAP account in the list is represented as a *relationship* and includes information about the date the accounts were linked or unlinked, and whether the account is currently active. For more information about relationship objects, see Section 8.5, "Managing Relationships Between Objects" in the *Integrator's Guide*.

This sample includes the following custom scripts, in its `script` directory:

- `onLink-managedUser_systemLdapAccounts.js`

  When a managed user object is linked to a target LDAP object, this script creates the relationship entry in the managed user's `historicalAccounts` property. The script adds two relationship properties:

  - `linkDate` — specifies the date that the link was created.
  - `active` — boolean true/false. When set to true, this property indicates that the target object is *currently* linked to the managed user account.

- `onUnlink-managedUser_systemLdapAccounts.js`

  When a managed user object is unlinked from a target LDAP object, this script updates that relationship entry's properties with an `unlinkDate` that specifies when the target was unlinked, and sets the `active` property to false, indicating that the target object is no longer linked.

- `check_account_state_change.js`

  During LiveSync or reconciliation, this script checks if the LDAP account state has changed. If the state has changed, the script updates the historical account properties to indicate the new state (enabled or disabled), and the date that the state was changed. This date can only be approximated and is set to the time that the change was detected by the script.

- `ldapBackCorrelationQuery.js`

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

This script correlates entries in the LDAP directory with managed user identities in OpenIDM.

3.8.1. Configuring the LDAP Server

This sample expects the configuration for the external LDAP server to be the same as described in Section 3.1.1, "LDAP Server Configuration".

The following steps provide setup instructions for an OpenDJ server. Adjust these instructions if you are using an alternative LDAP server.

The LDIF data for this sample is provided in the file `openidm/samples/historicalaccountlinking/data/Example.ldif`. You will need to import this data during your OpenDJ setup.

Although there is only one LDAP server in this example, you must enable *replication* on that server, so that the server has an external change log. The change log is required for LiveSync between OpenDJ and OpenIDM.

1. Download OpenDJ from ForgeRock's download site and extract the zip archive.
2. Install OpenDJ, using either the UI or the command-line setup.
   - If you use the UI, make sure that you enable replication, and import the Example.ldif file during the install.
   - If you use the command-line setup, import the Example.ldif file during the setup, then enable replication when the server has started:

```
$ cd /path/to/opendj
$ ./setup --cli
 \
--hostname localhost
 \
--ldapPort 1389
 \
--rootUserDN "cn=Directory Manager"
 \
--rootUserPassword password
 \
--adminConnectorPort 4444
 \
--baseDN dc=com
 \
--ldifFile /path/to/openidm/samples/historicalaccountlinking/data/Example.ldif
 \
--acceptLicense
 \
--no-prompt
...
Configuring Directory Server ..... Done.
Importing LDIF file /path/to/openidm/samples/sample2/data/Example.ldif ...... Done.
Starting Directory Server ...... Done.
.
...
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ bin/dsconfig create-replication-server
 \
--hostname localhost
 \
--port 4444
 \
-D "cn=Directory Manager"
 \
-w password
 \
--trustAll
 \
--no-prompt
 \
--provider-name "Multimaster Synchronization"
 \
--set replication-port:8989
 \
--set replication-server-id:2
 \
--type generic
$ bin/dsconfig create-replication-domain
 \
--hostname localhost
 \
--port 4444
 \
-D "cn=Directory Manager"
 \
-w password
 \
--trustAll
 \
--no-prompt
 \
--provider-name "Multimaster Synchronization"
 \
--set base-dn:dc=example,dc=com
 \
--set replication-server:localhost:8989
 \
--set server-id:3
 \
--type generic
 \
--domain-name example_com
```

3.8.2. Running the Historical Accounts Sample

This section walks you through each step of the sample to demonstrate how historical accounts are stored.

1. Prepare OpenIDM as described in Section 1.3, "Preparing OpenIDM", then start OpenIDM with the configuration for the historical accounts sample:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ cd /path/to/openidm
$ ./startup.sh -p samples/historicalaccountlinking/
Executing ./startup.sh...
Using OPENIDM_HOME:   /path/to/openidm
Using PROJECT_HOME:   /path/to/openidm/samples/historicalaccountlinking/
Using OPENIDM_OPTS:   -Xmx1024m -Xms1024m
Using LOGGING_CONFIG: -Djava.util.logging.config.file=
      /path/to/openidm/samples/historicalaccountlinking//conf/logging.properties
Using boot properties at
      /path/to/openidm/samples/historicalaccountlinking/conf/boot/boot
.properties
-> OpenIDM ready
```

2. Create a user, Joe Smith, in OpenIDM.

   The following command creates the user over REST, and assigns the user the ID `joesmith`:

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --header "Content-Type: application/json" \
 --request POST \
 --data '{
     "userName": "joe.smith",
     "givenName": "Joe",
     "sn" : "Smith",
     "password" : "Passw0rd",
     "displayName" : "Joe Smith",
     "mail" : "joe.smith@example.com",
     "_id" : "joesmith"
 }' \
 "http://localhost:8080/openidm/managed/user?_action=create"
{
  "_id": "joesmith",
  "_rev": "1",
  "userName": "joe.smith",
  "givenName": "Joe",
  "sn": "Smith",
  "displayName": "Joe Smith",
  "mail": "joe.smith@example.com",
  "accountStatus": "active",
  "effectiveRoles": [],
  "effectiveAssignments": []
}
```

3. Verify that the user Joe Smith was created in OpenDJ.

   Because implicit synchronization is enabled by default, any change to the managed/user repository should be propagated to OpenDJ. For more information about implicit synchronization, see Section 12.1, "Types of Synchronization" in the *Integrator's Guide*.

   The following command returns all IDs in OpenDJ and shows that user joesmith was created successfully:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/system/ldap/account?_queryId=query-all-ids"
{
  "result": [
    {
      "_id": "uid=jdoe,ou=People,dc=example,dc=com",
      "dn": "uid=jdoe,ou=People,dc=example,dc=com"
    },
    {
      "_id": "uid=bjensen,ou=People,dc=example,dc=com",
      "dn": "uid=bjensen,ou=People,dc=example,dc=com"
    },
    {
      "_id": "uid=joe.smith0,ou=People,dc=example,dc=com",
      "dn": "uid=joe.smith0,ou=People,dc=example,dc=com"
    }
  ]
,
...
}
```

Note that Joe Smith's `uid` in OpenDJ is appended with a `0`. The `onCreate` script, defined in the mapping (`sync.json`), increments the `uid` each time a new OpenDJ entry is linked to the same managed user object.

4. Verify that the historical account *relationship object* that corresponds to this linked LDAP account was created in the OpenIDM repository.

   The following command returns all of the `historicalAccounts` for user joesmith:

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/managed/user/joesmith/historicalAccounts?_queryId=query-all"
{
  "result": [
    {
      "_ref": "system/ldap/account/uid=joe.smith0,ou=People,dc=example,dc=com",
      "_refProperties": {
        "stateLastChanged": "Mon Nov 30 2015 14:45:22 GMT-0800 (PST)",
        "state": "enabled",
        "active": true,
        "linkDate": "Mon Nov 30 2015 14:45:22 GMT-0800 (PST)",
        "_id": "ff6913ce-a252-4dc9-a060-b8b56bb32bf4",
        "_rev": "1"
      }
    }
  ]
,
...
}
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

At this stage, Joe Smith has only one historical account link — the link to `uid=joe.smith0,ou=People,dc=example,dc=com`. Note that the relationship properties (`_refProperties`) show the following information about the linked accounts:

- The date on which the accounts were linked
- The fact that this link is currently active
- The state of the account in OpenDJ (`enabled`)

5. Enable the LiveSync schedule to propagate changes made in OpenDJ to the managed user repository.

   To start LiveSync, set `enabled` to `true` in the `conf/schedule-liveSync.json` file:

```
$ cd /path/to/openidm
$ more  samples/historicalaccountlinking/conf/schedule-liveSync.json
{
    "enabled" : true,
    "type" : "cron",
    "schedule" : "0/15 * * * * ?"
,
...
```

6. Use the `manage-account` command in the `opendj/bin` directory to disable Joe Smith's account in OpenDJ:

```
$ cd /path/to/opendj
$ bin/manage-account set-account-is-disabled
 \
--port 4444
 \
--bindDN "cn=Directory Manager"
 \
--bindPassword password
 \
--operationValue true
 \
--targetDN uid=joe.smith0,ou=people,dc=example,dc=com
 \
--trustAll
Account Is Disabled:  true
```

   Within 15 seconds, according to the configured schedule, LiveSync should pick up the change. OpenIDM should then adjust the `state` property in Joe Smith's managed user account.

7. Check Joe Smith's historical accounts again, to make sure that the state of this linked account has changed:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/managed/user/joesmith/historicalAccounts?_queryId=query-all"
{
  "result": [
    {
      "_ref": "system/ldap/account/uid=joe.smith0,ou=People,dc=example,dc=com",
      "_refProperties": {
        "stateLastChanged": "Mon Nov 30 2015 14:54:45 GMT-0800 (PST)",
        "state": "disabled",
        "active": true,
        "linkDate": "Mon Nov 30 2015 14:45:22 GMT-0800 (PST)",
        "_id": "ff6913ce-a252-4dc9-a060-b8b56bb32bf4",
        "_rev": "2"
      }
    }
  ]
,
...
}
```

8. Now, deactivate Joe Smith's managed user account by setting his `accountStatus` property to inactive.

   You can deactivate the account over the REST interface, or by using the Admin UI.

   To use the Admin UI, simply select Manage > User, select Joe Smith's account and change his Status to inactive, on his Details tab.

   The following command deactivates Joe Smith's account over REST:

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --header "Content-Type: application/json" \
 --request PATCH \
 --data '[
      { "operation" : "replace",
        "field" : "accountStatus",
        "value" : "inactive" }
 ]' \
 "http://localhost:8080/openidm/managed/user/joesmith"
{
  "_id": "joesmith",
  "_rev": "3",
  "userName": "joe.smith",
  ...
  "accountStatus": "inactive",
 ...
}
```

9. Request Joe Smith's historical accounts again:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/managed/user/joesmith/historicalAccounts?_queryId=query-all"
{
  "result": [
    {
      "_ref": "system/ldap/account/uid=joe.smith0,ou=People,dc=example,dc=com",
      "_refProperties": {
        "stateLastChanged": "Mon Nov 30 2015 14:54:45 GMT-0800 (PST)",
        "state": "disabled",
        "active": false,
        "linkDate": "Mon Nov 30 2015 14:45:22 GMT-0800 (PST)",
        "unlinkDate": "Mon Nov 30 2015 14:58:30 GMT-0800 (PST)",
        "_id": "ff6913ce-a252-4dc9-a060-b8b56bb32bf4",
        "_rev": "3"
      }
    }
  ]
,
...
}
```

10. Activate Joe Smith's managed user account by setting his `accountStatus` property to active. This action should create a new entry in OpenDJ (with `uid=joe.smith1`), and a new link from Joe Smith's managed user object to that OpenDJ entry.

    You can activate the account over the REST interface, or by using the Admin UI, as described previously.

    The following command activates Joe Smith's account over REST:

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --header "Content-Type: application/json" \
 --request PATCH \
 --data '[
      { "operation" : "replace",
        "field" : "accountStatus",
        "value" : "active" }
 ]' \
 "http://localhost:8080/openidm/managed/user/joesmith"
{
  "_id": "joesmith",
  "_rev": "4",
  "userName": "joe.smith",
  ...
  "accountStatus": "active",
 ...
}
```

11. Verify that a new LDAP entry for user Joe Smith was created in OpenDJ.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

The following command returns all IDs in OpenDJ and shows that two OpenDJ entries for Joe Smith `uid=joe.smith0` and `uid=joe.smith1`.

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/system/ldap/account?_queryId=query-all-ids"
{
  "result": [
    {
      "_id": "uid=jdoe,ou=People,dc=example,dc=com",
      "dn": "uid=jdoe,ou=People,dc=example,dc=com"
    },
    {
      "_id": "uid=bjensen,ou=People,dc=example,dc=com",
      "dn": "uid=bjensen,ou=People,dc=example,dc=com"
    },
    {
      "_id": "uid=joe.smith0,ou=People,dc=example,dc=com",
      "dn": "uid=joe.smith0,ou=People,dc=example,dc=com"
    },
    {
      "_id": "uid=joe.smith1,ou=People,dc=example,dc=com",
      "dn": "uid=joe.smith1,ou=People,dc=example,dc=com"
    }
  ]
,
...
}
```

12. Request Joe Smith's historical accounts again:

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/managed/user/joesmith/historicalAccounts?_queryId=query-all"
{
  "result": [
    {
      "_ref": "system/ldap/account/uid=joe.smith0,ou=People,dc=example,dc=com",
      "_refProperties": {
        "stateLastChanged": "Mon Nov 30 2015 14:54:45 GMT-0800 (PST)",
        "state": "disabled",
        "active": false,
        "linkDate": "Mon Nov 30 2015 14:45:22 GMT-0800 (PST)",
        "unlinkDate": "Mon Nov 30 2015 14:58:30 GMT-0800 (PST)",
        "_id": "ff6913ce-a252-4dc9-a060-b8b56bb32bf4",
        "_rev": "3"
      }
    },
    {
      "_ref": "system/ldap/account/uid=joe.smith1,ou=People,dc=example,dc=com",
      "_refProperties": {
        "stateLastChanged": "Mon Nov 30 2015 15:00:00 GMT-0800 (PST)",
        "state": "enabled",
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
        "active": true,
        "linkDate": "Mon Nov 30 2015 15:00:00 GMT-0800 (PST)",
        "_id": "08443775-7420-4994-bf86-9b29a986bfc9",
        "_rev": "1"
      }
    }
  ],
  ...
}
```

Note that Joe Smith's entry now shows two OpenDJ accounts, but that only the link to `uid=joe.smith1` is `enabled` and `active`.

3.9. Storing Multiple Passwords For Managed Users

This sample demonstrates how to set up multiple passwords for managed users and how to synchronize separate passwords to different external resources. The sample assumes the following scenario:

- The managed/user repository is the source system.
- There are two target LDAP servers — `ldap` and `ldap2`.

  For the purposes of this sample, the two servers are represented by two separate organizational units on a single OpenDJ instance.
- Managed user objects have two additional password fields, each mapped to one of the two LDAP servers.
- The two LDAP servers have different requirements for password policy and encryption.
- Both LDAP servers have a requirement for a password history policy, but the history size differs for the two policies.

  The sample shows how to extend the password history policy, described in Section 15.1.1, "Creating a Password History Policy" in the *Integrator's Guide*, to apply to multiple password fields.
- The value of a managed user's `password` field is used by default for the additional passwords *unless* the CREATE, UPDATE, or PATCH requests on the managed user explicitly contain a value for these additional passwords.

The sample includes the following customized configuration files in the `conf` directory:

- `provisioner.openicf-ldap.json` configures the first LDAP connection.
- `conf/provisioner.openicf-ldap2.json` configures the second LDAP connection.
- `sync.json` provides the mappings from the OpenIDM managed user repository to the respective LDAP servers.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

- `managed.json` contains a customized schema for managed users that includes the additional password fields.

For details of the customizations to these configuration files, see the `README` provided with the sample.

The sample includes the following customized scripts in the `script` directory:

- `onCreate-onUpdate-sync.js` performs custom mapping logic. Specifically, this script maps the pre-hashed password value and sets the target object DN on create events.
- `storeFields.groovy` stores the pre-hashed values of fields in the context chain, on validate events.
- `onCreate-user-custom.js` and `onUpdate-user-custom.js` are used for validation of the password history policy when a user is created or updated.
- `pwpolicy.js` is an additional policy script for the password history policy.
- `set-additional-passwords.js` populates the values of the additional password fields with the value of the main `password` field if the additional fields are not included in the request content.

Note

This sample does not support creation of new users in the Admin UI.

3.9.1. Understanding the Password History Policy

The sample includes a custom password history policy. Although the sample is concerned only about the history of passwords, you can use this policy to enforce history validation on any managed object property.

The following configuration changes set up the password history policy:

- A `fieldHistory` property is added to managed users. The value of this field is a map of field names to a list of historical values for that field. These lists of values are used by the policy to determine if a new value has previously been used.
- The `onCreate-user-custom.js` script performs the standard `onCreate` tasks for a managed user object but also stores the initial value of each of the fields that OpenIDM must keep a history of. The script is passed the following configurable properties:
  - `historyFields` — a list of the fields to store history on.
  - `historySize` — the number of historical fields to store.
- The `onUpdate-user-custom.js` compares the old and new values of the historical fields on update events, to determine if the values have changed. When a new value is detected, it is stored in the list of historical values for that field.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

This script also contains logic to deal with the comparison of encrypted or hashed field values. The script is passed the following configurable properties:

- `historyFields` — a list of the fields to store history on.
- `historySize` — the number of historical fields to store.

- The `pwpolicy.js` script contains the additional policy definition for the historical password policy. This script compares the new field value with the values in the list of historical values for each field.

  The `policy.json` configuration includes this script in its `additionalFiles` list, so that the policy service loads the new policy definition. The new policy takes a `historyLength` parameter, which indicates the number of historical values to enforce the policy on. This number must not exceed the `historySize` specified in the `onCreate` and `onUpdate` scripts.

- The `ldapPassword` and `ldap2Password` fields in the managed user schema have been updated with the policy. For the purposes of this sample the `historySize` has been set to 2 for `ldapPassword` and to 4 for `ldap2Password`.

3.9.2. Configuring the LDAP Server

This sample expects the configuration for the external LDAP server to be the same as described in Section 3.1.1, "LDAP Server Configuration".

The following steps provide setup instructions for an OpenDJ server. Adjust these instructions if you are using an alternative LDAP server.

The LDIF data for this sample is provided in the file `openidm/samples/multiplepasswords/data/Example.ldif`. You will need to import this data during your OpenDJ setup.

1. Download OpenDJ from ForgeRock's download site and extract the zip archive.

2. Install OpenDJ, using either the UI or the command-line setup.

   If you use the UI, import the Example.ldif file during the install.

   If you use the command-line setup, import the Example.ldif file during the setup as follows:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ cd /path/to/opendj
$ ./setup --cli
 \
--hostname localhost
 \
--ldapPort 1389
 \
--rootUserDN "cn=Directory Manager"
 \
--rootUserPassword password
 \
--adminConnectorPort 4444
 \
--baseDN dc=com
 \
--ldifFile /path/to/openidm/samples/multiplepasswords/data/Example.ldif
 \
--acceptLicense
 \
--no-prompt
...
Configuring Directory Server ..... Done.
Importing LDIF file /path/to/openidm/samples/multiplepasswords/data/Example.ldif ...... Done.
Starting Directory Server ...... Done
.
...
```

3. Run an `ldapsearch` on the LDAP directory and look at the organizational units:

```
$ cd /path/to/opendj
$ bin/ldapsearch
 \
--hostname localhost
 \
--port 1389
 \
--bindDN "cn=directory manager"
 \
--bindPassword password
 \
--baseDN "dc=example,dc=com" \
"ou=*" \
ou
dn: ou=People,dc=example,dc=com
ou: people

dn: ou=Customers,dc=example,dc=com
ou: people
ou: Customers
```

The organizational units, `ou=People` and `ou=Customers`, represent the two different target LDAP systems that our mappings point to.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

3.9.3. Demonstrating the Use of Multiple Accounts

This section starts OpenIDM with the sample configuration, then creates a user with multiple passwords, adhering to the different policies in the configured password policy. The section tests that the user was synchronized to two separate LDAP directories, with the different required passwords, and that the user can bind to each of these LDAP directories.

1. Prepare OpenIDM as described in Section 1.3, "Preparing OpenIDM", then start OpenIDM with the configuration for the multiple passwords sample.

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

2. Create a user in OpenIDM. Include a main `password` field but no additional password fields. The additional password fields (`ldapPassword` and `ldap2Password`) will be populated with the value of the main `password` field as a result of the script described previously.

   For the purposes of this example, we will not use the Admin UI, so that the result of each command can be clearly seen. Create the user over REST, by running the following command:

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --header "Content-Type: application/json" \
 --request PUT \
 --data '{
   "userName": "jdoe",
   "givenName": "John",
   "sn" : "Doe",
   "displayName" : "John Doe",
   "mail" : "john.doe@example.com",
   "password" : "Passw0rd"
 }' \
 "http://localhost:8080/openidm/managed/user/jdoe"
{
  "code": 403,
  "reason": "Forbidden",
  "message": "Policy validation failed",
  "detail": {
    "result": false,
    "failedPolicyRequirements": [
      {
        "policyRequirements": [
          {
            "policyRequirement": "AT_LEAST_X_CAPITAL_LETTERS",
            "params": {
              "numCaps": 2
            }
          }
        ],
        "property": "ldapPassword"
      },
      {
        "policyRequirements": [
          {
            "policyRequirement": "AT_LEAST_X_NUMBERS",
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
            "params": {
              "numNums": 2
            }
          }
        ],
        "property": "ldap2Password"
      }
    ]
  }
}
```

Notice that the create request failed with a policy validation failure on the two new password fields. Although the password met the requirement for the main `password` field, the user could not be created because the password did not meet the requirements of the `ldapPassword` and `ldap2Password` fields.

You can fix this problem either by updating the `password` field to one that passes both of the new requirements, or by updating the individual password fields to specifically pass their individual validation requirements.

3. Now, try to create user jdoe again, this time providing individual values for each of the different password fields, that comply with the three different password policies:

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --header "Content-Type: application/json" \
 --request PUT \
 --data '{
   "userName": "jdoe",
   "givenName": "John",
   "sn" : "Doe",
   "displayName" : "John Doe",
   "mail" : "john.doe@example.com",
   "password" : "Passw0rd",
   "ldapPassword" : "PPassw0rd",
   "ldap2Password" : "Passw00rd"
 }' \
 "http://localhost:8080/openidm/managed/user/jdoe"
{
  "_id": "jdoe",
  "_rev": "1",
  "userName": "jdoe",
  "givenName": "John",
  "sn": "Doe",
  "displayName": "John Doe",
  "mail": "john.doe@example.com",
  "ldapPassword": {
    "$crypto": {
      "value": {
        "algorithm": "SHA-256",
        "data": "CpbVZlXAEFL/LUqWyq9Bcks/tLVwJ0pHrc/smLWf8UmC/0BDtEKRo1o2IjB6mNFz"
      },
      "type": "salted-hash"
    }
  },
  "ldap2Password": {
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
    "$crypto": {
      "value": {
        "iv": "TbJlRF+cSFeOguclh8AZVg==",
        "data": "zQ250CXfR3QJ3cBKjpCQhQ==",
        "cipher": "AES/CBC/PKCS5Padding",
        "key": "openidm-sym-default"
      },
      "type": "x-simple-encryption"
    }
  }
,
...
}
```

The user has been created with three different passwords that comply with three distinct password policies. The passwords have been hashed or encrypted, as defined in the `managed.json` file.

4. As a result of implicit synchronization, two separate LDAP accounts should have been created for user jdoe on our two simulated LDAP servers. For more information about implicit synchronization, see Section 12.1, "Types of Synchronization" in the *Integrator's Guide*.

5. Query the IDs in the LDAP directory as follows:

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/system/ldap/account?_queryId=query-all-ids"
{
  "result" : [ {
    "_id" : "uid=jdoe,ou=People,dc=example,dc=com",
    "dn" : "uid=jdoe,ou=People,dc=example,dc=com"
  }, {
    "_id" : "uid=jdoe,ou=Customers,dc=example,dc=com",
    "dn" : "uid=jdoe,ou=Customers,dc=example,dc=com"
  } ]
,
...
}
```

Note that jdoe has two entries - one in `ou=People` and one in `ou=Customers`.

6. Now, see if you can search each LDAP server, as user jdoe, with the separate passwords that you created for each directory.

This step will indicate that the passwords were propagated correctly to the separate LDAP servers.

```
$ cd /path/to/opendj
$ bin/ldapsearch
 \
--hostname localhost
 \
--port 1389
 \
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
--bindDN uid=jdoe,ou=People,dc=example,dc=com
 \
--bindPassword PPassw0rd
 \
--baseDN dc=example,dc=com \
uid=jdoe
dn: uid=jdoe,ou=People,dc=example,dc=com
objectClass: organizationalPerson
objectClass: person
objectClass: inetOrgPerson
objectClass: top
uid: jdoe
mail: john.doe@example.com
sn: Doe
cn: John Doe
userPassword: {SSHA}ot11NT7zidSxXEDtNE+8qQjyfIE3CDbywKGYmQ==
givenName: John

dn: uid=jdoe,ou=Customers,dc=example,dc=com
objectClass: organizationalPerson
objectClass: person
objectClass: inetOrgPerson
objectClass: top
uid: jdoe
mail: john.doe@example.com
sn: Doe
cn: John Doe
givenName: John
$ bin/ldapsearch
 \
--hostname localhost
 \
--port 1389
 \
--bindDN uid=jdoe,ou=Customers,dc=example,dc=com
 \
--bindPassword Passw00rd
 \
--baseDN dc=example,dc=com \
uid=jdoe
dn: uid=jdoe,ou=People,dc=example,dc=com
objectClass: organizationalPerson
objectClass: person
objectClass: inetOrgPerson
objectClass: top
uid: jdoe
mail: john.doe@example.com
sn: Doe
cn: John Doe
userPassword: {SSHA}ot11NT7zidSxXEDtNE+8qQjyfIE3CDbywKGYmQ==
givenName: John

dn: uid=jdoe,ou=Customers,dc=example,dc=com
objectClass: organizationalPerson
objectClass: person
objectClass: inetOrgPerson
objectClass: top
uid: jdoe
mail: john.doe@example.com
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
sn: Doe
cn: John Doe
givenName: John
```

7. Patch jdoe's managed user entry to change his `ldapPassword`.

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --header "Content-Type: application/json" \
 --request PATCH \
 --data '[ {
   "operation" : "replace",
   "field" : "ldapPassword",
   "value" : "TTestw0rd"
 } ]' \
 "http://localhost:8080/openidm/managed/user/jdoe"
{
  "_id": "jdoe",
  "_rev": "2",
  "userName": "jdoe",
  "givenName": "John",
  "sn": "Doe",
  "displayName": "John Doe",
   ...
  "ldapPassword": {
    "$crypto": {
      "value": {
        "algorithm": "SHA-256",
        "data": "Vc6hvmzXaSSdG9WroqOTg3PQVdixhpg9tD/uKT610Z/H5iC6vsoOpE0/R5FaiDUg"
      },
      "type": "salted-hash"
    }
  },
 ...
}
```

8. Search the "first" LDAP server again, as user jdoe, with this new password to verify that the password change was propagated correctly to the LDAP server.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ cd /path/to/opendj
$ bin/ldapsearch
 \
--hostname localhost
 \
--port 1389
 \
--bindDN uid=jdoe,ou=People,dc=example,dc=com
 \
--bindPassword TTestw0rd
 \
--baseDN dc=example,dc=com \
uid=jdoe
dn: uid=jdoe,ou=People,dc=example,dc=com
objectClass: organizationalPerson
objectClass: person
objectClass: inetOrgPerson
objectClass: top
userPassword: {SSHA}pXV9/eZq6L30L/lGTsMV/39Dzjv/zHqIhWpLRw==
uid: jdoe
mail: john.doe@example.com
sn: Doe
givenName: John
cn: John Doe

dn: uid=jdoe,ou=Customers,dc=example,dc=com
objectClass: organizationalPerson
objectClass: person
objectClass: inetOrgPerson
objectClass: top
uid: jdoe
mail: john.doe@example.com
sn: Doe
cn: John Doe
givenName: John
```

3.9.4. Demonstrating the Use of the Password History Policy

This section patches managed user jdoe's entry, changing his `ldapPassword` a number of times, to demonstrate the application of the password history policy.

1. Send the following patch requests, changing the value of jdoe's `ldapPassword` each time:

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --header "Content-Type: application/json" \
 --request PATCH \
 --data '[ {
   "operation" : "replace",
   "field" : "ldapPassword",
   "value" : "TTestw0rd1"
 } ]' \
 "http://localhost:8080/openidm/managed/user/jdoe"
{
  "_id": "jdoe",
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
  "_rev": "3",
  "userName": "jdoe",
  "givenName": "John",
  "sn": "Doe",
  "displayName": "John Doe",
  "mail": "john.doe@example.com"
,
...
  "ldapPassword": {
    "$crypto": {
      "value": {
        "algorithm": "SHA-256",
        "data": "uFacwvr8JsiDwlfI7I/5M+q6jTmQT8e5BaNqxLRcVR+8JxA+/fqyOc8Wo0GhzIz6"
      },
      "type": "salted-hash"
    }
  },
}
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --header "Content-Type: application/json" \
 --request PATCH \
 --data '[ {
   "operation" : "replace",
   "field" : "ldapPassword",
   "value" : "TTestw0rd2"
 } ]' \
 "http://localhost:8080/openidm/managed/user/jdoe"
{
  "_id": "jdoe",
  "_rev": "4",
  "userName": "jdoe",
  "givenName": "John",
  "sn": "Doe",
  "displayName": "John Doe",
  ...
  "ldapPassword": {
    "$crypto": {
      "value": {
        "algorithm": "SHA-256",
        "data": "kzxz6Nc38srk28xhaBLNX1DDtVsauKnDERoXyVy/TSYtEiMWd2KitgTn7498lZs0"
      },
      "type": "salted-hash"
    }
  }
}
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --header "Content-Type: application/json" \
 --request PATCH \
 --data '[ {
   "operation" : "replace",
   "field" : "ldapPassword",
   "value" : "TTestw0rd3"
 } ]' \
 "http://localhost:8080/openidm/managed/user/jdoe"
{
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
  "_id": "jdoe",
  "_rev": "5",
  "userName": "jdoe",
  "givenName": "John",
  "sn": "Doe",
  "displayName": "John Doe",
 ...
  "ldapPassword": {
    "$crypto": {
      "value": {
        "algorithm": "SHA-256",
        "data": "5NEEkfSsUHFOyEHa/C6yXl9x8s3Q5yaLYJgF02Lp/hPQ8DBKmwUU0U37cqFlQLQX"
      },
      "type": "salted-hash"
    }
  }
}
```

User jdoe now has a *history* of `ldapPassword` values, that contains `TTestw0rd3`, `TTestw0rd2`, `TTestw0rd1`, and `TTestw0rd`, in that order. You can see the four separate hashed values in the `fieldHistory` property of the user:

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/managed/user/jdoe?_fields=fieldHistory"
{
  "_id": "jdoe",
  "_rev": "5",
  "fieldHistory": {
    ...
    "ldapPassword": [
      {
        "$crypto": {
          "value": {
            "algorithm": "SHA-256",
            "data": "k1A1udvQo2gAW/5HxFFjs+IG2p34prv36UsVP89YAVv/bALQTAUJjBhml+qrlLBx"
          },
          "type": "salted-hash"
        }
      },
      {
        "$crypto": {
          "value": {
            "algorithm": "SHA-256",
            "data": "LWHaTZYSUp6hP1RChZElfHmfFBQQV+FGtZuHJxsdA/j8sOvjyqeGxk+17IFCX/Ol"
          },
          "type": "salted-hash"
        }
      },
      {
        "$crypto": {
          "value": {
            "algorithm": "SHA-256",
            "data": "I4nR+Kkh0sO53Sy2V7SUc6Hv3eETC9d0HWopgDTBc9FqRZCV2C9ML0kXGJk8FhfV"
          },
          "type": "salted-hash"
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
        }
      },
      {
        "$crypto": {
          "value": {
            "algorithm": "SHA-256",
            "data": "um9iNdwU7XEVArep2X5I0wr4rRy7nacKXNuzzOc7Oa1f+lINHKwZKxaTyBwGbpX2"
          },
          "type": "salted-hash"
        }
      }
    ]
  }
}
```

2. The history size for the `ldapPassword` policy is set to 2. To demonstrate the history policy, attempt to patch jdoe's entry with a password value that was used in his previous 2 password resets: `TTestw0rd2`:

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --header "Content-Type: application/json" \
 --request PATCH \
 --data '[ {
   "operation" : "replace",
   "field" : "ldapPassword",
   "value" : "TTestw0rd2"
 } ]' \
 "http://localhost:8080/openidm/managed/user/jdoe"
{
  "code": 403,
  "reason": "Forbidden",
  "message": "Failed policy validation",
  "detail": {
    "result": false,
    "failedPolicyRequirements": [
      {
        "policyRequirements": [
          {
            "policyRequirement": "IS_NEW"
          }
        ],
        "property": "ldapPassword"
      }
    ]
  }
}
```

The password reset fails the `IS_NEW` policy requirement.

3. Now, reset jdoe's password to a value that was not used in the previous two updates:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --header "Content-Type: application/json" \
 --request PATCH \
 --data '[ {
   "operation" : "replace",
   "field" : "ldapPassword",
   "value" : "TTestw0rd"
 } ]' \
 "http://localhost:8080/openidm/managed/user/jdoe"
{
  "_id": "jdoe",
  "_rev": "5",
  "userName": "jdoe",
  "givenName": "John",
  "sn": "Doe",
  "displayName": "John Doe",
 ...
  "ldapPassword": {
    "$crypto": {
      "value": {
        "algorithm": "SHA-256",
        "data": "5NEEkfSsUHFOyEHa/C6yXl9x8s3Q5yaLYJgF02Lp/hPQ8DBKmwUU0U37cqFlQLQX"
      },
      "type": "salted-hash"
    }
  }
}
```

This time, the password reset succeeds.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

Chapter 4

Samples That Use the Groovy Connector Toolkit to Create Scripted Connectors

OpenIDM 4.5 includes a generic Groovy Connector Toolkit that enables you to run Groovy scripts on any external resource.

The Groovy Connector Toolkit is not a complete connector, in the traditional sense. Rather, it is a framework within which you must write your own Groovy scripts to address the requirements of your implementation. Specific scripts are provided within these samples, which demonstrate how the Groovy Connector Toolkit can be used. These scripts cannot be used "as is" in your deployment, but are a good starting point on which to base your customization.

To facilitate creating your own scripted connectors with the Groovy Connector Toolkit, OpenIDM provides a scripted connector *bundler*. The first sample in this chapter uses the connector bundler to create a new connector, and its configuration. The connector bundler is described in detail in the *OpenICF Developers Guide*.

4.1. Sample 3 - Using the Custom Scripted Connector Bundler to Build a ScriptedSQL Connector

This sample demonstrates the following OpenIDM functionality:

- Custom scripted connector bundler

  The sample uses the custom scripted connector bundler to create a new custom connector. The connector bundler generates a scripted connector, the connector configuration and the Groovy scripts required to communicate with an external MySQL database (HRDB).

- Complex data types

  Complex data types can be stored, retrieved and synchronized like any other object property. They are stored in the managed data as JSON objects, represented as a string, but can be mapped to external resources in any format required. You can customize the mapping to do additional work with or transformations on the complex data types.

  This sample defines one complex data type, `cars`, discussed in more detail later in this section.

- Event hooks to perform actions

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

The mapping from the internal repository to the external `hrdb` database (`managedUser_systemHrdb`), (defined in the `sync.json` file), includes two script hooks. The first hook is for an `onCreate` event and the second is for an `onUpdate` event. Using these event hooks, OpenIDM logs a statement to the log when a user is created or updated in the external system. In this sample, the script source is included in the mapping. However, a script can also be called from an external file. For more information on event hooks, see Section F.2, "Places to Trigger Scripts" in the *Integrator's Guide*.

- Custom scripted endpoints

  All scripted connectors support the configuration of custom scripted endpoints. These are configured in the provisioner configuration file and allow you to execute custom scripts over REST. This example uses a custom scripted endpoint to reset the database and populate it with data. Custom scripted endpoints are illustrated in the the custom script step of Section 4.1.2, "Building the Custom ScriptedSQL Connector".

Caution

Because MySQL cannot "un-hash" user passwords there is no way for a reconciliation operation to retrieve and store the password from MySQL and store it in the managed user object. This issue might impact configurations that support multiple external resources in that passwords might not be synchronized immediately after reconciliation from MySQL to the managed/user repository. Users who are missing from managed/user will be created by the reconciliation but their passwords will be empty. When those users are synchronized to other external resources, they will have empty passwords in those resources. Additional scripting might be required to handle this situation, depending on the requirements of your deployment.

The Groovy scripts required for the sample are located in the `sample3/tools` directory. You will need to customize these scripts to address the requirements of your specific deployment, however, the sample scripts are a good starting point on which to base your customization.

The scripted connector bundler takes a configuration file, in JSON format (`sample3/data/scriptedsql.json`). This file includes the details of the connection to the MySQL server, and the list of object types and properties that will be used in the sample. You can use this configuration file as the basis for creating your own connector with the custom scripted connector bundler.

4.1.1. Before You Start

Before you start with this sample, complete the following steps:

- Prepare a fresh installation of OpenIDM. (See Section 1.3, "Preparing OpenIDM").
- Download and install the Apache Maven build tool.
- Configure an external MySQL database, as follows:

  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/
     ```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

2. Set up MySQL to listen on localhost, port 3306. You will connect to the database as user `root` with password `password`.

   If want to use an existing MySQL instance that runs on a different host or port, adjust the configuration file for the sample (`sample3/data/scriptedsql.json`) before you launch the connector bundler. The default generated configuration is as follows:

```
"configurationProperties" : {
    "username" : "root",
    "password" : "password",
    "driverClassName" : "com.mysql.jdbc.Driver",
    "url" : "jdbc:mysql://localhost:3306/hrdb",
```

3. Create the `hrdb` database, with which OpenIDM will synchronize its managed user repository.

```
$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 58
Server version: 5.7.10

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> CREATE DATABASE hrdb CHARACTER SET utf8 COLLATE utf8_bin;
Query OK, 1 row affected (0.00 sec)
mysql> quit
Bye
```

4.1.2. Building the Custom ScriptedSQL Connector

This section uses the custom scripted connector bundler to generate the classes and configuration files required to build a new connector. The custom connector that you build in this section will be used to complete the sample.

1. Create a new directory named `create-connector` in the `openidm/samples/sample3` directory and change to that new directory.

```
$ mkdir /path/to/openidm/samples/sample3/create-connector
$ cd /path/to/openidm/samples/sample3/create-connector
```

2. Run the custom scripted connector bundler .jar, with the configuration file for this sample (`sample3/data/scriptedsql.json`).

```
$ java -jar ../../../tools/custom-scripted-connector-bundler-4.5.2-SNAPSHOT.jar -c ../data/scriptedsql
.json
Custom Scripted Connector Bundler for OpenIDM v4.5.2-SNAPSHOT
Generating connector sources for HRDB-ScriptedSQLConnector
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

This step generates a Maven project (`pom.xml` file) and a `src` directory that contains the packages to be bundled into the connector.

3. In addition to the generated packages, you must add the scripts required to perform operations on your resource. The scripts to access the resource illustrated in this sample are provided in the `sample3/tools` directory. Copy these scripts into the generated `resources/script/hrdb/` directory, so that they can be bundled with the connector.

```
$ cp ../tools/* src/main/resources/script/hrdb/
```

You can customize these scripts before you bundle them, to suit the requirements of your deployment. For more information about writing Groovy scripts to interact with a resource, see the OpenICF Developer's Guide.

4. Use the Maven build tool to build the custom connector, with the configuration and scripts that you provided in the previous steps.

To run this command, you must be in the `create-connector` directory, in which your Maven project (`pom.xml`) is located.

```
$ mvn install
[INFO] Scanning for projects...
Downloading: http://maven.forgerock.org/repo/releases/org/forgerock/openicf/connectors/
   connectors-parent/1.5.0.0/connectors-parent-1.5.0.0.pom
Downloaded: http://maven.forgerock.org/repo/releases/org/forgerock/openicf/connectors/
 connectors-parent/1.5.0.0/connectors-parent-1.5.0.0.pom (21 KB at 9.2 KB/sec)
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building  1.4.1.0
[INFO] ------------------------------------------------------------------------
-
...
[INFO] Writing OBR metadata
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 48.313 s
[INFO] Finished at: 2015-12-10T14:03:02+02:00
[INFO] Final Memory: 37M/320M
[INFO] ------------------------------------------------------------------------
```

This step generates a connector .jar file (`hrdb-connector-1.4.1.0.jar`) in the `target` directory. This connector .jar will be used in the rest of this sample.

5. Copy the new connector .jar file to the `openidm/connectors` directory, so that it can be picked up by OpenIDM.

```
$ cd /path/to/openim/samples/sample3
$ cp create-connector/target/hrdb-connector-1.4.1.0.jar ../../connectors/
```

You now have a custom-built connector that includes all the required files for it to be displayed in the OpenIDM Admin UI. The bundled connector also includes the scripts and provisioner configuration that enable it to be used with OpenIDM.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

6. Extract the connector configuration file (`provisioner.openicf-hrdb.json`) from the bundled connector into your sample's `conf` directory.

```
$ jar -xvf ../../connectors/hrdb-connector-1.4.1.0.jar conf/provisioner.openicf-hrdb.json
 inflated: conf/provisioner.openicf-hrdb.json
```

7. The generated connector configuration file includes no system actions by default.

   Edit the value of the `"systemActions"` property in the connector configuration file, to call a custom script (`tools/ResetDatabaseScript.groovy`) over the REST interface. This script will reset the `hrdb` database and populate it with sample data.

   The edited excerpt of the `conf/provisioner.openicf-hrdb.json` file should appear as follows:

```
"systemActions": [
  {
    "scriptId": "ResetDatabase",
    "actions": [
      {
        "systemType": ".*HRDBConnector",
        "actionType": "Groovy",
        "actionFile": "tools\/ResetDatabaseScript.groovy"
      }
    ]
  }
],
```

   Currently, only Groovy scripts are supported for these types of actions.

8. Finally, add the generated HTML template file to the UI extensions folder, to enable the new connector to be viewed and configured in the Admin UI.

   Inside the connector jar, locate the file that contains the string `1.4.html`.

```
$ cd /path/to/openidm
$ jar -tvf connectors/hrdb-connector-1.4.1.0.jar | grep "1.4.html"
 12775 Thu Dec 10 14:00:22 SAST 2015 ui/org.forgerock.openicf.connectors.hrdb.HRDBConnector_1.4.html
```

   Create a new extension directory for the connector template.

```
$ mkdir -p ui/admin/extension/templates/connector
```

   Extract the HTML template file that you found in the preceding step and then move it into that directory

```
$ jar -xvf connectors/hrdb-connector-1.4.1.0.jar ui/org.forgerock.openicf.connectors.hrdb.HRDBConnector_1
.4.html
inflated: ui/org.forgerock.openicf.connectors.hrdb.HRDBConnector_1.4.html
$ mv ui/org.forgerock.openicf.connectors.hrdb.HRDBConnector_1.4.html ui/admin/extension/templates/
connector
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

4.1.3. Run the Sample

1. Start OpenIDM with the configuration for sample 3.

```
$ cd /path/to/openidm
$ ./startup.sh -p samples/sample3
Executing ./startup.sh...
Using OPENIDM_HOME:   /path/to/openidm
Using PROJECT_HOME:   /path/to/openidm/samples/sample3/
Using OPENIDM_OPTS:   -Xmx1024m -Xms1024m
Using LOGGING_CONFIG: -Djava.util.logging.config.file=/path/to/openidm/samples/sample3//conf/logging
.properties
Using boot properties at /path/to/openidm/samples/sample3/conf/boot/boot
.properties
-> OpenIDM ready
```

2. Run the custom script described in the previous section to reset the database and populate it with sample data.

   You can run the script again, at any point, to reset the database.

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request POST \
 "http://localhost:8080/openidm/system/hrdb?_action=script&scriptId=ResetDatabase"
{
  "actions": [
    {
      "result": "Database reset successful."
    }
  ]
}
```

   The `hrdb` database should now be populated with sample data.

   You can review the contents of the database as follows:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ mysql -u root -p
Enter password:
...
mysql > use hrdb;
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 > select * from users;

+----+--------+--------------+-----------+----------+---------------+--------...
| id | uid    | password     | firstname | lastname | fullname      | email  ...
+----+--------+--------------------------------------+-----------+-------...
|  1 | bob    | e38ad2149... | Bob       | Fleming  | Bob Fleming   | Bob.Fle...
|  2 | rowley | 2aa60a8ff... | Rowley    | Birkin   | Rowley Birkin | Rowley....
|  3 | louis  | 1119cfd37... | Louis     | Balfour  | Louis Balfour | Louis.B...
|  4 | john   | a1d7584da... | John      | Smith    | John Smith    | John.Sm...
|  5 | jdoe   | edba955d0... | John      | Doe      | John Doe      | John.Do...
+----+--------+--------------------------------------+-----------+-------...
5 rows in set (0.00 sec)
```

Note

The passwords in the output shown above are hashed to the SHA-1 standard, as they cannot be read into OpenIDM as clear text. The SHA-1 Hash function is used for compatibility reasons. Use a more secure algorithm in a production database.

4.1.4. Reconciling the Repository

1. The mapping configuration file (`sync.json`) for this sample includes the mapping `systemHrdb_managedUser`, which synchronizes users from the source `hrdb` database with the target OpenIDM repository.

   You can test this part of the sample by using the `curl` command-line utility, or the OpenIDM Administration UI.

   - To reconcile the repository by using the Administration UI:

     1. Log in to the Admin UI at the URL `https://localhost:8443/admin` as the default administrative user (`openidm-admin`) with password `openidm-admin`.

        Warning

        To protect your deployment in production, change the default administrative password. To do so, select Self-Service from the dropdown list at the top right of the screen and click Change Password.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

Return to the Admin View to continue with the sample. (Select Admin View from the top right dropdown list.)

2. Select Configure > Mappings.

   The Mappings page shows two configured mappings, one from the `hrdb` database to the OpenIDM repository (`managed/user`), and one in the opposite direction.

3. Click the first mapping (systemHrdb_managedUser) and click Reconcile Now.

- To reconcile the repository by using the command-line, launch the reconciliation operation 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&waitForCompletion=true"
{
  "state": "SUCCESS",
  "_id": "f3c618aa-cc3b-49ed-9a3a-00b012db2513"
}
```

The reconciliation operation creates the five users from the MySQL database in the OpenIDM repository.

2. Retrieve the list of users from the repository.

- To retrieve the users in the repository from the Admin UI:

  1. Select Manage > User to display the User List.

     The five users from the `hrdb` database have been reconciled to the OpenIDM repository.

  2. To retrieve the details of a specific user, click that user entry.

- To retrieve the users from the repository by using the command-line, query the IDs in the repository as follows:

```
$ 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"
{
  "result": [
    {
      "_id": "9d7c304a-fd89-4b58-bd6a-99b2a6a94691",
      "_rev": "1"
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
    },
    {
      "_id": "53479e98-5460-421c-9e81-0f3a7cc45881",
      "_rev": "1"
    },
    {
      "_id": "4103b904-c7d6-45c2-a9ca-8e563a975fa8",
      "_rev": "1"
    },
    {
      "_id": "1ea17866-aaed-4c51-b3a8-5fa8eb600e04",
      "_rev": "1"
    },
    {
      "_id": "074588a6-64f8-4cce-bb2f-33490aab90ae",
      "_rev": "1"
    }
  ],
  "resultCount": 5,
  "pagedResultsCookie": null,
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

To retrieve a complete user record, query the userName of the individual user entry. The following query returns the record for the user `Rowley Birkin`:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/managed/user/?_queryId=for-userName&uid=rowley"
{
  "result": [
    {
      "_id": "53479e98-5460-421c-9e81-0f3a7cc45881",
      "_rev": "1",
      "mail": "Rowley.Birkin@example.com",
      "userName": "rowley",
      "sn": "Birkin",
      "organization": "SALES",
      "givenName": "Rowley",
      "cars": [
        {
          "year": "2013",
          "make": "BMW",
          "model": "328ci"
        },
        {
          "year": "2010",
          "make": "Lexus",
          "model": "ES300"
        }
      ],
      "accountStatus": "active"
,
...
 }
```

Regardless of how you have retrieved Rowley Birkin's entry, note the `cars` property in this user's entry. This property demonstrates a complex object, stored in JSON format in the user entry, as a list that contains multiple objects. In the MySQL database, the `car` table joins to the `users` table through a `cars.users_id` column. The Groovy scripts read this data from MySQL and repackage it in a way that OpenIDM can understand. With support for complex objects, the data is passed through to OpenIDM as a list of `car` objects. Data is synchronized from OpenIDM to MySQL in the same way. Complex objects can also be nested to any depth.

Group membership (not demonstrated here) is maintained with a traditional "join table" in MySQL (`groups_users`). OpenIDM does not maintain group membership in this way, so the Groovy scripts do the work to translate membership between the two resources.

4.1.5. Using Paging With Sample 3

All OpenICF connectors from version 1.4 onwards support the use of paging parameters to restrict query results. The following command indicates that only two records should be returned (`_pageSize=2`) and that the records should be sorted according to their `timestamp` and `_id` (`_sortKeys=timestamp,id`). Including the `timestamp` in the sort ensures that, as you page through the set, changes to records that have already been visited are not lost. Instead, those records are pushed onto the last page:

```
$ curl \
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/system/hrdb/account?_queryFilter=uid+sw+
%22%22&_pageSize=2&_sortKeys=timestamp,id"
{
  "result": [
    {
      "_id": "1",
      "email": "Bob.Fleming@example.com",
      "cars": [
        {
          "year": "1979",
          "make": "Ford",
          "model": "Pinto"
        }
      ],
      "uid": "bob",
      "organization": "HR",
      "firstName": "Bob",
      "fullName": "Bob Fleming",
      "lastName": "Fleming"
    },
    {
      "_id": "2",
      "email": "Rowley.Birkin@example.com",
      "cars": [
        {
          "year": "2013",
          "make": "BMW",
          "model": "328ci"
        }
      ],
      "uid": "rowley",
      "organization": "SALES",
      "firstName": "Rowley",
      "fullName": "Rowley Birkin",
      "lastName": "Birkin"
    }
  ],
  "resultCount": 2,
  "pagedResultsCookie": "2015-12-10 14:16:46.0,2",
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

The `pagedResultsCookie` is used by the server to keep track of the position in the search results. You can ignore the `"remainingPagedResults": -1` in the output. The real value of this property is not returned because the scripts that the connector uses do not do any counting of the records in the resource.

Using the `pagedResultsCookie` from the previous step, run a similar query, to retrieve the following set of records in the database. Note that the value of the `pagedResultsCookie` must be URL-encoded, as shown in the following example:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/system/hrdb/account?_queryId=query-all-ids&_pageSize=2&_sortKeys=timestamp
,id&_pagedResultsCookie=2015-12-10+14%3A16%3A46.0%2C2"
{
  "result": [
    {
      "_id": "3",
      "uid": "louis"
    },
    {
      "_id": "4",
      "uid": "john"
    }
  ],
  "resultCount": 2,
  "pagedResultsCookie": "2015-12-10 14:16:46.0,4",
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

For more information about paging support, see Section 7.3.5, "Paging and Counting Query Results" in the *Integrator's Guide*.

4.2. Sample - Using the Groovy Connector Toolkit to Connect to OpenDJ With ScriptedREST

This sample uses the Groovy Connector Toolkit to implement a ScriptedREST connector, which interacts with the OpenDJ REST API.

The Groovy Connector Toolkit is bundled with OpenIDM 4.5, in the JAR `openidm/connectors/groovy-connector-1.4.2.1.jar`.

The connector configuration file for this sample (`samples/scriptedrest2dj/conf/provisioner.openicf-scriptedrest.json`) indicates the ScriptedREST implementation of the Groovy connector as follows:

```
{
    "name": "scriptedrest",
    "connectorRef": {
        "connectorHostRef": "#LOCAL",
        "connectorName": "org.forgerock.openicf.connectors.scriptedrest.ScriptedRESTConnector",
        "bundleName": "org.forgerock.openicf.connectors.groovy-connector",
        "bundleVersion": "[1.4.0.0,2.0.0.0)"
    },
...
```

The Groovy scripts required for the sample are located in the `samples/scriptedrest2dj/tools` directory. You will need to customize these scripts to address the requirements of your specific deployment, however, the sample scripts are a good starting point on which to base your customization.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

Important

The Rest2ldap HTTP endpoint provided with OpenDJ is an evolving interface. As such, compatibility between versions is not guaranteed. This sample is designed to work with OpenDJ 3.0.0 and does not work, out of the box, with OpenDJ 3.5.0.

4.2.1. Setting Up OpenDJ

This sample assumes an OpenDJ server, running on the localhost. Follow these steps to install and configure an OpenDJ instance.

1. Download and extract the OpenDJ zip archive from https://forgerock.org/downloads/.

2. Install OpenDJ using the command-line setup, as follows:

```
$ cd /path/to/opendj
$ ./setup --cli \
  --hostname localhost \
  --ldapPort 1389 \
  --rootUserDN "cn=Directory Manager" \
  --rootUserPassword password \
  --adminConnectorPort 4444 \
  --addBaseEntry \
  --baseDN dc=com \
  --acceptLicense \
  --no-prompt
...
Configuring Directory Server ..... Done.
Creating Base Entry dc=com ..... Done.
Starting Directory Server ....... Done
.
...
```

   The sample assumes the following configuration:

   - The server is installed on the localhost.
   - The server listens for LDAP connections on port 1389.
   - The administration connector port is 4444.
   - The root user DN is `cn=Directory Manager`.
   - The root user password is `password`.

3. Configure the OpenDJ server for replication.

   To enable LiveSync, this server must be configured for replication, even if it does not actually participate in a replication topology. The following commands configure the server for replication.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ cd /path/to/opendj/bin
$ ./dsconfig create-replication-server \
  --hostname localhost \
  --port 4444 \
  --bindDN "cn=Directory Manager" \
  --bindPassword password \
  --provider-name "Multimaster Synchronization" \
  --set replication-port:8989 \
  --set replication-server-id:2 \
  --type generic \
  --trustAll \
  --no-prompt

$ ./dsconfig create-replication-domain \
  --hostname localhost \
  --port 4444 \
  --bindDN "cn=Directory Manager" \
  --bindPassword password \
  --provider-name "Multimaster Synchronization" \
  --domain-name example_com \
  --set base-dn:dc=example,dc=com \
  --set replication-server:localhost:8989 \
  --set server-id:3 \
  --type generic \
  --trustAll \
  --no-prompt
```

4. Enable HTTP access to the OpenDJ directory server as follows:

```
$ ./dsconfig set-connection-handler-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --handler-name "HTTP Connection Handler" \
 --set enabled:true \
 --set listen-port:8090 \
 --no-prompt \
 --trustAll
```

5. Enable the OpenDJ HTTP access log.

```
$ ./dsconfig set-log-publisher-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --publisher-name "File-Based HTTP Access Logger" \
 --set enabled:true \
 --no-prompt \
 --trustAll
```

6. Import the LDIF data required for the sample.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ ./ldapmodify \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --hostname localhost \
 --port 1389 \
 --filename /path/to/openidm/samples/scriptedrest2dj/data/ldap.ldif
Processing ADD request for dc=example,dc=com
ADD operation successful for DN dc=example,dc=com
Processing ADD request for ou=Administrators,dc=example,dc=com
ADD operation successful for DN ou=Administrators,dc=example,dc=com
Processing ADD request for uid=idm,ou=Administrators,dc=example,dc=com
ADD operation successful for DN uid=idm,ou=Administrators,dc=example,dc=com
Processing ADD request for ou=People,dc=example,dc=com
ADD operation successful for DN ou=People,dc=example,dc=com
Processing ADD request for ou=Groups,dc=example,dc=com
ADD operation successful for DN ou=Groups,dc=example,dc=com
```

7. To configure the mapping between JSON resources and LDAP entries, copy the the configuration file for the HTTP connection handler (`scriptedrest2dj/data/http-config.json`) to OpenDJ's configuration directory.

```
$ cd /path/to/opendj
$ cp /path/to/openidm/samples/scriptedrest2dj/data/http-config.json config/
```

8. Restart OpenDJ for the configuration change to take effect.

```
$ cd /path/to/opendj/bin
$ ./stop-ds --restart
Stopping Server...
The Directory Server has started successfully
```

OpenDJ is now configured for this sample.

4.2.2. Running the Sample

This section illustrates the basic CRUD operations on users and groups using the ScriptedREST connector and the OpenDJ REST API. Note that the power of the Groovy connector is in the associated Groovy scripts, and their application in your particular deployment. The scripts provided with this sample are specific to the sample and customization of the scripts is required.

1. Start OpenIDM with the configuration for the ScriptedREST sample.

```
$ cd /path/to/openidm
$ ./startup.sh -p samples/scriptedrest2dj/
```

2. Check the connector configuration is correct by obtaining the status of the connector, over REST.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request POST \
 "http://localhost:8080/openidm/system/scriptedrest?_action=test"
{
  "name": "scriptedrest",
  "enabled": true,
  "config": "config/provisioner.openicf/scriptedrest",
  "objectTypes": [
    "__ALL__",
    "account",
    "group"
  ],
  "connectorRef": {
    "bundleName": "org.forgerock.openicf.connectors.groovy-connector",
    "connectorName": "org.forgerock.openicf.connectors.scriptedrest.ScriptedRESTConnector",
    "bundleVersion": "[1.4.0.0,2.0.0.0)"
  },
  "displayName": "Scripted REST Connector",
  "ok": true
}
```

3. Create a group entry on the OpenDJ server.

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --header "Content-Type: application/json" \
 --request POST \
 --data '{
   "_id" : "group1"
 }' \
 "http://localhost:8080/openidm/system/scriptedrest/group?_action=create"
{
    "_id": "group1",
    "cn": "group1",
    "members": null,
    "lastModified": null,
    "created": "2014-09-24T17:34:27Z",
    "displayName": "group1"
}
```

4. Create a user entry on the OpenDJ server.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --header "Content-Type: application/json" \
 --request POST \
 --data '{
   "givenName" : "Steven",
   "familyName" : "Carter",
   "emailAddress" : "scarter@example.com",
   "telephoneNumber" : "444-444-4444",
   "password" : "Passw0rd",
   "displayName" : "Steven.Carter",
   "uid" : "scarter"
 }' \
 http://localhost:8080/openidm/system/scriptedrest/account?_action=create
{
    "_id": "scarter",
    "displayName": "Steven.Carter",
    "uid": "scarter",
    "groups": null,
    "familyName": "Carter",
    "emailAddress": "steven.carter@example.com",
    "givenName": "Steven",
    "created": "2014-09-24T17:35:46Z",
    "telephoneNumber": "444-444-4444"
}
```

Notice that at this stage, the user is not a member of any group.

5. Update Steven Carter's entry, by modifying his telephone number.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --header "Content-Type: application/json" \
 --header "If-Match: *" \
 --request PUT \
 --data '{
   "givenName" : "Steven",
   "familyName" : "Carter",
   "emailAddress" : "scarter@example.com",
   "telephoneNumber" : "555-555-5555",
   "password" : "Passw0rd",
   "displayName" : "Steven.Carter",
   "uid" : "scarter"
 }' \
 http://localhost:8080/openidm/system/scriptedrest/account/scarter
{
    "_id": "scarter",
    "displayName": "Steven.Carter",
    "uid": "scarter",
    "groups": null,
    "familyName": "Carter",
    "emailAddress": "steven.carter@example.com",
    "givenName": "Steven",
    "created": "2014-09-24T17:35:46Z",
    "telephoneNumber": "555-555-5555"
}
```

6. Add Steven Carter to the group you created previously, by updating the group entry.

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --header "Content-Type: application/json" \
 --header "If-Match: *" \
 --request PUT \
 --data '{
 "_id" : "group1",
 "members" : [{"_id" : "scarter"}]
 }' \
 http://localhost:8080/openidm/system/scriptedrest/group/group1
{
    "_id": "group1",
    "cn": "group1",
    "members": [
        {
            "displayName": "Steven.Carter",
            "_id": "scarter"
        }
    ],
    "lastModified": "2014-09-24T17:31:42Z",
    "created": "2014-09-24T17:27:37Z",
    "displayName": "group1"
}
```

7. Read Steven Carter's entry, to verify that he is now a member of group1.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 http://localhost:8080/openidm/system/scriptedrest/account/scarter
{
    "_id": "scarter",
    "displayName": "Steven.Carter",
    "uid": "scarter",
    "groups": [
        {
            "_id": "group1"
        }
    ],
    "familyName": "Carter",
    "emailAddress": "steven.carter@example.com",
    "givenName": "Steven",
    "created": "2014-09-24T17:31:04Z",
    "telephoneNumber": "555-555-5555"
}
```

8. Read the group entry to verify its members.

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 http://localhost:8080/openidm/system/scriptedrest/group/group1
{
    "_id": "group1",
    "cn": "group1",
    "members": [
        {
            "displayName": "Steven.Carter",
            "_id": "scarter"
            }
    ],
    "lastModified": "2014-09-24T17:31:42Z",
    "created": "2014-09-24T17:27:37Z",
    "displayName": "group1"
}
```

9. Delete the user and group entries, returning the OpenDJ server to its initial state.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request DELETE \
 http://localhost:8080/openidm/system/scriptedrest/account/scarter
{
    "_id": "scarter"
}
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request DELETE \
 http://localhost:8080/openidm/system/scriptedrest/group/group1
{
    "_id": "group1"
}
```

4.3. Using the Groovy Connector Toolkit to Connect to OpenDJ With ScriptedCREST

This sample uses the Groovy Connector Toolkit to implement a ScriptedCREST connector, which interacts with the ForgeRock Commons REST (CREST) API to connect to an OpenDJ instance. The main difference between a CREST-based API and a generic REST API is that the CREST API is inherently recognizable by all ForgeRock products. As such, the sample can leverage CREST resources in the groovy scripts, to create CREST requests.

The Groovy Connector Toolkit is bundled with OpenIDM 4.5, in the JAR `openidm/connectors/groovy-connector-1.4.2.1.jar`.

The connector configuration file for this sample (`samples/scriptedcrest2dj/conf/provisioner.openicf-scriptedcrest.json`) indicates the ScriptedCREST implementation of the Groovy Connector Toolkit as follows:

```
{
    "name": "scriptedcrest",
    "connectorRef": {
        "connectorHostRef": "#LOCAL",
        "connectorName": "org.forgerock.openicf.connectors.scriptedcrest.ScriptedCRESTConnector",
        "bundleName": "org.forgerock.openicf.connectors.groovy-connector",
        "bundleVersion": "[1.4.0.0,2.0.0.0)"
    },
...
```

The Groovy scripts required for the sample are located in the `samples/scriptedcrest2dj/tools` directory. You will need to customize these scripts to address the requirements of your specific deployment, however, the sample scripts are a good starting point on which to base your customization.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

Important

The Rest2ldap HTTP endpoint provided with OpenDJ is an evolving interface. As such, compatibility between versions is not guaranteed. This sample is designed to work with OpenDJ 3.0.0 and does not work, out of the box, with OpenDJ 3.5.0.

4.3.1. Setting Up OpenDJ

This sample assumes an OpenDJ server, running on the localhost. Follow these steps to install and configure an OpenDJ instance.

1. Download and extract the OpenDJ zip archive from https://forgerock.org/downloads/.

2. Install OpenDJ using the command-line setup, as follows:

```
$ cd /path/to/opendj
$ ./setup --cli \
  --hostname localhost \
  --ldapPort 1389 \
  --rootUserDN "cn=Directory Manager" \
  --rootUserPassword password \
  --adminConnectorPort 4444 \
  --addBaseEntry \
  --baseDN dc=com \
  --acceptLicense \
  --no-prompt
...
Configuring Directory Server ..... Done.
Creating Base Entry dc=com ..... Done.
Starting Directory Server ....... Done
.
...
```

   The sample assumes the following configuration:

   - The server is installed on the localhost.
   - The server listens for LDAP connections on port 1389.
   - The administration connector port is 4444.
   - The root user DN is `cn=Directory Manager`.
   - The root user password is `password`.

3. Configure the OpenDJ server for replication.

   To enable liveSync, this server must be configured for replication, even if it does not actually participate in a replication topology. The following commands configure the server for replication.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ cd /path/to/opendj/bin
$ ./dsconfig create-replication-server \
  --hostname localhost \
  --port 4444 \
  --bindDN "cn=Directory Manager" \
  --bindPassword password \
  --provider-name "Multimaster Synchronization" \
  --set replication-port:8989 \
  --set replication-server-id:2 \
  --type generic \
  --trustAll \
  --no-prompt

$ ./dsconfig create-replication-domain \
  --hostname localhost \
  --port 4444 \
  --bindDN "cn=Directory Manager" \
  --bindPassword password \
  --provider-name "Multimaster Synchronization" \
  --domain-name example_com \
  --set base-dn:dc=example,dc=com \
  --set replication-server:localhost:8989 \
  --set server-id:3 \
  --type generic \
  --trustAll \
  --no-prompt
```

4. Enable HTTP access to the OpenDJ directory server as follows:

```
$ ./dsconfig set-connection-handler-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --handler-name "HTTP Connection Handler" \
 --set enabled:true \
 --set listen-port:8090 \
 --no-prompt \
 --trustAll
```

5. Enable the OpenDJ HTTP access log.

```
$ ./dsconfig set-log-publisher-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --publisher-name "File-Based HTTP Access Logger" \
 --set enabled:true \
 --no-prompt \
 --trustAll
```

6. Import the LDIF data required for the sample.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ ./ldapmodify \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --hostname localhost \
 --port 1389 \
 --filename /path/to/openidm/samples/scriptedcrest2dj/data/ldap.ldif
Processing ADD request for dc=example,dc=com
ADD operation successful for DN dc=example,dc=com
Processing ADD request for ou=Administrators,dc=example,dc=com
ADD operation successful for DN ou=Administrators,dc=example,dc=com
Processing ADD request for uid=idm,ou=Administrators,dc=example,dc=com
ADD operation successful for DN uid=idm,ou=Administrators,dc=example,dc=com
Processing ADD request for ou=People,dc=example,dc=com
ADD operation successful for DN ou=People,dc=example,dc=com
Processing ADD request for ou=Groups,dc=example,dc=com
ADD operation successful for DN ou=Groups,dc=example,dc=com
```

7. To configure the mapping between JSON resources and LDAP entries, copy the the configuration file for the HTTP connection handler (`scriptedcrest2dj/data/http-config.json`) to OpenDJ's configuration directory.

```
$ cd /path/to/opendj
$ cp /path/to/openidm/samples/scriptedcrest2dj/data/http-config.json config/
```

8. Restart OpenDJ for the configuration change to take effect.

```
$ cd /path/to/opendj/bin
$ ./stop-ds --restart
Stopping Server...
The Directory Server has started successfully
```

OpenDJ is now configured for this sample.

4.3.2. Running the Sample

This section illustrates the basic CRUD operations on users and groups using the ScriptedCREST connector implementation and the OpenDJ REST API. Note that the power of the Groovy connector is in the associated Groovy scripts, and their application in your specific deployment. The scripts provided with this sample are specific to the sample and customization of the scripts is required.

1. Start OpenIDM with the configuration for the ScriptedCREST sample.

```
$ cd /path/to/openidm
$ ./startup.sh -p samples/scriptedcrest2dj/
```

2. Check the connector configuration is correct by obtaining the status of the connector, over REST.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request POST \
 "http://localhost:8080/openidm/system/scriptedcrest?_action=test"
{
  "ok": true,
  "connectorRef": {
    "bundleVersion": "[1.4.0.0,2.0.0.0)",
    "bundleName": "org.forgerock.openicf.connectors.groovy-connector",
    "connectorName": "org.forgerock.openicf.connectors.scriptedcrest.ScriptedCRESTConnector"
  },
  "objectTypes": [
    "groups",
    "users"
  ],
  "config": "config/provisioner.openicf/scriptedcrest",
  "enabled": true,
  "name": "scriptedcrest"
}
```

3. Create a group entry on the OpenDJ server.

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --header "Content-Type: application/json" \
 --request POST \
 --data '{
   "_id" : "group1"
 }' \
 "http://localhost:8080/openidm/system/scriptedcrest/groups?_action=create"
{
  "_rev": "0000000028f53bdf",
  "_id": "group1",
  "displayName": "group1",
  "meta": {
    "created": "2014-10-17T07:43:13Z"
  }
}
```

4. Create a user entry on the OpenDJ server.

```
$ curl \
 --header "Content-Type: application/json" \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request POST \
 --data '{
     "name": {
         "familyName": "Carter",
         "givenName" : "Steven"
     },
     "contactInformation": {
         "emailAddress" : "scarter@example.com",
         "telephoneNumber" : "444-444-4444"
     },
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
     "password" : "TestPassw0rd",
     "displayName" : "Steven.Carter",
     "_id" : "scarter"
 }' \
 "http://localhost:8080/openidm/system/scriptedcrest/users?_action=create"
{
  "_rev": "00000000d84482de",
  "meta": {
    "created": "2014-10-17T08:07:46Z"
  },
  "userName": "scarter@example.com",
  "contactInformation": {
    "emailAddress": "scarter@example.com",
    "telephoneNumber": "444-444-4444"
  },
  "name": {
    "givenName": "Steven",
    "familyName": "Carter"
  },
  "displayName": "Steven.Carter",
  "_id": "scarter"
}
```

Notice that at this stage, the user is not a member of any group.

5. Update Steven Carter's entry, by modifying his telephone number.

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --header "Content-Type: application/json" \
 --header "If-Match: *" \
 --request PUT \
 --data '{
     "name": {
         "familyName": "Carter",
         "givenName" : "Steven"
     },
     "contactInformation": {
         "emailAddress" : "scarter@example.com",
         "telephoneNumber" : "555-555-5555"
     },
     "password" : "TestPassw0rd",
     "displayName" : "Steven.Carter",
     "_id" : "scarter"
 }' \
 "http://localhost:8080/openidm/system/scriptedcrest/users/scarter"
{
  "_rev": "00000000eb8ba31c",
  "meta": {
    "created": "2014-10-17T08:07:46Z",
    "lastModified": "2014-10-17T08:25:05Z"
  },
  "userName": "scarter@example.com",
  "contactInformation": {
    "emailAddress": "scarter@example.com",
    "telephoneNumber": "555-555-5555"
  },
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
  "name": {
    "givenName": "Steven",
    "familyName": "Carter"
  },
  "displayName": "Steven.Carter",
  "_id": "scarter"
}
```

6. Add Steven Carter to the group you created previously, by updating the members of the group entry.

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --header "Content-Type: application/json" \
 --header "If-Match: *" \
 --request PUT \
 --data '{
 "_id" : "group1",
 "members" : [{"_id" : "scarter"}]
 }' \
 "http://localhost:8080/openidm/system/scriptedcrest/groups/group1"
{
  "_rev": "0000000011ed6ea1",
  "members": [
    {
      "displayName": "Steven.Carter",
      "_id": "scarter"
    }
  ],
  "_id": "group1",
  "displayName": "group1",
  "meta": {
    "created": "2014-10-17T07:43:13Z",
    "lastModified": "2014-10-17T08:26:41Z"
  }
}
```

7. Read Steven Carter's entry, to verify that he is now a member of group1.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/system/scriptedcrest/users/scarter"
{
  "_rev": "00000000eb8ba31c",
  "groups": [
    {
      "_id": "group1"
    }
  ],
  "meta": {
    "created": "2014-10-17T08:07:46Z",
    "lastModified": "2014-10-17T08:25:05Z"
  },
  "userName": "scarter@example.com",
  "contactInformation": {
    "emailAddress": "scarter@example.com",
    "telephoneNumber": "555-555-5555"
  },
  "name": {
    "givenName": "Steven",
    "familyName": "Carter"
  },
  "displayName": "Steven.Carter",
  "_id": "scarter"
}
```

8. Read the group entry to verify its members.

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/system/scriptedcrest/groups/group1"
{
  "_rev": "0000000011ed6ea1",
  "members": [
    {
      "displayName": "Steven.Carter",
      "_id": "scarter"
    }
  ],
  "_id": "group1",
  "displayName": "group1",
  "meta": {
    "created": "2014-10-17T07:43:13Z",
    "lastModified": "2014-10-17T08:26:41Z"
  }
}
```

9. Delete the user and group entries, returning the OpenDJ server to its initial state.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request DELETE \
 "http://localhost:8080/openidm/system/scriptedcrest/users/scarter"
{
    "_id": "scarter"
}
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request DELETE \
 "http://localhost:8080/openidm/system/scriptedcrest/groups/group1"
{
    "_id": "group1"
}
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

Chapter 5

Samples That Use the PowerShell Connector Toolkit to Create Scripted Connectors

OpenICF provides a generic PowerShell Connector Toolkit that enables you to run PowerShell scripts on any external resource. The PowerShell Connector Toolkit is not a complete connector, in the traditional sense. Rather, it is a framework within which you must write your own PowerShell scripts to address the requirements of your Microsoft Windows ecosystem. You can use the PowerShell Connector Toolkit to create connectors that can provision any Microsoft system. This chapter describes sample connector implementations that enable you to provision to an Active Directory server instance, and to Azure Active Directory (Azure AD).

The PowerShell Connector Toolkit is available, with a subscription, from the ForgeRock Backstage site.

5.1. Connect to Active Directory

This sample provides a number of PowerShell scripts that enable you to perform basic CRUD (create, read, update, delete) operations on an Active Directory server. The samples use the MS Active Directory PowerShell module. For more information on this module, see the corresponding Microsoft documentation.

This sample assumes that OpenIDM is running on a Windows system on the localhost. It also assumes that Active Directory and the OpenICF .NET connector server run on a remote Windows server. The PowerShell connector runs on the .NET connector server.

To use this sample for OpenIDM instances installed on UNIX systems, adjust the relevant commands shown with PowerShell prompts.

5.1.1. Setting Up the PowerShell Active Directory Sample

Run the commands in this procedure from the PowerShell command line. The continuation character used in the command is the back-tick (`).

1. Install, configure, and start the .NET connector server on the machine that is running an Active Directory Domain Controller or on a workstation on which the Microsoft Active Directory PowerShell module is installed.

   For instructions on installing the .NET connector server, see Procedure 11.3, "Installing the .NET Connector Server" in the *Integrator's Guide*.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

2. Configure OpenIDM to connect to the .NET connector server.

   To do so, copy the remote connector provisioner file from the `openidm\samples\provisioners` directory to your project's `conf\` directory, and edit the file according to your configuration.

```
PS C:\ cd \path\to\openidm
PS C:\path\to\openidm cp samples\provisioners\provisioner.openicf.connectorinfoprovider.json conf
```

   For instructions on editing this file, see Procedure 11.7, "Configuring OpenIDM to Connect to the .NET Connector Server" in the *Integrator's Guide*.

3. Download the PowerShell Connector Toolkit archive (`mspowershell-connector-1.4.2.0.zip`) from the ForgeRock Backstage site.

   Extract the archive and move the `MsPowerShell.Connector.dll` to the folder in which the connector server application (`connectorserver.exe`) is located.

4. Copy the PowerShell scripts from the `samples\powershell2AD\tools` directory, to the machine on which the connector server is installed.

```
PS C:\path\to\openidm>dir samples\powershell2AD\tools
Directory: C:\path\to\openidm\samples\powershell2AD\tools

Mode             LastWriteTime   Length Name
----             -------------   ------ ----

 -----     11/15/2015  01:55 PM   2813 ADAuthenticate.ps1
 -----     11/15/2015  01:55 PM  10019 ADCreate.ps1
 -----     11/15/2015  01:55 PM   2530 ADDelete.ps1
 -----     11/15/2015  01:55 PM   2617 ADResolveUsername.ps1
 -----     11/15/2015  01:55 PM   7998 ADSchema.ps1
 -----     11/15/2015  01:55 PM   3706 ADSearch.ps1
 -----     11/15/2015  01:55 PM   4827 ADSync.ps1
 -----     11/15/2015  01:55 PM   2075 ADTest.ps1
 -----     11/15/2015  01:55 PM  10044 ADUpdate.ps1

PS C:\path\to\openidm>
```

5. Copy the sample connector configuration for the PowerShell connector from the `samples\provisioners` directory to your project's `conf` directory.

```
PS C:\ cd \path\to\openidm
PS C:\ cp samples\provisioners\provisioner.openicf-adpowershell.json conf
```

   The following excerpt of the sample connector configuration shows the configuration properties:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
"configurationProperties" : {
  "AuthenticateScriptFileName" : "C:/openidm/samples/powershell2AD/tools/ADAuthenticate.ps1",
  "CreateScriptFileName" : "C:/openidm/samples/powershell2AD/tools/ADCreate.ps1",
  "DeleteScriptFileName" : "C:/openidm/samples/powershell2AD/tools/ADDelete.ps1",
  "ResolveUsernameScriptFileName" : "C:/openidm/samples/powershell2AD/tools/ADResolveUsername.ps1",
  "SchemaScriptFileName" : "C:/openidm/samples/powershell2AD/tools/ADSchema.ps1",
  "SearchScriptFileName" : "C:/openidm/samples/powershell2AD/tools/ADSearch.ps1",
  "SyncScriptFileName" : "C:/openidm/samples/powershell2AD/tools/ADSync.ps1",
  "TestScriptFileName" : "C:/openidm/samples/powershell2AD/tools/ADTest.ps1",
  "UpdateScriptFileName" : "C:/openidm/samples/powershell2AD/tools/ADUpdate.ps1",
  "VariablesPrefix" : "Connector",
  "QueryFilterType" : "AdPsModule",
  "ReloadScriptOnExecution" : true,
  "UseInterpretersPool" : true,
  "SubstituteUidAndNameInQueryFilter" : true,
  "UidAttributeName" : "ObjectGUID",
  "NameAttributeName" : "DistinguishedName",
  "PsModulesToImport" : [ "ActiveDirectory" ],
  "Host" : "",
  "Port" : null,
  "Login" : "",
  "Password" : null
},
```

The sample connector configuration assumes that the scripts are located in `C:/openidm/samples/powershell2AD/tools/`. If you copied your scripts to a different location, adjust your connector configuration file accordingly.

Note that the OpenICF framework requires the path to use forward slash characters and not the backslash characters that you would expect in a Windows path.

Add a `CustomProperties` section to your connector configuration file, and set your base context, for example:

```
"CustomProperties" : [
    "baseContext = CN=Users,DC=example,DC=com"
]
```

The PowerShell connector parses this string value to create a key=value pair. Because C# does not have built-in JSON support, this custom parser allows you to pass in configuration values beyond what is added to the static configuration.

The host, port, login and password of the machine on which Active Directory runs do not need to be specified here. By default the Active Directory cmdlets pick up the first available Domain Controller. In addition, the scripts are executed using the credentials of the .Net connector server.

Note

The `"ReloadScriptOnExecution"` property is set to `true` in this sample configuration. This setting causes script files to be reloaded each time the script is invoked. Having script files reloaded each time is suitable

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

for debugging purposes. However, this property should be set to `false` in production environments, as the script reloading can have a negative performance impact.

In addition, make sure that the value of the `"connectorHostRef"` property in the connector configuration file matches the value that you specified in the remote connector configuration file, in step 2 of this procedure. For example:

```
"connectorHostRef" : "dotnet",
```

5.1.2. Testing the PowerShell Active Directory Sample

Because you have copied all of the required configuration files into the default OpenIDM project, you can start OpenIDM with the default configuration (that is, without the `-p` option).

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

When OpenIDM has started, you can test the sample by using the `curl` command-line utility. The following examples test the scripts that were provided in the `tools` directory.

1. Test the connector configuration, and whether OpenIDM is able to connect to the .NET connector server with the following request.

```
PS C:\ curl `
 --header "X-OpenIDM-Username: openidm-admin" `
 --header "X-OpenIDM-Password: openidm-admin" `
 --request POST `
 "http://localhost:8080/openidm/system?_action=test"
[
  {
    "ok": true,
    "connectorRef": {
      "bundleVersion": "1.4.1.0",
      "bundleName": "MsPowerShell.Connector",
      "connectorName": "Org.ForgeRock.OpenICF.Connectors.MsPowerShell.MsPowerShellConnector"
    },
    "objectTypes": [
      "__ALL__",
      "group",
      "account"
    ],
    "config": "config/provisioner.openicf/adpowershell",
    "enabled": true,
    "name": "adpowershell"
  }
]
```

2. Query the users in your Active Directory with the following request:

```
PS C:\ curl `
 --header "X-OpenIDM-Username: openidm-admin" `
 --header "X-OpenIDM-Password: openidm-admin" `
 --request GET `
 "http://localhost:8080/openidm/system/adpowershell/account?_queryId=query-all-ids"
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
{
  "remainingPagedResults": -1,
  "pagedResultsCookie": null,
  "resultCount": 1257,
  "result": [
    {
      "_id": "7c41496a-9898-4074-a537-bed696b6be92",
      "distinguishedName": "CN=Administrator,CN=Users,DC=example,DC=com"
    },
    {
      "_id": "f2e08a5c-473f-4798-a2d5-d5cc27c862a9",
      "distinguishedName": "CN=Guest,CN=Users,DC=example,DC=com"
    },
    {
      "_id": "99de98a3-c125-48dd-a7c2-e21f1488ab06",
      "distinguishedName": "CN=Ben Travis,CN=Users,DC=example,DC=com"
    },
    {
      "_id": "0f7394cc-c66a-404f-ad6d-38dbb4b6526d",
      "distinguishedName": "CN=Barbara Jensen,CN=Users,DC=example,DC=com"
    },
    {
      "_id": "3e6fa858-ed3a-4b58-9325-1fca144eb7c7",
      "distinguishedName": "CN=John Doe,CN=Users,DC=example,DC=com"
    },
    {
      "_id": "6feef4a0-b121-43dc-be68-a96703a49aba",
      "distinguishedName": "CN=Steven Carter,CN=Users,DC=example,DC=com"
    }
,
...
```

3. To return the complete record of a specific user, include the ID of the user in the URL. The following request returns the record for Steven Carter.

```
PS C:\ curl `
 --header "X-OpenIDM-Username: openidm-admin" `
 --header "X-OpenIDM-Password: openidm-admin" `
 --request GET `
 "http://localhost:8080/openidm/system/adpowershell/account/6feef4a0-b121-43dc-be68-a96703a49aba"
{
  "_id": "6feef4a0-b121-43dc-be68-a96703a49aba",
  "postalCode": null,
  "passwordNotRequired": false,
  "cn": "Steven Carter",
  "name": "Steven Carter",
  "trustedForDelegation": false,
  "uSNChanged": "47219",
  "manager": null,
  "objectGUID": "6feef4a0-b121-43dc-be68-a96703a49aba",
  "modifyTimeStamp": "11/27/2014 3:37:16 PM",
  "employeeNumber": null,
  "sn": "Carter",
  "userAccountControl": 512,
  "passwordNeverExpires": false,
  "displayName": "Steven Carter",
  "initials": null,
  "pwdLastSet": "130615726366949784",
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
  "scriptPath": null,
  "badPasswordTime": "0",
  "employeeID": null,
  "badPwdCount": "0",
  "accountExpirationDate": null,
  "userPrincipalName": "steve.carter@ad0.example.com",
  "sAMAccountName": "steve.carter",
  "mail": "steven.carter@example.com",
  "logonCount": "0",
  "cannotChangePassword": false,
  "division": null,
  "streetAddress": null,
  "allowReversiblePasswordEncryption": false,
  "description": null,
  "whenChanged": "11/27/2014 3:37:16 PM",
  "title": null,
  "lastLogon": "0",
  "company": null,
  "homeDirectory": null,
  "whenCreated": "6/23/2014 2:50:48 PM",
  "givenName": "Steven",
  "telephoneNumber": "555-2518",
  "homeDrive": null,
  "uSNCreated": "20912",
  "smartcardLogonRequired": false,
  "distinguishedName": "CN=Steven Carter,CN=Users,DC=example,DC=com",
  "createTimeStamp": "6/23/2014 2:50:48 PM",
  "department": null,
  "memberOf": [
    "CN=employees,DC=example,DC=com"
  ],
  "homePhone": null
}
```

4. Test whether you can authenticate as one of the users in your Active Directory. The username that you specify here can be either an ObjectGUID, UPN, sAMAccountname or CN.

```
$ PS C:\ curl `
 --header "X-OpenIDM-Username: openidm-admin" `
 --header "X-OpenIDM-Password: openidm-admin" `
 --request POST `
 "http://localhost:8080/openidm/system/adpowershell/account?_action=authenticate&username=Steven
+Carter&password=Passw0rd"
{
  "_id": "6feef4a0-b121-43dc-be68-a96703a49aba"
}
```

The request returns the ObjectGUID if the authentication is successful.

5. You can return the complete record for a specific user, using the query filter syntax described in Section 7.3.4, "Constructing Queries" in the *Integrator's Guide*.

The following query returns the record for the guest user.

```
PS C:\ curl `
 --header "X-OpenIDM-Username: openidm-admin" `
 --header "X-OpenIDM-Password: openidm-admin" `
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
 --request GET `
 "http://localhost:8080/openidm/system/adpowershell/account?_queryFilter=cn+eq+guest"
{
  "remainingPagedResults": -1,
  "pagedResultsCookie": null,
  "resultCount": 1,
  "result": [
    {
      "_id": "f2e08a5c-473f-4798-a2d5-d5cc27c862a9",
      "postalCode": null,
      "passwordNotRequired": true,
      "cn": "Guest",
      "name": "Guest",
      "trustedForDelegation": false,
      "uSNChanged": "8197",
      "manager": null,
      "objectGUID": "f2e08a5c-473f-4798-a2d5-d5cc27c862a9",
      "modifyTimeStamp": "6/9/2014 12:35:16 PM",
      "employeeNumber": null,
      "userAccountControl": 66082,
      "whenChanged": "6/9/2014 12:35:16 PM",
      "initials": null,
      "pwdLastSet": "0",
      "scriptPath": null,
      "badPasswordTime": "0",
      "employeeID": null,
      "badPwdCount": "0",
      "accountExpirationDate": null,
      "sAMAccountName": "Guest",
      "logonCount": "0",
      "cannotChangePassword": true,
      "division": null,
      "streetAddress": null,
      "allowReversiblePasswordEncryption": false,
      "description": "Built-in account for guest access to the computer/domain",
      "userPrincipalName": null,
      "title": null,
      "lastLogon": "0",
      "company": null,
      "homeDirectory": null,
      "whenCreated": "6/9/2014 12:35:16 PM",
      "givenName": null,
      "homeDrive": null,
      "uSNCreated": "8197",
      "smartcardLogonRequired": false,
      "distinguishedName": "CN=Guest,CN=Users,DC=example,DC=com",
      "createTimeStamp": "6/9/2014 12:35:16 PM",
      "department": null,
      "memberOf": [
        "CN=Guests,CN=Builtin,DC=example,DC=com"
      ],
      "homePhone": null,
      "displayName": null,
      "passwordNeverExpires": true
    }
  ]
}
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

6. Test whether you are able to create a user on the Active Directory server by sending a POST request with the `create` action.

   The following request creates the user `Jane Doe` on the Active Directory server.

```
PS C:\ curl `
 --header "X-OpenIDM-Username: openidm-admin" `
 --header "X-OpenIDM-Password: openidm-admin" `
 --header "Content-Type: application/json" `
 --request POST `
 --data "{
  \"distinguishedName\" : \"CN=Jane Doe,CN=Users,DC=example,DC=com\",
  \"sn\" : \"Doe\",
  \"cn\" : \"Jane Doe\",
  \"sAMAccountName\" : \"sample\",
  \"userPrincipalName\" : \"janedoe@example.com\",
  \"__ENABLE__\" : true,
  \"__PASSWORD__\" : \"Passw0rd\",
  \"telephoneNumber\" : \"0052-611-091\"
 }" `
 "http://localhost:8080/openidm/system/adpowershell/account?_action=create"
{
  "_id": "42725210-8dce-4fdf-b0e0-393cf0377fdf",
  "title": null,
  "uSNCreated": "47244",
  "pwdLastSet": "130615892934093041",
  "cannotChangePassword": false,
  "telephoneNumber": "0052-611-091",
  "smartcardLogonRequired": false,
  "badPwdCount": "0",
  "department": null,
  "distinguishedName": "CN=Jane Doe,CN=Users,DC=example,DC=com",
  "badPasswordTime": "0",
  "employeeID": null,
  "cn": "Jane Doe",
  "division": null,
  "description": null,
  "userPrincipalName": "janedoe@example.com",
  "passwordNeverExpires": false,
  "company": null,
  "memberOf": [],
  "givenName": null,
  "streetAddress": null,
  "sn": "Doe",
  "initials": null,
  "logonCount": "0",
  "homeDirectory": null,
  "employeeNumber": null,
  "objectGUID": "42725210-8dce-4fdf-b0e0-393cf0377fdf",
  "manager": null,
  "lastLogon": "0",
  "trustedForDelegation": false,
  "scriptPath": null,
  "allowReversiblePasswordEncryption": false,
  "modifyTimeStamp": "11/27/2014 8:14:53 PM",
  "whenCreated": "11/27/2014 8:14:52 PM",
  "whenChanged": "11/27/2014 8:14:53 PM",
  "accountExpirationDate": null,
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
  "name": "Jane Doe",
  "displayName": null,
  "homeDrive": null,
  "passwordNotRequired": false,
  "createTimeStamp": "11/27/2014 8:14:52 PM",
  "uSNChanged": "47248",
  "sAMAccountName": "sample",
  "userAccountControl": 512,
  "homePhone": null,
  "postalCode": null
}
```

7. Test whether you are able to update a user object on the Active Directory server by sending a PUT request with the complete object, and including the user ID in the URL.

   The following request updates user `Jane Doe`'s entry, including her ID in the request. The update sends the same information that was sent in the `create` request, but adds an `employeeNumber`.

```
PS C:\ curl `
 --header "X-OpenIDM-Username: openidm-admin" `
 --header "X-OpenIDM-Password: openidm-admin" `
 --header "Content-Type: application/json" `
 --header "If-Match: *" `
 --request PUT `
 --data "{
  \"distinguishedName\" : \"CN=Jane Doe,CN=Users,DC=example,DC=com\",
  \"sn\" : \"Doe\",
  \"cn\" : \"Jane Doe\",
  \"sAMAccountName\" : \"sample\",
  \"userPrincipalName\" : \"janedoe@example.com\",
  \"__ENABLE__\" : true,
  \"__PASSWORD__\" : \"Passw0rd\",
  \"telephoneNumber\" : \"0052-611-091\",
  \"employeeNumber\": \"567893\"
 }" `
 "http://localhost:8080/openidm/system/adpowershell/account/42725210-8dce-4fdf-b0e0-393cf0377fdf"
{
  "_id": "42725210-8dce-4fdf-b0e0-393cf0377fdf",
  "title": null,
  "uSNCreated": "47244",
  "pwdLastSet": "130615906375709689",
  "cannotChangePassword": false,
  "telephoneNumber": "0052-611-091",
  "smartcardLogonRequired": false,
  "badPwdCount": "0",
  "department": null,
  "distinguishedName": "CN=Jane Doe,CN=Users,DC=example,DC=com",
  "badPasswordTime": "0",
  "employeeID": null,
  "cn": "Jane Doe",
  "division": null,
  "description": null,
  "userPrincipalName": "janedoe@example.com",
  "passwordNeverExpires": false,
  "company": null,
  "memberOf": [],
  "givenName": null,
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
  "streetAddress": null,
  "sn": "Doe",
  "initials": null,
  "logonCount": "0",
  "homeDirectory": null,
  "employeeNumber": "567893",
  "objectGUID": "42725210-8dce-4fdf-b0e0-393cf0377fdf",
  "manager": null,
  "lastLogon": "0",
  "trustedForDelegation": false,
  "scriptPath": null,
  "allowReversiblePasswordEncryption": false,
  "modifyTimeStamp": "11/27/2014 8:37:17 PM",
  "whenCreated": "11/27/2014 8:14:52 PM",
  "whenChanged": "11/27/2014 8:37:17 PM",
  "accountExpirationDate": null,
  "name": "Jane Doe",
  "displayName": null,
  "homeDrive": null,
  "passwordNotRequired": false,
  "createTimeStamp": "11/27/2014 8:14:52 PM",
  "uSNChanged": "47253",
  "sAMAccountName": "sample",
  "userAccountControl": 512,
  "homePhone": null,
  "postalCode": null
}
```

8. Test whether you are able to delete a user object on the Active Directory server by sending a DELETE request with the user ID in the URL.

   The following request deletes user `Jane Doe`'s entry.

```
PS C:\ curl `
 --header "X-OpenIDM-Username: openidm-admin" `
 --header "X-OpenIDM-Password: openidm-admin" `
 --request DELETE `
 "http://localhost:8080/openidm/system/adpowershell/account/42725210-8dce-4fdf-b0e0-393cf0377fdf"
```

   The response includes the complete user object that was deleted.

   You can you attempt to query the user object to confirm that it has been deleted.

```
PS C:\ curl `
 --header "X-OpenIDM-Username: openidm-admin" `
 --header "X-OpenIDM-Password: openidm-admin" `
 --request GET `
 "http://localhost:8080/openidm/system/adpowershell/account/42725210-8dce-4fdf-b0e0-393cf0377fdf"
{
  "message": "",
  "reason": "Not Found",
  "code": 404
}
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

5.2. Connect to Azure AD

This sample uses the Microsoft Azure Active Directory (Azure AD) PowerShell module. For more information about this module, see https://msdn.microsoft.com/en-us/library/jj151815.aspx.

The sample assumes that OpenIDM runs on a local UNIX/Linux machine and that the PowerShell Connector Toolkit (and the OpenICF .NET connector server) run on a remote Windows host with access to an instance of AzureAD. Adjust the command-line examples if your OpenIDM instance runs on Windows.

This sample demonstrates how you can synchronize managed object data such as users and groups with a Microsoft AzureAD deployment.

Note

This sample utilizes a connection between three systems: OpenIDM on UNIX/Linux, an OpenICF .NET connector server on Windows, and Azure AD "in the cloud". Internet connection times vary widely and performance is based on responses to external calls, including potential timeouts, if a command doesn't perform the first time, try again. OpenIDM's synchronization and reconciliation performance will be fully dependent on the performance of the managed resource.

5.2.1. Before You Start

Before you can run this sample, you need to meet several prerequisites. This section describes each of the prerequisites, and how to install or test them.

- You must have a Microsoft account, which gives you access to Microsoft Azure.

  You can set up a Microsoft account at https://signup.live.com/.

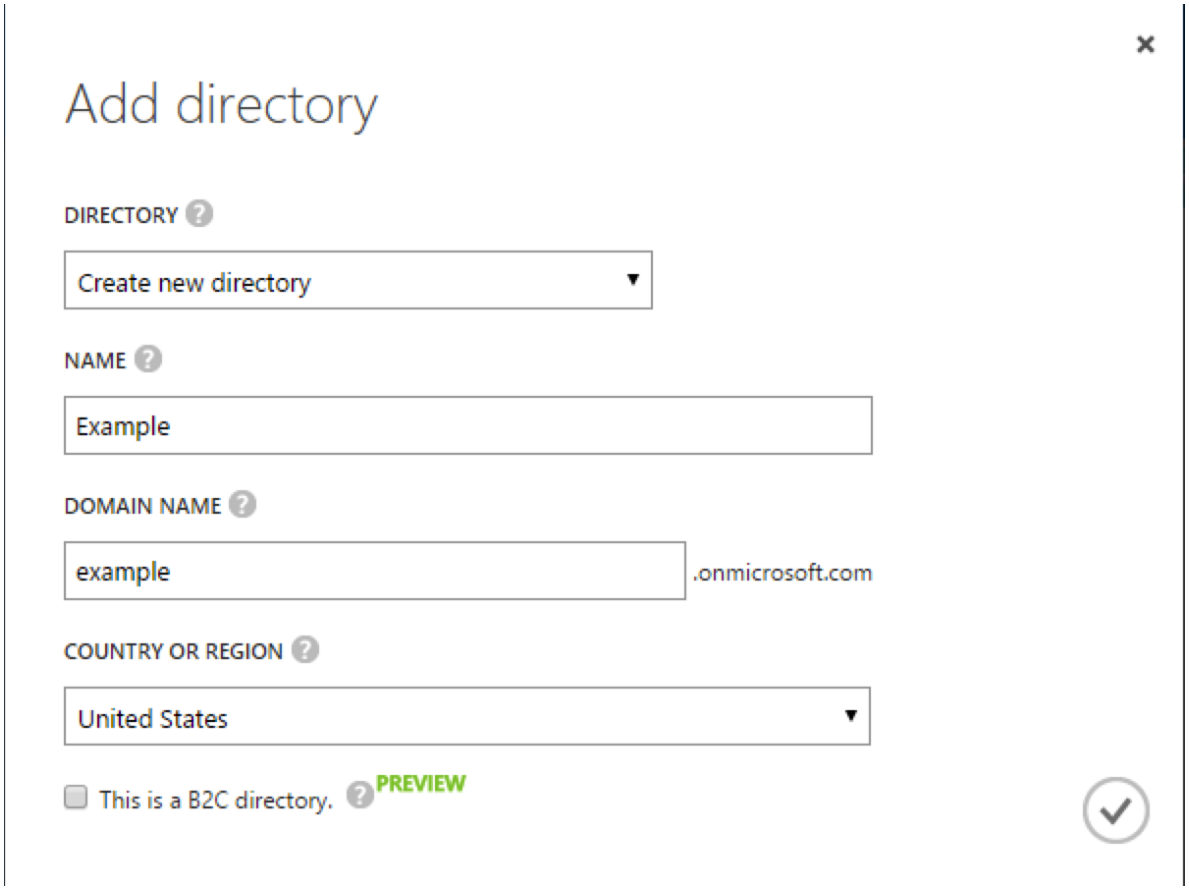
  With a Microsoft account, you can access the Azure portal at http://azure.microsoft.com.

- You must have an Azure AD cloud directory.

  If you do not have an existing Azure AD cloud, set one up as follows:

  1. Navigate to https://account.windowsazure.com/signup. Once you log in with your Microsoft credentials, fill in the prompts and Microsoft will create an Azure subscription.
  2. Navigate to http://portal.azure.com, log in with your Microsoft account.
  3. In the Microsoft Azure screen, select New on the left hand menu.
  4. From the New list, select Security + Identity > Active Directory.
  5. Complete the Add Directory form with the details of your directory, and select the check mark at the bottom of the form to submit.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.



Your directory should now be created and listed.

- Apart from your default Microsoft Azure account, you must have an *administrative user account* for your Azure AD.

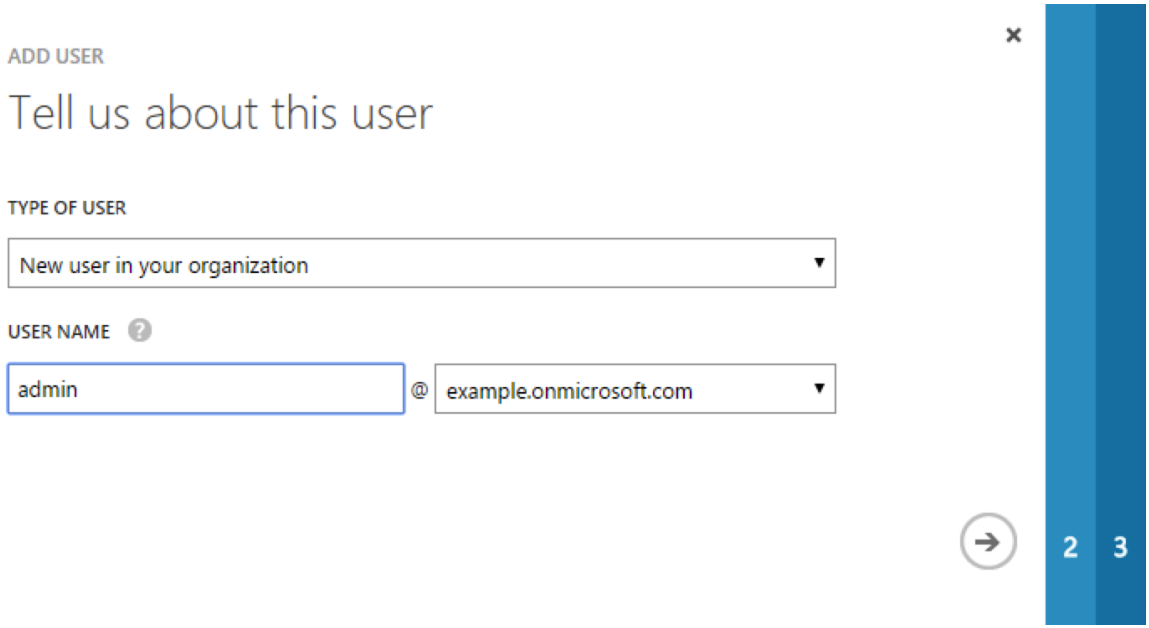
  By default your directory will have a single identity, your Microsoft Azure account. You cannot use this account to run the PowerShell Connector scripts that administer the Azure AD.

  If your Azure AD does not already include other administrative accounts, create a local administrative identity that is native to your directory as follows:

  1. Log in to https://portal.azure.com/ with your Microsoft Azure credentials.
  2. From the left-hand menu, select Browse > Active Directory.
  3. Select your cloud directory from the left-hand menu and select USERS in the top navigation bar.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

4. At the bottom of the page select Add User and enter the details of the new administrative user.



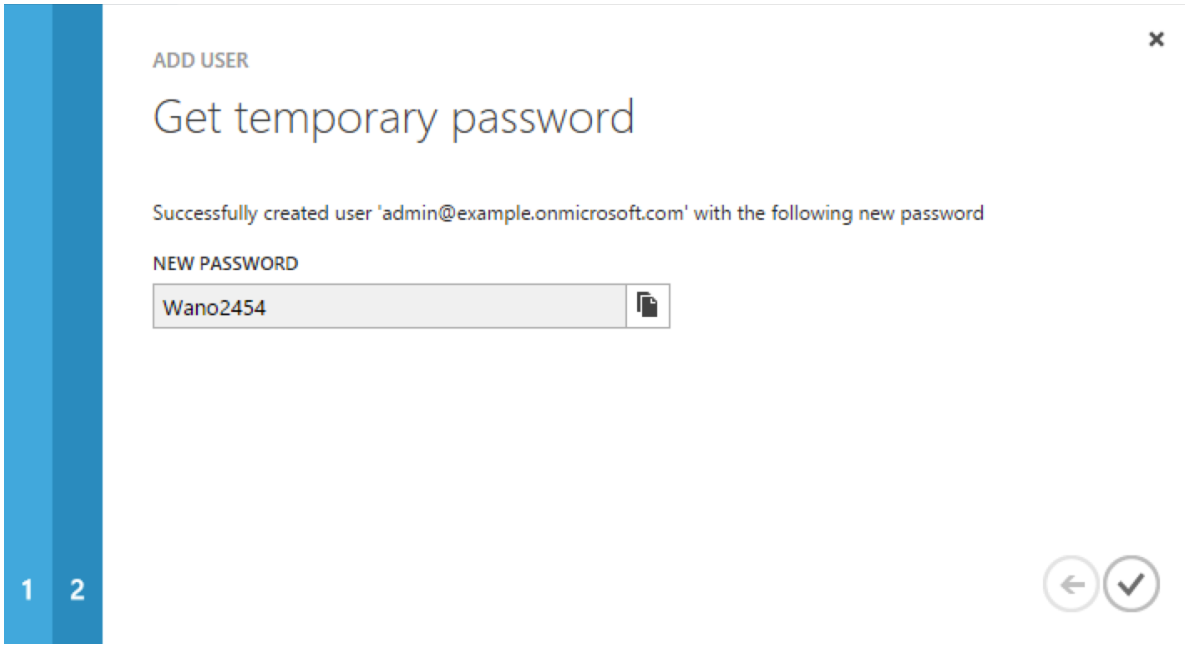
   Select the arrow to continue.

5. On the User Profile screen, enter the details of this administrative user. Make sure that the user's Role is at least User Admin.

   Select the arrow to continue.

6. On the final screen, select Create and note the temporary password that is assigned to the user.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.



Because new administrative users are forced to change their password on first login, you should log in as this user to change the password.

Select the check mark to complete the new user creation process.

7. Select the username at the top right of the screen and select Sign-out to sign out of your Microsoft Azure account, then select SIGN IN > Use Another Account to sign in as your new administrative user.

8. Enter the email address of the new administrative user and select Continue.

9. Enter the temporary password that you received and select Sign In.

10. On the Update Your Password screen, enter a new password, then select Update password and sign in.

    You now have a new administrative user account that the PowerShell scripts will use to access your Azure AD.

- The Windows Azure AD Module for Windows PowerShell must be installed on the Windows host that connects to Azure.

  If needed, install the Azure AD Module as described in the following Microsoft article.

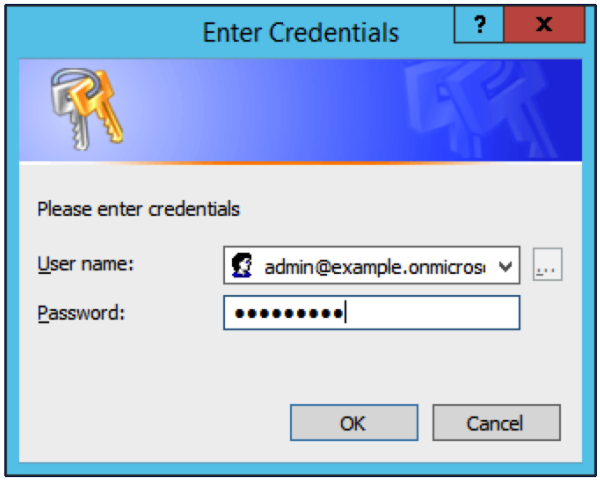
- Your Windows host must be able to contact your Azure AD deployment.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

Verify the connection as follows:

1. Open a PowerShell window and type `Connect-Msolservice` at the command prompt.

2. On the Enter Credentials screen, enter the credentials of the administrative account that you created for the Azure directory.



   If the PowerShell command returns with no error, you have successfully connected to your remote Azure AD deployment.

- The OpenICF .NET connector server must be installed on your Windows host.

  If you have not yet installed the .NET connector server, follow the instructions in Section 11.4.1, "Installing and Configuring a .NET Connector Server" in the *Integrator's Guide*. The connector server must be running in legacy mode (see Procedure 11.4, "Running the .NET Connector Server in Legacy Mode" in the *Integrator's Guide*.

- The PowerShell Connector Toolkit must be installed on your Windows host.

  If you have not yet installed the PowerShell Connector Toolkit, follow the instructions in Chapter 5, "*PowerShell Connector Toolkit*" in the *Connectors Guide*. In these instructions, you will use a command with a **/setkey** option to create a password key for your .NET connector server. You will use that key in Section 5.2.2, "Setting Up the PowerShell Azure AD Sample on OpenIDM".

Important

Before you continue, check that the OpenICF .NET connector server is still running. If it is not running, restart the connector server and check the logs. In some cases, Windows blocks the PowerShell connector dll. If the connector server fails to start, right-click on `MsPowerShell.Connector.dll` and select Properties > Security. If you see the following text on that tab:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
This file came from another computer and might be blocked to help protect
this computer.
```

Select the Unblock button to unblock the connector dll. Then restart the connector server.

When all of the above elements are in place, you can proceed with running the sample, as described in Section 5.2.2, "Setting Up the PowerShell Azure AD Sample on OpenIDM".

5.2.2. Setting Up the PowerShell Azure AD Sample on OpenIDM

This section assumes that OpenIDM is installed on the local UNIX/Linux machine.

1. On the Windows host, create a directory for the PowerShell scripts.

   The sample connector configuration expects the scripts in the directory `C:/openidm/samples/powershell2AzureAD/tools/`. If you put them in a different location, adjust your connector configuration accordingly.

   ```
   PS C:\> mkdir -Path openidm\samples\powershell2AzureAD\azureADScripts

       Directory: C:\openidm\samples\powershell2AzureAD

   Mode                LastWriteTime     Length Name
   ----                -------------     ------ ----
   d----          5/4/2016   11:26 AM           azureADScripts

   PS C:\>
   ```

2. Copy the PowerShell sample scripts from the OpenIDM instance on your UNIX/Linux host to the new directory on the remote Windows server.

   One way to do this is to run an `scp` client, such as `pscp` in your Windows terminal. The following command copies the PowerShell scripts from the OpenIDM installation to the Windows machine:

   ```
   PS C:\> cd openidm\samples\powershell2AzureAD\tools
   PS C:\> pscp -r username@openidm-host:path/to/openidm/samples/powershell2AzureAD/azureADScripts/*.ps .
   ```

   The following scripts should now be in the `azureADScripts` directory on your Windows system:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
PS C:\openidm\samples\powershell2AzureAD\azureADScripts> ls

    Directory: C:\openidm\samples\powershell2AzureAD\azureADScripts

Mode                LastWriteTime     Length Name
----                -------------     ------ ---
-
-a---         5/4/2016  11:26 AM       7258 AzureADCreate
.ps1
-a---         5/4/2016  11:26 AM       3208 AzureADDelete
.ps1
-a---         5/4/2016  11:26 AM       6952 AzureADSchema
.ps1
-a---         5/4/2016  11:26 AM       8149 AzureADSearch
.ps1
-a---         5/4/2016  11:26 AM       2465 AzureADTest
.ps1
-a---         5/4/2016  11:26 AM      10840 AzureADUpdate.ps1
```

Note

You need to set the execution policy, as Windows by default does not trust downloaded scripts. For more information, see the following article: Using the Set-ExecutionPolicy Cmdlet

You can then run the `Unblock-File` cmdlet to allow OpenIDM to run the scripts on your Windows system. For more information, see the following article: Unblock-File.

3. On the Linux/UNIX machine on which OpenIDM is installed, navigate to the `path/to/openidm/samples/powershell2AzureAD` directory, and open the `provisioner.openicf.connectorinfoprovider.json conf` file.

4. Edit the remote connector server configuration file to match the settings of the remote .NET connector server.

   Change the port to `8760`, and the password (`key`) that you configured for the .NET connector server.

   The following example assumes that the .NET connector server is running on the host `198.51.100.1`, listening on the default port, and configured with a secret key of `Passw0rd`:

```
{
   "remoteConnectorServers" :
      [
         {
            "name" : "dotnet",
            "host" : "198.51.100.1",
            "port" : 8760,
            "useSSL" : false,
            "timeout" : 0,
            "key" : "Passw0rd"
         }
      ]
}
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

5. Open the sample Azure AD PowerShell connector configuration file, `provisioner.openicf-azureadpowershell.json`, and edit it to match your deployment. In particular, set the following properties in that file:

```
"Host" : "198.51.100.1",
"Port" : 8760,
"Login" : "admin@example.onmicrosoft.com",
"Password" : "Passw0rd",
```

`Host`

The hostname or IP address on which the .NET connector server is running.

`Port`

The port on which the .NET connector server is listening (`8760` by default in legacy mode).

`Login`

The username of the administrative account you created for the Azure directory in the previous section.

`Password`

The password of the administrative account you created for the Azure directory in the previous section.

If you have placed the PowerShell scripts in a directory other than the default (`C:\openidm\samples\powershell2AzureAD\azureADScripts`) you must also update those paths in the PowerShell connector configuration file.

6. Start OpenIDM with the PowerShell AzureAD sample configuration:

```
$ cd path/to/openidm
$ ./startup.sh -p samples/powershell2AzureAD
```

5.2.3. Managing Users and Groups with the PowerShell Azure AD Sample

This section walks you through several REST commands that enable you to test the connector configuration, and perform basic CRUD operations in your Azure AD, through the PowerShell connector.

1. Test that the connector has been configured correctly and that the Azure AD resource can be reached:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request POST \
 "http://localhost:8080/openidm/system/azureadpowershell?_action=test"
{
  "name": "azureadpowershell",
  "enabled": true,
  "config": "config/provisioner.openicf/azureadpowershell",
  "objectTypes": [
    "__ALL__",
    "account",
    "group"
  ],
  "connectorRef": {
    "bundleName": "MsPowerShell.Connector",
    "connectorName": "Org.ForgeRock.OpenICF.Connectors.MsPowerShell.MsPowerShellConnector",
    "bundleVersion": "1.4.2.0"
  },
  "displayName": "PowerShell Connector ",
  "ok": true
}
```

If you see no response from this connector test, review any changes that you made to the `provisioner-openicf*` files in your project's `conf/` subdirectory. If you've made changes appropriate for your deployment, wait a couple of minutes and try again.

2. Query the IDs of the existing users in your Azure AD deployment:

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/system/azureadpowershell/account?_queryId=query-all-ids"
{
  "result": [ {
      "_id": "51560d42-e60e-49a8-855b-42b6eca35ca6",
      "UserPrincipalName": "admin@example.onmicrosoft.com"
    },
    {
      "_id": "5e63b42f-c93a-466f-af86-f0a8d00f2491",
      "UserPrincipalName": "scarter@example.onmicrosoft.com"
    } ]
,
...
}
```

3. Use a query filter to return all details of all existing users in your Azure AD:

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/system/azureadpowershell/account?_queryFilter=true"
{
  "result": [
    {
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
      "_id": "51560d42-e60e-49a8-855b-42b6eca35ca6",
      "LiveId": "10033FFF96C5186D",
      "FirstName": "Barbara",
      "LastName": "Jensen",
      "UserPrincipalName": "admin@example.onmicrosoft.com",
      "AlternateEmailAddress" : [ "bjensen@example.com" ],
      "LastPasswordChangeTimestamp": "3/15/2016 11:02:19 AM",
      "DisplayName": "Barbara Jensen",
      "PasswordNeverExpires": false,
      "MobilePhone" : "+1 3602297105"
    },
    {
      "_id": "5e63b42f-c93a-466f-af86-f0a8d00f2491",
      "LiveId": "1003BFFD96A4CFBA",
      "FirstName": "Sam",
      "LastName": "Carter",
      "UserPrincipalName": "scarter@example.onmicrosoft.com"
      "AlternateEmailAddresses": [ "scarter@example.com" ],
      "LastPasswordChangeTimestamp": "3/7/2016 1:09:31 PM",
      "DisplayName": "Sam Carter",
      "PasswordNeverExpires": false,
      "MobilePhone" : "+1 3602297105"
    }
  ]
,
...}
```

4. Return details for a specific user account, by its `_id`

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/system/azureadpowershell/account/51560d42-e60e-49a8-855b-42b6eca35ca6"
```

5. Create a new user in Azure AD. Substitute the domain for your Azure AD deployment for `example.onmicrosoft.com`:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request POST \
 --header "content-type: application/json" \
 --data '{
   "PasswordNeverExpires": false,
   "AlternateEmailAddresses": ["John.Bull@example.com"],
   "LastName": "Bull",
   "PreferredLanguage": "en-GB",
   "FirstName": "John",
   "UserPrincipalName": "Dev_John.Bull@example.onmicrosoft.com",
   "DisplayName": "John Bull"
 }' \
 "http://localhost:8080/openidm/system/azureadpowershell/account?_action=create"
{
  "_id" : "d4aac947-2037-4f29-b0f5-d404fd99938c",
  "LiveId" : "10037FFE979FB2C1",
  "FirstName" : "John",
  "LastName" : "Bull",
  "UserPrincipalName" : "Dev_John.Bull@example.onmicrosoft.com",
  "AlternateEmailAddresses" : [ "John.Bull@example.com" ],
  "LastPasswordChangeTimestamp" : "5/5/2016 3:52:43 PM",
  "DisplayName" : "John Bull",
  "PasswordNeverExpires" : false,
  "PreferredLanguage" : "en-GB"
}
```

Rerun the same command. You should see the following error:

```
{
  "code" : 500,
  "reason" : "Internal Server Error",
  "message" : "Operation CREATE failed with ConnectorException on system object:
      Dev_John.Bull@example.onmicrosoft.com"
}
```

6. Update the user entry that you have just created with a patch request. Include the `_id` of the new user in the URL. Save that `_id` value for a later step.

   The following example updates the user's display name:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --header "if-match: *" \
 --header "content-type: application/json" \
 --request PATCH \
 --data '[
    {
       "operation": "replace",
       "field": "DisplayName",
       "value": "John P. Bull"
    }
 ]' \
 "http://localhost:8080/openidm/system/azureadpowershell/account/d4aac947-2037-4f29-b0f5-d404fd99938c"
{
  "_id" : "d4aac947-2037-4f29-b0f5-d404fd99938c",
  "LiveId" : "10037FFE979FB2C1",
  "FirstName" : "John",
  "LastName" : "Bull",
  "UserPrincipalName" : "Dev_John.Bull@mikejangfr.onmicrosoft.com",
  "AlternateEmailAddresses" : [ "John.Bull@example.com" ],
  "LastPasswordChangeTimestamp" : "5/5/2016 3:52:43 PM",
  "DisplayName" : "John P. Bull",
  "PasswordNeverExpires" : false,
  "PreferredLanguage" : "en-GB"
}
```

7. Now create a group:

```
$ curl \
--header "X-OpenIDM-Username: openidm-admin"
 \
--header "X-OpenIDM-Password: openidm-admin"
 \
--header 'content-type: application/json'
 \
--request POST
 \
--data '{
      "DisplayName" : "Dev Testers group",
      "Description" : "Description of a Dev Group"
}' \
'http://localhost:8080/openidm/system/azureadpowershell/group?_action=create'
     {
  "_id" : "9091be74-f37e-408d-9198-2d2b5f4b4cdd",
  "Members" : [ ],
  "DisplayName" : "Dev Testers Group",
  "GroupType" : "Security",
  "Description" : "Description of a Dev Group",
  "objectId" : "9091be74-f37e-408d-9198-2d2b5f4b4cdd"
}
```

8. Add your recently created user to this new group. Use the `_id` of that user, as the `ObjectId`. Use the `_id` of the newly created group in the endpoint:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
--header "X-OpenIDM-Username: openidm-admin"
 \
--header "X-OpenIDM-Password: openidm-admin"
 \
--header "Content-Type: application/json"
 \
--header "If-Match: *"
 \
--request PUT
 \
--data '{
   "Members" : [
      {
         "ObjectId" : "d4aac947-2037-4f29-b0f5-d404fd99938c"
      }
   ]
}' \
"http://localhost:8080/openidm/system/azureadpowershell/group/9091be74-f37e-408d-9198-2d2b5f4b4cdd"
     {
   "_id" : "9091be74-f37e-408d-9198-2d2b5f4b4cdd",
   "Members" : [ {
      "ObjectId" : "d4aac947-2037-4f29-b0f5-d404fd99938c",
      "DisplayName" : "John P. Bull",
      "GroupMemberType" : "User",
      "EmailAddress" : "Dev_John.Bull@example.onmicrosoft.com"
   } ],
   "DisplayName" : "Testing Devs Group",
   "GroupType" : "Security",
   "Description" : "Description of a Dev Group",
   "objectId" : "9091be74-f37e-408d-9198-2d2b5f4b4cdd"
}
```

9. Confirm the result, by the `_id` of the group:

```
$ curl \
--header "X-OpenIDM-Username: openidm-admin"
 \
--header "X-OpenIDM-Password: openidm-admin"
 \
--request GET \
"http://localhost:8080/openidm/system/azureadpowershell/group/9091be74-f37e-408d-9198-2d2b5f4b4cdd"
```

10. Update a label for the group. Use the same group `_id`:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
--header "X-OpenIDM-Username: openidm-admin"
 \
--header "X-OpenIDM-Password: openidm-admin"
 \
--header "Content-Type: application/json"
 \
--header "If-Match: *"
 \
--request PUT
 \
--data '{
   "_id" : "9091be74-f37e-408d-9198-2d2b5f4b4cdd",
   "Description" : "Dev Masters Group",
   "Members" : [
      {
         "ObjectId" : "d4aac947-2037-4f29-b0f5-d404fd99938c",
         "DisplayName" : "John P. Bull",
         "GroupMemberType" : "User",
         "EmailAddress" : "Dev_John.Bull@example.onmicrosoft.com"
      }
   ],
   "DisplayName" : "Testing Devs Group",
   "GroupType" : "Security",
   "objectId" : "9091be74-f37e-408d-9198-2d2b5f4b4cdd"
}' \
"http://localhost:8080/openidm/system/azureadpowershell/group/9091be74-f37e-408d-9198-2d2b5f4b4cdd"
```

You should see the new `Description` in the output.

11. Remove the user from the new group. Use the same group `_id` Note how the `Members` in the `--data` block, and the output, are blank:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
--header "X-OpenIDM-Username: openidm-admin"
 \
--header "X-OpenIDM-Password: openidm-admin"
 \
--header "Content-Type: application/json"
 \
--header "If-Match: *"
 \
--request PUT
 \
--data '{
   "_id" : "9091be74-f37e-408d-9198-2d2b5f4b4cdd",
   "Description" : "Dev Masters Group",
   "Members" : [ ],
   "DisplayName" : "Testing Devs Group",
   "GroupType" : "Security",
   "objectId" : "9091be74-f37e-408d-9198-2d2b5f4b4cdd"
}' \
"http://localhost:8080/openidm/system/azureadpowershell/group/9091be74-f37e-408d-9198-2d2b5f4b4cdd"
     {
  "_id" : "9091be74-f37e-408d-9198-2d2b5f4b4cdd",
  "Members" : [ ],
  "DisplayName" : "Testing Devs Group",
  "GroupType" : "Security",
  "Description" : "Dev Masters Group",
  "objectId" : "9091be74-f37e-408d-9198-2d2b5f4b4cdd"
}
```

12. Delete the user that you created earlier:

```
$ curl \
--header "X-OpenIDM-Username: openidm-admin"
 \
--header "X-OpenIDM-Password: openidm-admin"
 \
--request DELETE \
"http://localhost:8080/openidm/system/azureadpowershell/account/d4aac947-2037-4f29-b0f5-d404fd99938c"
```

To verify that the user was deleted, run the REST call to `query-all-ids` shown earlier in this section. The ID associated with that user should have been removed.

5.2.4. Reconciling Users Between OpenIDM and Azure AD

In this section, you'll run commands that demonstrate reconciliation mappings between OpenIDM managed users and your remote instance of Azure AD.

In preparation, create a new user on the Azure AD system:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
--header "X-OpenIDM-Username: openidm-admin"
 \
--header "X-OpenIDM-Password: openidm-admin"
 \
--request POST
 \
--header "content-type: application/json"
 \
--data '{
    "UserPrincipalName": "CEO@example.onmicrosoft.com",
    "LastName": "Officer",
    "FirstName": "Chief",
    "DisplayName": "Chief Executive Officer",
    "PasswordNeverExpires": false
}' \
"http://localhost:8080/openidm/system/azureadpowershell/account?_action=create"
```

In the steps that follow, you'll run reconciliations to see what happens to that user in the OpenIDM data store.

1. Review the list of current managed users in the OpenIDM repository, filtered for the `userName` that starts with (`sw`) CEO:

   ```
   $ curl \
   --header "X-OpenIDM-Username: openidm-admin"
    \
   --header "X-OpenIDM-Password: openidm-admin"
    \
   --request GET \
   "http://localhost:8080/openidm/managed/user?_queryFilter=userName+sw+'CEO'"
   ```

   Until you reconcile the Azure AD repository to OpenIDM, the output should be empty:

   ```
   {
      "result" : [ ],
      "resultCount" : 0,
      "pagedResultsCookie" : null,
      "totalPagedResultsPolicy" : "NONE",
      "totalPagedResults" : -1,
      "remainingPagedResults" : -1
   }
   ```

2. Run a reconciliation from Azure AD to OpenIDM:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

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

     {
   "_id" : "71811f1c-2ec0-47ae-ba47-d62c7094201b-1105",
   "state" : "SUCCESS"
}
```

3. Now rerun the command to list of current managed users in the OpenIDM repository, filtered for the `userName` that starts with (`sw`) CEO:

```
$ curl \
--header "X-OpenIDM-Username: openidm-admin"
 \
--header "X-OpenIDM-Password: openidm-admin"
 \
--request GET \
"http://localhost:8080/openidm/managed/user?_queryFilter=userName+sw+'CEO'"
      {
   "result" : [ {
   "_id" : "3a012a60-19c2-4fb4-99cc-0bb82dc4588c",
   "_rev" : "1",
   "userName" : "CEO@example.onmicrosoft.com",
   "mail" : "CEO@example.onmicrosoft.com",
   "sn" : "Officer",
   "givenName" : "Chief",
   "accountStatus" : "active",
   "effectiveRoles" : [ ],
   "effectiveAssignments" : [ ]
   } ],
   "resultCount" : 1,
   "pagedResultsCookie" : null,
   "totalPagedResultsPolicy" : "NONE",
   "totalPagedResults" : -1,
   "remainingPagedResults" : -1
}
```

4. Delete that CEO user from the Azure AD system, by the `_id` shown earlier when you searched that system:

```
$ curl \
--header "X-OpenIDM-Username: openidm-admin"
 \
--header "X-OpenIDM-Password: openidm-admin"
 \
--request DELETE \
"http://localhost:8080/openidm/system/azureadpowershell/account/3a012a60-19c2-4fb4-99cc-0bb82dc4588c"
```

   If successful, you'll see the data for the CEO user one last time.

5. Run a second reconciliation from the remote Azure AD repository to OpenIDM:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

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

6. Rerun the command to search the OpenIDM repository for a `userName` that starts with 'CEO' one more time, to confirm that user has been reconciled out of the OpenIDM repository:

```
$ curl \
--header "X-OpenIDM-Username: openidm-admin"
 \
--header "X-OpenIDM-Password: openidm-admin"
 \
--request GET \
"http://localhost:8080/openidm/managed/user?_queryFilter=userName+sw+'CEO'"
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

Chapter 6

Audit Samples

This chapter demonstrates two ways to configure OpenIDM to save audit data. In Section 6.1, "Directing Audit Information To a MySQL Database", you'll see how to direct information for OpenIDM audit events to a MySQL database server. In Section 6.2, "Show Audit Events Published on a JMS Topic", you'll see how to configure an audit event handler for the Java Message Service (JMS).

6.1. Directing Audit Information To a MySQL Database

This sample demonstrates how you can use the audit features provided with OpenIDM and directs audit information to a MySQL database. The sample is closely related to Section 2.1, "First OpenIDM Sample - Reconciling an XML File Resource" but includes a ScriptedSQL implementation of the Groovy Connector Toolkit to connect to the MySQL database.

To use any of the audit features that are demonstrated in this sample, copy information from files in the `samples/audit-sample` directory. This can help you to set up auditing for any other sample.

6.1.1. Audit Sample Configuration Files

Review the configuration files used in this sample. They can help you understand the functionality of the data sets being audited.

The key configuration files, in the `samples/audit-sample` directory, are as follows:

- `conf/provisioner.openicf-scriptedsql.json` shows the configuration of the Scripted SQL implementation of the Groovy Connector. For more information, see Chapter 6, "*Groovy Connector Toolkit*" in the *Connectors Guide*.
- `conf/provisioner.openicf-xml.json` shows the configuration of the Chapter 14, "*XML File Connector*" in the *Connectors Guide*.
- `conf/audit.json` configures audit logging on the router, to a remote system, as discussed in Section 19.1, "Configuring the Audit Service" in the *Integrator's Guide*.
- `conf/sync.json` shows mappings between managed users and the data set attached through the XML File Connector.
- `data/sample_audit_db.mysql` includes a schema that supports tables in the external MySQL database.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

- Groovy scripts in the `tools/` subdirectory supports communications between the Scripted SQL connector and the MySQL database.

6.1.2. Configuration with MySQL

You need to set up communications between OpenIDM and an external MySQL database server.

Make sure MySQL is running. Follow the configuration instructions described in Section 2.2, "To Set Up OpenIDM With MySQL" in the *Installation Guide*.

The sample expects the following configuration for MySQL:

- The database is available on the local system.
- The database listens on the standard MySQL port, 3306.
- You can connect to the MySQL database over the network.
- OpenIDM should include the MySQL connector .jar file in the `/path/to/openidm/bundle` directory. If you need to download this file, see Section 2.2, "To Set Up OpenIDM With MySQL" in the *Installation Guide* for instructions.
- MySQL serves a database called `audit` with six tables: `auditaccess`, `auditactivity`, `auditauthentication`, `auditconfig`, `auditrecon`, and `auditsync`.
- For more information on the database schema, examine the following data definition language file: `openidm/samples/audit-sample/data/sample_audit_db.mysql`. Import the file into MySQL before running the sample.

```
$ mysql -u root -p < /path/to/openidm/samples/audit-sample/data/sample_audit_db.mysql

Enter password:
$
```

Now review the format of the audit database sample, created from the `sample_audit_db.mysql` file, at the MySQL prompt. To access that prompt, run the following command:

```
$ mysql -u root -p
mysql > use audit;
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
```

You can now review the current state of each noted audit log with the following commands:

```
select * from auditaccess;
select * from auditactivity;
select * from auditauthentication;
select * from auditconfig;
select * from auditrecon;
select * from auditsync;
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

Unless you enable scheduled reconciliation, you will not see audit reconciliation data until you run reconciliation manually.

6.1.3. Install the Sample

Prepare OpenIDM as described in Section 1.3, "Preparing OpenIDM", then start OpenIDM with the configuration for the audit sample.

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

6.1.4. 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&waitForCompletion=true"
```

You can also run this reconciliation from the Admin UI, at `https://localhost:8443/admin`.

Warning

The default password for the OpenIDM administrative user, `openidm-admin`, is `openidm-admin`.

To protect your deployment in production, change the default administrative password. To do so, navigate to the Self-Service UI at `https://localhost:8443/` and click Change Password.

You can now review the results in the MySQL database, from the MySQL prompt, using the commands described earlier.

If you have retained the default `"useForQueries" : true` option in the `conf/audit.json` file, you can also `GET` the same data with a REST call. For examples on how you can query audit logs, see Section 19.9, "Querying Audit Logs Over REST" in the *Integrator's Guide*.

6.1.5. Additional Audit Configuration

You can configure a variety of audit event handlers, event topics, audit filter policies, and scripts. You can even set up auditing, by topic, in CSV files. For more information, see Chapter 19, "*Using Audit Logs*" in the *Integrator's Guide*.

You can see how this works from the Admin UI. After you log in with the OpenIDM administrative account, click Configure > System Preferences > Audit.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

6.2. Show Audit Events Published on a JMS Topic

Starting with OpenIDM 4.5.0, you can configure a Java Message Service (JMS) Audit Event Handler. JMS is an API that supports Java-based peer-to-peer messages between clients. The JMS API can create, send, receive, and read messages, reliably and asynchronously. You can now set up a JMS audit event handler, which can publish messages that comply with the *Java(TM) Message Service Specification Final Release 1.1* .

Warning

JMS topics are not related to ForgeRock audit event topics. The ForgeRock implementation of JMS topics use the publish/subscribe messaging domain, and may direct messages to the JMS audit event handler. In contrast, ForgeRock audit event topics specify categories of events.

In this sample, we demonstrate the use of the JMS audit event handler. This sample is based on Section 2.1, "First OpenIDM Sample - Reconciling an XML File Resource". You will set up communications between OpenIDM and an external JMS Message Broker, as well as *Apache Active MQ* as the JMS provider and message broker.

6.2.1. Adding Required Bundles for the JMS Audit Event Handler

To test this sample, you'll download a total of five JAR files. The first four are OSGi Bundles:

- *The ActiveMQ Client*, JAR, which you can download from https://repository.apache.org/content/repositories/releases/org/apache/activemq/activemq-client/.
- The *bnd* JAR for working with OSGi bundles, which you can download from bnd-1.50.0.jar.
- The Apache Geronimo J2EE management bundle, `geronimo-j2ee-management_1.1_spec-1.0.1.jar`, which you can download from http://central.maven.org/maven2/org/apache/geronimo/specs/geronimo-j2ee-management_1.1_spec/1.0.1/.
- The *hawtbuf* Maven-based protocol buffer compiler JAR, which you can download from hawtbuf-1.11.jar.
- The ActiveMQ 5.13.2 binary, which you can download from http://activemq.apache.org/activemq-5132-release.html.

Note

The JMS audit event handler has been tested and documented with the noted versions of the JAR files that you've just downloaded.

Make sure at least the first two JAR files, for *the Active MQ Client* and *bnd*, are in the same directory. Navigate to that directory, and create an OSGi bundle with the following steps:

1. Create a BND file named `activemq.bnd` with the following contents:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
version=5.13.2
Export-Package: *;version=${version}
Bundle-Name: ActiveMQ :: Client
Bundle-SymbolicName: org.apache.activemq
Bundle-Version: ${version}
```

2. Run the following command to create the OSGi bundle archive file:

```
$ java \
-jar \
bnd-1.50.0.jar \
wrap \
-properties \
activemq.bnd \
activemq-client-5.13.2.jar
```

3. Rename the `activemq-client-5.13.2.bar` file that appears to `activemq-client-5.13.2-osgi.jar` and copy it to the `/path/to/openidm/bundle` directory.

Copy the other two bundle files, *Apache Geronimo* and *hawtbuf*, to the `/path/to/openidm/bundle` directory.

6.2.2. Starting the ActiveMQ Broker and OpenIDM

With the appropriate bundles in the `/path/to/openidm/bundles` directory, you're ready to start the ActiveMQ message broker, as well as OpenIDM with the JMS Audit Sample.

Navigate to the directory where you unpacked the ActiveMQ binary, possibly `/path/to/apache-activemq-5.13.0/`. If you need SSL protection for your audit data, edit the ActiveMQ configuration file, `activemq.xml`, in the `conf/` subdirectory. Find the code block associated with `<transportConnectors>`, and add the following line within that block:

```
<transportConnector name="ssl"
uri="ssl://0.0.0.0:61617?transport.enabledCipherSuites=
    SSL_RSA_WITH_RC4_128_SHA,SSL_DH_anon_WITH_3DES_EDE_CBC_SHA
&amp;maximumConnections=1000&amp;wireFormat.maxFrameSize=104857600&amp;transport.daemon=true"/>
```

To start the ActiveMQ broker, navigate to the directory where you unpacked the ActiveMQ binary, and run the following command:

```
$ bin/activemq start
INFO: Loading '/path/to/apache-activemq-5.13.0/bin/env'
INFO: Using java '/usr/bin/java'
INFO: Starting - inspect logfiles specified in logging.properties and log4j.properties to get details
INFO: pidfile created : '/path/to/apache-activemq-5.13.0/data/activemq.pid' (pid '22671')
```

Now start OpenIDM, with the sample in the `/path/to/openidm/samples/audit-jms-sample` directory:

```
$ cd /path/to/openidm
$ ./startup.sh -p samples/audit-jms-sample
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

Note

If you see the following error in the OpenIDM console, you may have forgotten to go through the steps shown in Section 6.2.1, "Adding Required Bundles for the JMS Audit Event Handler"; you also need to start the ActiveMQ broker.

```
SEVERE: Unable to create JmsAuditEventHandler 'jms': null
```

6.2.3. Configuring and Using a JMS Consumer Application

To take advantage of the Apache ActiveMQ event broker, the JMS audit sample includes a Java consumer in the following directory: `/path/to/openidm/samples/audit-jms-sample/consumer/`

Assuming you have Apache Maven installed on the local system, you can compile that sample consumer with the following commands:

```
$ cd /path/to/openidm/samples/audit-jms-sample/consumer/
$ mvn clean install
```

When the build process is complete, you'll see a `BUILD SUCCESS` message:

```
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 12.638 s
[INFO] Finished at: 2016-04-15T15:18:31-07:00
[INFO] Final Memory: 13M/119M
[INFO] ------------------------------------------------------------------------
```

Note

You may see `[WARNING]` messages during the build. As long as the messages end with `BUILD SUCCESS`, you can proceed with the JMS consumer application.

You can then run the following command to output audit messages related to OpenIDM actions:

```
$ mvn \
exec:java \
-Dexec.mainClass="SimpleConsumer"
 \
-Dexec.args="tcp://localhost:61616"
   [INFO] ------------------------------------------------------------------------
[INFO] Building SimpleConsumer 1.0-SNAPSHOT
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] --- exec-maven-plugin:1.4.0:java (default-cli) @ SimpleConsumer ---
Connection factory=org.apache.activemq.ActiveMQConnectionFactory
READY, listening for messages. (Press 'Enter' to exit)
```

If you've configured ActiveMQ on a secure port, as described in Section 6.2.2, "Starting the ActiveMQ Broker and OpenIDM", you can run this alternative command:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ mvn \
exec:java \
-Dexec.mainClass="SimpleConsumer"
 \
-Dexec.args="ssl://localhost:61617?daemon=true&socket.enabledCipherSuites=
    SSL_RSA_WITH_RC4_128_SHA,SSL_DH_anon_WITH_3DES_EDE_CBC_SHA"
```

Try some actions on OpenIDM, either in a different console or in the Admin UI. Watch the output in the `SimpleConsumer` console. As an example, you might see output similar to the following when you are Section 2.1.4, "Running Reconciliation" on the data in this sample:

```
{
  "event": {
    "_id": "88b3da4d-e427-4f21-881c-036d7a854ccc-2559",
    "reconId": "88b3da4d-e427-4f21-881c-036d7a854ccc-2546",
    "mapping": "systemXmlfileAccounts_managedUser",
    "linkQualifier": "default",
    "exception": null,
    "action": "UPDATE",
    "userId": "openidm-admin",
    "eventName": "recon",
    "timestamp": "2016-04-16T13:40:35.974Z",
    "transactionId": "88b3da4d-e427-4f21-881c-036d7a854ccc-2546",
    "message": null,
    "situation": "CONFIRMED",
    "sourceObjectId": "system/xmlfile/account/scarter",
    "status": "SUCCESS",
    "targetObjectId": "managed/user/scarter",
    "reconciling": "source",
    "ambiguousTargetObjectIds": "",
    "entryType": "entry"
  },
  "auditTopic": "recon"
}
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

Chapter 7

Roles Samples - Demonstrating the OpenIDM Roles Implementation

This chapter illustrates how roles are managed in OpenIDM, and how they can be used to provision to external systems, based on certain criteria.

OpenIDM supports two types of roles:

- *Provisioning roles* - used to specify how objects are provisioned to an external system.
- *Authorization roles* - used to specify the authorization rights of a managed object internally, within OpenIDM.

Both provisioning roles and authorization roles use the relationships mechanism to link the role object, and the managed object to which the role applies. For more information about relationships between objects, see Section 8.5, "Managing Relationships Between Objects" in the *Integrator's Guide*.

There are three samples in this chapter:

- The *provisioning* sample (in `openidm/samples/roles/provrole`) demonstrates the operations that can be performed over the REST interface, or by using the Admin UI, to manage roles in OpenIDM.
- The *crudops* sample (in `openidm/samples/roles/crudops`) demonstrates how attributes are provisioned to an external system (an LDAP directory), based on role membership.
- The *temporalConstraints* sample (in `openidm/samples/roles/temporalConstraints`) builds on the other two roles samples and demonstrates how to apply time-based restrictions to role grants.

The separate samples are described in the sections that follow.

Important

Most of the commands in this sample can be run by using the command-line but it is generally much easier to use the Admin UI. In some cases, the command-line version makes it easier to explain what is happening

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

in OpenIDM. Therefore, in all steps, the sample first shows the command-line version, and then provides the equivalent method of running the command in the Admin UI.

7.1. CRUDOPS Role Sample - Working With Managed Roles

This sample demonstrates how to manage roles by using the REST interface, or the Admin UI. The sample covers the tasks discussed in the following sections:

- Section 7.1.2, "Creating a Managed Role"
- Section 7.1.3, "Reading and Searching Managed Roles"
- Section 7.1.4, "Granting a Managed Role to a User"
- Section 7.1.5, "Removing a Managed User's Roles"
- Section 7.1.6, "Deleting a Managed Role"

7.1.1. Before You Start

This sample does not include its own configuration. Before you work through the sample, start OpenIDM with the default configuration, as follows:

```
$ cd /path/to/openidm
$ ./startup.sh
Executing ./startup.sh...
Using OPENIDM_HOME:   /path/to/openidm
Using PROJECT_HOME:   /path/to/openidm
Using OPENIDM_OPTS:   -Xmx1024m -Xms1024m
Using LOGGING_CONFIG: -Djava.util.logging.config.file=/path/to/openidm/conf/logging.properties
Using boot properties at /path/to/openidm/conf/boot/boot
.properties
-> OpenIDM ready
```

7.1.2. Creating a Managed Role

In this section, you will create two managed roles - an *Employee* role and a *Contractor* role. The sample shows how to create the first role directly over the REST interface and the second by using the Admin UI. Choose whichever method is easiest for you to create both roles.

To create the Employee role by using the REST interface:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
 --header "Content-type: application/json" \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request POST \
 --data '{
   "name" : "Employee",
   "description": "Role granted to workers on the payroll."
 }' \
 "http://localhost:8080/openidm/managed/role?_action=create"
{
  "_id": "ad19979e-adbb-4d35-8320-6db50646b432",
  "_rev": "1",
  "name": "Employee",
  "description": "Role granted to workers on the payroll."
}
```

Take note of the role's identifier (ad19979e-adbb-4d35-8320-6db50646b432 in this example). Although you can create roles with a PUT request and specify your own ID, you should use system-generated IDs in a production environment to avoid conflicts and referential integrity issues.

To create the Contractor role by using the Admin UI:

1. Point your browser to the Admin UI URL (for example `https://localhost:8443/admin/`).
2. On the Dashboard, click Manage Role > New Role.
3. Enter a name for the role (Contractor) and a description.
4. Click Save.

   The managed role has been created, but currently has no assignments. These will be added in the second roles sample.

   Note

   The Admin UI creates managed objects with an unmutable, system-generated identifier. You will see this in the next section, in which you read or query the role object.

7.1.3. Reading and Searching Managed Roles

This section shows how to read, and search managed role objects.

The easiest way to see a list of managed role objects is by using the Admin UI:

1. Navigate to the Admin UI URL.
2. On the Dashboard, click Manage Role.

   The list of roles (currently only Contractor and Employee) is displayed.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

If you know the identifier, it is easy to read a managed role object over the REST interface. The following command reads the Employee role, that you created in the previous section:

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/managed/role/ad19979e-adbb-4d35-8320-6db50646b432"
{
  "_id": "ad19979e-adbb-4d35-8320-6db50646b432",
  "_rev": "1",
  "name": "Employee",
  "description": "Role granted to workers on the payroll."
}
```

You can also *search* for the role object, with a filtered query, if you know the role name. For more information about filtered queries, see Section 7.3.1, "Common Filter Expressions" in the *Integrator's Guide*.

The following query retrieves all roles with a name equal to "Contractor". The `_prettyPrint=true` parameter displays the result in a format that is easy to read:

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/managed/role?_queryFilter=/name+eq+'Contractor'&_prettyPrint=true"

{
  "result": [
    {
      "_id": "b02d2531-5066-415e-bc90-31fe57e02322",
      "_rev": "1",
      "name": "Contractor",
      "description": "Role granted to contract workers."
    }
  ],
 ...
}
```

To retrieve a list of all the managed roles that have been defined, query the `managed/role` endpoint, as follows:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/managed/role?_queryFilter=true&_prettyPrint=true"

{
  "result" : [ {
    "_id" : "ad19979e-adbb-4d35-8320-6db50646b432",
    "_rev" : "1",
    "name" : "Employee",
    "description" : "Role granted to workers on the payroll."
  }, {
    "_id" : "b02d2531-5066-415e-bc90-31fe57e02322",
    "_rev" : "1",
    "name" : "Contractor",
    "description" : "Role granted to contract workers."
  } ],
  ...
}
```

7.1.4. Granting a Managed Role to a User

For a role to be useful, it must be granted to a managed user. This section creates a new managed user entry, Felicitas Doe, then assigns the Employee role, created in the previous section, to Felicitas's user entry.

1. Create the user entry over REST, as follows:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
 --header "Content-type: application/json" \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request POST \
 --data '{
   "mail":"fdoe@example.com",
   "sn":"Doe",
   "telephoneNumber":"555-1234",
   "userName":"fdoe",
   "givenName":"Felicitas",
   "description":"Felicitas Doe",
   "displayName":"fdoe"
   }' \
 "http://localhost:8080/openidm/managed/user?_action=create"
{
  "_id": "837085ae-766e-417c-9b7e-c36eee4352a3",
  "_rev": "1",
  "mail": "fdoe@example.com",
  "sn": "Doe",
  "telephoneNumber": "555-1234",
  "userName": "fdoe",
  "givenName": "Felicitas",
  "description": "Felicitas Doe",
  "displayName": "fdoe",
  "accountStatus": "active",
  "effectiveRoles": [],
  "effectiveAssignments": []
}
```

Note that Felicitas has no `effectiveRoles` or `effectiveAssignments` by default.

Tip

Create the user entry in the Admin UI:

- Click Manage User > New User from the Dashboard.

2. Grant the Employee role to Felicitas's entry by sending a PATCH request to update her entry, and providing a pointer to the role ID:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
 --header "Content-type: application/json" \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request PATCH \
 --data '[
   {
     "operation" : "add",
     "field" : "/roles/-",
     "value" : {"_ref": "managed/role/ad19979e-adbb-4d35-8320-6db50646b432"}
   }
 ]' \
 "http://localhost:8080/openidm/managed/user/837085ae-766e-417c-9b7e-c36eee4352a3"
{
  "_id": "837085ae-766e-417c-9b7e-c36eee4352a3",
  "_rev": "2",
  "mail": "fdoe@example.com",
  "sn": "Doe",
  "telephoneNumber": "555-1234",
  "userName": "fdoe",
  "givenName": "Felicitas",
  "description": "Felicitas Doe",
  "displayName": "fdoe",
  "accountStatus": "active",
  "effectiveRoles": [
    {
      "_ref": "managed/role/ad19979e-adbb-4d35-8320-6db50646b432"
    }
  ],
  "effectiveAssignments": []
}
```

Tip

Grant the role in the Admin UI:

1. Select Manage > User and click on fdoe's entry.
2. Click Provisioning Roles, select the Employee role and click Add Role.

OR

1. Select Manage > Role and click on the Employee Role.
2. Select the Role Members tab, click Add Role Members, browse for fdoe's entry, and click Add.

3. Now, query Felicitas's entry to return her `roles` and *effectiveRoles*.

   The `effectiveRoles` property is a virtual property whose value is calculated based on the roles that have been granted to a user, either manually as in this example, or dynamically, through a script or condition. For more information about dynamically granted roles, see Section 8.4.3.2, "Granting Roles Dynamically" in the *Integrator's Guide*.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/managed/user?_queryFilter=/givenName+eq+'Felicitas'&_fields=_id,userName
,roles,effectiveRoles"
{
  "result" : [ {
    "_id" : "837085ae-766e-417c-9b7e-c36eee4352a3",
    "_rev" : "2",
    "userName" : "fdoe",
    "roles" : [ {
      "_ref": "managed/role/ad19979e-adbb-4d35-8320-6db50646b432",
      "_refProperties": {
        "_id": "4a42cd0b-d5d0-47e9-81e7-513aed74f6bc",
        "_rev": "1"
      } ],
    "effectiveRoles" : [ {
      "_ref" : "managed/role/ad19979e-adbb-4d35-8320-6db50646b432"
    } ]
  } ]
,
...
}
```

Note that Felicitas's `roles` and `effectiveRoles` attributes both show the reference to the `Employee` role ID.

7.1.5. Removing a Managed User's Roles

Imagine that the Employee role was erroneously granted to Felicitas, and must be removed from her user entry.

To remove a granted role from a managed user, send a DELETE request to the user entry, specifying the ID of the *relationship* that must be removed. Note that this is not the ID of the role.

The request to remove the Employee role from Felicitas's entry is as follows:

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request DELETE \
 "http://localhost:8080/openidm/managed/user/837085ae-766e-417c-9b7e-c36eee4352a3/roles/4a42cd0b-d5d0-47e9
-81e7-513aed74f6bc"
{
  "_ref": "managed/role/ad19979e-adbb-4d35-8320-6db50646b432",
  "_refProperties": {
    "_id": "4a42cd0b-d5d0-47e9-81e7-513aed74f6bc",
    "_rev": "1"
  }
}
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

If you query Felicitas's entry again, you will notice that her `roles` and `effectiveRoles` properties are now empty:

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/managed/user?_queryFilter=/givenName+eq+'Felicitas'&_fields=_id,userName
,roles,effectiveRoles"
{
  "result" : [ {
    "_id" : "837085ae-766e-417c-9b7e-c36eee4352a3",
    "_rev" : "2",
    "userName" : "fdoe",
    "roles": [],
    "effectiveRoles": []
  } ]
,
...
}
```

There are other methods to remove a granted role from a user over the REST interface. These are described in Section 8.4.6, "Deleting a User's Roles" in the *Integrator's Guide*.

Tip

You can also remove the granted role from fdoe's entry in the Admin UI as follows:

1. Select Manage > User and click on fdoe's entry.
2. On the Provisioning Roles tab, click the checkbox next to the Employee role and click Remove Selected Provisioning Roles.

7.1.6. Deleting a Managed Role

The final step in basic role management is to remove an existing role. In this section, we will remove the Contractor role we created previously.

Note that it is not possible to remove a role that is already granted to a user. To demonstrate this, we temporarily grant the Contractor role to Felicitas.

1. Assign the Contractor role to Felicitas, either by using the Admin UI, as described previously, or over the REST interface. If you use the REST interface, you will need to remember the system-generated identifier that was output when you queried the roles.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
 --header "Content-type: application/json" \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request PATCH \
 --data '[
   {
     "operation" : "add",
     "field" : "/roles/-",
     "value" : {"_ref": "managed/role/b02d2531-5066-415e-bc90-31fe57e02322"}
   }
 ]' \
 "http://localhost:8080/openidm/managed/user/837085ae-766e-417c-9b7e-c36eee4352a3"
{
  "_id": "837085ae-766e-417c-9b7e-c36eee4352a3",
  "_rev": "4",
  "mail": "fdoe@example.com",
  "sn": "Doe",
  "telephoneNumber": "555-1234",
  "userName": "fdoe",
  "givenName": "Felicitas",
  "description": "Felicitas Doe",
  "displayName": "fdoe",
  "accountStatus": "active",
  "effectiveRoles": [
    {
      "_ref": "managed/role/b02d2531-5066-415e-bc90-31fe57e02322"
    }
  ],
  "effectiveAssignments": []
}
```

2. Now, try to delete the Contractor role:

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request DELETE \
 "http://localhost:8080/openidm/managed/role/b02d2531-5066-415e-bc90-31fe57e02322"
{
  "code": 409,
  "reason": "Conflict",
  "message": "Cannot delete a role that is currently granted"
}
```

   As you can see from the previous output, it is not possible to delete a role that is still granted to a user.

3. Remove the role from Felicitas's entry, either by using the Admin UI (as described previously), or by using a DELETE request, specifying the ID of the *relationship* that must be removed. Note that this is not the ID of the role. You will need to read Felicitas's `roles` property to obtain the relationship ID:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/managed/user?_queryFilter=/givenName+eq+'Felicitas'&_fields=_id,userName
,roles,effectiveRoles"
{
  "result" : [ {
    "_id" : "837085ae-766e-417c-9b7e-c36eee4352a3",
    "_rev" : "3",
    "userName" : "fdoe",
    "roles" : [ {
      "_ref": "managed/role/b02d2531-5066-415e-bc90-31fe57e02322",
      "_refProperties": {
        "_id": "93703eff-a7ef-4cf3-80b1-f86fa3607978",
        "_rev": "1"
      } ],
    "effectiveRoles" : [ {
      "_ref" : "managed/role/b02d2531-5066-415e-bc90-31fe57e02322"
    } ]
  } ]
,
...
}
```

In this example, the relationship ID is 93703eff-a7ef-4cf3-80b1-f86fa3607978.

The following request removes the Contractor role from Felicitas's entry:

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request DELETE \
 "http://localhost:8080/openidm/managed/user/837085ae-766e-417c-9b7e-c36eee4352a3/roles/93703eff-a7ef-4cf3
-80b1-f86fa3607978"
{
  "_ref": "managed/role/b02d2531-5066-415e-bc90-31fe57e02322",
  "_refProperties": {
    "_id": "93703eff-a7ef-4cf3-80b1-f86fa3607978",
    "_rev": "1"
  }
}
```

4. Now that no users are granted the Contractor role, you can delete the role as follows:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request DELETE \
 "http://localhost:8080/openidm/managed/role/b02d2531-5066-415e-bc90-31fe57e02322"
{
  "_id": "b02d2531-5066-415e-bc90-31fe57e02322",
  "_rev": "1",
  "name": "Contractor",
  "description": "Role granted to contract workers."
}
```

The DELETE operation returns the complete role entry.

Tip

Delete the Contractor role by using the Admin UI:

1. Select Manage > Role.
2. Select the Contractor role and click Delete Selected.

5. Verify that the Contractor role has been deleted by querying the list of managed role objects:

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/managed/role?_queryFilter=true&_prettyPrint=true"
{
  "result": [
    {
      "_id": "ad19979e-adbb-4d35-8320-6db50646b432",
      "_rev": "1",
      "name": "Employee",
      "description": "Role granted to workers on the payroll."
    }
  ],
  ...
}
```

Note that only the Employee role remains.

Tip

List the remaining role objects in the Admin UI by clicking Manage > Role.

This concludes the basic role management operations. In the next section, you will see how to add assignments to roles, and how those assignments are used to provision users to external systems.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

7.2. Provisioning Role Sample - Provisioning to an LDAP Server

The main purpose of OpenIDM roles is to provision a set of attributes, based on a managed user's role membership.

This sample builds on what you learnt in the previous sample, and you will create the same Employee and Contractor roles that were described in that sample. This sample also builds on Sample 2b (described in Section 3.2, "Sample 2b - LDAP Two Way"), and provisions users from the managed user repository to an OpenDJ directory.

The sample assumes a company, example.com. As an *Employee* of example.com, a user should be added to two groups in OpenDJ - the Employees group and the Chat Users group (presumably to access certain internal applications). As a *Contractor*, a user should be added only to the Contractors group in OpenDJ. A user's employee type must also be set correctly in OpenDJ, based on the role that is granted to the user.

7.2.1. External LDAP Configuration

Configure OpenDJ as for sample 2 (see Section 3.1.1, "LDAP Server Configuration"). The LDAP user must have write access to create users from OpenIDM on the LDAP server. When you set up the LDAP server, import the LDIF file for this sample (`openidm/samples/roles/provrole/data/Example.ldif).`

7.2.2. Before You Start

This section sets up the scenario by performing the following tasks:

1. Start OpenIDM with the configuration for the provisioning roles sample.
2. Create the Employee and Contractor roles that you created in the previous sample.
3. Reconcile the managed user repository with the user entries in the LDAP server.

1. Prepare OpenIDM as described in Section 1.3, "Preparing OpenIDM", then start OpenIDM with the configuration for the Provisioning sample.

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

```
$ startup.sh -p samples/roles/provrole
```

2. Create the Employee and Contractor roles, either by using the Admin UI (as described in the previous sample), or by running the following commands:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
 --header "Content-type: application/json" \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request POST \
 --data '{
   "name" : "Employee",
   "description": "Role granted to workers on the payroll."
 }' \
 "http://localhost:8080/openidm/managed/role?_action=create"
{
  "_id" : "2902afd5-155a-49c0-9dd9-7e6bfcf1708a",
  "_rev" : "1",
  "name" : "Employee",
  "description" : "Role granted to workers on the payroll."
}
```

```
$ curl \
 --header "Content-type: application/json" \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request POST \
 --data '{
   "name" : "Contractor",
   "description": "Role granted to contract workers."
 }' \
 "http://localhost:8080/openidm/managed/role?_action=create"
{
  "_id": "e7f649ad-8013-4673-a52a-bdcac7483111",
  "_rev": "1",
  "name": "Contractor",
  "description": "Role granted to contract workers."
}
```

Note the IDs of these two roles because you will use them in the commands that follow.

3. Reconcile the repository.

   The `sync.json` configuration file for this sample includes two mappings: `systemLdapAccounts_managedUser`, which synchronizes users from the source LDAP server with the target OpenIDM repository; and `managedUser_systemLdapAccounts`, which synchronizes changes from the OpenIDM repository with the LDAP server.

   Run a reconciliation operation for the first mapping, either by using the Admin UI, or over the REST interface:

   - To use the Admin UI, select Configure > Mapping, click on the first mapping (System/Ldap/Account --> Managed User) and click Reconcile Now.

   - To use the REST interface, run the following command:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ 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&waitForCompletion=true"
{
  "_id": "b5c535f8-5c1f-44dc-afa3-40d4f9984925-24",
  "state": "SUCCESS"
}
```

The sample is now ready to demonstrate provisioning roles.

7.2.3. Run the Sample

This section assumes that you have reconciled the managed user repository to populate it with the users from the LDAP server, and that you have created the Employee and Contractor roles.

This part of the sample demonstrates the following features of the OpenIDM roles implementation:

- Section 7.2.3.1, "Adding Assignments to a Role Definition"
- Section 7.2.3.2, "Granting a Role to a User and Observing that User's Role Assignments"
- Section 7.2.3.3, "Propagating Assignments to an External System"
- Section 7.2.3.4, "Removing a Role Grant From a User and Observing That User's Role Assignments"

7.2.3.1. Adding Assignments to a Role Definition

A role *assignment* is the logic that provisions a managed user to an external system, based on some criteria. The most common use case of a role assignment is the provisioning of specific attributes to an external system, based on the role or roles that the managed user has been granted. Assignments are sometimes called *entitlements*. For more information about assignments, see Section 8.4.8, "Working With Role Assignments" in the *Integrator's Guide*.

In this section, you will create assignments and add them to the two roles that you created previously. This section assumes the following scenario:

example.com's policy requires that every employee has the correct value for their `employeeType` in their corporate directory (OpenDJ).

1. Display the roles that you created in the previous section:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/managed/role?_queryFilter=true"
{
  "result" : [
   {
      "_id" : "2902afd5-155a-49c0-9dd9-7e6bfcf1708a",
      "_rev" : "1",
      "name" : "Employee",
      "description" : "Role granted to workers on the payroll."
   },
   {
     "_id" : "e7f649ad-8013-4673-a52a-bdcac7483111",
     "_rev" : "1",
     "name" : "Contractor",
     "description" : "Role granted to contract workers."
   }
 ]
,
...
}
```

Tip

Display the roles in the Admin UI by selecting Manage > Role.

2. Create a new managed assignment named Employee.

   The assignment is specifically for the mapping from the managed user repository to the LDAP server. The assignment sets the value of the `employeeType` attribute on the LDAP server to `Employee`:

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --header "Content-type: application/json" \
 --request POST \
 --data '{
   "name" : "Employee",
   "description": "Assignment for employees.",
   "mapping" : "managedUser_systemLdapAccounts",
   "attributes": [
     {
       "name": "employeeType",
       "value": "Employee",
       "assignmentOperation" : "mergeWithTarget",
       "unassignmentOperation" : "removeFromTarget"
     }
   ]
 }' \
 "http://localhost:8080/openidm/managed/assignment?_action=create"
{
  "_id": "f2830b80-6ab8-416b-b219-d3bf2efd0ed3",
  "_rev": "1",
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
  "name": "Employee",
  "description": "Assignment for employees.",
  "mapping": "managedUser_systemLdapAccounts",
  "attributes": [
    {
      "name": "employeeType",
      "value": "Employee",
      "assignmentOperation": "mergeWithTarget",
      "unassignmentOperation": "removeFromTarget"
    }
  ]
}
```

Tip

Create the assignment in the Admin UI:

1. Select Manage > Assignment, and click New Assignment.
2. Enter a name and description for the assignment, and select the mapping for which the assignment is applied (managedUser_systemLdapAccounts).
3. Click Add Assignment.
4. Select the Attributes tab, and click Add an Attribute.
5. Select employeeType, and enter the value Employee.
6. Click Save.

3. Add the assignment to the Employee role that you created previously.

   Assignments are implemented as *relationship objects*. This means that you add an assignment to a role by *referencing* the assignment in the role's `assignments` field:

   This command patches the Employee role to update its `assignments` field.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
 --header "Content-type: application/json" \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request PATCH \
 --data '[
   {
     "operation" : "add",
     "field" : "/assignments/-",
     "value" : { "_ref": "managed/assignment/f2830b80-6ab8-416b-b219-d3bf2efd0ed3"}
   }
 ]' \
 "http://localhost:8080/openidm/managed/role/2902afd5-155a-49c0-9dd9-7e6bfcf1708a"
{
  "_id": "2902afd5-155a-49c0-9dd9-7e6bfcf1708a",
  "_rev": "2",
  "name": "Employee",
  "description": "Role granted to workers on the payroll."
}
```

Tip

Add the assignment to the role in the Admin UI:

1. Select Manage > Role, and select the Employee role.
2. On the Managed Assignments tab, click Add Managed Assignments.
3. Select the Employee assignment and click Add.

7.2.3.2. Granting a Role to a User and Observing that User's Role Assignments

When a role is granted to a user (by updating the users `roles` property), any assignments that are referenced by the role are automatically referenced in the user's `assignments` property.

In this section, we will grant the Employee role we created previously to the user Barbara Jensen, who was created in the managed/user repository during the reconciliation from OpenDJ.

1. Before you can update Barbara Jensen's entry, determine the identifier of her entry by querying her username, `bjensen`, and requesting only her `_id` field:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/managed/user?_queryFilter=/userName+eq+'bjensen'&_fields=_id"

{
  "result" : [ {
    "_id" : "2c7daf46-d3ce-4bc5-9790-b44113bca8e7",
    "_rev" : "1"
  } ],
  ...
}
```

From the output, you can see that bjensen's `_id` is `2c7daf46-d3ce-4bc5-9790-b44113bca8e7`. (This unique ID will obviously be different in your command output.)

2. Update bjensen's entry by adding a reference to the ID of the Employee role as a value of her `roles` attribute:

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --header "Content-type: application/json" \
 --request PATCH \
 --data '[
     {
         "operation" : "add",
         "field" : "/roles/-",
         "value" : { "_ref": "managed/role/2902afd5-155a-49c0-9dd9-7e6bfcf1708a" }
     }
 ]' \
 "http://localhost:8080/openidm/managed/user/2c7daf46-d3ce-4bc5-9790-b44113bca8e7"
{
  "_id": "2c7daf46-d3ce-4bc5-9790-b44113bca8e7",
  "_rev": "4",
  "displayName": "Barbara Jensen",
  "description": "Created for OpenIDM",
  "givenName": "Barbara",
  "mail": "bjensen@example.com",
  "telephoneNumber": "1-360-229-7105",
  "sn": "Jensen",
  "userName": "bjensen",
  "accountStatus": "active",
  "effectiveRoles": [
    {
      "_ref": "managed/role/2902afd5-155a-49c0-9dd9-7e6bfcf1708a"
    }
  ],
  "effectiveAssignments": [
    {
      "name": "Employee",
      "description": "Assignment for employees.",
      "mapping": "managedUser_systemLdapAccounts",
      "attributes": [
        {
          "name": "employeeType",
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
          "value": "Employee",
          "assignmentOperation": "mergeWithTarget",
          "unassignmentOperation": "removeFromTarget"
        }
      ],
      "_id": "f2830b80-6ab8-416b-b219-d3bf2efd0ed3",
      "_rev": "1"
    }
  ]
}
```

Tip

Assign the role to bjensen by using the Admin UI:

1. Select Manage > User, and click on bjensen's entry.
2. On the Provisioning Roles tab, click Add Provisioning Roles.
3. Select the Employee role and click Add.

3. Take a closer look at bjensen's entry, specifically at her roles, effective roles and effective assignments:

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/managed/user?_queryFilter=/userName+eq+'bjensen'&_fields=_id,userName
,roles,effectiveRoles,effectiveAssignments"
{
  "result": [
    {
      "_id": "2c7daf46-d3ce-4bc5-9790-b44113bca8e7",
      "_rev": "4",
      "userName": "bjensen",
      "roles": [
        {
          "_ref": "managed/role/2902afd5-155a-49c0-9dd9-7e6bfcf1708a",
          "_refProperties": {
            "_id": "b1c29213-e726-466a-9051-e9bb4e593331",
            "temporalConstraints": [],
            "_grantType": "",
            "_rev": "2"
          }
        }
      ],
      "effectiveRoles": [
        {
          "_ref": "managed/role/2902afd5-155a-49c0-9dd9-7e6bfcf1708a"
        }
      ],
      "effectiveAssignments": [
        {
          "name": "Employee",
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
          "description": "Assignment for employees.",
          "mapping": "managedUser_systemLdapAccounts",
          "attributes": [
            {
              "name": "employeeType",
              "value": "Employee",
              "assignmentOperation": "mergeWithTarget",
              "unassignmentOperation": "removeFromTarget"
            }
          ],
          "_id": "f2830b80-6ab8-416b-b219-d3bf2efd0ed3",
          "_rev": "1"
        }
      ]
    }
  ],
  ...
}
```

Note that bjensen now has the calculated property `effectiveAssignments`, which includes the set of assignments that pertains to any user with the Employee role. Currently the assignment lists the `employeeType` attribute.

In the next section, you will see how the assignment is used to set the value of the `employeeType` attribute in the LDAP server.

7.2.3.3. Propagating Assignments to an External System

This section provides a number of steps that show how effective assignments are propagated out to the external systems associated with their mappings.

1. Verify that bjensen's `employeeType` has been set correctly in the OpenDJ.

   Because implicit synchronization is enabled by default in OpenIDM, any changes made to a managed user object are pushed out to all the external systems for which mappings are configured.

   So, because bjensen has an effective assignment that sets an attribute in her LDAP entry, you should immediately see the resulting change in her LDAP entry.

   To verify that her entry has changed, run an `ldapsearch` on her entry and check the value of her `employeeType` attribute.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

Note

This command assumes that you are using the **ldapsearch** provided with OpenDJ (`opendj/bin/ldapsearch`).

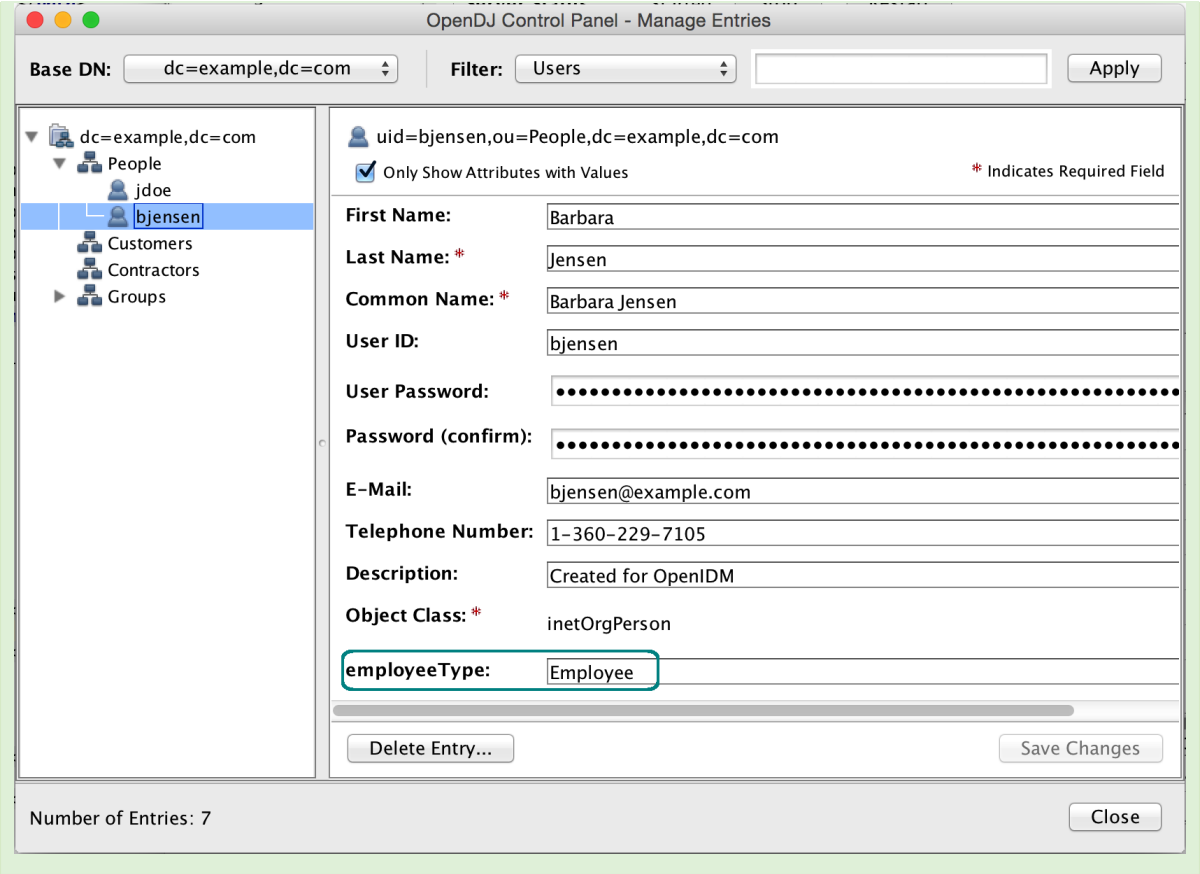
```
$ ldapsearch \
 --port 1389 \
 --hostname localhost \
 --baseDN "dc=example,dc=com" \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --searchScope sub \
 "(uid=bjensen)" dn uid employeeType
dn: uid=bjensen,ou=People,dc=example,dc=com
uid: bjensen
employeeType: Employee
```

Note that bjensen's `employeeType` attribute is correctly set to `Employee`.

Tip

You can also check bjensen's LDAP entry by using the OpenDJ Control Panel (`opendj/bin/control-panel`):

Copyright © 2011-2016 ForgeRock AS. All rights reserved.



2. To observe how a managed user's roles can be used to provision group membership in an external directory, we add the groups that an Employee and a Contractor should have in the corporate directory (OpenDJ) as assignment attributes of the respective roles.

   First, look at the current `assignments` of the Employee role again:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/managed/role/2902afd5-155a-49c0-9dd9-7e6bfcf1708a?_fields=assignments
,name"
{
  "_id": "2902afd5-155a-49c0-9dd9-7e6bfcf1708a",
  "_rev": "2",
  "assignments": [
    {
      "_ref": "managed/assignment/f2830b80-6ab8-416b-b219-d3bf2efd0ed3",
      "_refProperties": {
        "_id": "c0005ecb-9dda-4db1-8660-a723b8237f16",
        "temporalConstraints": [],
        "_grantType": "",
        "_rev": "1"
      }
    }
  ],
  "name": "Employee"
}
```

To update the `groups` attribute in bjensen's LDAP entry, you do not need to create a *new* assignment. You simply need to add the attribute for LDAP groups to the assignment with ID `f2830b80-6ab8-416b-b219-d3bf2efd0ed3`:

```
$ curl \
 --header "Content-type: application/json" \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request PATCH \
 --data '[
     {
         "operation" : "add",
         "field" : "/attributes/-",
         "value" : {
             "name": "ldapGroups",
             "value": [
                 "cn=Employees,ou=Groups,dc=example,dc=com",
                 "cn=Chat Users,ou=Groups,dc=example,dc=com"
             ],
             "assignmentOperation" : "mergeWithTarget",
             "unassignmentOperation" : "removeFromTarget"
         }
     }
 ]' \
 "http://localhost:8080/openidm/managed/assignment/f2830b80-6ab8-416b-b219-d3bf2efd0ed3"

{
  "_id": "f2830b80-6ab8-416b-b219-d3bf2efd0ed3",
  "_rev": "2",
  "name": "Employee",
  "description": "Assignment for employees.",
  "mapping": "managedUser_systemLdapAccounts",
  "attributes": [
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
    {
      "name": "employeeType",
      "value": "Employee",
      "assignmentOperation": "mergeWithTarget",
      "unassignmentOperation": "removeFromTarget"
    },
    {
      "name": "ldapGroups",
      "value": [
        "cn=Employees,ou=Groups,dc=example,dc=com",
        "cn=Chat Users,ou=Groups,dc=example,dc=com"
      ],
      "assignmentOperation": "mergeWithTarget",
      "unassignmentOperation": "removeFromTarget"
    }
  ]
}
```

So, the Employee assignment now sets two attributes on the LDAP system - the `employeeType` attribute, and the `ldapGroups` attribute.

Tip

To add more attributes to the Employee assignment in the Admin UI:

1. Select Manage > Assignment, and click on the Employee assignment.
2. On the Attributes tab, select Add an attribute and select the ldapGroups attribute.
3. Enter the values

   ```
   cn=Employees,ou=Groups,dc=example,dc=com
   ```

   and

   ```
   cn=Chat Users,ou=Groups,dc=example,dc=com
   ```

   and click Save.

3. With the implicit synchronization between the managed user repository and OpenDJ, bjensen should now be a member of the `cn=Employees` and `cn=Chat Users` groups in LDAP.

   You can verify this with the following `ldapsearch` command. This command returns bjensen's group membership, in her `isMemberOf` attribute.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ ldapsearch \
 --port 1389 \
 --hostname localhost \
 --baseDN "dc=example,dc=com" \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --searchScope sub \
 "(uid=bjensen)" dn uid employeeType isMemberOf

dn: uid=bjensen,ou=People,dc=example,dc=com
uid: bjensen
employeeType: Employee
isMemberOf: cn=openidm2,ou=Groups,dc=example,dc=com
isMemberOf: cn=Chat Users,ou=Groups,dc=example,dc=com
isMemberOf: cn=Employees,ou=Groups,dc=example,dc=com
```

You can also check bjensen's group membership by using the OpenDJ Control Panel (as shown previously), or by querying her object in the LDAP system, over the REST interface:

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/system/ldap/account?_queryFilter=/uid+sw+'bjensen'&_fields=dn,uid
,employeeType,ldapGroups"
{
  "result" : [ {
    "_id" : "uid=bjensen,ou=People,dc=example,dc=com",
    "dn" : "uid=bjensen,ou=People,dc=example,dc=com",
    "uid" : "bjensen",
    "employeeType" : "Employee",
    "ldapGroups" : [
       "cn=openidm2,ou=Groups,dc=example,dc=com",
       "cn=Employees,ou=Groups,dc=example,dc=com",
       "cn=Chat Users,ou=Groups,dc=example,dc=com"
    ]
  } ],
  ...
}
```

In the original LDIF file, bjensen was already a member of the openidm2 group. You can ignore this group for the purposes of this sample.

Tip

Use the Admin UI to see bjensen's LDAP groups as follows:

1. Select Manage > User, and select bjensen's entry.
2. On the Linked Systems tab, scroll down to the ldapGroups item.

4. Now, create a new assignment that will apply to Contract employees, and add that assignment to the Contractor role.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

Create the Contractor assignment with the following command. This assignment sets the value of the `employeeType` attribute to `Contractor`, and updates the user's `ldapGroups` attribute to include the `cn=Contractors` group:

```
$ curl \
 --header "Content-type: application/json" \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request POST \
 --data '{
     "name" : "Contractor",
     "description": "Contractor assignment for contract workers.",
     "mapping": "managedUser_systemLdapAccounts",
     "attributes": [
         {
             "name": "ldapGroups",
             "value": [
                 "cn=Contractors,ou=Groups,dc=example,dc=com"
             ],
             "assignmentOperation" : "mergeWithTarget",
             "unassignmentOperation" : "removeFromTarget"
         },
         {
             "name": "employeeType",
             "value": "Contractor",
             "assignmentOperation" : "mergeWithTarget",
             "unassignmentOperation" : "removeFromTarget"
         }
     ]
 }' \
 "http://localhost:8080/openidm/managed/assignment?_action=create"
{
  "_id": "7536e234-1268-482d-8459-24c8ef832def",
  "_rev": "1",
  "name": "Contractor",
  "description": "Contractor assignment for contract workers.",
  "mapping": "managedUser_systemLdapAccounts",
  "attributes": [
    {
      "name": "ldapGroups",
      "value": [
        "cn=Contractors,ou=Groups,dc=example,dc=com"
      ],
      "assignmentOperation": "mergeWithTarget",
      "unassignmentOperation": "removeFromTarget"
    },
    {
      "name": "employeeType",
      "value": "Contractor",
      "assignmentOperation": "mergeWithTarget",
      "unassignmentOperation": "removeFromTarget"
    }
  ]
}
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

Note the ID of the Contractor assignment (7536e234-1268-482d-8459-24c8ef832def in this example).

Tip

Create the assignment by using the Admin UI, as described in Section 7.2.3.1, "Adding Assignments to a Role Definition").

5. Now, add the Contractor assignment to the Contractor role (ID e7f649ad-8013-4673-a52a-bdcac7483111 in this example):

```
$ curl \
 --header "Content-type: application/json" \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request PATCH \
 --data '[
     {
         "operation" : "add",
         "field" : "/assignments/-",
         "value" : {
             "_ref" : "managed/assignment/7536e234-1268-482d-8459-24c8ef832def"
         }
     }
 ]' \
 "http://localhost:8080/openidm/managed/role/e7f649ad-8013-4673-a52a-bdcac7483111"
{
  "_id": "e7f649ad-8013-4673-a52a-bdcac7483111",
  "_rev": "2",
  "name": "Contractor",
  "description": "Role granted to contract workers."
}
```

Tip

Add the Contractor assignment to the Contractor role in the Admin UI, as follows:

1. Select Manage > Role, and select the Contractor role.
2. On the Managed Assignments tab, click Add Managed Assignment.
3. Select the Contractor assignment from the dropdown list, and click Add.

6. Next, we need to grant the Contractor role to user jdoe. Before we can patch jdoe's entry, we need to know his system-generated ID. To obtain the ID, query jdoe's entry as follows:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/managed/user?_queryFilter=/userName+eq+'jdoe'&_fields=_id"

{
  "result": [
    {
      "_id": "92680be0-82f9-4297-9e00-c35c7cf700d2",
      "_rev": "2"
    }
  ],
  ...
}
```

From the output, you can see that jdoe's `_id` is `92680be0-82f9-4297-9e00-c35c7cf700d2`. (This unique ID will obviously be different in your command output.)

7. Update jdoe's entry by adding a reference to the ID of the Contractor role (e7f649ad-8013-4673-a52a-bdcac7483111) as a value of his `roles` attribute:

```
$ curl \
 --header "Content-type: application/json" \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request PATCH \
 --data '[
     {
         "operation" : "add",
         "field" : "/roles/-",
         "value" : {
             "_ref": "managed/role/e7f649ad-8013-4673-a52a-bdcac7483111"
         }
     }
 ]' \
 "http://localhost:8080/openidm/managed/user/92680be0-82f9-4297-9e00-c35c7cf700d2"
{
  "_id": "92680be0-82f9-4297-9e00-c35c7cf700d2",
  "_rev": "4",
  "displayName": "John Doe",
  "description": "Created for OpenIDM",
  "givenName": "John",
  "mail": "jdoe@example.com",
  "telephoneNumber": "1-415-599-1100",
  "sn": "Doe",
  "userName": "jdoe",
  "accountStatus": "active",
  "effectiveRoles": [
    {
      "_ref": "managed/role/e7f649ad-8013-4673-a52a-bdcac7483111"
    }
  ],
  "effectiveAssignments": [
    {
      "name": "Contractor",
      "description": "Contractor assignment for contract workers.",
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
      "mapping": "managedUser_systemLdapAccounts",
      "attributes": [
        {
          "name": "ldapGroups",
          "value": [
            "cn=Contractors,ou=Groups,dc=example,dc=com"
          ],
          "assignmentOperation": "mergeWithTarget",
          "unassignmentOperation": "removeFromTarget"
        },
        {
          "name": "employeeType",
          "value": "Contractor",
          "assignmentOperation": "mergeWithTarget",
          "unassignmentOperation": "removeFromTarget"
        }
      ],
      "_id": "7536e234-1268-482d-8459-24c8ef832def",
      "_rev": "1"
    }
  ]
}
```

Tip

Grant the Contractor role to jdoe by using the Admin UI, as follows:

1. Select Manage > User, and click on jdoe's entry.
2. On the Provisioning Roles tab, click Add Provisioning Roles.
3. Select the Contractor role and click Add.
4. Click Save.

8. Check jdoe's entry on the LDAP system.

   With the implicit synchronization between the managed user repository and OpenDJ, jdoe should now be a member of the `cn=Contractors` group in LDAP. In addition, his `employeeType` should have been set to `Contractor`.

   You can verify this with the following REST query. This command returns jdoes's group membership, in his `isMemberOf` attribute, and his `employeeType`:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/system/ldap/account?_queryFilter=/uid+sw+'jdoe'&_fields=dn,uid
,employeeType,ldapGroups"
{
  "result": [
    {
      "_id": "uid=jdoe,ou=People,dc=example,dc=com",
      "givenName": "John",
      "ldapGroups": [
        "cn=openidm,ou=Groups,dc=example,dc=com",
        "cn=Contractors,ou=Groups,dc=example,dc=com"
      ],
      "mail": "jdoe@example.com",
      "employeeType": "Contractor",
      "uid": "jdoe",
      "telephoneNumber": "1-415-599-1100",
      "sn": "Doe",
      "disabled": null,
      "cn": "John Doe",
      "description": "Created for OpenIDM",
      "dn": "uid=jdoe,ou=People,dc=example,dc=com"
    }
  ],
  ...
}
```

Tip

Use the Admin UI to see jdoe's LDAP groups as follows:

1. Select Manage > User, and select jdoe's entry.
2. On the Linked Systems tab, scroll down to the ldapGroups item.

7.2.3.4. Removing a Role Grant From a User and Observing That User's Role Assignments

In this section, you will remove the Contractor role from jdoe's managed user entry and observe the subsequent change to jdoe's managed assignments, and to the corresponding attributes in OpenDJ.

1. Before you change jdoe's roles, view his entry again to examine his current roles.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/managed/user?_queryFilter=/userName+eq+'jdoe'&_fields=_id,roles"
{
  "result": [
    {
      "_id": "92680be0-82f9-4297-9e00-c35c7cf700d2",
      "_rev": "4",
      "roles": [
        {
          "_ref": "managed/role/e7f649ad-8013-4673-a52a-bdcac7483111",
          "_refProperties": {
            "_id": "093fc34b-0694-478e-952e-98d0a828b1ac",
            "_rev": "2"
          }
        }
      ]
    }
  ],
  ...
}
```

Note that jdoe's ID is 92680be0-82f9-4297-9e00-c35c7cf700d2 and the ID of the *relationship* that expresses the role grant is 093fc34b-0694-478e-952e-98d0a828b1ac. You will need these IDs in the next step.

Tip

View jdoe's current roles in the Admin UI:

1. Select Manage > User, and select jdoe's entry.
2. The Provisioning Roles tab lists the current roles.

2. Remove the Contractor role from jdoe's entry by sending a DELETE request to the user entry, specifying the relationship ID:

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request DELETE \
 "http://localhost:8080/openidm/managed/user/92680be0-82f9-4297-9e00-c35c7cf700d2/roles/093fc34b-0694-478e
-952e-98d0a828b1ac"
{
  "_ref": "managed/role/e7f649ad-8013-4673-a52a-bdcac7483111",
  "_refProperties": {
    "_id": "093fc34b-0694-478e-952e-98d0a828b1ac",
    "_rev": "2"
  }
}
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

Tip

Use the Admin UI to remove the Contractor role from jdoe's entry as follows:

1. Select Manage > User, and select jdoe's entry.
2. On the Provisioning Roles tab, check the box next to the Contractor role and click Remove Selected Provisioning Roles.

3. Verify jdoe's `employeeType` and `ldapGroups`.

   The removal of the Contractor role causes a synchronization operation to be run on jdoe's entry. His `employeeType` and `ldapGroups` attributes in OpenDJ should be reset to what they were before he was granted the Contractor role.

   You can check jdoe's attributes by querying his object in the LDAP directory, over the REST interface:

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/system/ldap/account?_queryFilter=/uid+sw+'jdoe'&_fields=dn,uid
,employeeType,ldapGroups"
{
  "result" : [ {
    "sn" : "Doe",
    "telephoneNumber" : "1-415-599-1100",
    "employeeType" : null,
    "dn" : "uid=jdoe,ou=People,dc=example,dc=com",
    "cn" : "John Doe",
    "uid" : "jdoe",
    "ldapGroups" : [ "cn=openidm,ou=Groups,dc=example,dc=com" ],
    "givenName" : "John",
    "mail" : "jdoe@example.com",
    "description" : "Created for OpenIDM",
    "_id" : "uid=jdoe,ou=People,dc=example,dc=com"
  } ],
  ...
}
```

Tip

Use the Admin UI to see jdoe's LDAP groups as follows:

1. Select Manage > User, and select jdoe's entry.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

2. On the Linked Systems tab, scroll down to the ldapGroups item.

This concludes the provisioning with roles sample. For more information about roles, assignments, and how to manipulate them, see Section 8.4, "Working With Managed Roles" in the *Integrator's Guide*.

7.3. Temporal Constraints Sample - Applying Time-Based Constraints to Roles

To restrict the period during which a role is effective, you can set a *temporal constraint* on the role itself, or on the role grant. A temporal constraint that is set on a role definition applies to all grants of that role. A temporal constraint that is set on a role grant enables you to specify the period that the role is valid per user. For more information about temporal constraints, see Section 8.4.4, "Using Temporal Constraints to Restrict Effective Roles" in the *Integrator's Guide*.

This sample builds on what you learned in the previous two roles samples. The sample demonstrates how to add a temporal constraint to the Contractor role to restrict the period that it is effective. The previous sample ended with jdoe's contract ending because he finished up the work that was required for the company. The company now has additional work that requires jdoe's expertise and has rehired him as a contractor. This time however, the company requires that jdoe's contractor role be terminated as soon as his contract expires.

7.3.1. Before You Start

This sample assumes that you have already run through the previous two roles samples, and have installed and configured OpenDJ as for Sample 2 (see Section 3.1.1, "LDAP Server Configuration"). The LDAP user must have write access to create users from OpenIDM on the LDAP server. When you set up the LDAP server, import the LDIF file for this sample (`openidm/samples/roles/temporalConstraints/data/Example.ldif).`

1. Prepare OpenIDM as described in Section 1.3, "Preparing OpenIDM", then start OpenIDM with the configuration for the temporal constraints sample.

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

2. Create the Employee and Contractor roles, and their assignments, either by using the Admin UI, or by using the REST interface, as described in the previous sample

   Note the IDs of the roles and assignments because you will use them in the commands that follow.

3. Reconcile the repository, as described in the previous sample.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

The sample is now ready to demonstrate temporal constraints on roles.

7.3.2. Run the Sample

This section assumes that you have reconciled the managed user repository to populate it with the users from the LDAP server, and that you have created the Employee and Contractor roles and their assignments.

This part of the sample demonstrates the following features of the OpenIDM roles implementation:

- Section 7.3.2.1, "Adding a Temporal Constraint to a Role Definition"
- Section 7.3.2.2, "Granting a Role With a Temporal Constraint to a User"
- Section 7.3.2.3, "Observing the Effects of a Temporal Role"
- Section 7.3.2.4, "Removing the Role Grant From the User"

7.3.2.1. Adding a Temporal Constraint to a Role Definition

In this section, you will update the definition of the Contractor role to add a temporal constraint.

1. If you are running the sample directly over the REST interface, obtain the ID of the Contractor role:

   You can skip this step if you are using the Admin UI.

   ```
   $ curl \
    --header "X-OpenIDM-Username: openidm-admin" \
    --header "X-OpenIDM-Password: openidm-admin" \
    --request GET \
    "http://localhost:8080/openidm/managed/role?_queryFilter=true&_prettyPrint=true"
   {
     "result" : [ {
       "_id" : "1a4c9047-1ce3-4f39-8901-e4f60176330e",
       "_rev" : "1",
       "name" : "Employee",
       "description" : "Role granted to workers on the payroll."
     }, {
       "_id" : "06128fc1-b89b-4fe8-b9b3-4de30fd17b9e",
       "_rev" : "1",
       "name" : "Contractor",
       "description" : "Role granted to contract workers."
     } ],
     ...
   }
   ```

   In this example, the ID of the Contractor role is 06128fc1-b89b-4fe8-b9b3-4de30fd17b9e.

2. Patch the contractor role with a temporal constraint by adding a start and end date to the role definition.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

For the start date, use the current time plus 5 minutes. This will demonstrate that you can add a future temporal constraint on a user before their contract even starts. For the end date, use 10 minutes from the current time.

In this example, the current time is 15:30:00. Adding 5 minutes results in a start time of 15:35:00. Adding 10 minutes results in an end time of 15:40:00. Adjust these values for the current time that you run the sample.

To add the temporal constraint over REST:

```
$ curl \
 --header "Content-type: application/json" \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request PATCH \
 --data '[
   {
     "operation" : "add",
     "field" : "/temporalConstraints",
     "value" : [{"duration": "2016-05-05T15:35:00.000Z/2016-05-05T15:40:00.000Z"}]
   }
 ]' \
 "http://localhost:8080/openidm/managed/role/06128fc1-b89b-4fe8-b9b3-4de30fd17b9e"
{
  "_id": "06128fc1-b89b-4fe8-b9b3-4de30fd17b9e",
  "_rev": "2",
  "name": "Contractor",
  "description": "Role granted to contract workers.",
  "temporalConstraints": [
    {
      "duration": "2016-05-05T15:35:00.000Z\/2016-05-05T15:40:00.000Z"
    }
  ]
}
```

To add the temporal constraint by using the Admin UI:

1. Select Manage > Role and click the Contractor role.
2. Select Temporal Constraint, then select a time zone, start date, and end date.
3. Click Save.

7.3.2.2. Granting a Role With a Temporal Constraint to a User

In this section, you will grant the Contractor role to jdoe.

1. If you are running this sample over REST, determine the identifier of jdoe's entry querying the existing managed users.

   If you are using the Admin UI you can skip this step.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/managed/user?_queryFilter=true"
{
  "result" : [ {
    ...
    }, {
    "_id" : "0c470c71-bb4e-4cf1-bc9d-77d7b35b180b",
    "_rev" : "2",
    "displayName" : "John Doe",
    "description" : "Created for OpenIDM",
    "givenName" : "John",
    "mail" : "jdoe@example.com",
    "telephoneNumber" : "1-415-599-1100",
    "sn" : "Doe",
    "userName" : "jdoe",
    "accountStatus" : "active",
    "effectiveRoles" : [ ],
    "effectiveAssignments" : [ ]
  } ],
  ...
}
```

Note that jdoe's ID is 0c470c71-bb4e-4cf1-bc9d-77d7b35b180b.

2. Update jdoe's entry to grant him the Contractor role:

```
$ curl \
 --header "Content-type: application/json" \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request PATCH \
 --data '[
   {
     "operation" : "add",
     "field" : "/roles/-",
     "value" : {
       "_ref": "managed/role/06128fc1-b89b-4fe8-b9b3-4de30fd17b9e"
     }
   }
 ]' \
 "http://localhost:8080/openidm/managed/user/0c470c71-bb4e-4cf1-bc9d-77d7b35b180b"
{
  "_id": "0c470c71-bb4e-4cf1-bc9d-77d7b35b180b",
  "_rev": "4",
  "displayName": "John Doe",
  "description": "Created for OpenIDM",
  "givenName": "John",
  "mail": "jdoe@example.com",
  "telephoneNumber": "1-415-599-1100",
  "sn": "Doe",
  "userName": "jdoe",
  "accountStatus": "active",
  "effectiveRoles": [

  ],
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
  "effectiveAssignments": [

  ]
}
```

Note that, at this stage, jdoe does not have any effective roles or assignments because his contract has not yet started. Check jdoe's entry again between the start and end date of the temporal constraint to observe the assignment.

To grant the Contractor role to jdoe by using the Admin UI:

1. Select Manage > User and click jdoe's entry.
2. On the Provisioning Roles tab, click Add Provisioning Roles.
3. Select the Contractor role and click Add.

7.3.2.3. Observing the Effects of a Temporal Role

During the contract period, query jdoe again to see when the Contractor role and assignments are present in his list of effective roles and effective assignments.

1. When the start time that you set has been reached, query jdoe's entry as follows:

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/managed/user?_queryFilter=/userName+sw+"jdoe"&_fields=roles
,effectiveRoles,effectiveAssignments"
{
  "result": [
    {
      "_id": "0c470c71-bb4e-4cf1-bc9d-77d7b35b180b",
      "_rev": "6",
      "roles": [
        {
          "_ref": "managed/role/06128fc1-b89b-4fe8-b9b3-4de30fd17b9e",
          "_refProperties": {
            "_id": "6774696d-999e-4483-87a9-02f501752b29",
            "_rev": "4"
          }
        }
      ],
      "effectiveRoles": [
        {
          "_ref": "managed/role/06128fc1-b89b-4fe8-b9b3-4de30fd17b9e"
        }
      ],
      "effectiveAssignments": [
        {
          "name": "Contractor",
          "description": "Contractor assignment for contract workers.",
          "mapping": "managedUser_systemLdapAccounts",
          "attributes": [
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
            {
              "name": "ldapGroups",
              "value": [
                "cn=Contractors,ou=Groups,dc=example,dc=com"
              ],
              "assignmentOperation": "mergeWithTarget",
              "unassignmentOperation": "removeFromTarget"
            },
            {
              "name": "employeeType",
              "value": "Contractor",
              "assignmentOperation": "mergeWithTarget",
              "unassignmentOperation": "removeFromTarget"
            }
          ],
          "_id": "3e61a1db-202d-4761-aeb7-b617d3376a79",
          "_rev": "1"
        }
      ]
    }
  ],
 ...
}
```

There is not really a comparable way to perform this step in the UI.

2. Between the start and end times, look at jdoe's entry in the LDAP server. At this point, the Contractor group should be effective for jdoe's entry, so his LDAP groups should include the Contractors group:

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/system/ldap/account?_queryFilter=/uid+sw+"jdoe"&_fields=dn,uid
,employeeType,ldapGroups"
{
  "result": [
    {
      "_id": "uid=jdoe,ou=People,dc=example,dc=com",
      "dn": "uid=jdoe,ou=People,dc=example,dc=com",
      "uid": "jdoe",
      "employeeType": "Contractor",
      "ldapGroups": [
        "cn=openidm,ou=Groups,dc=example,dc=com",
        "cn=Contractors,ou=Groups,dc=example,dc=com"
      ]
    }
  ]
}
```

To look at jdoe's LDAP groups in the Admin UI:

1. Select Manage > User, and select jdoe's entry.
2. On the Linked Systems tab, scroll down to the ldapGroups item.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

3. Jdoe's contract expires when the current date is greater than the end date of the temporal constraint.

   Query jdoe's entry at that point to see that he no longer has the Contractor roles and assignments in his effective roles and assignments lists (although the role is still there in his roles list, it is no longer effective).

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/managed/user?_queryFilter=/userName+sw+"jdoe"&_fields=roles
,effectiveRoles,effectiveAssignments"
{
  "result": [
    {
      "_id": "0c470c71-bb4e-4cf1-bc9d-77d7b35b180b",
      "_rev": "8",
      "roles": [
        {
          "_ref": "managed/role/06128fc1-b89b-4fe8-b9b3-4de30fd17b9e",
          "_refProperties": {
            "_id": "6774696d-999e-4483-87a9-02f501752b29",
            "_rev": "6"
          }
        }
      ],
      "effectiveRoles": [

      ],
      "effectiveAssignments": [

      ]
    }
  ],
 ...
}
```

   There is not really comparable way to view this information in the Admin UI. Take note of the ID of the *relationship* that expresses the role grant (6774696d-999e-4483-87a9-02f501752b29 in this example). You will need this ID in the next step.

4. Now verify that jdoe is no longer part of the Contractors group in OpenDJ:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/system/ldap/account?_queryFilter=/uid+sw+"jdoe"&_fields=dn,uid
,employeeType,ldapGroups"
{
  "result": [
    {
      "_id": "uid=jdoe,ou=People,dc=example,dc=com",
      "dn": "uid=jdoe,ou=People,dc=example,dc=com",
      "uid": "jdoe",
      "employeeType": null,
      "ldapGroups": [
        "cn=openidm,ou=Groups,dc=example,dc=com"
      ]
    }
  ],
 ...
}
```

In the Admin UI:

1. Select Manage > User, and select jdoe's entry.
2. On the Linked Systems tab, scroll down to the ldapGroups item.

7.3.2.4. Removing the Role Grant From the User

To finish up this sample, remove the Contractor role from jdoe by sending a DELETE request to his managed user entry, specifying the relationship ID (6774696d-999e-4483-87a9-02f501752b29 in this example):

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request DELETE \
 "http://localhost:8080/openidm/managed/user/0c470c71-bb4e-4cf1-bc9d-77d7b35b180b/roles/6774696d-999e-4483
-87a9-02f501752b29"
{
  "_ref": "managed/role/06128fc1-b89b-4fe8-b9b3-4de30fd17b9e",
  "_refProperties": {
    "_id": "6774696d-999e-4483-87a9-02f501752b29",
    "_rev": "6"
  }
}
```

Alternatively, use the Admin UI to remove the Contractor role from jdoe's entry as follows:

1. Select Manage > User, and select jdoe's entry.
2. On the Provisioning Roles tab, check the box next to the Contractor role and click Remove Selected Provisioning Roles.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

Chapter 8

The Multi-Account Linking Sample

The sample provided in the `samples/multiaccountlinking` directory illustrates how OpenIDM addresses links from multiple accounts to one identity.

This sample is based on a common use case in the insurance industry, where a company (Example.com) employs agents to sell policies to their insured customers. Most of their agents are also insured. These different roles are sometimes known as the multi-account linking conundrum.

With minor changes, this sample works for other use cases. For example, you may have a hospital that employs doctors who treat patients. Some of their doctors are also patients of that hospital.

8.1. External LDAP Configuration

Configure the LDAP server as for sample 2, (see Section 3.1.1, "LDAP Server Configuration").

The LDAP user must have write access to create users from OpenIDM on the LDAP server. When you configure the LDAP server, import the appropriate LDIF file, in this case, *openidm/samples/multiaccountlinking/data/Example.ldif*.

8.2. Install the Sample

Prepare OpenIDM as described in Section 1.3, "Preparing OpenIDM", then start OpenIDM with the following configuration for the Multi-Account Linking sample.

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

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

8.3. Create New Identities for the Sample

For the purpose of this sample, create identities for users John Doe and Barbara Jensen. To create these identities from the Admin UI, navigate to `https://localhost:8443/admin` and click Manage > User > New User.

Alternatively, use the following REST calls to set up identities for the noted users:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
--cacert self-signed.crt
 \
--header "Content-Type: application/json"
 \
--header "X-OpenIDM-Username: openidm-admin"
 \
--header "X-OpenIDM-Password: openidm-admin"
 \
--request POST
 \
--data '{
  "displayName" : "Barbara Jensen",
  "description" : "Created for OpenIDM",
  "givenName" : "Barbara",
  "mail" : "bjensen@example.com",
  "telephoneNumber" : "1-360-229-7105",
  "sn" : "Jensen",
  "userName" : "bjensen",
  "accountStatus" : "active"
}' \
"https://localhost:8443/openidm/managed/user?_action=create"
```

```
$ curl \
--cacert self-signed.crt
 \
--header "Content-Type: application/json"
 \
--header "X-OpenIDM-Username: openidm-admin"
 \
--header "X-OpenIDM-Password: openidm-admin"
 \
--request POST
 \
--data '{
  "displayName" : "John Doe",
  "description" : "Created for OpenIDM",
  "givenName" : "John",
  "mail" : "jdoe@example.com",
  "telephoneNumber" : "1-415-599-1100",
  "sn" : "Doe",
  "userName" : "jdoe",
  "accountStatus" : "active"
}' \
"https://localhost:8443/openidm/managed/user?_action=create"
```

In the output, you will see an ID number associated with each user, in the following format:

```
 "_id" : "35d0a49d-2571-401f-b429-96c66b23a1c0",
```

Record the `_id` number for each user. You will use those numbers to assign desired roles for each user.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

8.4. Create New Roles for the Sample

For this sample, to set up links for multiple accounts on OpenIDM, you need to set up roles. To do so, set up roles for `Agent` and `Insured`. To create these roles in the Admin UI, navigate to `https://localhost:8443/admin` and click Manage > Role > New Role.

Alternatively, use the following REST calls to set up the `Agent` and `Insured` roles:

```
$ curl \
--cacert self-signed.crt
 \
--header "Content-Type: application/json"
 \
--header "X-OpenIDM-Username: openidm-admin"
 \
--header "X-OpenIDM-Password: openidm-admin"
 \
--request POST
 \
--data '{
  "name" : "Agent",
  "description" : "Role assigned to insurance agents."
}' \
"https://localhost:8443/openidm/managed/role?_action=create"
```

```
$ curl \
--cacert self-signed.crt
 \
--header "Content-Type: application/json"
 \
--header "X-OpenIDM-Username: openidm-admin"
 \
--header "X-OpenIDM-Password: openidm-admin"
 \
--request POST
 \
--data '{
  "name" : "Insured",
  "description" : "Role assigned to insured customers."
}' \
"https://localhost:8443/openidm/managed/role?_action=create"
```

Do record the `_id` output for the `Agent` and `Insured` roles. You will use those numbers to assign desired roles for each user.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

Note

While you could use `PUT` to create roles with descriptive names, we recommend that you use `POST` to create roles with immutable IDs.

8.5. Assign Roles to Appropriate Users

Now you can assign roles to appropriate users. To review, user `jdoe` is an `Agent` and user `bjensen` is `Insured`.

You will need the `_id` value for each user. The `_id` values shown in the following commands are random; substitute the `_id` values that you collected when creating users.

The following command adds the `Agent` role to user `jdoe`, by their `_id` values:

```
$ curl \
--cacert self-signed.crt
 \
--header "Content-type: application/json"
 \
--header "X-OpenIDM-Username: openidm-admin"
 \
--header "X-OpenIDM-Password: openidm-admin"
 \
--header "If-Match: *"
 \
--request PATCH
 \
--data '[
    {
      "operation" : "add",
      "field" : "/roles/-",
      "value" : {
        "_ref" : "managed/role/287dc4b1-4b19-49ec-8b4c-28a6c12ede34"
      }
    }
  ]' \
"https://localhost:8443/openidm/managed/user/8fae84ed-1f30-4542-8087-e7fa6e89541c"
```

And this next command adds the `Insured` role to user `bjensen`:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
--cacert self-signed.crt
 \
--header "Content-type: application/json"
 \
--header "X-OpenIDM-Username: openidm-admin"
 \
--header "X-OpenIDM-Password: openidm-admin"
 \
--header "If-Match: *"
 \
--request PATCH
 \
--data '[
    {
      "operation" : "add",
      "field" : "/roles/-",
      "value" : {
        "_ref" : "managed/role/bb9302c4-5fc1-462c-8be2-b17c87175d1b"
      }
    }
  ]' \
"https://localhost:8443/openidm/managed/user/d0b79f30-946f-413a-b7d1-d813034fa345"
```

Now assign the `Insured` role to user `jdoe`, as that user is both an insured customer and an agent:

```
$ curl \
--cacert self-signed.crt
 \
--header "Content-type: application/json"
 \
--header "X-OpenIDM-Username: openidm-admin"
 \
--header "X-OpenIDM-Password: openidm-admin"
 \
--header "If-Match: *"
 \
--request PATCH
 \
--data '[
    {
      "operation" : "add",
      "field" : "/roles/-",
      "value" : {
        "_ref" : "managed/role/006935c2-b080-45cd-8347-881df42cae0c"
      }
    }
  ]' \
"https://localhost:8443/openidm/managed/user/a3335177-7366-4656-a66c-8d6e77a5786f"
```

User `jdoe` should now have two managed roles:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
...
"effectiveRoles" : [ {
  "_ref" : "managed/role/6aabe990-ec05-403e-bc5d-ff9b217ba571",
  "_refProperties" : {
    "_id" : "687714b8-5854-42c7-a190-c781ea5174c5",
    "_rev" : "1"
  }
}, {
  "_ref" : "managed/role/844110ce-3686-43bb-aabf-46b17a14abaa"
} ],
...
```

8.6. Background: Link Qualifiers, Agents, and Insured Customers

This is a good moment to take a step back, to see how this sample works, based on custom options in the `sync.json` configuration file.

OpenIDM defines mappings between source and target accounts in the `sync.json` file, which allows you to create a link between one source entry and multiple target entries using a concept known as a "link qualifier," which enables one-to-many relationships in mappings and policies.

For more information on resource mappings and link qualifiers, see the following sections of the Integrator's Guide:

Section 12.3.2, "Mapping Source Objects to Target Objects" in the *Integrator's Guide*.

Section 12.3.2.5, "Mapping a Single Source Object to Multiple Target Objects" in the *Integrator's Guide*.

In this sample, we use two link qualifiers:

- `insured` represents the insured customer accounts associated with Example.com, created under the following LDAP container: `ou=Customers,dc=example,dc=com`
- `agent` represents agent accounts, considered independent contractors, and created under the following LDAP container: `ou=Contractors,dc=example,dc=com`

Assume that agents and insured customers connect via two different portals. Each group gets access to different features, depending on the portal.

Agents may have two different accounts; one each for professional and personal use. While the accounts are different, the identity information for each agent should be the same for both accounts.

To that end, this sample sets up link qualifiers for two categories of users: `insured` and `agent`, under the `managedUser_systemLdapAccounts` mapping:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
{
  "name" : "managedUser_systemLdapAccounts",
  "source" : "managed/user",
  "target" : "system/ldap/account",
  "linkQualifiers" : [
    "insured",
    "agent"
  ],
  .....
}
```

You can verify this in the Admin UI. Click Configure > Mappings > `managedUser_systemLdapAccounts` > Properties > Link Qualifiers. You should see `insured` and `agent` in the list of configured Link Qualifiers.

In addition, this sample also includes a transformation script between an LDAP Distinguished Name (`dn`) and the two categories of users. The following excerpt of the `sync.json` file includes that script:

```
{
   "target" : "dn",
   "transform" : {
      "type" : "text/javascript",
      "globals" : { },
      "source" :
         "if (linkQualifier === 'agent') {
            'uid=' + source.userName + ',ou=Contractors,dc=example,dc=com';
         } else if (linkQualifier === 'insured') {
            'uid=' + source.userName + ',ou=Customers,dc=example,dc=com';
         }"
},
```

The following validSource script looks through the effective roles of a user, with two objectives:

- Determine whether the user has an `Agent` or `Insured` role, and segregates accounts based on link qualifiers and roles.
- Uses the `effectiveRoles` property to determine whether a user has an `Agent` or `Insured` role, based on the effective roles of that user.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
"validSource" : {
  "type" : "text/javascript",
  "globals" : { },
  "source" : "var res = false;
    var i=0;

    while (!res && i < source.effectiveRoles.length) {
      var roleId = source.effectiveRoles[i];
      if (roleId != null && roleId.indexOf("/") != -1) {
        var roleInfo = openidm.read(roleId);
        logger.warn("Role Info : {}",roleInfo);
        res = (((roleInfo.properties.name === 'Agent')
          &&(linkQualifier ==='agent'))
        || ((roleInfo.properties.name === 'Insured')
          &&(linkQualifier ==='insured')));
        }
        i++;
      }
      res"
}
```

You can see how correlation queries are configured in the `sync.json` file.

The structure for the correlation query specifies one of two link qualifiers: `insured` or `agent`. For each link qualifier, the correlation query defines a script that verifies if the subject `dn` belongs in a specific container. For this sample, the container (`ou`) may be Customers or Contractors.

You can can avoid specifying the structure of the `dn` attribute in two places in the `sync.json` file with the following code, which leverages the expression builder to reuse the construct defined in the `dn` mapping:

```
"correlationQuery" : [
  {
    "linkQualifier" : "insured",
    "expressionTree" : {
      "all" : [
        "dn"
      ]
    },
    "mapping" : "managedUser_systemLdapAccounts",
    "type" : "text/javascript",
    "file" : "ui/correlateTreeToQueryFilter.js"
  },
  {
    "linkQualifier" : "agent",
    "expressionTree" : {
      "all" : [
        "dn"
      ]
    },
    "mapping" : "managedUser_systemLdapAccounts",
    "type" : "text/javascript",
    "file" : "ui/correlateTreeToQueryFilter.js"
  }
],
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

You can also leverage the expression builder in the UI. Review how the UI illustrates the expression builder. To do so, click Configure > Mapping > select a mapping > Association > Association Rules. Edit either link qualifier. You will see how the expression builder is configured for this sample.

8.7. Update Roles With Desired LDAP Attributes

This use case illustrates how accounts frequently have different functions on target systems. For example, while agents may be members of a Contractor group, insured customers may be part of a Chat Users group (possibly for access to customer service).

While an `agent` may also be an insured customer, you do not want other `insured` accounts to have the same properties (or memberships) as the `agent` account. In this sample, we ensure that OpenIDM limits role based assignments to the correct account.

With the following commands you will create two managed assignments which will be used by the `agent` and `insured` roles.

Record the `_id` number for each user. You will use those numbers to assign desired roles for each user.

The following command will create an `agent` assignment.

```
$ curl \
  --cacert self-signed.crt \
  --header "Content-Type: application/json" \
  --header "X-OpenIDM-Username: openidm-admin" \
  --header "X-OpenIDM-Password: openidm-admin" \
  --request POST \
  --data '{
    "name" : "ldapAgent",
    "description" : "LDAP Agent Assignment",
    "mapping" : "managedUser_systemLdapAccounts",
    "attributes": [
      {
        "name": "ldapGroups",
        "value": [
          "cn=Contractors,ou=Groups,dc=example,dc=com"
        ],
        "assignmentOperation" : "mergeWithTarget",
        "unassignmentOperation" : "removeFromTarget"
      }
    ],
    "linkQualifiers": ["agent"]
  }' \
"https://localhost:8443/openidm/managed/assignment?_action=create"
```

Now repeat the process for the `insured` assignment, with the value set to the `Chat Users` group:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
  --cacert self-signed.crt \
  --header "Content-Type: application/json" \
  --header "X-OpenIDM-Username: openidm-admin" \
  --header "X-OpenIDM-Password: openidm-admin" \
  --request POST \
  --data '{
    "name" : "ldapCustomer",
    "description" : "LDAP Customer Assignment",
    "mapping" : "managedUser_systemLdapAccounts",
    "attributes": [
      {
        "name": "ldapGroups",
        "value": [
          "cn=Chat Users,ou=Groups,dc=example,dc=com"
        ],
        "assignmentOperation" : "mergeWithTarget",
        "unassignmentOperation" : "removeFromTarget"
      }
    ],
    "linkQualifiers": ["insured"]
  }' \
"https://localhost:8443/openidm/managed/assignment?_action=create"
```

Now you can add the created assignments to their respective roles.

Add the `insured` assignment to the insured customer role:

```
$ curl \
--cacert self-signed.crt
 \
--header "Content-type: application/json"
 \
--header "X-OpenIDM-Username: openidm-admin"
 \
--header "X-OpenIDM-Password: openidm-admin"
 \
--header "If-Match: *"
 \
--request PATCH
 \
--data '[
    {
      "operation" : "add",
      "field" : "/assignments/-",
      "value" : {
        "_ref" : "managed/assignment/ee5241b2-e571-4736-8fb2-6b9caa9d0554"
      }
    }
  ]' \
"https://localhost:8443/openidm/managed/role/287dc4b1-4b19-49ec-8b4c-28a6c12ede34"
```

Add the `agent` assignment to the `agent` role:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
--cacert self-signed.crt
 \
--header "Content-type: application/json"
 \
--header "X-OpenIDM-Username: openidm-admin"
 \
--header "X-OpenIDM-Password: openidm-admin"
 \
--header "If-Match: *"
 \
--request PATCH
 \
--data '[
    {
      "operation" : "add",
      "field" : "/assignments/-",
      "value" : {
        "_ref" : "managed/assignment/12927c5d-576f-491e-ba65-e228cd218947"
      }
    }
  ]' \
"https://localhost:8443/openidm/managed/role/bb9302c4-5fc1-462c-8be2-b17c87175d1b"
```

8.8. Reconciling Managed Users to the External LDAP Server

Now that you have loaded `Example.ldif` into OpenDJ, and have started OpenIDM, you can perform a reconciliation from the internal Managed Users repository to the external OpenDJ data store:

```
$ curl \
 --cacert self-signed.crt \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request POST \
 "https://localhost:8443/openidm/recon?_action=recon&mapping=managedUser_systemLdapAccounts"
```

With all of the preparation work that you have done, this reconciliation will create three new accounts on the external LDAP server:

- Two accounts under the `ou=Customers,dc=example,dc=com` branch `dn` under the insured customers role, `bjensen` and `jdoe`.
- One account under the `ou=Contractors,dc=example,dc=com` branch `dn` under the insurance agents role, `jdoe`.

Congratulations, you have just created accounts in two different areas of the LDAP Directory Information Tree.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

8.9. Reviewing the Result

You have already confirmed that user `bjensen` has a `insured` role, and user `jdoe` has both a `insured` and `agent` role. You can confirm the same result in the Admin UI:

1. Click Manage > Role. You should see both Agent and Insured in the Role List window that appears.
2. Click Agent > Users. You should see that user `jdoe` is included as an Agent.
3. Click Back to Roles > Insured > Users. You should see that users `bjensen` and `jdoe` are included in the Insured role.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

Chapter 9

The Trusted Servlet Filter Sample

This sample demonstrates how to use a custom servlet filter and the "Trusted Request Attribute Authentication Module" in OpenIDM. Once configured, OpenIDM can use the servlet filter to authenticate through another service.

If you want to set up authentication through OpenAM, refer to Chapter 10, "*Full Stack Sample - Using OpenIDM in the ForgeRock Identity Platform*".

9.1. Before You Start

Before you start this sample, complete the following steps:

- Prepare a fresh installation of OpenIDM. (See Section 1.3, "Preparing OpenIDM").
- Download and install the Apache Maven build tool.
- Build the custom servlet filter bundle file:

```
$ cd /path/to/openidm/samples/trustedservletfilter/filter
$ mvn clean install
```

- Copy the newly built servlet bundle file to the `openidm/bundle` directory:

```
$ cp target/sample-trusted-servletfilter-1.0.jar /path/to/openidm/bundle
```

9.2. The Sample Servlet Filter

You just built a bundle file from a Java file in the following `trustedservletfilter/filter` subdirectory: `src/main/java/org/forgerock/openidm/sample/trustedservletfilter`. The file is named `SampleTrustedServletFilter.java`.

The following line looks for the `X-Special-Trusted-User` header, to identify a specific User ID as a "trusted" user.

```
final String specialHeader = ((HttpServletRequest) servletRequest).getHeader("X-Special-Trusted-User");
```

The next line sets the special Servlet attribute `X-ForgeRock-AuthenticationId` to this trusted User ID.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
servletRequest.setAttribute("X-ForgeRock-AuthenticationId", specialHeader);
```

The rest of the servlet filter chain continues request processing:

```
filterChain.doFilter(servletRequest, servletResponse);
```

This sample includes a `servletfilter-trust.json` file that calls the compiled OpenIDM trusted servlet `filterClass`:

```
{
   "classPathURLs" : [ ],
   "systemProperties" : { },
   "requestAttributes" : { },
   "scriptExtensions" : { },
   "initParams" : { },
   "urlPatterns" : [
      "/*"
   ],
   "filterClass" : "org.forgerock.openidm.sample.trustedservletfilter.SampleTrustedServletFilter"
}
```

9.3. Run the Sample

Start OpenIDM with the configuration for the trusted filter sample.

```
$ cd /path/to/openidm
$ ./startup.sh -p samples/trustedservletfilter
Executing ./startup.sh...
Using OPENIDM_HOME:   /path/to/openidm
Using PROJECT_HOME:   /path/to/openidm
Using OPENIDM_OPTS:   -Xmx1024m -Xms1024m
Using LOGGING_CONFIG: -Djava.util.logging.config.file=/path/to/openidm/samples/trustedservletfilter/conf/
logging.properties
Using boot properties at /path/to/openidm/samples/trustedservletfilter/conf/boot/boot
.properties
-> OpenIDM ready
```

9.4. Create a Trusted User

In this section, you will create a user, and then apply the special request header `X-Special-Trusted-User` to authenticate that user.

Create user Barbara Jensen in OpenIDM, with `userName` `bjensen`:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
 --cacert self-signed.crt \
 --header "X-OpenIDM-Password: openidm-admin" \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "Content-Type: application/json" \
 --request PUT \
 --data '
   {
      "userName": "bjensen",
      "telephoneNumber": "6669876987",
      "givenName": "Barbara",
      "sn": "Jensen",
      "description": "Example User",
      "mail": "bjensen@example.com",
      "authzRoles" : [
         {
            "_ref" : "repo/internal/role/openidm-authorized"
         }
      ]
   }' \
 "https://localhost:8443/openidm/managed/user/bjensen"
```

Now you can demonstrate the servlet filter by configuring it with the noted special header. Normally, a servlet filter used for authentication does not allow a client to masquerade as any user. This sample demonstrates a basic use of a servlet filter by establishing the authentication ID.

```
$ curl \
 --cacert self-signed.crt \
 --header "X-Special-Trusted-User: bjensen" \
 --request GET \
 "https://localhost:8443/openidm/info/login?_fields=authenticationId,authorization"
```

The output should include a JSON structure with the user's authentication and authorization details. In this case, user `bjensen` is authenticated with the "openidm-authorized" role.

```
{
   "_id" : "",
   "authenticationId" : "bjensen",
   "authorization" : {
      "id" : "bjensen",
      "component" : "managed/user",
      "roles" : [ "openidm-authorized" ]
   }
}
```

9.5. Customizing the Sample for an External System

To customize this sample for an external authentication/authorization system, you need a servlet filter which authenticates against that external system. You may use a third-party supplied filter, or develop your own filter, using the one in this sample as a model.

The filter you use should have at least the following capabilities:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

- Perform REST calls to another system.
- Search through databases.
- Inspect headers related to authentication and authorization requests.

This servlet filter must set the username of the authenticated user in a special request attribute. You need to configure that same attribute name in the `TRUSTED_ATTRIBUTE` authentication module, specifically the value of `authenticationIdAttribute`.

It is helpful if you have a filter that returns an object with the `userRoles` property. If your filter does not support queries using the following parameter:

```
queryOnResource + "/" + authenticationId
```

You will need to provide a security context augmentation script that populates the following authorization properties in the "security" object:

- `security.authorization.component`
- `security.authorization.roles`

The value for the `security.authorization.component` is automatically set to the value specified in any exisitng `queryOnResource` property.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

Chapter 10

Full Stack Sample - Using OpenIDM in the ForgeRock Identity Platform

This sample demonstrates the integration of three ForgeRock products: OpenIDM, OpenDJ, and OpenAM. With this sample, you can see how you can use OpenAM for authentication, for users maintained with OpenIDM, based on a data store of users in OpenDJ.

It may take some time to set up this sample. The instructions that follow describe how you set up OpenDJ with a custom data store, sync that to OpenIDM. You will also configure OpenAM to use that same instance of OpenDJ. When you finish this sample, you will know how make OpenIDM, OpenDJ, and OpenAM work together. When your setup is complete, OpenIDM will authorize, and OpenAM will protect your users.

Now let us get started. In this sample, you will integrate OpenDJ as the data store for both OpenIDM and OpenAM.

OpenAM requires the use of a Fully-Qualified Domain Name (FQDN). For this sample, you may set up FQDNs for OpenAM, OpenDJ, and OpenIDM on either an appropriate DNS server or the `hosts` file for each system.

This sample assumes that you have assigned the following FQDNs to the OpenAM, OpenDJ, and OpenIDM systems, respectively:

- `openam.example.com`
- `opendj.example.com`
- `openidm.example.com`

This sample assumes that you set up OpenIDM in a "two-way" mapping with an instance of OpenDJ, in a fashion similar to Sample 2b or 2c. It also assumes that you configure OpenAM to use the same instance of OpenDJ as its data store.

To prepare this sample, you should first prepare OpenDJ and OpenAM. You will then start and customize OpenIDM configuration files via the Admin UI, or alternatively, via a text editor.

10.1. External OpenDJ Configuration

Configure the OpenDJ server as described in Section 3.1.1, "LDAP Server Configuration".

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

You need to configure the OpenDJ server with write access. This allows you to create users from OpenIDM or OpenAM on the same LDAP server. When you configure the LDAP server, import the LDIF file associated with the Full Stack sample: (`openidm/samples/fullStack/data/Example.ldif.`)

When you configure OpenAM, you need the following information to configure OpenDJ as an external data store:

- Access URL and port for the LDAP server, such as opendj.example.com:1389.
- LDAP Bind DN, normally `cn=Directory Manager`
- LDAP Bind Password, which should match the password configured the LDAP server.
- LDAP SSL/TLS, which assumes that you've configured OpenDJ to communicate over the secure LDAP port, such as 1636.

10.2. OpenAM Server Configuration

This sample assumes that you will configure OpenAM on a separate system from OpenIDM.

Install OpenAM as described in *To Custom Configure OpenAM*.

During the installation process, include the parameters described in Section 10.1, "External OpenDJ Configuration". Alternatively, if you already have an operational instance of OpenAM, set up an external instance of OpenDJ as described in *To Configure a Data Store*.

10.3. Install the Sample

Prepare OpenIDM, as described in Section 1.3, "Preparing OpenIDM", then start OpenIDM with the configuration for the Full Stack Sample.

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

10.4. Configuring OpenIDM for the Full Stack Sample

You will need to change three configuration files, related to authentication and provisioning. These files are

- `fullStack/conf/authentication.json`
- `fullStack/conf/ui-configuration.json`
- `fullStack/conf/provisioner.openicf-ldap.json`

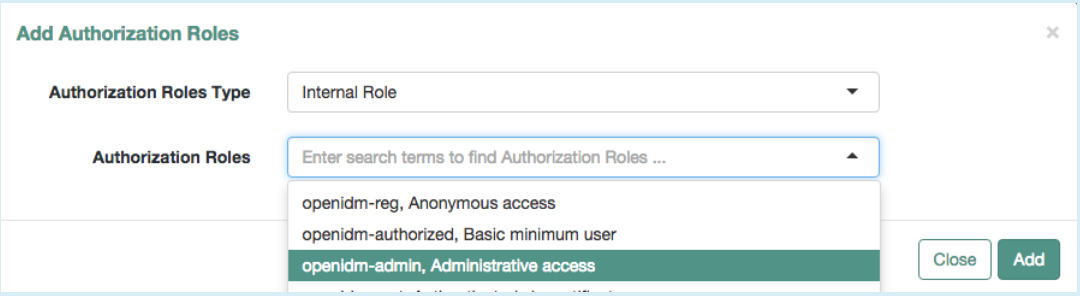
Copyright © 2011-2016 ForgeRock AS. All rights reserved.

You can make these changes on configuration files or via the Admin UI. In the following sections, you will see how to do so using both methods.

Note

Before configuring the `OPENAM_SESSION` module, be sure to configure at least one regular user with the `openidm-admin` authorization role.

In the Admin UI, click Manage > User > select a user > Authorization Roles > Add Authorization Roles.



Making this change ensures that you can still access OpenIDM after you have activated the `OPENAM_SESSION` module.

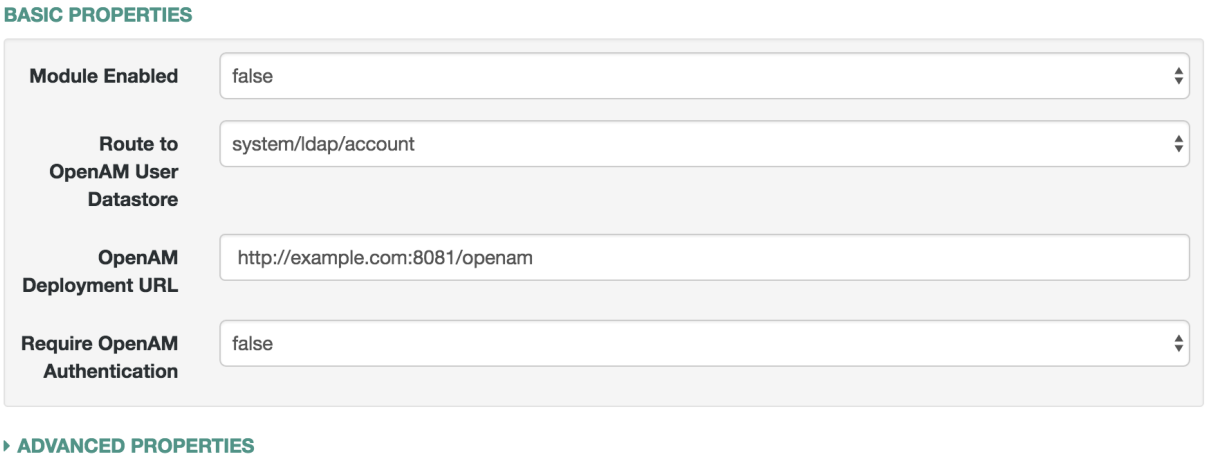
You'll also have to modify the session module as described in the following section: Section 16.1.2.1, "Supported Session Module" in the *Integrator's Guide*. Specifically, you'll need to limit token lifetime properties to 5 seconds.

10.4.1. Configuring the OPENAM_SESSION Module in the UI

Now you can configure the `OPENAM_SESSION` module. To do so, take the following steps:

- Navigate to `https://openidm.example.com:8443/admin`.
- Click Configure > System Preferences > Authentication.
- Scroll down to the OpenAM Session module. Click the Edit icon to view Basic Properties for that module.

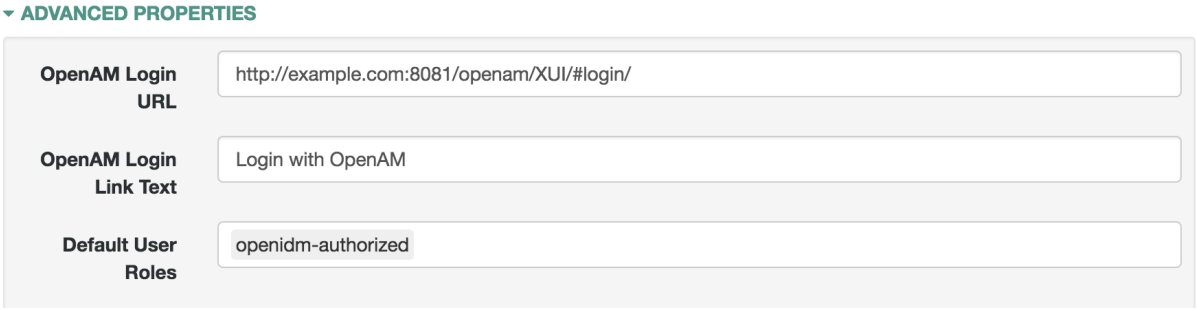
Copyright © 2011-2016 ForgeRock AS. All rights reserved.



- OpenIDM does not use OpenAM for authentication until you set `Module Enabled` to true. Make sure that the `OpenAM Deployment Url` matches the FQDN of your instance of OpenAM, such as `https://openam.example.com:8443/openam`.

  For detailed information on the options associated with the `OPENAM_SESSION` module, see Section I.1, "OPENAM_SESSION Module Configuration Options" in the *Integrator's Guide*.

- Select Advanced Properties to access additional options. Typically, you should only have to modify the `OpenAM Login URL` to match the Login URL of the OpenAM server. If you are working with the OpenAM Top-Level Realm, as described in *Configuring Realms*, and the given FQDN (`openam.example.com`), with an OpenAM web archive file of `openam.war` the following URL should work in this text box: `https://openam.example.com:8443/openam/XUI/#login`



When configured, the Admin UI writes the properties to the `authentication.json` and `ui-configuration.json` files. To identify which property is saved in which file, see Section I.1, "OPENAM_SESSION Module Configuration Options" in the *Integrator's Guide*.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

10.4.2. Configuring the OPENAM_SESSION Module with Configuration Files

To configure the `OPENAM_SESSION` module, you will need to configure two different files: `authentication.json` and `ui-configuration.json`. You can find those files in the `samples/fullStack/conf` directory.

10.4.2.1. Changes to authentication.json

Under `"authModules"`, find the `"OPENAM_SESSION"` authentication module. The default version of the `authentication.json` file includes one entry which you will probably change, to match the URL of your instance of OpenAM:

```
"authModules" : [
...
   {
      "name" : "OPENAM_SESSION",
      "properties" : {
         "openamDeploymentUrl" : "http://example.com:8081/openam",
         "groupRoleMapping" : {
            "openidm-admin" : [
               "cn=idmAdmins,ou=Groups,dc=example,dc=com"
            ]
         },
         "openamSSOTokenCookieName" : "iPlanetDirectoryPro",
         "openamUserAttribute" : "uid",
         "queryOnResource" : "system/ldap/account",
         "propertyMapping" : {
            "authenticationId" : "uid",
            "groupMembership" : "ldapGroups"
         },
         "defaultUserRoles" : [
            "openidm-authorized"
         ],
         "groupComparisonMethod" : "ldap",
         "augmentSecurityContext" : {
            "type" : "text/javascript",
            "file" : "auth/populateAsManagedUser.js"
         },
         "truststoreType" : "&{openidm.truststore.type}",
         "truststoreFile" : "&{openidm.truststore.location}",
         "truststorePassword" : "&{openidm.truststore.password}"
      },
      "enabled" : false
   }
]
```

Based on a standard `openidm-admin` user and a URL for OpenAM of openam.example.com, you would change the first part of the code snippet to:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
"authModules" : [
...
   {
      "name" : "OPENAM_SESSION",
      "properties" : {
         "openamDeploymentUrl" : "https://openam.example.com:8443/openam",
         "groupRoleMapping" : {
            "openidm-admin" : [
               "cn=idmAdmins,ou=Groups,dc=example,dc=com"
         ]
      },
```

Remember to include the configured OpenAM webapps subdirectory, typically `/openam`, in the `"openamDeploymentUrl"`. After the Java EE container used for OpenAM starts, it unpacks a file such as `openam.war` so that you can access it on the `/openam` endpoint.

The `openamDeploymentUrl` shown above assumes that you are using SSL. If you have a signed certificate, you should import that into the OpenIDM truststore file. For more information, see Section 17.1, "Accessing the Security Management Service" in the *Integrator's Guide*.

Look at the `enabled` property. By default the `OPENAM_SESSION` module is disabled, as shown here:

```
"enabled" : false
```

To enable the module, change `false` to `true`.

10.4.2.2. Changes to ui-configuration.json

For the `OPENAM_SESSION` module, you may want to modify some of the properties in the following excerpt of the `ui-configuration.json` configuration file.

```
"defaultNotificationType" : "info",
"openamLoginUrl" : "http://example.com:8081/openam/XUI/#login/",
"openamUseExclusively" : false,
"openamAuthEnabled" : true,
"openamLoginLinkText" : "Login with OpenAM"
```

When `openamAuthEnabled` is true, you should also change the `"openamLoginUrl"` URL to match the login URL of your instance of OpenAM. If you want users to connect, securely, to the openam.example.com FQDN, on the top-level OpenAM realm, change this property to `"https://openam.example.com:8443/openam/XUI/#/login/"`.

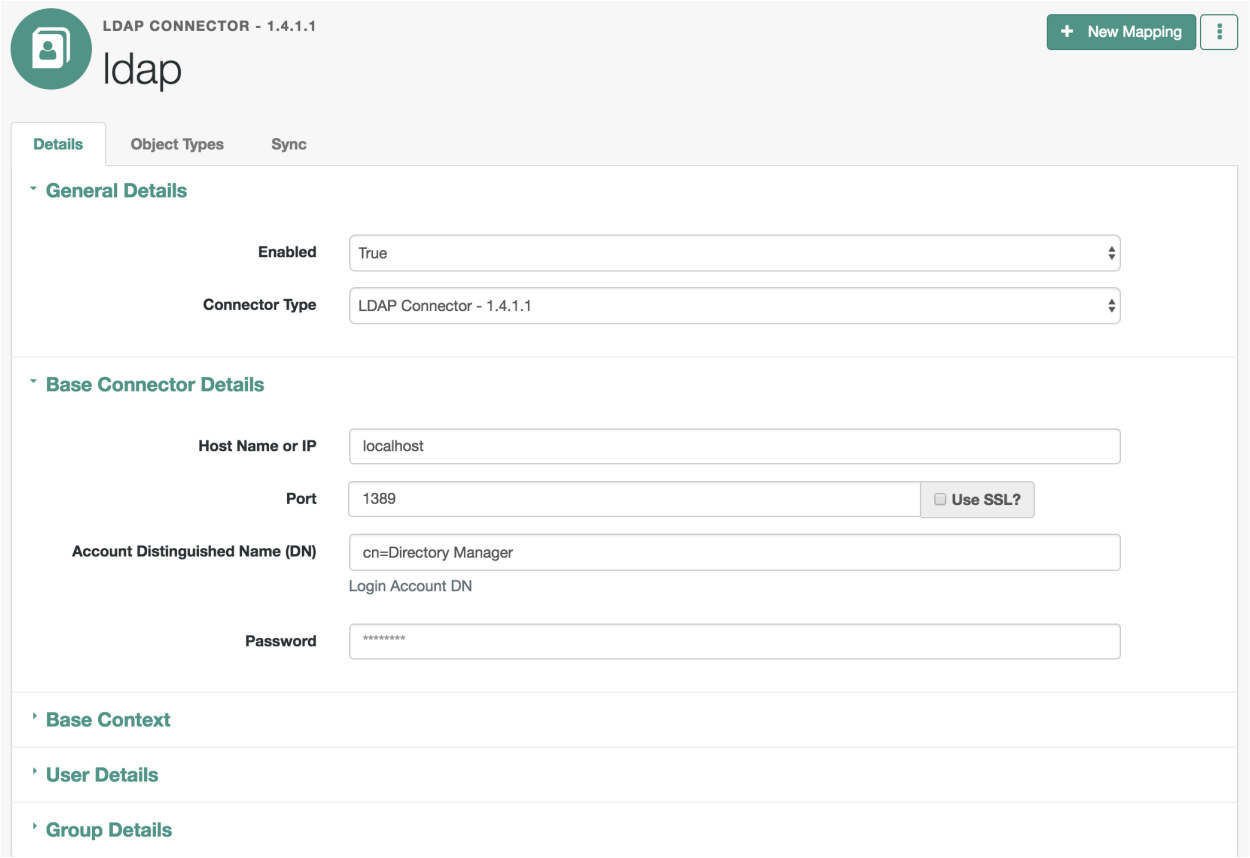
For details on each of these properties, see Section I.1, "OPENAM_SESSION Module Configuration Options" in the *Integrator's Guide*.

10.4.3. Configure Provisioning for the Full Stack Sample

This section describes how you might customize the `provisioner.openicf-ldap.json` file.

If you want to configure this provisioner from the Admin UI, navigate to `https://openidm.example.com:8443/admin`, and edit the LDAP connector.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.



Edit connector details as required. For consistency, with OpenAM requirements, change the `Host name or IP` to match the FQDN of your configured instance of OpenDJ, opendj.example.com. Be consistent with the `Port` number; if you set this port to 1389, configure OpenDJ to communicate over the same port. Configure OpenAM to use the same data store.

Open the noted provisioner file from the `samples/fullStack/conf` directory. The default version of this file should look similar to the following:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
"configurationProperties" : {
   "host" : "localhost",
   "port" : 1389,
   "ssl" : false,
   "principal" : "cn=Directory Manager",
   "credentials" : {
      "$crypto" : {
         "value" : {
            "iv" : "XUfvN7eE471b/1MG8bF60g==",
            "data" : "Y4M22LQehQ95MRQTJCmKdw==",
            "cipher" : "AES/CBC/PKCS5Padding",
            "key" : "openidm-sym-default"
         },
      "type" : "x-simple-encryption"
   }
},
"baseContexts" : [
   "dc=example,dc=com"
],
"baseContextsToSynchronize" : [
   "dc=example,dc=com"
],
```

This snippet already matches the noted base context of `"dc=example,dc=com"` with a principal of `"cn=Directory Manager"`.

Make sure that the following settings are consistent with the way you have configured OpenDJ and OpenAM.

Change the `"localhost"` entry to the FQDN where OpenDJ is installed. In this case, that FQDN is `opendj.example.com`. Depending on whether you want to set up communications over a regular or secure LDAP port, you might change the `"port"` number to to something like 1389 or 1636. The following excerpt illustrates the change to an LDAP connector configuration:

```
"configurationProperties" : {
   "host" : "opendj.example.com",
   "port" : 1389,
   "ssl" : false,
   "principal" : "cn=Directory Manager",
   "credentials" : "password",
      "$crypto" : {
         "value" : {
            "iv" : "XUfvN7eE371b/1MG8bF60g==",
            "data" : "Y4M77LQehQ95MRQTJCmKdw==",
            "cipher" : "AES/CBC/PKCS5Padding",
            "key" : "openidm-sym-default"
         },
         "type" : "x-simple-encryption"
      }
   },
   "baseContexts" : [
      "dc=example,dc=com"
   ],
   "baseContextsToSynchronize" : [
      "dc=example,dc=com"
   ],
...
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

If you want to configure secure communications between OpenIDM and OpenDJ, do remember to configure OpenDJ to communicate securely.

When configuring OpenDJ, you may have a self-signed certificate. You may even have a certificate from a Certificate Authority. In either case, import that OpenDJ certificate into the OpenIDM truststore. For more information, see Section 17.1, "Accessing the Security Management Service" in the *Integrator's Guide*.

10.5. Run the Sample

The mapping configuration file (`sync.json`) for this sample includes two mappings, `systemLdapAccounts_managedUser`, which synchronizes users from the source LDAP server with the target OpenIDM repository, and `managedUser_systemLdapAccounts`, which synchronizes changes from the OpenIDM repository to the LDAP server.

You can run this part of the sample by using the `curl` command-line utility, or by using the OpenIDM Administration UI.

This part of the operation is identical to that shown in the relevant part of Sample 2b, Section 3.2.3, "Run the Sample".

After you complete the steps described in Sample 2b, do one more thing. Navigate to the Admin UI at `https://openidm.example.com:8443/admin`, and select Manage > User. Select a specific user, such as `bjensen`. Click Change Password and change that password.

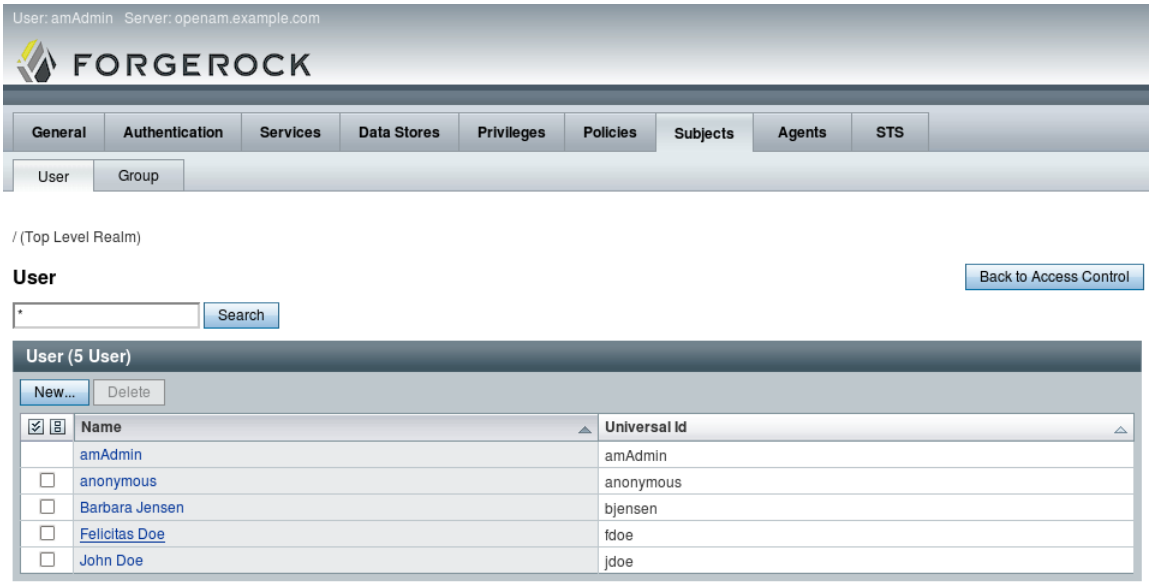
Return to the Admin UI, select Mappings, and select the managedUser_systemLdapAccounts mapping. Click Reconcile Now to propagate the password you just changed to OpenDJ. You are now ready for the next section.

10.6. Verify the Sample on OpenAM

When you reconciled data stores for this OpenIDM sample, you should see the standard users for this sample reconciled into the OpenAM Data store.

In OpenAM, access the list of users. Navigate to OpenAM at `https://openam.example.com:8443/openam`. Log in with the administrative account, which by default is `amadmin`. Navigate to Access Control > Realm > Subjects > User. You should see the same users as you see in the OpenIDM Self-Service UI.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.



Log out of OpenAM.

Return to OpenIDM. In the login window that appears, click Login with OpenAM.

You should be redirected to to the OpenAM login screen at `https://openam.example.com:8443/openam`.

Login as user `bjensen`, with the password that you just changed in OpenIDM. If successful, you should now be logged into the *OpenIDM* Self-Service UI screen, as user `bjensen`.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

Chapter 11

Workflow Samples

This chapter walks you through the workflow provisioning sample (in the `openidm/samples/workflow` directory) and the workflow use cases (in the `openidm/samples/usecase` directories). For a complete list of the samples provided with OpenIDM, and an overview of each sample, see Chapter 1, "*Overview of the OpenIDM Samples*".

11.1. Sample Workflow - Provisioning User Accounts

This sample, provided in `openidm/samples/workflow`, demonstrates a typical use case of a workflow — provisioning new users.

The sample demonstrates the use of the Admin UI, to configure user self-service and the Self-Service UI that enables users to complete their registration process.

This sample simulates the following scenario:

- An existing employee requests that an outside contractor be granted access to an organization's system.
- The *system* in this case, is OpenIDM's managed user repository and a remote datasource, represented by an XML file.
- User roles are stored separately, in a CSV file.

The sample has three mappings — two for the bidirectional synchronization of the managed user repository and the XML data store, and one for the synchronization of the roles data (stored in the CSV file) to the managed repository.

11.1.1. Prepare OpenIDM For the Provisioning Sample

In this section, you start OpenIDM, configure the outbound email service, and reconcile user and role data. The reconciliation operations create two managed users, `user1` and `manager1`, and two managed roles, `employee` (assigned to `user1`) and `manager` (assigned to `manager1`).

1. Start OpenIDM with the configuration for the provisioning sample.

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

2. Log in to the Admin UI (`https://localhost:8443/admin`) with the default username (`openidm-admin` and password (`openidm-admin`).

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

3. Configure the outbound email service.

   An email configuration is required to send a password-reset email to the user, at the end of the sample.

   Select Configure > System Preferences > Email and provide connection information for your preferred email host.

   Gmail is configured by default but you must still supply credentials to use Gmail.

4. Reconcile the role data, and the user data.

   a. Select Configure > Mappings.

   b. Click the first mapping (`systemRolesFileRole_managedRole`) and click Reconcile Now.

      This reconciliation operation creates two roles in the managed repository (`employee` and `manager`). You can check the result of the reconciliation by selecting Manage > Role.

   c. Go back to the Mappings page (Configure > Mappings), select the second mapping (`systemXmlfileAccounts_managedUser`) and click Reconcile Now.

      This reconciliation operation creates the top-level managers (users who do not have their own `manager` property) in the managed user repository. In this sample, there is only one top-level manager (`manager1`).

   d. Click Reconcile Now a second time.

      This reconciliation operation creates the employees of the managers that were created by the previous reconciliation. In this sample, there is only one employee (`employee1`).

      Check that the manager and employee entries were created correctly by selecting Manage > User. You should have two users — `manager1` and `user1`.

5. Verify the relationships between your new user and role objects:

   a. Click on `user1`. Note that the `Manager` field shows `manager1` for this user.

   b. On the Authorization Roles tab, note that user1 has two roles — `employee` and `openidm-authorized`.

   c. Click Back to User List then click on `manager1`. Note that the `Manager` field is empty for this user.

   d. On the Authorization Roles tab, note that manager1 has two roles — `manager` and `openidm-authorized`.

6. Verify the available workflows:

   a. Select Manage > Processes > Definitions.

   b. Click the Contractor onboarding process and look at the sequence diagram for this workflow.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

7. Log out of the Admin UI by selecting Log Out from the top right dropdown menu.

OpenIDM is now prepared for the Provisioning sample. In the next section you will walk through the workflow whose sequence diagram you saw in the previous step.

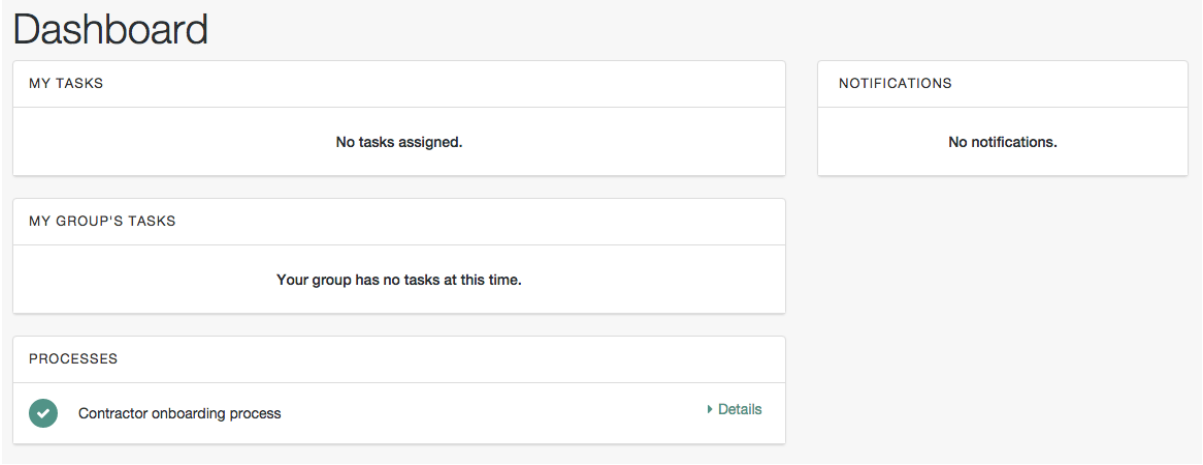
11.1.2. Running the Provisioning Sample

As part of provisioning, employees are required to initiate a *Contractor Onboarding* process. This process is a request to add a contractor to the managed user repository, with an option to include the contractor in the original data source (the XML file).

When the employee has completed the required form, the request is sent to the manager for approval. Any user with the role `manager` can claim the approval task. If the request is approved, an email is sent to the address provided in the initial form, with a request for the contractor to reset their password. When the password reset has been completed, the contractor is created in the managed user repository. If a request was made to add the contractor to the original data source (the XML file) this is done when the manager approves the request.

1. Log in to the Self-Service UI (`https://localhost:8443/`) as the user you created in the previous section (`user1`), with password `Welcome1`.

   The user Dashboard is displayed:



2. Initiate the provisioning workflow as user1:

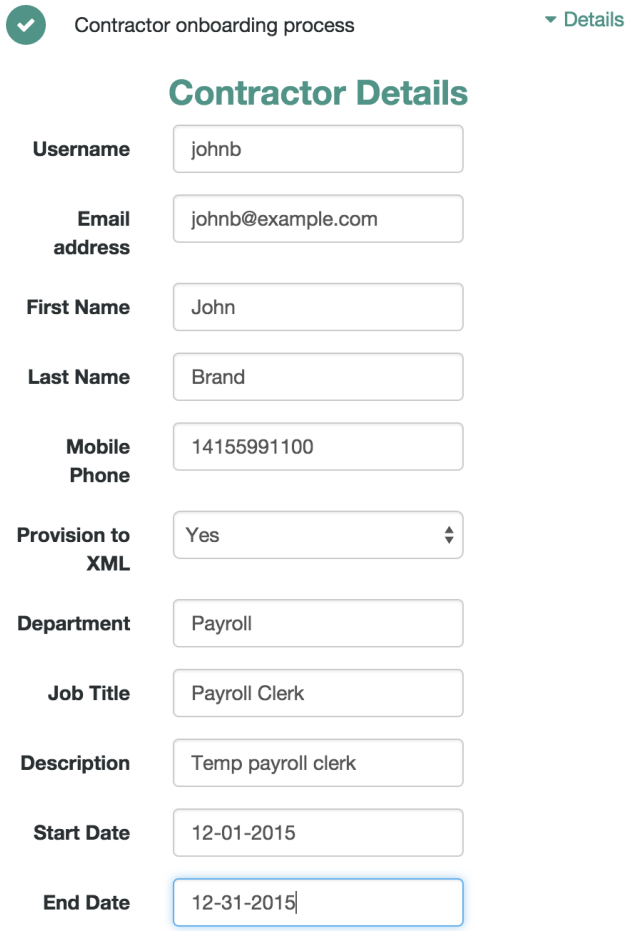
   a. Under Processes, click the Details link next to the Contractor onboarding process and complete the form for the sample user you will be creating.

      Use an email address that you have access to because you will need the email that is sent to this address to complete the workflow.

      In the Provision to XML field, select Yes.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

This selection enables implicit synchronization from the managed user repository to the XML file.



Note that user1 does not provide the password for this user. This is to ensure that only the actual contractor can log in with this account

b. Click Start to initiate the process.

c. Log out of the Self-Service UI.

3. Approve the workflow task as manager1:

a. Log in to the Self-Service UI as `manager1`, with password `Welcome1`.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

b. Under My Group's Tasks, locate the Approve Contractor task and select Assign to Me.

c. Approve Contractor is now listed under My Tasks.

   Click the Details link.

d. Check the form content. (It is the same content that you provided as `user1`, along with a Decision field.)

   Select Accept and click Complete to finish the task.

e. Log out of the Self-Service UI.

4. Verify that the contractor has been created in the XML file:

   Open `openidm/samples/workflow/data/xmlConnectorData.xml` and note the addition of the new contractor entry. Note that there is no value for `<icf:__PASSWORD__/>`. Note that `user1` is the contractor's manager.

   The following excerpt of the `xmlConnectorData.xml` shows a new user entry for bjensen, after the implicit synchronization to the XML file:

```
<ri:__ACCOUNT__>
...
    <ri:roles>openidm-authorized</ri:roles>
    <ri:firstname>Barbara</ri:firstname>
    <ri:manager>user1</ri:manager>
    <icf:__UID__>a02536cf-84b2-4d5c-a3ae-480f5e0899e9</icf:__UID__>
    <icf:__NAME__>bjensen</icf:__NAME__>
    <ri:email>bjensen@example.com</ri:email>
    <icf:__PASSWORD__/>
    <ri:lastname>Jensen</ri:lastname>
</ri:__ACCOUNT__>
```

5. Complete the password reset process:

   a. Check the inbox for the email address that you provided when you completed the initial form.

      You should have received an email with the subject "Reset your password".

   b. Open the password reset email and click Password reset link.

      The link takes you to the Self-Service UI, with the option to Reset Your Password.

   c. Enter a new password and confirmation password, submit the form.

      The password that you enter here must comply with the default password policy for managed users, described in Section 15.1, "Enforcing Password Policy" in the *Integrator's Guide*.

   d. Click Return to Login Page and log in with the username that you provided when you completed the initial form, and the new password you have just set.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

Notice the Welcome message under Notifications.

6. Verify that the password reset has been propagated to the XML file:

   Open `openidm/samples/workflow/data/xmlConnectorData.xml` and note that the password for the contractor has been added to their entry.

If you declined the approval request, the user is not created in either data source.

11.2. Workflow Use Cases

This section describes a number of sample workflows, that demonstrate typical use cases for OpenIDM. The use cases, provided in `/path/to/openidm/samples/usecase`, work together to describe a complete business story, with the same set of sample data. Each of the use cases is integrated with the Self-Service UI.

These use cases use OrientDB as a repository by default. Alternative repository configuration files are provided in `/path/to/openidm/samples/usecase/db`. If you want to use one of these alternative repositories, remove the `repo.orientdb.json` file from the `conf/` directory of the use case you are testing (for example, `samples/usecase/usecase1/conf/repo.orientdb.json`) and copy the appropriate JDBC repository configuration files (`datasource.jdbc-default.json` and `repo.jdbc.json`) into that `conf/` directory. For more information on using an alternative repository, see Chapter 2, "*Installing a Repository For Production*" in the *Installation Guide*.

Each use case builds on the previous one. You *must* run the use cases in order, from use case 1 through 3, before you try the remaining use cases. Use cases 2 onwards depend on the `hr_data.ldif` file that you import and reconcile when you run use case 1.

The use cases assume an initial data set of twenty *ordinary* managed users in OpenIDM (user.0 - user.19). The users are divided as follows:

Users	Department	Manager	Employees	Contractors
user.0-user.4	Human Resources	user.0	user.0-user.3	user.4
user.5-user.9	Production Planning	user.5	user.5-user.8	user.9
user.10-user.14	Sales & Distribution	user.10	user.10-user.13	user.14
user.15-user.19	Treasury & Payments	user.15	user.15-user.18	user.19

In addition, the following *special* users are defined:

- `hradmin` - represents the human interaction of the HR department
- `systemadmin` - represents the human interaction of the populated systems (Business and Project)
- `superadmin` - represents the manager of the managers

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

Note

Note that the **curl** commands in this section use the secure port for OpenIDM (8443) and assume a self-signed certificate named `self-signed.crt`, located in the directory from which the command is launched. For instructions on using the self-signed certificate that is generated when OpenIDM first starts up, see Section 17.2.2, "Restrict REST Access to the HTTPS Port" in the *Integrator's Guide*.

11.2.1. Use Case 1 - Initial Reconciliation

This use case assumes an OpenDJ server and populates the managed user repository with users from OpenDJ.

To set up the sample, install and configure OpenDJ, as follows:

1. Download and install OpenDJ, as described in To Install OpenDJ Directory Server With the GUI.

   This sample assumes that OpenDJ is listening on port 1389, the standard LDAP port for users who cannot use privileged ports.

2. The use case assumes a user with DN `cn=Directory Manager` and password `password` who will bind to the directory server.

3. During the install, import the user data from the LDIF file `/path/to/openidm/samples/usecase/data/hr_data.ldif`.

   The OpenDJ server now contains the users required for all the workflow use cases.

Procedure 11.1. Running Use Case 1

1. Start OpenIDM with the configuration for use case 1.

   ```
   $ cd /path/to/openidm
   $ ./startup.sh -p samples/usecase/usecase1
   ```

2. Run reconciliation to populate the managed user repository with the users from the OpenDJ server.

   The validation rules in this workflow require a managed user's *manager* entry to exist before that user can be created. You must therefore run three consecutive reconciliation operations:

   - The first reconciliation creates the `superadmin` user. This user has no manager entry, so is unaffected by the validation rules. Creation fails for the remaining users, because their manager does not yet exist.

   - The second reconciliation creates the 12 users who have the `superadmin` user as their manager. Creation fails for the remaining users, because they require the 12 manager users to exist before they can be created.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

- The third reconciliation creates the remaining 10 users, bringing the total number of users to 23.

The easiest way to run reconciliation is from the Admin UI:

1. Log in to the Admin UI (`https://localhost:8443/admin`) as the administrative user (`openidm-admin`) with password `openidm-admin`.

2. Select Configure > Mappings.

3. Click on the only configured mapping (`systemHRAccounts_managedUser`) and click Reconcile Now.

To run reconciliation from the command line, use the following command:

```
$ curl \
 --cacert self-signed.crt \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request POST \
 "https://localhost:8443/openidm/recon?_action=recon&mapping=systemHRAccounts_managedUser"
{
 "_id": "376b3290-24f0-47a9-8a9e-bba025536c39",
 "state": "ACTIVE"
 }
```

3. Run reconciliation twice more to create all the users in the repository.

4. Query the managed users that were created by the three reconciliation operations.

   In the Admin UI, select Manage > User and note the new entries in the User List.

   Alternatively, run the following command to view the managed user entries over REST:

```
$ curl \
 --cacert self-signed.crt \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "https://localhost:8443/openidm/managed/user?_queryId=query-all-ids"
{
  "result": [
    {
      "_id": "user.7",
      "_rev": "1"
    },
    {
      "_id": "user.3",
      "_rev": "1"
    },
    {
      "_id": "user.4",
      "_rev": "1"
    },
 ...
    {
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
      "_id": "systemadmin",
      "_rev": "1"
    },
    {
      "_id": "hradmin",
      "_rev": "1"
    },
    {
      "_id": "superadmin",
      "_rev": "1"
    }
  ],
  "resultCount": 23,
  ...
}
```

Your managed user repository should now contain 23 users. The default password of all the newly created users is `Passw0rd`.

5. Shut down OpenIDM before you proceed with the next use case.

```
$ cd /path/to/openidm
$ ./shutdown.sh
```

11.2.2. Use Case 2 - New User Onboarding

This use case demonstrates a new user onboarding process. The process can be initiated by any of the users created in the previous reconciliation process. In this example, we use `user.1` to initiate the process. `user.1` captures the details of a new user, and then submits the new user entry for approval by the prospective manager of that new user.

The use case includes three separate workflows - onboarding (creation of the new user), sunrise (new user start date) and sunset (user end date).

The use case also demonstrates email notification with the configuration of an external email service. You must configure the external email service, as described in Procedure 11.2, "Configuring Email Notification", *before you start the workflow*.

The use case demonstrates the OpenIDM Self-Service UI. If you deploy OpenIDM on the local system, you can access the UI at the following URL: `https://localhost:8443`.

Procedure 11.2. Configuring Email Notification

1. Start OpenIDM with the configuration for use case 2.

```
$ cd /path/to/openidm
$ ./startup.sh -p samples/usecase/usecase2
```

2. Configure the outbound email service.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

Log in to the Admin UI (`https://localhost:8443/admin/`) as the default administrative user (`openidm-admin` with password `openidm-admin`).

Select Configure > System Preferences > Email and provide connection information for your preferred email host.

Gmail is configured by default but you must still supply credentials to use Gmail.

3. Log out of the Admin UI.

4. Change the notification email parameters in the workflow definition file. To edit the workflow definition file:

   1. Copy the workflow archive (`.bar`) file (`newUserCreate.bar`) to a temporary location, such as a `/tmp` directory:

      ```
      $ cd /path/to/openidm/samples/usecase/usecase2/workflow
      $ cp newUserCreate.bar /tmp/
      ```

   2. Unzip the temporary workflow `.bar` file.

      This step extracts the workflow definition file (`newUserCreate.bpmn20.xml`) and two xhtml templates required by the workflow:

      ```
      $ unzip /tmp/newUserCreate.bar
      Archive:  newUserCreate.bar
        inflating: nUCDecideApprovalForm.xhtml
        inflating: nUCStartForm.xhtml
        inflating: newUserCreate.bpmn20.xml
      ```

   3. Edit the extracted workflow definition file (`newUserCreate.bpmn20.xml`). The email parameters are towards the end of this file:

      ```
      $ cd /tmp
      $ grep emailParams newUserCreate.bpmn20.xml
      emailParams = [from : 'usecasetest@forgerock.com', to : 'notification@example.com'
      ,
      ...
      ```

      Change the `from` and `to` parameters to reflect valid email addresses.

   4. Zip up the amended workflow definition file, and the xhtml templates into a workflow `.bar` file.

      ```
      $ zip newUserCreate.bar newUserCreate.bpmn20.xml nUCDecideApprovalForm.xhtml nUCStartForm.xhtml
      updating: nUCDecideApprovalForm.xhtml (deflated 82%)
      updating: nUCStartForm.xhtml (deflated 82%)
      updating: newUserCreate.bpmn20.xml (deflated 85%)
      ```

   5. Copy the new `.bar` file to the workflow directory, overwriting the existing `.bar` file.

      ```
      $ cp /tmp/newUserCreate.bar /path/to/openidm/samples/usecase/usecase2/workflow
      ```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

Procedure 11.3. Initiating the Onboarding Workflow

1. Start OpenIDM with the configuration for use case 2 (if you have not already done so when you configured the email service).

```
$ cd /path/to/openidm
$ ./startup.sh -p samples/usecase/usecase2
```

2. Log in to the Self-Service UI (`https://localhost:8443`) as `user.1` with password `Passw0rd`.

   In this sample, any user who logs into the Self-Service UI can view the new User Onboarding Process workflow.

3. Click Details next to User Onboarding Process link and complete the fields for a sample new user.

   Department. Specifies one of four departments that the new user will belong to (Human Resources, Production Planning, Sales & Distribution, or Treasury & Payments). The value you select here determines the *manager* of the new user, to which the request will be sent for approval. (See the previous table of users for a list of the managers of each department.)

   User Type. Governs user access to specific accounts. If the User Type is Employee, the new user will have access to an account named Business. This access is represented as an attribute of the managed user entry in the OpenIDM repository, as follows: `accounts : ["Business"]`. If the User Type is Contractor, the new user will have no accounts associated with its managed user entry in OpenIDM.

   Send Email Notification. Indicates whether an email should be sent to alert the manager of the new required approval. The email details used here are defined when you configure email notification, as described in Procedure 11.2, "Configuring Email Notification". If you select not to send an email notification, the notification is simply added to the OpenIDM repository, and appears when the manager logs into the UI.

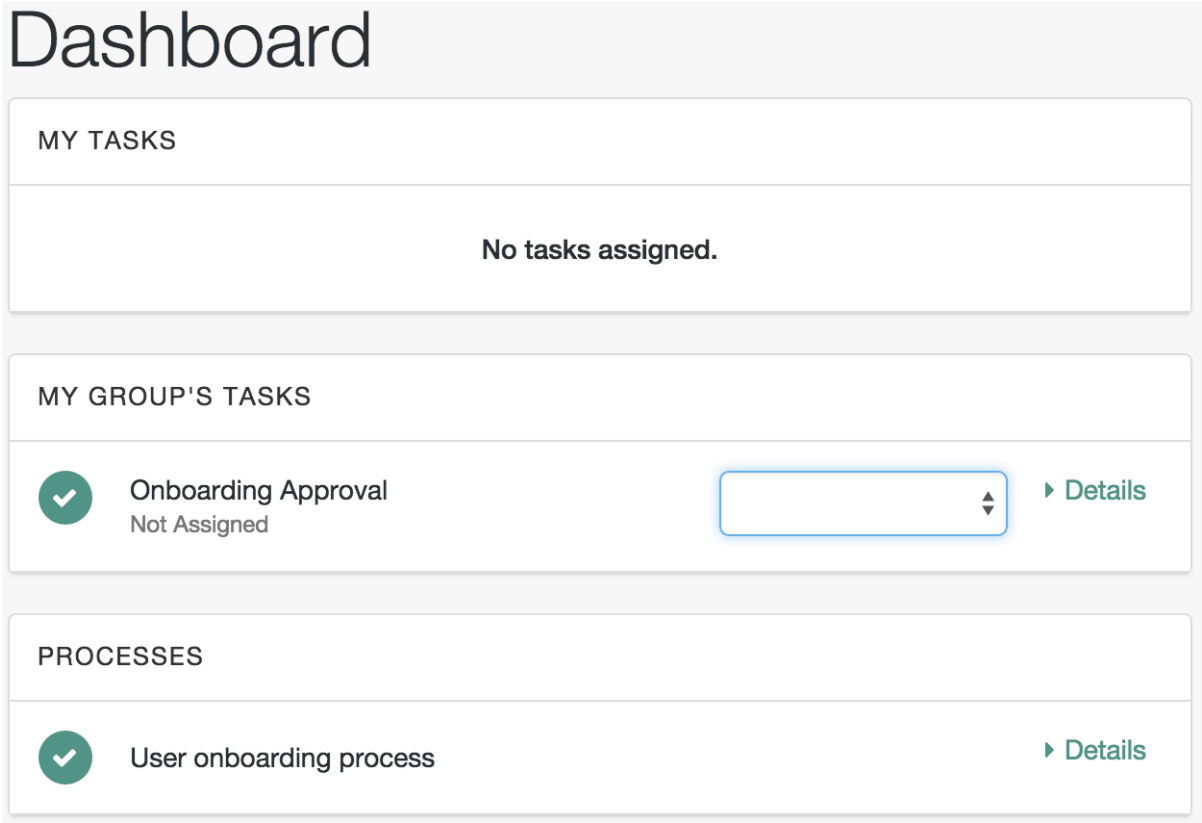
4. Click Start to initiate the onboarding workflow.

   This action sends the new user request to the corresponding *management* users (the department manager, as well as the `superadmin` user, who is an overall manager).

5. Log out of the UI, and log back in as the management user of the department that you selected when you completed the new user form. For example, if you selected Human Resources, log in as `user.0`, which simulates the management user for the HR department. All users have the password `Passw0rd`.

   Notice that the management user now has an Onboarding Approval task in the queue of tasks assigned to that user's group.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.



6. Select Assign to Me from list next to the Onboarding Approval task.

   This action *claims* the task for `user.0`, removes it from the group queue, and places it in the list of pending tasks for `user.0`.

7. Select Details next to the Onboarding Approval task under My Tasks.

   The complete new user request is displayed for the manager's approval. As the manager, you can add any information that was missing from the original request.

   In addition, you can specify the following information for the new user.

   - *Start Date*. Completing this field results in the user being created, with a `startDate` added to that user's managed user entry. The status of the user is `inactive`. This field is optional, and is used by the task scanner to trigger the Sunrise workflow.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

- *End Date*. Completing this field results in the user being created, with an `endDate` added to that user's managed user entry. The field is optional, and is used by the task scanner to trigger the Sunset workflow.
- *Decision*. Selecting Reject here terminates the workflow and sends a notification to the user who initiated the workflow. Selecting Accept creates the managed user entry in OpenIDM. The password of the new user is `Passw0rd`.

  Two notifications are created when the request is accepted - one for the user who initiated the workflow, and one for the newly created user. The notifications are visible in the UI after login. If you selected email notification, one email is sent to the user that you defined when you configured email notification, as described in Procedure 11.2, "Configuring Email Notification".

8. At the bottom of the form, there is an option either to Requeue the request or to Complete it. Click Complete.

   If you click Requeue here, the task is removed from the list of that user's tasks, and returned to the list of tasks pending for that group. The task can then be claimed by any member of that group.

   When the new user request has been approved, the user is created in the OpenIDM repository. If you did not include a Start Date in the manager approval, you should now be able to log into the UI with the details of the new user. If you included a Start Date, you need to complete the sunrise workflow before the user account is active (which will enable you to log in as this user).

Procedure 11.4. Initiating the Sunrise Workflow

If a sunrise date is specified for the new user, the user is created in the repository, with an `inactive` account status.

- To trigger the sunrise workflow (which activates the account), enable the sunrise task scanning schedule. The schedule is disabled by default.

  Modify the schedule configuration file (`schedule-taskscan_sunrise.json`), setting the `enabled` property to `true`.

  ```
  $ cd /path/to/openidm
  $ grep "enabled" samples/usecase/usecase2/conf/schedule-taskscan_sunrise.json
  "enabled" : true,
  ```

  The scan runs every minute, and checks the repository for users that have a sunrise date that is anything up to one day after the current date. When the scan is triggered, it locates the newly created user and starts the sunrise workflow on this user. The workflow takes the following actions:

  - Changes the account status of the user to `active`.

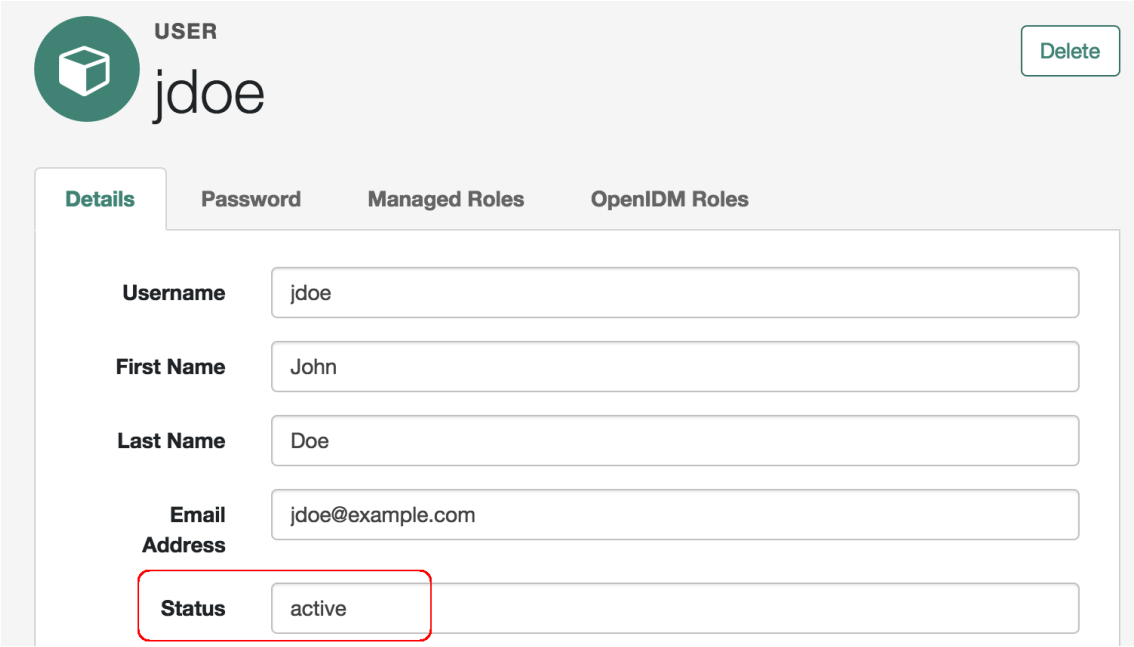
    You can check that this part of the workflow has completed by looking at the user's account in the Admin UI:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

1. Log in to the Admin UI as `openidm-admin` with password `openidm-admin`, then select Manage > User.

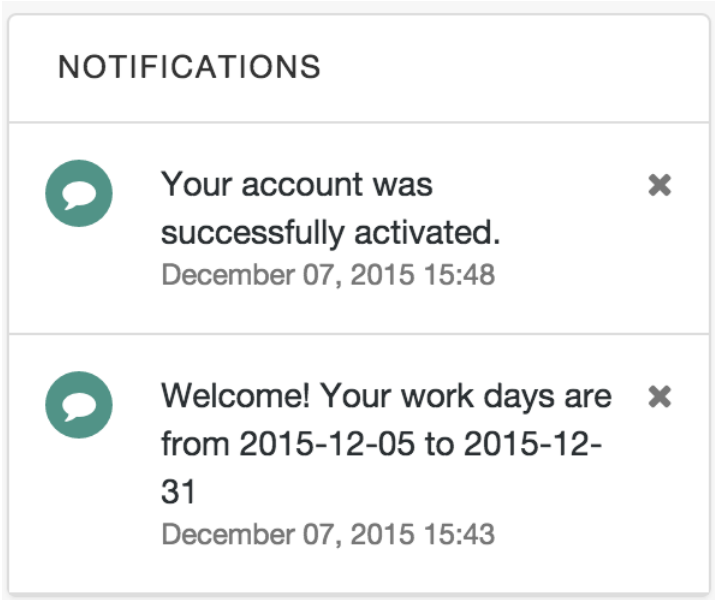
2. Click on the account of the new user that was created in the previous section and note their Status.

   The following image shows the status for the new user jdoe, who was created by the previous workflow:



- Generates a notification for the new user, which is visible when the user logs into the Self-Service UI.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.



Procedure 11.5. Initiating the Sunset Workflow

If a sunset date is set for the new user, you can trigger the sunset workflow to deactivate the user account when the end of his work period is reached.

1. To trigger the sunset workflow, enable the sunset task scanning schedule. The schedule is disabled by default.

   Modify the schedule configuration file (`schedule-taskscan_sunset.json`), setting the `enabled` property to `true`.

   ```
   $ cd /path/to/openidm
   $ grep "enabled" samples/usecase/usecase2/conf/schedule-taskscan_sunset.json

   "enabled" : true,
   ```

   The scan runs every minute, and checks the repository for users that have a sunset date that is anything up to one day after the current date. When the scan is triggered, it locates users whose contracts are about to end, and starts the sunset workflow on these users. When the workflow is initiated, it assigns a task to the manager of the affected user. In our example, the task is assigned to `user.0`.

2. When the sunset schedule has been enabled, log in to the Self-Service UI as `user.0` (with password `Passw0rd`). If the user's sunset date is within one day of the current date, a Contract Termination task becomes available under the manager's My Group's Tasks section.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

Select the contract termination task and click Details.

3. In the Decision field, select either Accept termination or Modify date, then click Complete.

   When you accept the termination, the user's account status is set to `inactive` and the HR administrative user receives a notification to that effect, the next time that user logs into the UI. The deactivated user is no longer able to log into the UI.

   If you select to modify the date, the sunset date of that user is changed to the value that you specify in the End Date field on that form. The management user receives a UI notification that the employee's contract has been extended.

4. Shut down OpenIDM before you proceed with the next use case.

```
$ cd /path/to/openidm
$ ./shutdown.sh
```

11.2.3. Use Case 3 - User Access Request

This use case simulates a user access request, with two levels of approval for the request.

If you want to use email notification with this workflow, start OpenIDM with the configuration for Use Case 3, then follow the instructions in Procedure 11.2, "Configuring Email Notification", substituting and `usecase3/workflow/accessRequest.bpmn20.xml` for the file described in that procedure.

1. Start OpenIDM with the configuration for use case 3, if you have not already started the server to configure email notification.

```
$ cd /path/to/openidm
$ ./startup.sh -p samples/usecase/usecase3
```

2. Log into the Self-Service UI as `user.1` with password `Passw0rd`.

   `user.1` belongs to the HR department and, in this workflow, is requesting access to a Project system.

3. Click Details next to the Access request process and click Start to start the workflow.

   A User Access Request appears in the list of tasks for `user.1`.

4. Click Details next to the User Access Request task.

   The resulting form indicates the various systems to which the user may request access.

   Access to Business system. This field reflects the current value of the `accounts` property for that user in the repository. If the value includes `Business` this field is True.

   Access to Project system. Set this field to True to request Project access for `user.1`.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

Send Email Notification. Set to True to send an email to alert the manager of the new access request. The email details used here are defined when you configure email notification, as described in Procedure 11.2, "Configuring Email Notification". If you select not to send an email notification, the notification appears when the manager logs into the UI.

Select either Cancel, to terminate the process, or Request, to start a user task, assigned to the manager of the user who is requesting access (`user.0` in this example), and select Complete.

5. Log out of the Self-Service UI and log back in as the manager (`user.0` with password `Passw0rd`).

6. Under My Group's Tasks, locate the User Access Request Approval task and select Assign to me.

   Note that the User Access Request Approval task has moved under My Tasks.

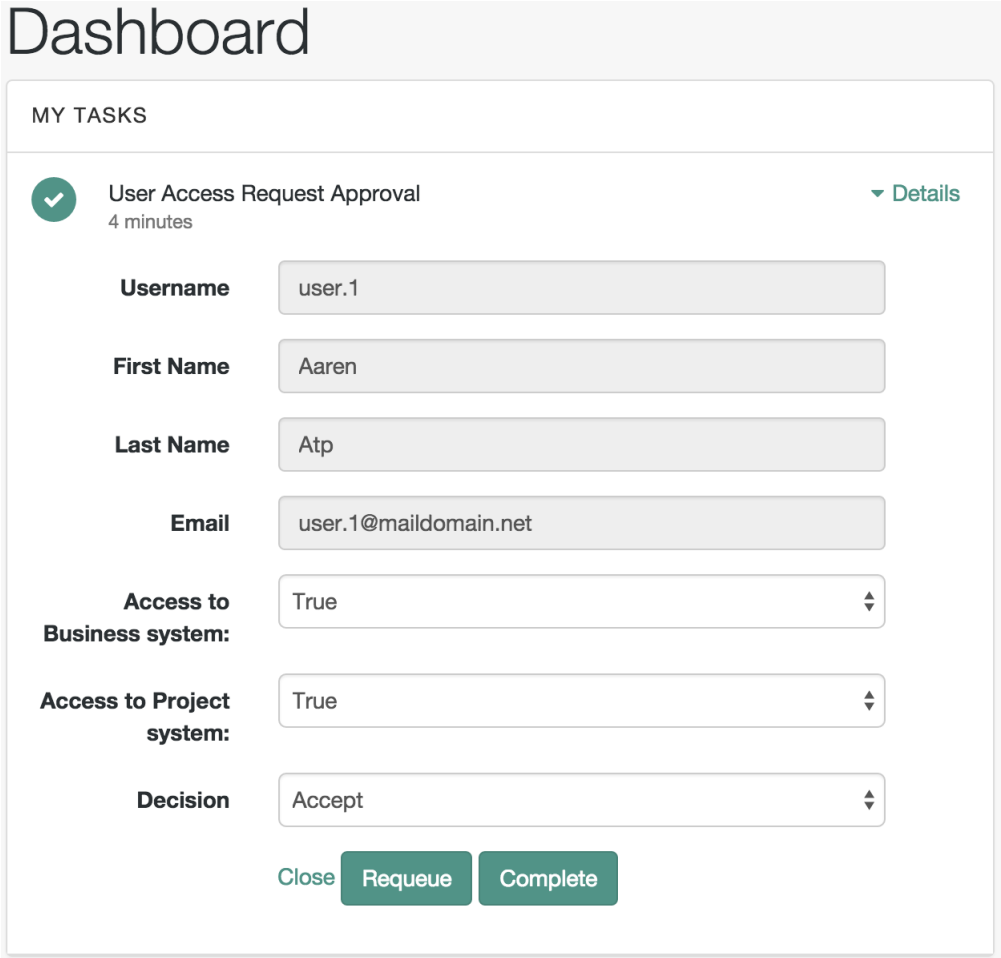
7. Click Details next to the User Access Request Approval task.

8. The details of the access request are displayed. The manager is able to modify the access rights. Select Accept or Reject to approve or deny the request.

   Rejecting the request results in a notification being sent to the user who made the request. If you have enabled email notification, a single email is sent to the account that you specified when you configured email notification.

   Accepting the request initiates a second approval task, assigned to the `systemadmin` user.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.



Click Complete to complete the task.

9. Log out of the UI and log in as the `systemadmin` user (with password `Passw0rd`).

   This user now has one User Access Request Approval task in his queue.

10. Select the task and click Details.

    This task interface is similar to that of the task that was assigned to the manager.

    Rejecting the request results in a notification being sent to the user who made the request.

    Accepting the request updates the managed/user record in OpenIDM, to reflect the approved access changes.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

If you have enabled email notification, a single email is sent to the account defined when you configured the external email service (Procedure 11.2, "Configuring Email Notification"), indicating whether the request has been accepted or rejected.

Note that this sample includes an *escalation* step that is attached to the manager approval task. If the manager does not complete assessment of the user task within ten minutes of its initiation, a new user task is created and assigned to the `superadmin` user. This task has the same interface and functionality as the task assigned to the manager. Accordingly, when the `superadmin` user completes the task, the execution is passed to the `systemadmin` user for approval.

Shut down OpenIDM before you proceed with the next use case.

```
$ cd /path/to/openidm
$ ./shutdown.sh
```

11.2.4. Use Case 4 - Orphan Account Detection

This use case demonstrates two asynchronous tasks, started from a reconciliation process:

- Detecting orphan accounts on a target object set
- Handling ambiguous results during correlation

This use case relies on a customized synchronization configuration (mapping) file, named `syncManagedBusiness.json`, in the `/path/to/openidm/samples/usecase/usecase4/conf` directory.

This file defines a mapping (`recon_managedUser_systemBusiness`) between a source (managed users) and a target object set. The target object set is defined in the file `samples/usecase/usecase4/data/business.csv`. The `business.csv` file includes all users from the initial reconciliation (described in Section 11.2.1, "Use Case 1 - Initial Reconciliation"). These users are categorized as `employees`, and therefore include the property `"accounts" : ["Business"]` in their managed user entry (see Section 11.2.2, "Use Case 2 - New User Onboarding" for an explanation of the User Type).

The mapping includes the following `"validSource"` field:

```
"validSource" : {
    "type" : "text/javascript",
    "file" : "script/isSourceValidBusiness.js"
},
```

This field references a script which specifies that only those users who are employees are taken into account during the reconciliation.

In addition, the `business.csv` file includes the following users:

- `user.50`. This user is defined *only* in the .csv file, and not in the managed/user repository. When a reconciliation operation is run, this user is detected as an *orphan account*. The orphan account workflow is triggered when an "UNQUALIFIED" or "UNASSIGNED" situation is encountered, as indicated in this section of the mapping:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
{
    "situation" : "UNQUALIFIED",
    "action" : {
        "workflowName" : "orphanAccountReport",
        "type" : "text/javascript",
        "file" : "workflow/triggerWorkflowFromSync.js"
    }
},
{
    "situation" : "UNASSIGNED",
    "action" : {
        "workflowName" : "orphanAccountReport",
        "type" : "text/javascript",
        "file" : "workflow/triggerWorkflowFromSync.js"
    }
}
```

- `user.33`. This user has a `"userName"` attribute of `"user.3"` (which is the same as the `"userName"` attribute of the user, `user.3`). The correlation query of the reconciliation operation is based on the `"userName"` attribute. During the correlation query, two candidate users are therefore correlated with the same managed user (`user.3`), and the result is ambiguous. The manual match workflow is triggered when an "AMBIGUOUS" situation is encountered, as indicated in this section of the mapping:

```
{
    "situation" : "AMBIGUOUS",
    "action" : {
        "workflowName" : "manualMatch",
        "type" : "text/javascript",
        "file" : "workflow/triggerWorkflowFromSync.js"
    }
}
```

1. Before you start with this use case, rename the mapping file to `sync.json`.

   ```
   $ cd /path/to/openidm/samples/usecase/usecase4/conf
   $ mv syncManagedBusiness.json sync.json
   ```

2. Start OpenIDM with the configuration for use case 4.

   ```
   $ cd /path/to/openidm
   $ ./startup.sh -p samples/usecase/usecase4
   ```

3. Reconcile the managed user repository with the CSV file, either by using the Admin UI or over the command line.

   To use the Admin UI, log in to the Admin UI (`https://localhost:8443/admin/`) as `openidm-admin` with password `openidm-admin`. Select Configure > Mappings, click on the mapping `recon_managedUser_systemBusiness` and click Reconcile Now.

   To use the command line, run the following command:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
 --cacert self-signed.crt \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request POST \
 "https://localhost:8443/openidm/recon?_action=recon&mapping=recon_managedUser_systemBusiness"
```

When the reconciliation operation finds the ambiguous entry (`user.3`) and the orphan entry (`user.50`) in the CSV file, two asynchronous workflows are launched (`manualMatch` and `orphanAccountReport`), as indicated in the mapping file, described previously.

4. Log in to the Self-Service UI (`https://localhost:8443`) as the `systemadmin` user, with password `Passw0rd`.

5. Next to the Manual Linking Task in the My Tasks list, click Details.

   The *Possible targets* field presents a list of target entries to which the ambiguous record can be linked. In this example, `user.3 - Atrc, Aaron` and `user.33 - Atrc, Aaron` are the two candidate users found in the target object set by the correlation query. When you select one of these values, the workflow manually links the managed user (`user.3`) to the selected user.

   Click complete to finish the manual account linking task.

   If you select Ignore, here, no action is taken (no link is created), and the workflow terminates.

6. Next to the Orphan Account Task in the My Tasks list, click Details.

   The *Link to* field enables you to enter an existing managed user ID to which this orphan account should be linked. For the purposes of this example, enter `user.5`.

   In the Decision field, select Link to link the orphan account to the ID that you entered in the previous step. Click Complete to complete the task.

   Selecting Delete here deletes the user from the target object set (the CSV file in this case) and terminates the workflow.

7. Shut down OpenIDM before you proceed with the next use case.

```
$ cd /path/to/openidm
$ ./shutdown.sh
```

Note

Use Case 5 has been removed from the sample use cases.

11.2.5. Use Case 6 - Password Change Reminder

This use case demonstrates using the task scanner to trigger a password change reminder workflow for managed users.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

In this example, each managed user entry in OpenIDM has a dedicated attribute, `lastPasswordSet`, that stores the date on which the password was last changed. The value of this attribute is updated by an `onStore` script, defined in the managed user configuration file (`conf/managed.json`), as follows:

```
"onStore" : {
    "type" : "text/javascript",
    "file" : "script/onStoreManagedUser.js"
},
```

When a new password is stored for a user, the script sets the date on which this change was made. The task scanner periodically scans the `lastPasswordSet` attribute, and starts the workflow if the password was changed more than an hour ago. This condition is configured in the schedule configuration file (`schedule-taskscan_passwordchange.json`):

```
$ cd /path/to/openidm
$ more samples/usecase/usecase6/conf/schedule-taskscan_passwordchange.json

...
"condition" : {
    "before" : "${Time.now - 1h}"
}
,
....
```

Obviously, in a real deployment, the period between required password changes would be longer, and this value would need to be set accordingly. For the purposes of testing this use case, you might want to set the value to a shorter period, such as `"${Time.now - 1m}"`, which will send the notification one minute after a password change.

By default, the workflow sends notifications to the user entry, visible when the user logs into the UI. If you want notifications sent by email, configure the external email service, as follows:

1. Set up outbound email, as described in Procedure 11.2, "Configuring Email Notification".

2. Enable email notification in the script file that starts the workflow (`samples/usecase/usecase6/script/passwordchange.js`). For example:

   ```
   $ cd /path/to/openidm
   $ more samples/usecase/usecase6/script/passwordchange.js

   /*global  objectID*/

   (function () {
       var params = {
    "userId" : objectID,
    "emailEnabled" : "true",
    "_key": "passwordChangeReminder"
   };
   ```

3. Make sure that all managed users have a valid email address as the value of their `mail` attribute.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

The task scanning schedule is disabled by default. To test this use case, follow these steps:

1. Enable the task scanning schedule by setting `enabled` to `true` in the schedule configuration file (`schedule-taskscan_passwordchange.json`).

```
$ cd /path/to/openidm
$ more samples/usecase/usecase6/conf/schedule-taskscan_passwordchange.json
{
    "enabled" : true
,
...
```

2. Start OpenIDM with the configuration for use case 6.

```
$ cd /path/to/openidm
$ ./startup.sh -p samples/usecase/usecase6
```

3. Log in to the Self-Service UI as any of the users listed in the introduction to this section (for example, `user.4`, with password `Passw0rd`).

   You will see a password expiration notice.

   If you ignore the change for five minutes, you will see a second warning, that the password is about to expire.

   If you ignore that second warning for another two minutes, the user's account is deactivated.

4. (Optional) To avoid the second notification, or the account deactivation, you can change the user password through the UI, as follows:

   a. Log in to the UI as the user whose password you want to change and click Change Password at the top right of the page.

   b. Enter a new password that conforms to the requirements of the password policy.

   c. Enter the old password (in this case `Passw0rd`).

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

Chapter 12

Google Sample - Connecting to Google With the Google Apps Connector

OpenICF provides a Google Apps Connector that enables you to interact with Google's web applications.

Note

The Google Apps Connector, and this corresponding sample, are provided only with the OpenIDM Enterprise build, available on the ForgeRock Backstage site.

This sample demonstrates the creation of users and groups on an external Google system, using OpenIDM's REST interface. The sample requires that you have a Google Apps account. Obtaining a Google Apps account is described in the Google documentation.

12.1. Before You Start

To set up OpenIDM to connect to your Google Apps account, you must have a Google Apps project (or create a new project) that authorizes consent for OpenIDM.

1. Log in to the Google Apps Developers Console (at https://console.developers.google.com/start) and update your project or create a new project for OpenIDM.
2. Enable the following APIs for your OpenIDM project:
   - Admin SDK API
   - Enterprise License Manager API
3. Set up an OAuth2 Client ID.

   The Google Apps connector uses OAuth2 to authorize the connection to the Google service. Set up an OAuth2 Client ID as follows:

   a. In the Google Apps Developers Console, select Credentials > New Credentials > OAuth Client ID.

   b. Click Configure Consent Screen and enter a Product Name.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

This is the name that will be shown for all applications registered in this project.

For the purposes of this example, we use the Product Name `OpenIDM`.

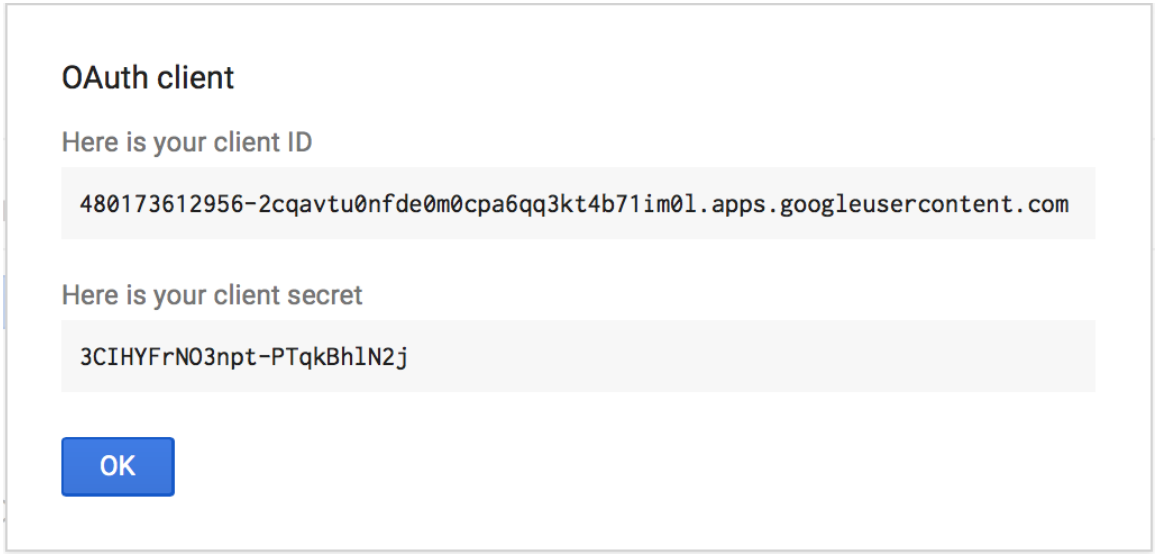
Click Save.

c. Select Credentials > OAuth Client ID > Web application.

Under Authorized redirect URIs, enter the callback URL (the URL at which your clients will access your application). The default OpenIDM callback URL is `https://localhost:8443/admin/oauth.html`. Click Create to set up the callback URL.

Click Create again to set up the client ID.

This step generates an OAuth Client ID and Client, similar to the following:



Copy and paste these values into a text file as you will need them when you configure the Google Apps connector.

12.2. Configuring the Google Apps Connector

This procedure uses the OpenIDM Admin UI to set up the Google Apps connector.

1. To configure the connector, start OpenIDM with the Google Apps sample configuration:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ cd /path/to/openidm
$ ./startup.sh -p samples/google-connector
Executing ./startup.sh...
Using OPENIDM_HOME:   /path/to/openidm
Using PROJECT_HOME:   /path/to/openidm/samples/google-connector/
Using OPENIDM_OPTS:   -Xmx1024m -Xms1024m
Using LOGGING_CONFIG: -Djava.util.logging.config.file=/path/to/openidm/conf/logging.properties
Using boot properties at /path/to/openidm/samples/google-connector/conf/boot/boot
.properties
-> OpenIDM ready
```

2. Log in to the Admin UI at the URL `https://localhost:8443/admin` as the default administrative user (`openidm-admin`) with password `openidm-admin`.

   This URL reflects the host on which OpenIDM is installed and corresponds to the callback URL that you specified in the previous section. The URL must be included in the list of Authorized redirect URIs for your project.

3. Select Configure > Connectors and click on the Google Apps connector.

4. On the Details tab, set the Enabled field to True.

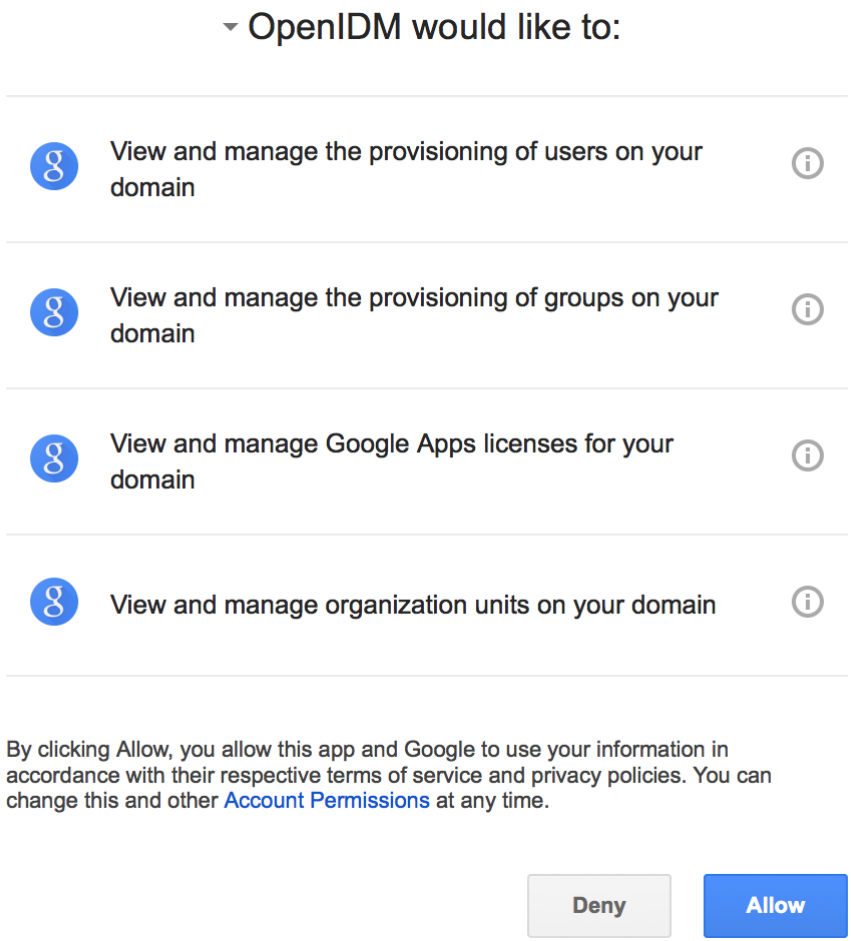
5. Enter the Oauth2 Client ID and Client Secret that you obtained in the previous section.

6. Click Save Connector Changes.

7. You are redirected to Google's Login page.

   When you have logged in, Google requests that you allow access from your project, in this case, OpenIDM.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

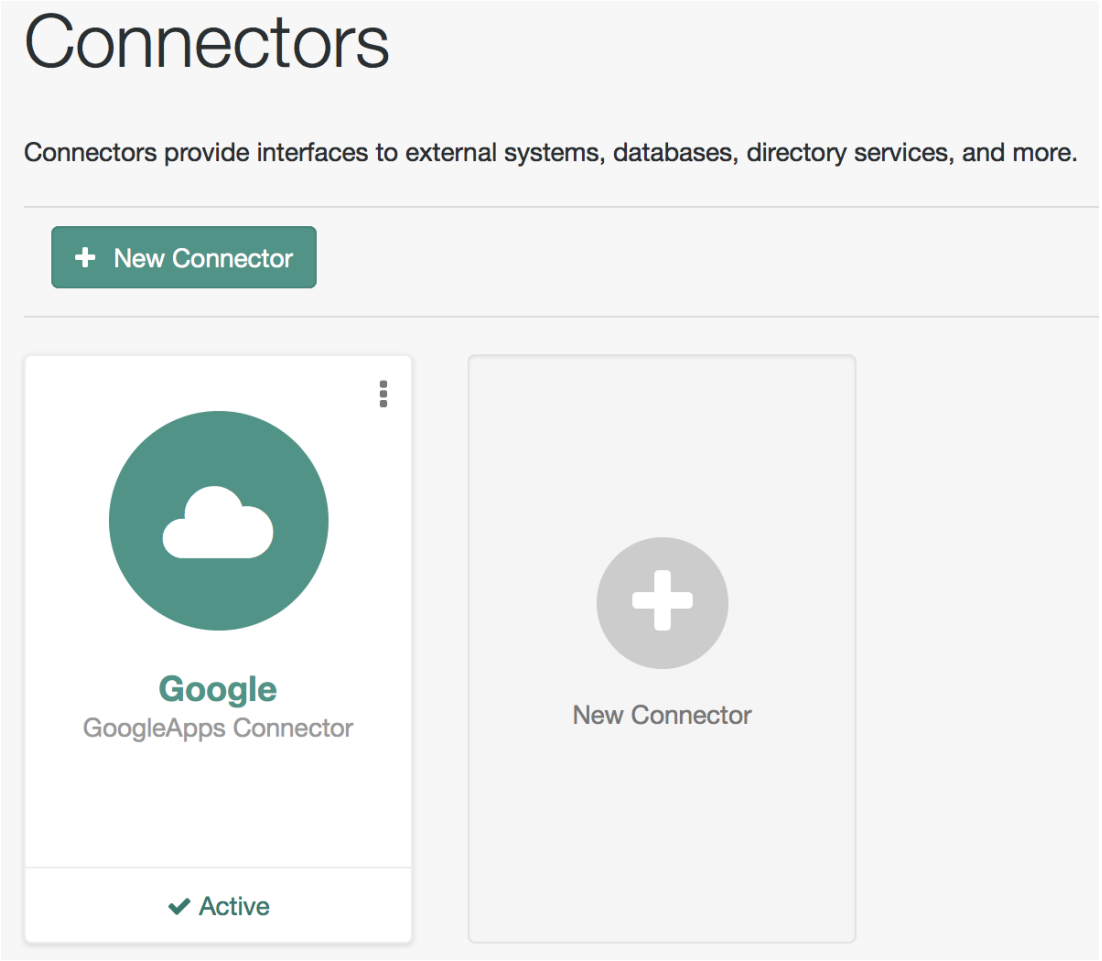


Click Allow.

If you click Deny here, you will need to return to the Connector Configuration > Details tab in the Admin UI and save your changes again.

When you allow access, you are redirected to the Connectors page in the OpenIDM Admin UI, where the Google Apps Connector should now be Active.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.



12.3. Running the Google Apps Sample

This procedure uses create, read, update, and delete (CRUD) operations to the Google resource, to verify that the connector is working as expected. The procedure uses a combination of REST commands, to manage objects on the Google system, and the Admin UI, to manage reconciliation from the Google system to the manage user repository.

The sample configuration has one mapping *from* the Google system *to* the managed user repository.

All of the commands shown here assume that your domain is `example.com`. Adjust the examples to manage your domain.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

1. Create a user entry on your Google resource, over REST.

   When you create resources for Google, note that the equals (`=`) character cannot be used in any attribute value.

   The following command creates an entry for user `Sam Carter`:

```
$  curl \
 --cacert self-signed.crt \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --header "Content-Type: application/json" \
 --request POST \
 --data '{
  "__NAME__": "samcarter@example.com",
  "__PASSWORD__"  : "password",
  "givenName" : "Sam",
  "familyName": "Carter",
  "agreedToTerms": true,
  "changePasswordAtNextLogin" : false
 }' \
 "https://localhost:8443/openidm/system/google/__ACCOUNT__?_action=create"
{
  "_id": "10356743525525123355 1",
  "_rev": "\"iwpzoDgSq9BJw-XzORg0bILYPVc/LWHPMXXG8M0cjQAPITM95Y636cM\"",
  "orgUnitPath": "/",
  "isAdmin": false,
  "fullName": "Sam Carter",
  "customerId": "C02rsqddz",
  "relations": null,
  "nonEditableAliases": null,
  "suspensionReason": null,
  "includeInGlobalAddressList": true,
  "givenName": "Sam",
  "addresses": null,
  "isDelegatedAdmin": false,
  "changePasswordAtNextLogin": false,
  "isMailboxSetup": true,
  "__NAME__": "samcarter@example.com",
  "agreedToTerms": true,
  "externalIds": null,
  "ipWhitelisted": false,
  "aliases": null,
  "lastLoginTime": [
    "1970-01-01T00:00:00.000Z"
  ],
  "organizations": null,
  "suspended": false,
  "deletionTime": null,
  "familyName": "Carter",
  "ims": null,
  "creationTime": [
    "2016-02-02T12:52:30.000Z"
  ],
  "thumbnailPhotoUrl": null,
  "emails": [
    {
      "address": "samcarter@example.com",
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
      "primary": true
    }
  ],
  "phones": null
}
```

Note the ID of the new user (`103567435255251233551` in this example). You will need this ID for the update commands in this section.

2. Reconcile the Google resource with the managed user repository.

   This step should create the new user, Sam Carter (and any other users in your Google resource) in the OpenIDM managed user repository.

   To run reconciliation follow these steps:

   a. In the Admin UI, select Configure > Mappings.

   b. Click on the sourceGoogle__ACCOUNT___managedUser mapping, and click Reconcile Now.

   c. Select Manage > User and verify that the user Sam Carter has been created in the repository.

3. Update Sam Carter's phone number in your Google resource by sending a PUT request with the updated data, and specifying the user `_id` in the request:

```
$ curl \
 --cacert self-signed.crt \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --header "Content-Type: application/json" \
 --request PUT \
 --header "If-Match : *" \
 --data '{
  "__NAME__": "samcarter@example.com",
  "__PASSWORD__"  : "password",
  "givenName" : "Sam",
  "familyName": "Carter",
  "agreedToTerms": true,
  "changePasswordAtNextLogin" : false,
  "phones" :
   [
    {
     "value": "1234567890",
     "type": "home"
    },
    {
     "value": "0987654321",
     "type":"work"
    }
   ]
  }' \
 "https://localhost:8443/openidm/system/google/__ACCOUNT__/103567435255251233551"
{
  "_id": "103567435255251233551",
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
  "_rev": "\"iwpzoDgSq9BJw-XzORg0bILYPVc/vfSJgHt-STUUto4lM_4ESO9izR4\""
,
...
 "emails": [
    {
      "address": "samcarter@example.com",
      "primary": true
    }
  ],
  "phones": [
    {
      "value": "1234567890",
      "type": "home"
    },
    {
      "value": "0987654321",
      "type": "work"
    }
  ]
}
```

4. Read Sam Carter's entry from your Google resource by including his `_id` in the URL:

```
$ curl \
 --cacert self-signed.crt \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "https://localhost:8443/openidm/system/google/__ACCOUNT__/103567435255251233551"
{
  "_id": "103567435255251233551",
  "__NAME__": "samcarter@example.com"
,
...
 "phones": [
    {
      "value": "1234567890",
      "type": "home"
    },
    {
      "value": "0987654321",
      "type": "work"
    }
  ]
}
```

5. Create a group entry on your Google resource:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
 --cacert self-signed.crt \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --header "Content-Type: application/json" \
 --request POST \
 --data '{
  "__NAME__": "testGroup@example.com",
  "__DESCRIPTION__": "Group used for google-connector sample.",
  "name": "TestGroup"
 }' \
 "https://localhost:8443/openidm/system/google/__GROUP__?_action=create"

{
  "_id": "00meukdy40gpg98",
  "_rev": "\"iwpzoDgSq9BJw-XzORg0bILYPVc/LLhHx2plMJPKeY1-h6eX_OVDi4c\"",
  "adminCreated": true,
  "__NAME__": "testgroup@example.com",
  "aliases": null,
  "nonEditableAliases": null,
  "__DESCRIPTION__": "Group used for google-connector sample.",
  "name": "TestGroup",
  "directMembersCount": 0
}
```

6. Add Sam Carter to the test group you have just created. Include the `Member` endpoint, and Sam Carter's `_id` in the URL. Specify the `_id` of the group you created as the value of the `groupKey` in the JSON payload:

```
$ curl \
 --cacert self-signed.crt \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --header "Content-Type: application/json" \
 --request PUT \
 --data '{
  "groupKey" : "00meukdy40gpg98",
  "role": "MEMBER",
  "__NAME__": "samcarter@example.com",
  "email": "samcarter@example.com",
  "type": "MEMBER"
 }' \
 "https://localhost:8443/openidm/system/google/Member/103567435255251233551"
{
  "_id": "00meukdy40gpg98/samcarter@example.com",
  "_rev": "\"iwpzoDgSq9BJw-XzORg0bILYPVc/CPNpkRnowkGWRvNQvUK9ev6gQ90\"",
  "__NAME__": "00meukdy40gpg98/samcarter@example.com",
  "role": "MEMBER",
  "email": "samcarter@example.com",
  "type": "USER",
  "groupKey": "103567435255251233551"
}
```

7. Read the group entry by specifying the group `_id` in the request URL. Notice that the group has one member (`"directMembersCount": 1`).

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
 --cacert self-signed.crt \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "https://localhost:8443/openidm/system/google/__GROUP__/00meukdy40gpg98"

{
  "_id": "00meukdy40gpg98",
  "_rev": "\"iwpzoDgSq9BJw-XzORg0bILYPVc/chUdq5m5_cycV2G4sdl7ZKAF75A\"",
  "adminCreated": true,
  "__NAME__": "testgroup@example.com",
  "aliases": null,
  "nonEditableAliases": [
    "testGroup@example.test-google-a.com"
  ],
  "__DESCRIPTION__": "Group used for google-connector sample.",
  "name": "TestGroup",
  "directMembersCount": 1
}
```

8. Delete the group entry.

```
$ curl \
 --cacert self-signed.crt \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request DELETE \
 "https://localhost:8443/openidm/system/google/__GROUP__/00meukdy40gpg98"
{
  "_id": "00meukdy40gpg98",
  "_rev": "\"iwpzoDgSq9BJw-XzORg0bILYPVc/chUdq5m5_cycV2G4sdl7ZKAF75A\"",
  "adminCreated": true,
  "__NAME__": "testgroup@example.com",
  "aliases": null,
  "nonEditableAliases": [
    "testGroup@example.com.test-google-a.com"
  ],
  "__DESCRIPTION__": "Group used for google-connector sample.",
  "name": "TestGroup",
  "directMembersCount": 1
}
```

The delete request returns the complete group object.

9. Delete Sam Carter, to return your Google resource to its original state.

```
$ curl \
 --cacert self-signed.crt \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request DELETE \
 "https://localhost:8443/openidm/system/google/__ACCOUNT__/103567435255251233551"
{
  "_id": "103567435255251233551",
  "_rev": "\"iwpzoDgSq9BJw-XzORg0bILYPVc/ah6xBLujMAHieSWSisPa1CV6T3Q\"",
  "orgUnitPath": "/",
  "isAdmin": false,
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
  "fullName": "Sam Carter",
  "customerId": "C02rsqddz",
  "relations": null,
  "nonEditableAliases": [
    "samcarter@example.com.test-google-a.com"
  ],
  "suspensionReason": null,
  "includeInGlobalAddressList": true,
  "givenName": "Sam",
  "addresses": null,
  "isDelegatedAdmin": false,
  "changePasswordAtNextLogin": false,
  "isMailboxSetup": true,
  "__NAME__": "samcarter@example.com",
  "agreedToTerms": true,
  "externalIds": null,
  "ipWhitelisted": false,
  "aliases": null,
  "lastLoginTime": [
    "1970-01-01T00:00:00.000Z"
  ],
  "organizations": null,
  "suspended": false,
  "deletionTime": null,
  "familyName": "Carter",
  "ims": null,
  "creationTime": [
    "2016-02-02T12:52:30.000Z"
  ],
  "thumbnailPhotoUrl": null,
  "emails": [
    {
      "address": "samcarter@example.com",
      "primary": true
    }
  ],
  "phones": [
    {
      "value": "1234567890",
      "type": "home"
    },
    {
      "value": "0987654321",
      "type": "work"
    }
  ]
}
```

In this sample, you used the Google Apps connector to add and delete user and group objects in your Google application, and to reconcile users from your Google application to the OpenIDM managed user repository. You can expand on this sample by customizing the connector configuration to provide additional synchronization functionality between OpenIDM and your Google applications. For more information on configuring connectors, see Chapter 11, "*Connecting to External Resources*" in the *Integrator's Guide*.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

Chapter 13

Salesforce Sample - Salesforce With the Salesforce Connector

OpenIDM 4.5 provides a Salesforce Connector that enables provisioning, reconciliation, and synchronization with a Salesforce organization. The Salesforce Connector is not an OpenICF connector, but a separate OpenIDM module, based on the ForgeRock Common Resource API.

Note

The Salesforce Connector, and this corresponding sample, are provided only with the OpenIDM Enterprise build, available on the ForgeRock Backstage site.

This sample demonstrates the creation and update of users from OpenIDM to Salesforce, and from Salesforce to OpenIDM. You can use either the Admin UI, or the command line to run this sample. Both methods are outlined in the sections that follow.

13.1. Before you Start

This sample requires that you have a Salesforce account, and a Connected App with OAuth enabled. For more information about Connected Apps, see the Connected Apps Overview in the Salesforce documentation.

To set up a Connected App for OpenIDM, follow these steps:

1. Log in to salesforce.com with your Salesforce credentials.
2. Click *Setup* in the top right corner.
3. In the left hand menu, under *Build*, expand the *Create* item and click *Apps*.
4. On the right hand panel, scroll down to *Connected Apps* and click *New*.
5. In the *New Connected App* panel, enter the following Basic Information:
   - *Connected App Name*. Enter a name that you will recognize as the OpenIDM App, for example, `OpenIDM`.
   - *API Name*. This field defaults to the Connected App Name, but you can change it. Note that the Application API Name can only contain underscores and alphanumeric characters. The name

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

must be unique, begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.

- *Contact Email*. Enter the email address of the person responsible for this Connected App within your organization.

6. Select *Enable OAuth Settings* and enter the following information:

   - *Callback URL*. Enter the OpenIDM URL, to which the requested OAuth token will be sent, for example `https://localhost:8443/admin/oauth.html`.

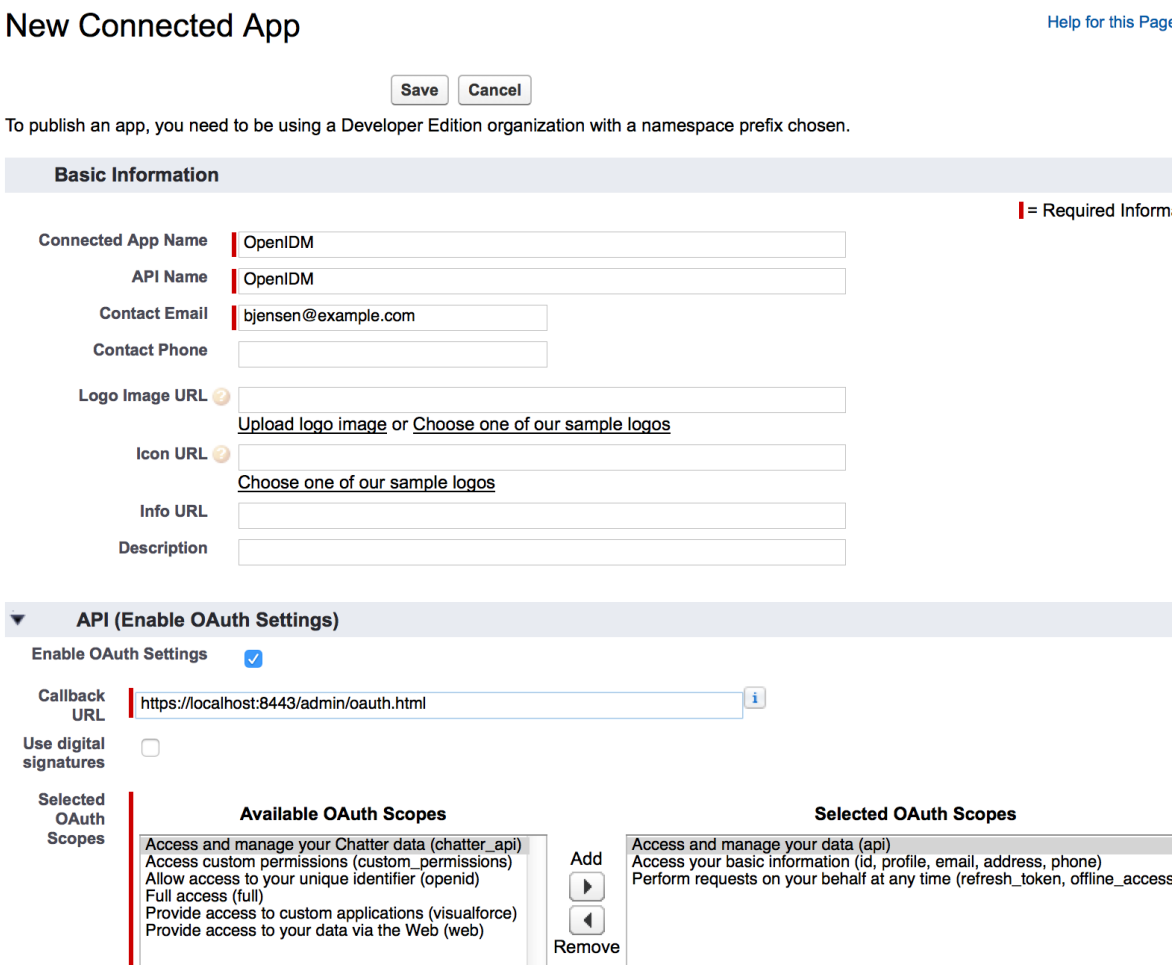
   - *Selected OAuth Scopes*. Click the *Add* button to add the following *Available Auth Scopes* to the *Selected OAuth Scopes* column:

     - Access and manage your data

     - Access your basic information

     - Perform requests on your behalf at any time

Copyright © 2011-2016 ForgeRock AS. All rights reserved.



You can leave the remaining fields blank.

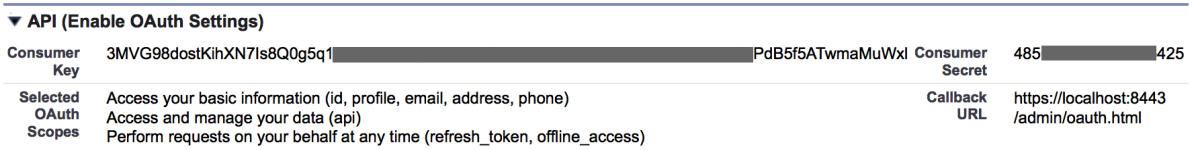
7. Click *Save* to create the new Connected App.

8. The next window displays your new Connected App.

   Under the *API (Enable OAuth Settings)* item, the Consumer Key and a link to the Consumer Secret are displayed.

   Click the link to reveal the Consumer Secret.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.



OpenIDM requires the Consumer Key and Secret to obtain an access token and a refresh token for access to salesforce.com.

Copy and paste both the key and the secret into a file for use when you set up the sample.

9. To demonstrate the reconciliation of users from Salesforce to OpenIDM, your Salesforce organization should contain at least a few users. Add these users now if your Salesforce organization is empty.

13.2. Install the Sample

Prepare OpenIDM as described in Section 1.3, "Preparing OpenIDM", then start OpenIDM with the configuration for the Salesforce sample.

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

13.3. Running the Sample by Using the Admin UI

The Admin UI is the recommended way to test this sample.

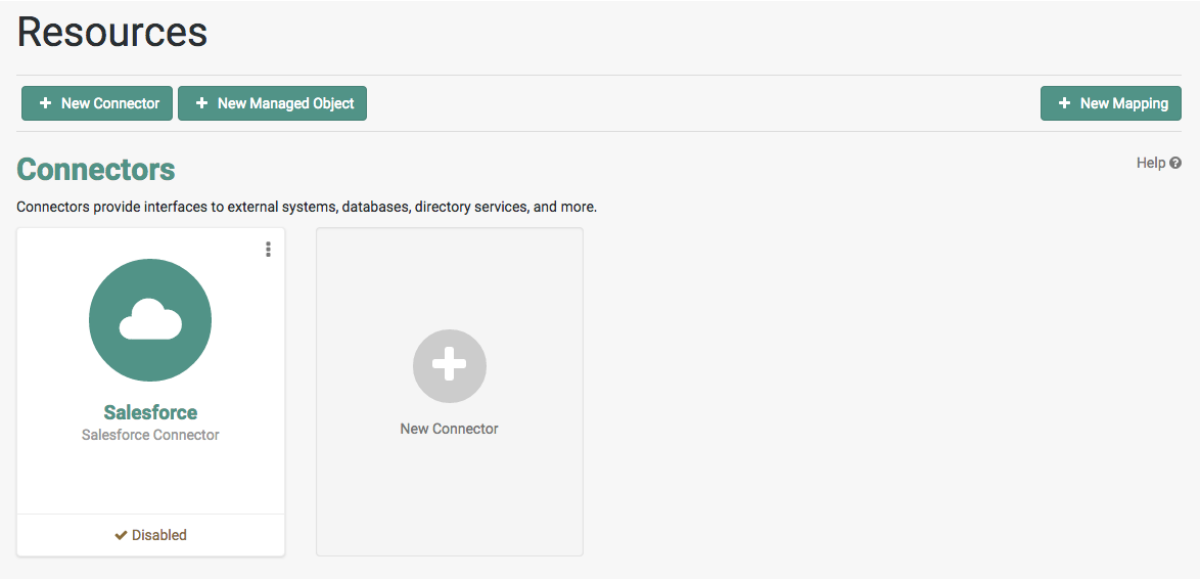
1. Log in to the Admin UI at the URL `https://localhost:8443/admin` as the default administrative user (`openidm-admin`) with password `openidm-admin`.

   Warning

   To protect your deployment in production, change the default administrative password. To do so, navigate to the Self-Service UI at `https://localhost:8443/` and click Change Password.

   The Resources tab shows the Salesforce connector, which is currently disabled.

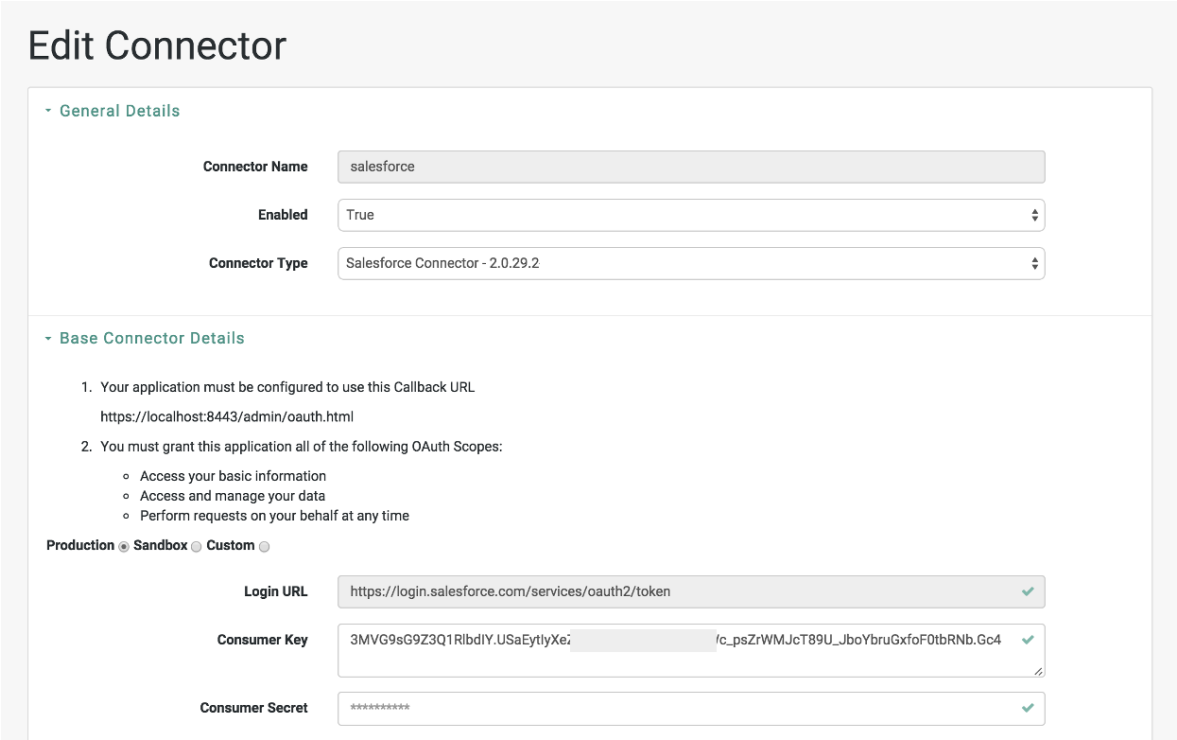
Copyright © 2011-2016 ForgeRock AS. All rights reserved.



2. Enable the Salesforce connector by completing the authentication details as follows. You will need the Consumer Key and Consumer Secret that you obtained in the previous section.

   - Click on the Salesforce connector and click Edit to open the Edit Connector dialog.
   - Select True from the Enabled list.
   - Select Salesforce Connector from the Connector Type list.
   - Under Basic Connector Details select the Sandbox URL (for the purposes of testing the sample) and enter the Connector Key and Secret that you obtained in the previous section.
   - You can leave the default LiveSync details for now. Click Update to update the connector configuration.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

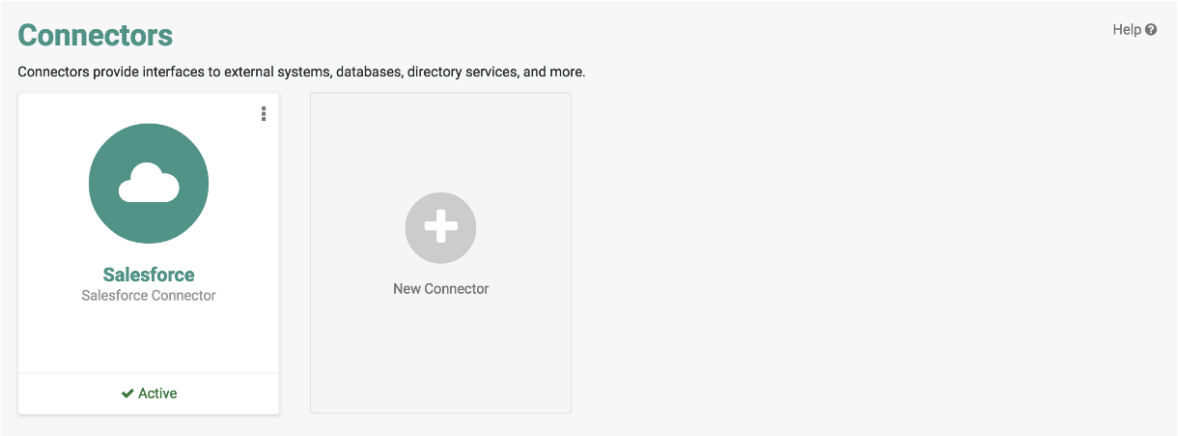


- On the permission request screen click Allow, to enable OpenIDM to access your Salesforce Connected App.

> **Note**
>
> In the current OpenIDM release, an issue is occasionally seen where the system appears to time out while retrieving the refresh token from Salesforce, at this stage. If this happens, refresh your browser and attempt the connector setup again.

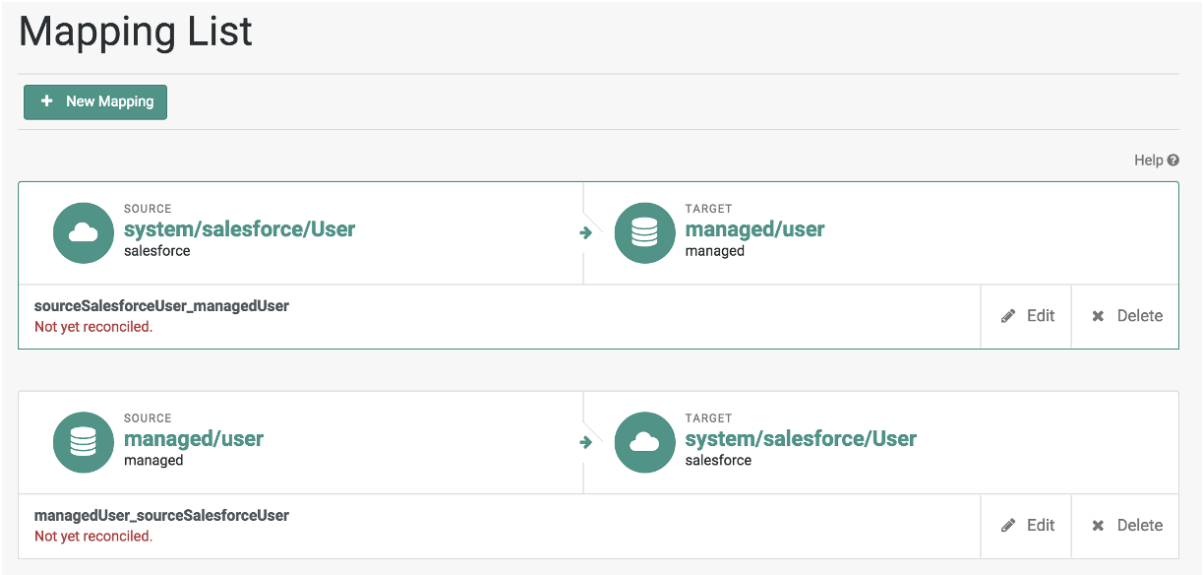
On the Resources tab, your Salesforce Connector should now be Active.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.



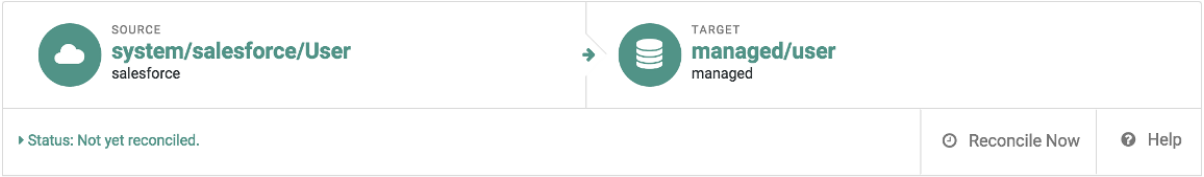
3. To test the reconciliation process, select the Mappings tab.

   This tab shows two configured mappings, one from Salesforce to the OpenIDM repository (`managed/user`) and one from the OpenIDM repository to Salesforce.



4. Click anywhere on the first mapping and click Reconcile Now.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

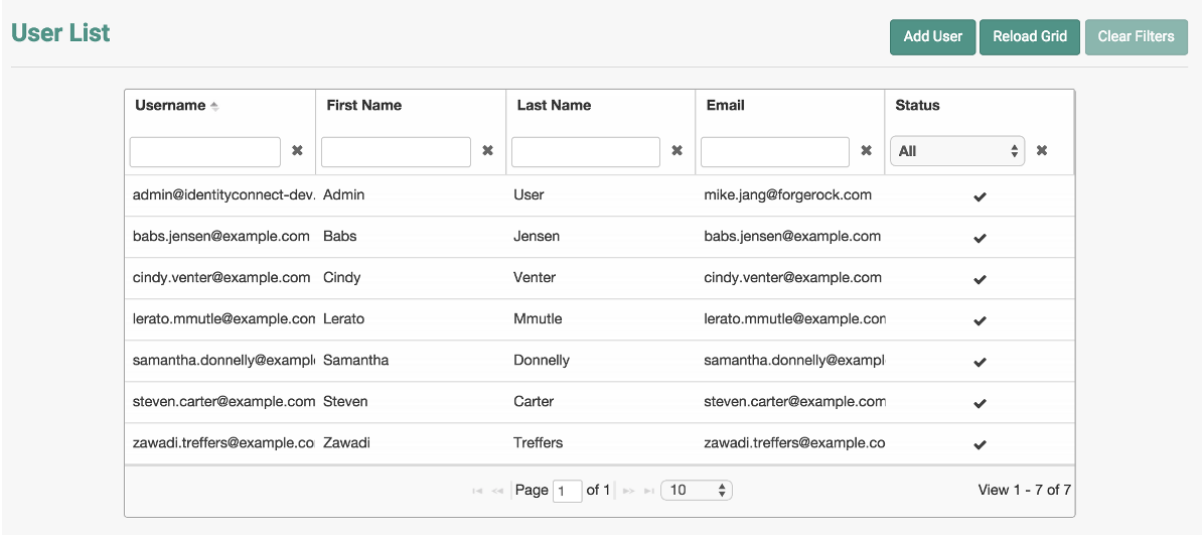


The reconciliation operation creates the users that were present in your Salesforce organization in the OpenIDM repository.

5. Retrieve the users in the repository. In the upper-right of the screen, click the `openidm-admin` link. In the pop-up menu that appears, click the Data Management View link.

   This link opens the Self-Service UI. If you did not change your password in the first step, you are prompted to change your password again. You can bypass this by clicking X to close the password prompt window.

6. Select the Users tab.

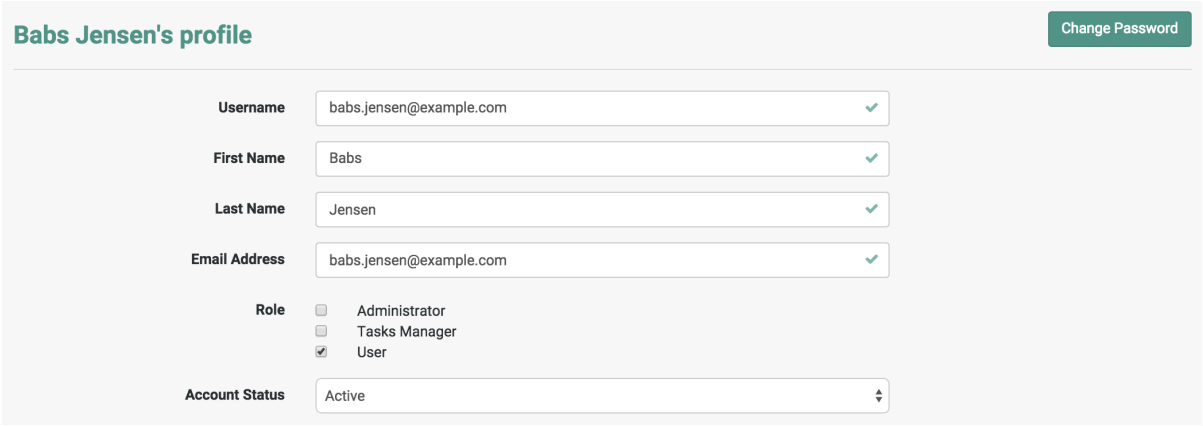


   The users from the Salesforce organization have been reconciled to the OpenIDM repository. If the reconciliation was successful, the list of users displayed here should reflect what was in your Salesforce organization.

7. To retrieve the details of a specific user, click that username on the Users tab.

   The following image shows the details of user `bjensen`. Scroll down. Note the Linked Systems panel that shows the corresponding user record in Salesforce.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.



8. To test the second mapping (from OpenIDM to Salesforce), update any user in the OpenIDM repository. For example, update Babs Jensen's username.

9. By default, *implicit synchronization* is enabled for mappings *from* the `managed/user` repository *to* any external resource. This means that when you update a managed object, any mappings defined in the `sync.json` file that have the managed object as the source are automatically executed to update the target system. For more information, see Section 12.3.2, "Mapping Source Objects to Target Objects" in the *Integrator's Guide*.

   To test that the implicit synchronization has been successful, look at Babs Jensen's record in the Self-Service UI. At the bottom of the user profile, the Linked Systems panel indicates Babs Jensen's record in the Salesforce data store. Note the changed Username.

   Alternatively, check the updated user record in Salesforce.

13.4. Running the Sample by Using the Command Line

Running the sample by using the command line is a little more complex. This section breaks the sample into two tasks - configuring the connector, and then testing the configuration by running reconciliation operations between the two systems.

Procedure 13.1. To Set Up the Salesforce Connector

Before you start, you will need the Consumer Key and Consumer Secret that you obtained in the previous section.

1. Obtain the refresh token from salesforce.com by pointing your browser to the following URL. Substitute your Consumer Key for `CLIENT_ID`. If OpenIDM is not running on the localhost, substitute the appropriate hostname and port number in the value of the `redirect_uri` parameter.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

https://login.salesforce.com/services/oauth2/authorize?response_type=code&client_id=CLIENT_ID&redirect_uri=https://localhost:8443/admin/oauth.html&scope=id+api+refresh_token

2. You are redirected to Salesforce, and prompted to give this application access to your Salesforce account. When you have given consent, you should receive a response URL that looks similar to the following:

```
https://localhost:8443/admin/index.html#connectors/edit//&code=aPrxJZTK7Rs03PU634VK8Jn9o_U3ZY1ERxM7IiklF.
..
```

The `&code` part of this URL is an authorization code, that you need for the following step.

> **Caution**
>
> Note that this authorization code expires after 10 minutes. If you do not complete the OAuth flow within that time, you will need to start this process again.

3. Copy the authorization code from the response URL and use it as the value of the `"code"` parameter in the following REST call. You will also need to supply your Consumer Key and Consumer Secret in this call.

```
$  curl \
 --verbose \
 --data "grant_type=authorization_code" \
 --data "client_id=consumer-key" \
 --data "client_secret=consumer-secret" \
 --data "redirect_uri=https://localhost:8443/admin/oauth.html" \
 --data "code=access-token-code" \
 "https://login.salesforce.com/services/oauth2/token"
{
  "access_token": "00DS0000003K4fU!AQMAQOzEU
.8tCjg8Wk79yKPKCtrtaszX5jrHtoT4NBpJ8x2NFZGjg3PNuc0TWq0EgiGS_mVkfg5f4pVN5...",
  "signature": "2uREX1lseXdg3Vng/2+Hrlo/KHOWYoim+poj74wKFtw=",
  "refresh_token": "5Aep861KIwKdekr90I4iHdtDgWwRoG7O_6uHrgJ
.yVtMS0UaGxRqE6WFM77W7wCV4muVMgdqKjuWI2i5S6sjN2X",
  "token_type": "Bearer",
  "instance_url": "https://example-com.cs1.my.salesforce.com",
  "scope": "id api refresh_token",
  "issued_at": "1417182949781",
  "id": "https://login.salesforce.com/id/00DS0000003K4fUMAS/00530000009hWLcAAM"
}
```

The output includes an `access_token` and a `refresh_token`. You will need the `refresh_token` in the following step.

4. Edit the `configurationProperties` in your Salesforce connector configuration file (`openidm/samples/salesforce-connector/conf/provisioner.salesforce-salesforce.json`) to include your Consumer Key (`clientID`), Consumer Secret (`clientSecret`), and refresh token.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

In addition, set the `"instanceUrl"` to the value returned in the previous step, and set the `"enabled"` property to `true`.

The relevant excerpts of the `provisioner.salesforce-salesforce.json` file are as follows:

```
{
    "name" : "salesforce",
    "enabled" : true,
    "connectorRef" : {
...
    "configurationProperties" : {
        "connectTimeout" : 120000,
        "loginUrl" : null,
        "idleCheckInterval" : 10000,
        "refreshToken"
 : "5Aep861KIwKdekr90I4iHdtDgWwRoG7O_6uHrgJ.yVtMS0UaGxRqE6WFM77W7wCV4muVMgdqKjuWI2i5S6sjN2X",
        "clientSecret" : "4850xxxxxxxxxxxxx425",
        "clientId" : "3MVG98dostKihXN7Is8Q0g5q1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxPdB5f5ATwmaMuWxl",
        "instanceUrl" : "https://example-com.cs1.my.salesforce.com",
        "version" : 29
    }
...
```

5. Check that your connector configuration is correct by testing the status of the connector, over REST.

```
$ curl \
 --cacert self-signed.crt \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request POST \
 "https://localhost:8443/openidm/system?_action=test"
[
  {
    "ok": true,
    "connectorRef": {
      "bundleVersion": "2.0.29.2",
      "systemType": "provisioner.salesforce",
      "displayName": "Salesforce Connector",
      "bundleName": "org.forgerock.openidm.salesforce",
      "connectorName": "org.forgerock.openidm.salesforce.Salesforce"
    },
    "objectTypes": [
      "User",
      "PermissionSet",
      "PermissionSetAssignment",
      "Profile",
      "PermissionSetLicenseAssign",
      "Organization",
      "PermissionSetLicense",
      "Group",
      "GroupMember"
    ],
    "config": "config/provisioner.salesforce/salesforce",
    "enabled": true,
    "name": "salesforce"
  }
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
]
```

Procedure 13.2. Run Reconciliation by Using the Command Line

The mapping configuration file (`sync.json`) for this sample includes two mappings, `sourceSalesforceUser_managedUser`, which synchronizes users from the Salesforce with the OpenIDM repository, and `managedUser_sourceSalesforceUser`, which synchronizes changes from the OpenIDM repository to Salesforce.

1. Reconcile the repository over the REST interface by running the following command:

```
$ curl \
 --cacert self-signed.crt \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request POST \
 "https://localhost:8443/openidm/recon?
_action=recon&mapping=sourceSalesforceUser_managedUser&waitForCompletion=true"
{
  "state": "SUCCESS",
  "_id": "8a6281ef-6faf-43dd-af5c-3a842b38c468"
}
```

   The reconciliation operation returns a reconciliation run ID and the status of the operation. Reconciliation creates user objects from LDAP in the OpenIDM repository, assigning the new objects random unique IDs.

2. View the recon entry over REST for an indication of the actions that were taken on the OpenIDM repository.

```
$ curl \
 --cacert self-signed.crt \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "https://localhost:8443/openidm/recon/8a6281ef-6faf-43dd-af5c-3a842b38c468"
{
  "duration": 6447,
  "ended": "2014-11-28T15:01:38.399Z",
  "started": "2014-11-28T15:01:31.952Z",
  "parameters": {
    "null": false,
    "boolean": false,
    "number": false,
    "list": false,
    "object": {
      "targetQuery": {
        "_queryId": "query-all-ids",
        "resourceName": "managed/user"
      },
      "sourceQuery": {
        "_queryId": "query-all-ids",
        "resourceName": "system/salesforce/User"
      }
    },
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
    "pointer": {
      "empty": true
    },
    "transformers": [],
    "set": false,
    "map": true,
    "string": false,
    "collection": false,
    "wrappedObject": {
      "targetQuery": {
        "resourceName": "managed/user",
        "_queryId": "query-all-ids"
      },
      "sourceQuery": {
        "_queryId": "query-all-ids",
        "resourceName": "system/salesforce/User"
      }
    }
  },
  "_id": "8a6281ef-6faf-43dd-af5c-3a842b38c468",
  "mapping": "sourceSalesforceUser_managedUser",
  "state": "SUCCESS",
  "stage": "COMPLETED_SUCCESS",
  "stageDescription": "reconciliation completed.",
  "progress": {
    "links": {
      "created": 8,
      "existing": {
        "total": "0",
        "processed": 0
      }
    },
    "target": {
      "created": 8,
      "existing": {
        "total": "0",
        "processed": 0
      }
    },
    "source": {
      "existing": {
        "total": "9",
        "processed": 9
      }
    }
  },
  "situationSummary": {
    "FOUND_ALREADY_LINKED": 0,
    "UNASSIGNED": 0,
    "TARGET_IGNORED": 0,
    "SOURCE_IGNORED": 0,
    "MISSING": 0,
    "FOUND": 0,
    "AMBIGUOUS": 0,
    "UNQUALIFIED": 0,
    "CONFIRMED": 0,
    "SOURCE_MISSING": 0,
    "ABSENT": 9
  },
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
  "statusSummary": {
    "SUCCESS": 8,
    "FAILURE": 1
  }
}
```

The output shows that eight entries were created on the target (`managed/user`).

3. You can display those users by querying the IDs in the managed/user repository.

```
$ curl \
 --cacert self-signed.crt \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "https://localhost:8443/openidm/managed/user?_queryId=query-all-ids"
{
  "remainingPagedResults": -1,
  "pagedResultsCookie": null,
  "resultCount": 8,
  "result": [
    {
      "_rev": "0",
      "_id": "f15322f2-5873-4e5f-a4e5-2d4bc03dd190"
    },
    {
      "_rev": "0",
      "_id": "85879c60-afa1-4425-8c7a-5cccbbaff587"
    },
    {
      "_rev": "0",
      "_id": "ed3fe655-29a6-4016-b6bc-4b2356911fd1"
    },
    {
      "_rev": "0",
      "_id": "34678464-c080-41b1-8da6-d5fde9d35aeb"
    },
    {
      "_rev": "0",
      "_id": "02d5da29-8349-4f35-affc-5f6c331307ef"
    },
    {
      "_rev": "0",
      "_id": "f91d6fce-bf27-4379-9411-fd626f8a9528"
    },
    {
      "_rev": "0",
      "_id": "6ace9220-59e7-4d97-8683-e03362a9150c"
    },
    {
      "_rev": "0",
      "_id": "56863eea-35d7-4aeb-a017-74ef28fd3116"
    }
  ]
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

Chapter 14

Scripted Kerberos Connector Sample

New in OpenIDM 4.5.0, the scripted Kerberos connector sample demonstrates how to manage Kerberos user principals and how to reconcile user principals with OpenIDM managed user objects.

The connector configuration (`/path/to/openidm/samples/kerberos/conf/provisioner.openicf-kerberos.json)`) assumes that OpenIDM is running on a host that is separate from the Kerberos host.

This sample assumes that the default realm is `EXAMPLE.COM` and that there is an existing user principal `openidm/admin`. Adjust the sample to match your Kerberos realm and principals.

14.1. Editing the Kerberos Connector Configuration

Before you run this sample, edit the connector configuration file to match your Kerberos environment. Specifically, set the correct values for the following properties:

`host`

The host name or IP address of the machine on which Kerberos is running.

`port`

The SSH port on that machine.

Default: `22` (the default SSH port)

`user`

The username of the account that is used to connect to the SSH server.

`password`

The password of the account that is used to connect to the SSH server.

`prompt`

A string that represents the remote SSH session prompt. This must be the exact prompt string, in the format `username@target:`, for example `root@localhost:~$` . The easiest way to obtain this string is to `ssh` into the machine and copy paste the prompt.

`customConfiguration`

The details of the admin user principal and the default realm.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

This example assumes an admin user principal of `openidm/admin`.

For more information on setting this property, see `customConfiguration` in the *Connectors Guide*.

`customSensitiveConfiguration`

The password for the user principal.

For more information on setting this property, see `customSensitiveConfiguration` in the *Connectors Guide*.

Your connector configuration should look something like the following:

```
...
  "configurationProperties" : {
    "host" : "192.0.2.0",
    "port" : 22,
    "user" : "admin",
    "password" : "Passw0rd",
    "prompt" : "admin@myhost:~$",
    "sudoCommand" : "/usr/bin/sudo",
    "echoOff" : true,
    "terminalType" : "vt102",
    "setLocale" : false,
    "locale" : "en_US.utf8",
    "connectionTimeout" : 5000,
    "expectTimeout" : 5000,
    "authenticationType" : "PASSWORD",
    "throwOperationTimeoutException" : true,
    "customConfiguration" : "kadmin { cmd = '/usr/sbin/kadmin.local'; user='openidm/admin';
 default_realm='EXAMPLE.COM' }",
    "customSensitiveConfiguration" : "kadmin { password = 'Passw0rd'}",
  ...
```

OpenIDM will encrypt all passwords in the configuration when it starts up, or whenever it reloads the configuration file.

For information about the complete Kerberos connector configuration, see Section 10.2, "Configuring the Kerberos Connector" in the *Connectors Guide*.

Caution

Do not modify the value of the `scriptRoots` or `classpath` properties unless you have extracted the scripts from the connector bundle and placed them on the filesystem.

14.2. Running the Kerberos Sample

The commands in this section achieve the following:

1. Start OpenIDM and check that the connector can reach the Kerberos server.
2. Create two users in the OpenIDM managed repository.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

3. Reconcile the managed repository with the Kerberos server so that the new users are created in Kerberos.
4. Retrieve the details of one of the new Kerberos principals from the server.
5. Delete one of the managed users.
6. Reconcile the managed repository again and note that the corresponding Kerberos principal has been deleted.

1. Start OpenIDM with the configuration for the Kerberos sample:

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

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

2. Test that your connector configuration is correct and that OpenIDM can reach your Kerberos server, with the following command:

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request POST \
 "http://localhost:8080/openidm/system?_action=test"
[
  {
    "name": "kerberos",
    "enabled": true,
    "config": "config/provisioner.openicf/kerberos",
    "objectTypes": [
      "__ALL__",
      "account"
    ],
    "connectorRef": {
      "bundleName": "org.forgerock.openicf.connectors.kerberos-connector",
      "connectorName": "org.forgerock.openicf.connectors.kerberos.KerberosConnector",
      "bundleVersion": "1.4.0.0"
    },
    "displayName": "Kerberos Connector",
    "ok": true
  }
]
```

If the command returns `"ok": true`, as in the preceding output, your configuration is correct and you can continue with the sample.

3. Retrieve a list of the existing user principals in the Kerberos database:

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/system/kerberos/account?_queryId=query-all-ids"
{
  "result": [
    {
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
      "_id": "K/M@EXAMPLE.COM",
      "principal": "K/M@EXAMPLE.COM"
    },
    {
      "_id": "kadmin/admin@EXAMPLE.COM",
      "principal": "kadmin/admin@EXAMPLE.COM"
    },
    {
      "_id": "kadmin/changepw@EXAMPLE.COM",
      "principal": "kadmin/changepw@EXAMPLE.COM"
    },
    {
      "_id": "kadmin/krb1.example.com@EXAMPLE.COM",
      "principal": "kadmin/krb1.example.com@EXAMPLE.COM"
    },
    {
      "_id": "kiprop/krb1.example.com@EXAMPLE.COM",
      "principal": "kiprop/krb1.example.com@EXAMPLE.COM"
    },
    {
      "_id": "krbtgt/EXAMPLE.COM@EXAMPLE.COM",
      "principal": "krbtgt/EXAMPLE.COM@EXAMPLE.COM"
    },
    {
      "_id": "openidm/admin@EXAMPLE.COM",
      "principal": "openidm/admin@EXAMPLE.COM"
    }
  ],
  ...
}
```

4. Create two new managed users, either over REST or by using the Admin UI.

   The following command creates users bjensen and scarter over REST. To create similar users by using the Admin UI, select Managed > User and click New User:

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --header "Content-type: application/json" \
 --request POST \
 --data '{
  "userName": "bjensen",
  "givenName": "Barbara",
  "sn" : "Jensen",
  "password" : "Passw0rd",
  "displayName" : "Barbara Jensen",
  "mail" : "bjensen@example.com"
 }' \
 "http://localhost:8080/openidm/managed/user?_action=create"
{
  "_id": "ce3d9b8f-1d15-4950-82c1-f87596aadcb6",
  "_rev": "2",
  "userName": "bjensen",
  "givenName": "Barbara",
  "sn": "Jensen",
  "displayName": "Barbara Jensen",
  "mail": "bjensen@example.com",
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
  "accountStatus": "active",
  "effectiveRoles": [],
  "effectiveAssignments": []
}
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --header "Content-type: application/json" \
 --request POST \
 --data '{
  "userName": "scarter",
  "givenName": "Steven",
  "sn" : "Carter",
  "password" : "Passw0rd",
  "displayName" : "Steven Carter",
  "mail" : "scarter@example.com"
 }' \
 "http://localhost:8080/openidm/managed/user?_action=create"
{
  "_id": "a204ca60-b0fc-42f8-bf93-65bb30131361",
  "_rev": "2",
  "userName": "scarter",
  "givenName": "Steven",
  "sn": "Carter",
  "displayName": "Steven Carter",
  "mail": "scarter@example.com",
  "accountStatus": "active",
  "effectiveRoles": [],
  "effectiveAssignments": []
}
```

5. Run a reconciliation operation between the managed user repository and the Kerberos database to create the new users bjensen and scarter in Kerberos. You can run the reconciliation over REST, or using the Admin UI.

   The following command creates runs the reconciliation over REST:

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request POST \
 "http://localhost:8080/openidm/recon?_action=recon&mapping=managedUser_systemKerberos"
{
  "_id": "862ab9ba-d1d9-4058-b6bc-a23a94b68776-234",
  "state": "ACTIVE"
}
```

   To run the reconciliation by using the Admin UI, select Configure > Mappings, click on the `managedUser_systemKerberos` mapping, and click Reconcile Now.

6. Retrieve the list of Kerberos user principals again. You should now see bjensen and scarter in this list:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/system/kerberos/account?_queryId=query-all-ids"
{
  "result": [
    {
      "_id": "bjensen@EXAMPLE.COM",
      "principal": "bjensen@EXAMPLE.COM"
    },
    {
      "_id": "scarter@EXAMPLE.COM",
      "principal": "scarter@EXAMPLE.COM"
    },
    ...
    {
      "_id": "openidm/admin@EXAMPLE.COM",
      "principal": "openidm/admin@EXAMPLE.COM"
    }
  ],
  ...
}
```

7. Retrieve bjensen's complete user principal from the Kerberos server:

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/system/kerberos/account/bjensen@EXAMPLE.COM"
{
  "_id": "bjensen@EXAMPLE.COM",
  "lastFailedAuthentication": "[never]",
  "passwordExpiration": "[none]",
  "lastSuccessfulAuthentication": "[never]",
  "maximumTicketLife": "0 days 10:00:00",
  "lastModified": "Tue May 24 04:05:45 EDT 2016 (openidm/admin@EXAMPLE.COM)",
  "policy": "user [does not exist]",
  "expirationDate": "[never]",
  "failedPasswordAttempts": "0",
  "maximumRenewableLife": "7 days 00:00:00",
  "principal": "bjensen@EXAMPLE.COM",
  "lastPasswordChange": "Tue May 24 04:05:45 EDT 2016"
}
```

Note the default values for properties such as `maximumRenewableLife`. These values are set in your connector configuration. For more information, see Section 10.2, "Configuring the Kerberos Connector" in the *Connectors Guide*.

To perform this step in the Admin UI, select Manage > User, click bjensen's entry, and click the Linked Systems tab to display her corresponding entry on the Kerberos server.

8. Delete the managed user bjensen by specifying her managed object ID in the DELETE request.

First, obtain her ID by querying for her userName:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/managed/user?_queryFilter=userName+eq+'bjensen'"
{
  "result": [
    {
      "_id": "ce3d9b8f-1d15-4950-82c1-f87596aadcb6",
      "_rev": "3",
      "userName": "bjensen",
      "givenName": "Barbara",
      "sn": "Jensen",
      "displayName": "Barbara Jensen",
      "mail": "bjensen@example.com",
      "accountStatus": "active",
      "effectiveRoles": [],
      "effectiveAssignments": []
    }
  ],
  ...
}
```

Now delete the user with ID `ce3d9b8f-1d15-4950-82c1-f87596aadcb6`. This ID will obviously be different in your example.

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request DELETE \
 "http://localhost:8080/openidm/managed/user/ce3d9b8f-1d15-4950-82c1-f87596aadcb6"
{
  "_id": "ce3d9b8f-1d15-4950-82c1-f87596aadcb6",
  "_rev": "3",
  "userName": "bjensen",
  "givenName": "Barbara",
  "sn": "Jensen",
  "displayName": "Barbara Jensen",
  "mail": "bjensen@example.com",
  "accountStatus": "active",
  "effectiveRoles": [],
  "effectiveAssignments": []
}
```

To delete bjensen's managed user entry by using the Admin UI, select Manage > User, click on bjensen's entry, select the checkbox next to her entry, and click Delete Selected.

9. Reconcile the managed user repository and the Kerberos database again:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request POST \
 "http://localhost:8080/openidm/recon?_action=recon&mapping=managedUser_systemKerberos"
{
  "_id": "862ab9ba-d1d9-4058-b6bc-a23a94b68776-584",
  "state": "ACTIVE"
}
```

10. Retrieve the list of Kerberos user principals again. The Kerberos principal for bjensen should have been been removed from the list:

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/system/kerberos/account?_queryId=query-all-ids"
{
  "result": [
    {
      "_id": "K/M@EXAMPLE.COM",
      "principal": "K/M@EXAMPLE.COM"
    },
    {
      "_id": "kadmin/admin@EXAMPLE.COM",
      "principal": "kadmin/admin@EXAMPLE.COM"
    },
    {
      "_id": "kadmin/changepw@EXAMPLE.COM",
      "principal": "kadmin/changepw@EXAMPLE.COM"
    },
    {
      "_id": "kadmin/krb1.example.com@EXAMPLE.COM",
      "principal": "kadmin/krb1.example.com@EXAMPLE.COM"
    },
    {
      "_id": "kiprop/krb1.example.com@EXAMPLE.COM",
      "principal": "kiprop/krb1.example.com@EXAMPLE.COM"
    },
    {
      "_id": "krbtgt/EXAMPLE.COM@EXAMPLE.COM",
      "principal": "krbtgt/EXAMPLE.COM@EXAMPLE.COM"
    },
    {
      "_id": "scarter@EXAMPLE.COM",
      "principal": "scarter@EXAMPLE.COM"
    },
    {
      "_id": "openidm/admin@EXAMPLE.COM",
      "principal": "openidm/admin@EXAMPLE.COM"
    }
  ],
  ...
}
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

Note

Some user IDs in Kerberos include characters such as a forward slash (`/`) and an "at sign" (`@`) that prevent them from being used directly in a REST URL. For example, `openidm/system/kerberos/account/kadmin/admin@EXAMPLE.COM`, where the ID is `kadmin/admin@EXAMPLE.COM`. To retrieve such entries directly over REST, you must URL-encode the Kerberos ID as follows:

```
"http://localhost:8080/openidm/system/kerberos/account/kadmin%2Fadmin%40EXAMPLE.COM"
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

Chapter 15
Custom Endpoint Sample

This chapter describes the custom endpoint sample delivered with OpenIDM.

OpenIDM supports scriptable custom endpoints that enable you to launch arbitrary scripts through an OpenIDM REST URI. For information about how custom endpoints are configured, see Section 13.2, "Creating Custom Endpoints to Launch Scripts" in the *Integrator's Guide*.

The sample endpoint provided in `/path/to/openidm/samples/customendpoint` illustrates the configuration of a custom endpoint, and the structure of custom endpoint scripts.

The purpose of this custom endpoint is to return a list of variables available to each method used in a script. The scripts show the complete set of methods that can be used. These methods map to the standard HTTP verbs - create, read, update, delete, patch, query, and action. A sample JavaScript and Groovy script is provided.

To run the sample:

1. Copy the endpoint configuration file (`samples/customendpoint/conf/endpoint-echo.json` to your project's `conf` directory.
2. Copy either the JavaScript file (`samples/customendpoint/script/echo.js`) or Groovy script file (`samples/customendpoint/script/echo.groovy`) to your project's `script` directory.
3. Open the endpoint configuration file in a text editor:

   ```
   {
       "file" : "echo.groovy",
       "type" : "groovy",
       "_file" : "echo.js",
       "_type" : "text/javascript"
   }
   ```

   The configuration file contains nothing more than a reference to the endpoint scripts. In this case, the JavaScript script is commented out (with an underscore before the `file` and `type` properties. If you want to use the JavaScript endpoint script, uncomment these lines and comment out the lines that correspond to the Groovy script in the same way.

   Endpoint configuration files can include a `context` property that specifies the route to the endpoint, for example:

   ```
   "context" : "endpoint/linkedView/*"
   ```

   If no `context` is specified, the route to the endpoint is taken from the file name, in this case `endpoint/echo`.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

4. Test each method in succession to return the expected request structure of that method. The following examples show the request structure of the read, create and patch methods. The configuration file has been edited to use the JavaScript file, rather than the Groovy file. The output shown in these examples has been cropped for legibility. For a description of each parameter, see Section 13.2.2, "Writing Custom Endpoint Scripts" in the *Integrator's Guide*.

   The following command performs a read on the echo endpoint and returns the request structure of a read request:

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/endpoint/echo
{
  "_id": "",
  "method": "read",
  "resourceName": "",
  "parameters": {},
  "context": {
  ...
  }
}
```

   The following command performs a query on the echo endpoint and returns the request structure of a query request:

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/endpoint/echo?_queryId=query-all-ids"
{
  "result": [
    {
      "method": "query",
      "resourceName": "",
      "pagedResultsCookie": null,
      "pagedResultsOffset": 0,
      "pageSize": 0,
      "queryExpression": null,
      "queryId": "query-all-ids",
      "queryFilter": "null",
      "parameters": {},
      "content": null,
      "context": {
      ...
      }
    }
  ],
  ...
}
```

   The following command sends a create request to the echo endpoint. No user is actually created. The endpoint script merely returns the request structure of a create request. The `content` parameter in this case provides the JSON object that was sent with the request:

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
 --header "X-OpenIDM-Password: openidm-admin" \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "Content-Type: application/json" \
 --data '{
       "userName":"steve",
       "givenName":"Steve",
       "sn":"Carter",
       "telephoneNumber":"0828290289",
       "mail":"scarter@example.com",
       "password":"Passw0rd"
       }' \
 --request POST \
 "http://localhost:8080/openidm/endpoint/echo?_action=create"
{
  "_id": "",
  "method": "create",
  "resourceName": "",
  "newResourceId": null,
  "parameters": {},
  "content": {
    "userName": "steve",
    "givenName": "Steve",
    "sn": "Carter",
    "telephoneNumber": "0828290289",
    "mail": "scarter@example.com",
    "password": "Passw0rd"
  },
  "context": {
  ...
  }
}
```

The following command sends a patch request to the echo endpoint.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

```
$ curl \
 --header "X-OpenIDM-Password: openidm-admin" \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "Content-Type: application/json" \
 --data '[
    {
      "operation":"replace",
      "field":"/givenName",
      "value":"Steven"
    }
 ]' \
 --request PATCH \
 "http://localhost:8080/openidm/endpoint/echo"
{
  "_id": "",
  "method": "patch",
  "resourceName": "",
  "revision": null,
  "patch": [
    {
      "operation": "replace",
      "field": "/givenName",
      "value": "Steven"
    }
  ],
  "parameters": {},
  "context": {
  ...
  }
}
```


Copyright © 2011-2016 ForgeRock AS. All rights reserved.

OpenIDM Glossary

correlation query	A correlation query specifies an expression that matches existing entries in a source repository to one or more entries on a target repository. While a correlation query may be built with a script, it is *not* a correlation script. As noted in Section 12.3.2.6, "Correlating Source Objects With Existing Target Objects" in the *Integrator's Guide*, you can set up a query definition, such as `_queryId`, `_queryFilter`, or `_queryExpression`, possibly with the help of a `linkQualifier`.
correlation script	A correlation script matches existing entries in a source repository, and returns the IDs of one or more matching entries on a target repository. While it skips the intermediate step associated with a `correlation query`, a correlation script can be relatively complex, based on the operations of the script.
entitlement	An entitlement is a collection of attributes that can be added to a user entry via roles. As such, it is a specialized type of `assignment`. A user or device with an entitlement gets access rights to specified resources. An entitlement is a property of a managed object.
JSON	JavaScript Object Notation, a lightweight data interchange format based on a subset of JavaScript syntax. For more information, see the JSON site.
JWT	JSON Web Token. As noted in the *JSON Web Token draft IETF Memo*, "JSON Web Token (JWT) is a compact URL-safe means of representing claims to be transferred between two parties." For OpenIDM, the JWT is associated with the `JWT_SESSION` authentication module.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

managed object	An object that represents the identity-related data managed by OpenIDM. Managed objects are configurable, JSON-based data structures that OpenIDM stores in its pluggable repository. The default configuration of a managed object is that of a user, but you can define any kind of managed object, for example, groups or roles.
mapping	A policy that is defined between a source object and a target object during reconciliation or synchronization. A mapping can also define a trigger for validation, customization, filtering, and transformation of source and target objects.
OSGi	A module system and service platform for the Java programming language that implements a complete and dynamic component model. For a good introduction, see the OSGi site. While OpenIDM services are designed to run in any OSGi container, currently only Apache Felix is supported.
reconciliation	During reconciliation, comparisons are made between managed objects and objects on source or target systems. Reconciliation can result in one or more specified actions, including, but not limited to, synchronization.
resource	An external system, database, directory server, or other source of identity data to be managed and audited by the identity management system.
REST	Representational State Transfer. A software architecture style for exposing resources, using the technologies and protocols of the World Wide Web. REST describes how distributed data objects, or resources, can be defined and addressed.
role	OpenIDM includes two different types of provisioning roles and authorization roles. For more information, see Section 8.4, "Working With Managed Roles" in the *Integrator's Guide*.
source object	In the context of reconciliation, a source object is a data object on the source system, that OpenIDM scans before attempting to find a corresponding object on the target system. Depending on the defined mapping, OpenIDM then adjusts the object on the target system (target object).
synchronization	The synchronization process creates, updates, or deletes objects on a target system, based on the defined mappings from the source system. Synchronization can be scheduled or on demand.
system object	A pluggable representation of an object on an external system. For example, a user entry that is stored in an external LDAP directory is represented as a system object in OpenIDM for the period during which OpenIDM requires access to that entry.System objects follow

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

	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.

Copyright © 2011-2016 ForgeRock AS. All rights reserved.

Index

G

Getting started, 6

I

Installing
- Samples, 5

R

Repository database
- Evaluation version, 6

S

Samples
- Full Stack Sample (OpenAM, OpenDJ, OpenIDM), 197
- Google Apps Connector, 230
- Salesforce Connector, 241
- Sample 1 - XML file, 6
- Sample 2 - LDAP one way, 22
- Sample 2b - LDAP two way, 27
- Sample 2c - Synchronizing LDAP Group Membership, 31
- Sample 2d - Synchronizing LDAP Groups, 37
- Sample 5 - Synchronization of two resources, 41
- Sample 5b - Failure Compensation With Multiple Resources, 44
- Sample 6 - LiveSync between two LDAP servers, 47
- Sample 8 - Logging in Scripts, 17
- Sample 9 - asynchronous reconciliation, 18
- ScriptedCREST Connector, 96
- ScriptedREST Connector, 88

Copyright © 2011-2016 ForgeRock AS. All rights reserved.