



Connectors Guide

/ ForgeRock Identity Management 7

Latest update: 7.0.4

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

Copyright © 2011-2021 ForgeRock AS.

Abstract

Installation and configuration reference for the connectors that are supported with ForgeRock® Identity Management software. This reference includes installation and configuration instructions for each connector, and examples that demonstrate how to use the connectors in a deployment.



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, <https://fontawesome.com/>.

This Font Software is licensed under the SIL Open Font License, Version 1.1. See <https://opensource.org/licenses/OFL-1.1>.

Table of Contents

Overview	v
1. The ForgeRock Identity Connector Framework (ICF)	1
2. Supported Connectors	3
Adobe Marketing Cloud Connector	6
AS400 connector	11
Cerner Connector	28
CSV File Connector	45
Database Table Connector	48
DocuSign Connector	59
Google Cloud Platform Connector	72
Google Apps Connector	82
Groovy Connector Toolkit	88
HubSpot Connector	101
Kerberos Connector	106
LDAP Connector	115
Marketo Connector	139
MongoDB Connector	147
MS Graph API Java Connector	154
PeopleSoft Connector	165
PowerShell Connector Toolkit	185
IBM RACF Connector	196
Salesforce Connector	212
SAP Connector	219
SCIM Connector	251
Scripted REST Connector	257
Scripted SQL Connector	266
ServiceNow Connector	278
SSH Connector	287
SAP SuccessFactors Connector	296
Workday Connector	313
3. Configure Connectors	325
Sample Provisioner Files	326
Creating Connector Configurations With the Admin UI	326
Configure Connectors Over REST	328
Setting the Connector Reference Properties	334
Setting the Pool Configuration	335
Setting the Operation Timeouts	336
Setting the Connection Configuration	337
Setting the Synchronization Failure Configuration	337
Configuring How Results Are Handled	338
Specifying What Attributes are Updated	339
Specifying the Supported Object Types	340
Configuring the Operation Options	346
4. Remote Connectors	349

Install a Remote Connector Server	349
Configure IDM to Connect to a Remote Connector Server	361
Secure the Connection to the Connector Server With SSL	371
Install Connector Dependencies	377
Example: Use the CSV Connector to Reconcile Users in a Remote CSV Data Store	379
5. Check External System Status Using REST	384
6. Remove a Connector	388
A. ICF Interfaces	389
B. ICF Operation Options	392
C. Connection Pooling Configuration	394
IDM Glossary	396







Overview

Important

Connectors continue to be updated and released outside IDM. The latest connectors guide for all ICF connectors is available [here](#).

Connectors let you connect to external resources such as LDAP, Active Directory, flat files, and others. This guide describes all the connectors supported with IDM, and how to configure them.

Quick Start

 ICF Overview Learn about the ICF framework, and how it fits into the ForgeRock Identity Management service.	 Connectors Learn about the connectors supported with IDM.	 Configure Connectors Learn how to configure connectors, and how to control what the connector synchronizes.
 Remote Connectors Manage connectors on remote systems, with connector servers.	 ICF Interfaces Discover the ICF interfaces implemented by each connector.	 Operations & Options Discover the operations and options implemented by each connector.

Configurations shown in this guide are simplified to show essential aspects. Not all resources support all IDM operations; however, the resources shown here support most of the CRUD operations, reconciliation, and liveSync.

Resources are external systems, databases, directory servers, and other sources of identity data, that are managed and audited by IDM. To connect to resources, IDM loads the ForgeRock Open Identity Connector Framework (ICF). ICF avoids the need to install agents to access resources, instead using the resources' native protocols. For example, ICF connects to database resources using the database's Java connection libraries or JDBC driver, to directory servers over LDAP, and to UNIX systems over **ssh**.

ForgeRock Identity Platform™ serves as the basis for our simple and comprehensive Identity and Access Management solution. We help our customers deepen their relationships with their

customers, and improve the productivity and connectivity of their employees and partners. For more information about ForgeRock and about the platform, see <https://www.forgerock.com>.

The ForgeRock Common REST API works across the platform to provide common ways to access web resources and collections of resources.

Chapter 1

The ForgeRock Identity Connector Framework (ICF)

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

ICF provides a common interface to allow identity services access to the resources that contain user information. IDM loads the ICF API as one of its OSGi modules. ICF uses *connectors* to separate the IDM implementation from the dependencies of the resource to which IDM is connecting. A specific connector is required for each remote resource. Connectors can run locally (on the IDM host) or remotely.

Local connectors are loaded by ICF as regular bundles in the OSGi container. Most connectors run locally. Remote connectors must be executed on a remote *connector server*. If a resource requires access libraries that cannot be included as part of the IDM process, you must use a connector server. For example, ICF connects to Microsoft Active Directory through a remote connector server that is implemented as a .NET service.

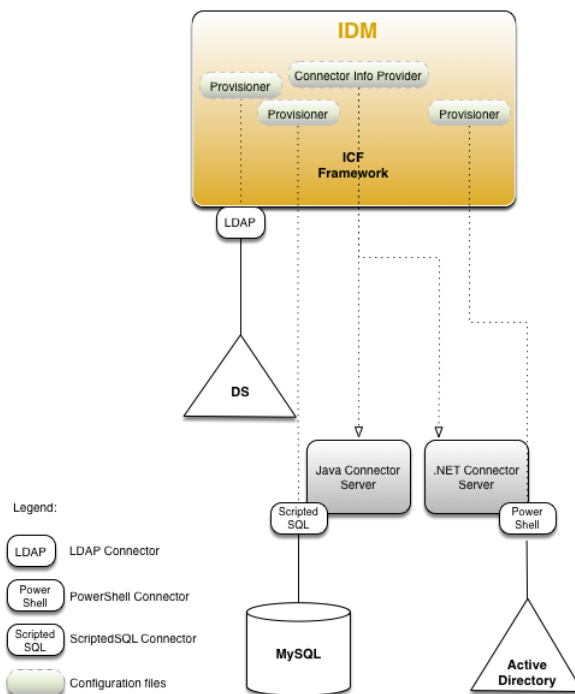
Connections to remote connector servers are configured in a single *connector info provider* configuration file, located in your project's `conf/` directory.

Connectors themselves are configured through *provisioner* files. One provisioner file must exist for each connector. Provisioner files are named `provisioner.openicf-name` where *name* corresponds to the name of the connector, and are also located in the `conf/` directory.

A number of sample connector configurations are available in the `openidm/samples/example-configurations/provisioners` directory. To use these connectors, edit the configuration files as required, and copy them to your project's `conf/` directory.

The following figure shows how IDM connects to resources by using connectors and remote connector servers. The figure shows one local connector (LDAP) and two remote connectors (Scripted SQL and PowerShell). In this example, the remote Scripted SQL connector uses a remote Java connector server. The remote PowerShell connector always requires a remote .NET connector server.

How IDM Uses the ICF Framework and Connectors



Tip

Connectors that use the .NET framework *must* run remotely. Java connectors can be run locally or remotely. You might run a Java connector remotely for security reasons (firewall constraints), for geographical reasons, or if the JVM version that is required by the connector conflicts with the JVM version that is required by IDM.

Chapter 2

Supported Connectors

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

IDM bundles connectors in the `/path/to/openidm/connectors` directory. ForgeRock supports a number of additional connectors that you can download from the [ForgeRock Download Center](#).

All the connectors described in this guide are supported. This list indicates the connectors that are *bundled* with IDM 7.0.4:

<p>+ <i>Adobe Marketing Cloud Connector</i></p> <p>The Adobe Marketing Cloud connector lets you manage profiles in an Adobe Campaign data store.</p>	<p>+ <i>AS400 Connector</i></p> <p>The AS400 connector lets you interact with AS400.</p>	<p>+ <i>CSV File Connector</i></p> <p>The CSV file connector is useful when importing users, either for initial provisioning or for ongoing updates. When used continuously in production, a CSV file serves as a change log, often containing only user records that have changed.</p>
<p>+ <i>Database Table Connector</i></p> <p>The Database Table connector enables provisioning to a single table in a JDBC database.</p>	<p>+ <i>Google Apps Connector</i></p> <p>The Google Apps connector lets you interact with Google's web applications.</p>	<p>+ <i>Groovy Connector</i></p> <p>The scripted Groovy Connector lets you run a Groovy script for any ICF operation, such as search, update, create, and others, on any external resource.</p>
<p>+ <i>Kerberos Connector</i></p> <p>The Kerberos connector is an implementation of the SSH</p>	<p>+ <i>LDAP Connector</i></p> <p>The LDAP connector is based on JNDI, and can</p>	<p>+ <i>Marketo Connector</i></p> <p>The Marketo connector lets you synchronize between IDM</p>

connector, and is based on Java Secure Channel (JSch) and the Java implementation of the Expect library (Expect4j). This connector lets you manage Kerberos user principals from IDM.	be used to connect to any LDAPv3-compliant directory server, such as ForgeRock Directory Services (DS), Active Directory, SunDS, Oracle Directory Server Enterprise Edition, IBM Security Directory Server, and OpenLDAP.	managed users and a Marketo Leads Database.
+ <i>MongoDB Connector</i>	+ <i>MS Graph API Connector</i>	+ <i>Salesforce Connector</i>
The MongoDB connector is an implementation of the Scripted Groovy Connector. This connector lets you interact with a MongoDB document database, using Groovy scripts for the ICF operations.	The MS Graph API connector lets you manage users and groups in a Microsoft Azure tenant, and lets you synchronize users and groups between IDM and Azure.	The Salesforce connector enables provisioning, reconciliation, and synchronization between Salesforce and the IDM repository.
+ <i>SCIM Connector</i>	+ <i>Scripted REST Connector</i>	+ <i>Scripted SQL Connector</i>
The SCIM connector is based on the Simple Cloud Identity Management (SCIM) protocol and lets you manage user and group accounts on any SCIM-compliant resource provider, such as Slack, Facebook or Salesforce.	The Scripted REST connector is an implementation of the Scripted Groovy Connector. This connector lets you interact with any REST API, using Groovy scripts for the ICF operations.	The Scripted SQL connector is an implementation of the Scripted Groovy Connector. This connector lets you interact with any SQL database, using Groovy scripts for the ICF operations.
+ <i>ServiceNow Connector</i>	+ <i>SSH Connector</i>	
The ServiceNow connector lets you manage objects in the ServiceNow platform,	The SSH connector is an implementation of the Scripted Groovy Connector, and	

integrating with ServiceNow's REST API.	is based on Java Secure Channel (JSch) and the Java implementation of the Expect library (Expect4j). This connector lets you interact with any SSH server, using Groovy scripts for the ICF operations.	
---	---	--

This list indicates the connectors that are not bundled with IDM 7.0.4 but available from the [ForgeRock Download Center](#):

<p>+ <i>Cerner Connector</i></p> <p>The Cerner connector lets you interact with Cerner healthcare IT systems.</p>	<p>+ <i>DocuSign Connector</i></p> <p>The DocuSign connector lets you manage DocuSign service accounts and synchronize accounts between DocuSign and the IDM managed user repository.</p>	<p>+ <i>GCP Connector</i></p> <p>The GCP connector lets you interact with the Google Cloud Platform service.</p>
<p>+ <i>HubSpot Connector</i></p> <p>The HubSpot connector lets you synchronize HubSpot contacts and companies with managed objects in an IDM repository.</p>	<p>+ <i>Peoplesoft Connector</i></p> <p>The Peoplesoft connector lets you interact with Oracle PeopleSoft systems.</p>	<p>+ <i>PowerShell Connector</i></p> <p>The PowerShell connector is not a complete connector in the traditional sense, but a framework within which you write your own PowerShell scripts to address the requirements of your Microsoft Windows ecosystem. Use this connector to create custom connectors that can provision any Microsoft system, such as Active Directory, Microsoft SQL, MS Exchange, SharePoint, Azure, and Office365.</p>
<p>+ <i>RACF Connector</i></p>	<p>+ <i>SAP Connector</i></p>	<p>+ <i>SAP SuccessFactors Connector</i></p>

<p>The RACF connector lets you interact with IBM RACF systems.</p>	<p>The SAP connector is an implementation of the Scripted Groovy Connector that connects to any SAP system using the SAP JCo Java libraries.</p>	<p>The SAP SuccessFactors connector lets you synchronize SAP SuccessFactors users with IDM managed users.</p>
<p>+ <i>Workday Connector</i></p> <p>The Workday connector lets you synchronize user accounts between IDM and Workday's cloud-based HR system.</p>		

Adobe Marketing Cloud Connector

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

The Adobe Marketing Cloud connector enables you to manage profiles in an Adobe Campaign data store. The connector supports a subset of the OpenICF operations, as listed in "OpenICF Interfaces Implemented by the Adobe Marketing Cloud Connector".

To use this connector, you need an Adobe ID.

Before You Start

The Adobe Marketing Cloud connector requires the [JSON Web Token](#) library. Before you start, download this library and copy it to the `/path/to/openidm/lib` directory.

You must also configure a new integration on AdobeIO, as shown in the following steps. Note that these steps assume a specific version of the AdobeIO user interface. For information on the current version, see the corresponding Adobe documentation.

1. The integration requires a public certificate and private key that will be used to sign the JWT token.

You can use IDM's generated self-signed certificate and private key to test the connector. In a production environment, use a CA-signed certificate and key.

Export IDM's self-signed certificate as follows:

- a. Export the certificate and key from JCEKS to standardized format PKCS #12:

```
keytool \
-importkeystore \
-srckeystore /path/to/openidm/security/keystore.jceks \
-srcstoretype jceks \
-destkeystore /path/to/keystore.p12 \
-deststoretype PKCS12 \
-srcalias openidm-localhost \
-deststorepass changeit \
-destkeypass changeit
```

- b. Export the certificate:

```
openssl pkcs12 \
-in /path/to/keystore.p12 \
-nokeys \
-out /path/to/cert.pem
```

- c. Export the unencrypted private key:

```
openssl pkcs12 \
-in /path/to/keystore.p12 \
-nodes \
-nocerts \
-out /path/to/key.pem
```

2. Log in to <https://console.adobe.io/> and select Integrations > New Integration.
3. Select Access an API > Continue.
4. Under the Experience Cloud item, select Adobe Campaign > Continue, then select New integration > Continue.
5. Enter a name for the new integration, for example, **IDM-managed**, and a short description.
6. Drag the public certificate that you exported previously into the Public keys certificates box.
7. Select a license, then select Create Integration.
8. Select Continue to integration details to obtain the Client Credentials required by the connector.

You will need these details for the connector configuration.

Configuring the Adobe Marketing Cloud Connector

Create a connector configuration file for the Adobe Marketing Cloud connector and place it in your project's `conf/` directory.

IDM bundles a sample configuration file (`/path/to/openidm/samples/example-configurations/provisioners/provisioner.openicf-adobe.json`) that you can use as a starting point. Alternatively, you can create the

configuration by using the Admin UI. Select Configure > Connectors > New Connector and select Adobe Marketing Cloud Connector - 1.5.20.11 as the connector type.

The following example shows an excerpt of the provisioner configuration. Enable the connector (set "enabled" : true) then edit at least the `configurationProperties` to match your Adobe IO setup:

```
"configurationProperties" : {  
  "endpoint" : "mc.adobe.io",  
  "imsHost" : "ims-nal.adobelogin.com",  
  "tenant" : "https://example.adobesandbox.com/",  
  "apiKey" : "",  
  "techAccId" : "example@techacct.adobe.com",  
  "orgId" : "example@AdobeOrg",  
  "clientSecret" : "CLIENT_SECRET",  
  "privateKey" : "PRIVATE_KEY"  
},  
...
```

endpoint

The Adobe IO endpoint for Marketing Cloud. `mc.adobe.io` by default - you should not have to change this value.

imsHost

The Adobe Identity Management System (IMS) host. `ims-nal.adobelogin.com` by default - you should not have to change this value.

tenant

Your tenant (organization) name or sandbox host.

apiKey

The API key (client ID) assigned to your API client account.

techAccId

Your Technical account ID, required to generate the JWT.

orgId

Your organization's unique ID, for example `12345@AdobeOrg`.

clientSecret

The client secret assigned to your API client account.

privateKey

The private key used to sign the JWT token, corresponds to the public key certificate that you attached to the integration.

For a list of all the configurable properties, see "Adobe Marketing Cloud Connector Configuration".

When your connector is configured correctly, you can test its status by running the following command:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system?_action=test"
[
  {
    "name": "adobe",
    "enabled": true,
    "config": "config/provisioner.openicf/adobe",
    "connectorRef": {
      "bundleName": "org.forgerock.openicf.connectors.adobecm-connector",
      "connectorName": "org.forgerock.openicf.acm.ACMConnector",
      "bundleVersion": "1.5.20.11"
    },
    "displayName": "Adobe Marketing Cloud Connector",
    "objectTypes": [
      "__ALL__",
      "account"
    ],
    "ok": true
  }
]
```

A status of `"ok": true` indicates that the connector can reach the configured Adobe integration.

OpenICF Interfaces Implemented by the Adobe Marketing Cloud Connector

The Adobe Marketing Cloud Connector implements the following OpenICF interfaces.

Create

Creates an object and its `uid`.

Delete

Deletes an object, referenced by its `uid`.

Schema

Describes the object types, operations, and options that the connector supports.

Script on Connector

Enables an application to run a script in the context of the connector. Any script that runs on the connector has the following characteristics:

- The script runs in the same execution environment as the connector and has access to all the classes to which the connector has access.

- The script has access to a **connector** variable that is equivalent to an initialized instance of the connector. At a minimum, the script can access the connector configuration.
- The script has access to any script-arguments passed in by the application.

Search

Searches the target resource for all objects that match the specified object class and filter.

Test

Tests the connector configuration. Testing a configuration checks all elements of the environment that are referred to by the configuration are available. For example, the connector might make a physical connection to a host that is specified in the configuration to verify that it exists and that the credentials that are specified in the configuration are valid.

This operation might need to connect to a resource, and, as such, might take some time. Do not invoke this operation too often, such as before every provisioning operation. The test operation is not intended to check that the connector is alive (that is, that its physical connection to the resource has not timed out).

You can invoke the test operation before a connector configuration has been validated.

Update

Updates (modifies or replaces) objects on a target resource.

Adobe Marketing Cloud Connector Configuration

The Adobe Marketing Cloud Connector has the following configurable properties.

Basic configuration properties

Property	Type	Default	Encrypted ^a	Required ^b
endpoint	String	mc.adobe.io		Yes
The Adobe IO endpoint for Marketing Cloud. mc.adobe.io by default - you should not have to change this.				
imsHost	String	ims-na1.adobelogin.com		Yes
Adobe Identity Management System (IMS) host. ims-na1.adobelogin.com by default - you should not have to change this.				
tenant	String	null		Yes
Your tenant (organization) name or sandbox host.				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

Adobe Integration Properties

Property	Type	Default	Encrypted ^a	Required ^b
apiKey	GuardedString	null	Yes	Yes
The API key (client ID) assigned to your API client account				
technicalAccountID	String	null		Yes
Your Technical account ID, required to generate the JWT				
organizationID	String	null		Yes
Your organizations unique ID, for example 12345@AdobeOrg				
clientSecret	GuardedString	null	Yes	Yes
The client secret assigned to your API client account				
privateKey	GuardedString	null	Yes	Yes
The private key used to sign the JWT token, corresponds to the public key certificate attached to the integration				
accessToken	GuardedString	null	Yes	No
The OAuth Access Token for the application				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

AS400 connector

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

The AS400 connector enables you to manage and synchronize users between AS400 and the IDM managed user repository.

Before You Start

These instructions assume you have an AS400 administrator account and you have access to AS400. You need the following information to configure the connector:

Host Name

The name or IP address of the host where AS400 is running.

Username

The AS400 Organizational Admin username.

Password

The AS400 Organizational Admin password.

Is Secure

Whether or not to enable a secure connection to AS400.

Install the AS400 connector

Download the connector .jar file from the [ForgeRock BackStage](#) download site.

- If you are running the connector locally, place it in the `/path/to/openidm/connectors` directory, for example:

```
mv ~/Downloads/as400-connector-1.5.20.12.jar /path/to/openidm/connectors/
```

- If you are using a remote connector server (RCS), place it in the `/path/to/openicf/connectors` directory on the RCS.

Configure the AS400 connector

Create a connector configuration using the admin UI:

1. Select **Configure > Connectors** and click **New Connector**.
2. Enter a **Connector Name**.
3. Select **AS400 Connector - 1.5.20.12** as the **Connector Type**.
4. Provide the **Base Connector Details**.
5. Click **Save**.

When your connector is configured correctly, the connector displays as **Active** in the admin UI.

Alternatively, test that the configuration is correct by running the following command:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system/as400?_action=test"
{
  "name": "as400",
  "enabled": true,
  "config": "config/provisioner.openicf/as400",
  "connectorRef": {
    "bundleVersion": "${bundleVersion}",
    "bundleName": "org.forgerock.openicf.connectors.as400-connector",
    "connectorName": "org.forgerock.openicf.connectors.as400.As400Connector"
  },
  "displayName": "AS400 Connector",
  "objectTypes": [
    "_ACCOUNT_",
    "_ALL_",
    "_GROUP_"
  ],
  "ok": true
}
```

If the command returns **"ok": true**, your connector has been configured correctly, and can authenticate to the AS400 system.

Use the AS400 connector

The following resources are supported by AS400:

ICF Native Type	AS400 Resource Type
ACCOUNT	Users
GROUP	Groups

The following filter operators and attributes are supported by AS400:

Object Type	Operators	Attributes
GROUP	id filter	Id

You can perform the following actions with the AS400 connector:

+ Create an AS400 user

The following example creates a user with all available attributes:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
```

```
--header "Content-Type: application/json"\  
--request POST \  
--data "{  
  "_NAME_": "BJENSEN",  
  "_PASSWORD_": "ASDE1234",  
  "PWDEXP": false,  
  "_ENABLE_": true,  
  "USRCLS": "*USER",  
  "ASTLVL": "*BASIC",  
  "CURLIB": "*CRTDFT",  
  "INLPGM": "*NONE",  
  "INLMNU": "MAIN",  
  "TEXT": "TEXTFILEDVALUE",  
  "SPCAUT": ["*AUDIT"],  
  "SPCENV": "*S36",  
  "DSPSGNINF": "*YES",  
  "PWDEXPITV": "323",  
  "PWDCHGBLK": "93",  
  "LCLPWDGMT": true,  
  "LMTDEVSSN": "*NO",  
  "MAXSTG": "10000",  
  "PTYLMT": 8,  
  "JOBID": "QDFTJOBID",  
  "OWNER": "*USRPRF",  
  "ACGCDE": "*BLANK",  
  "DOCPWD": "W12345",  
  "MSGQ": "*USRPRF",  
  "DLVRY": "*HOLD",  
  "SEV": "50",  
  "PRTDEV": "*SYSVAL",  
  "OUTQ": "*DEV",  
  "ATNPGM": "*ASSIST",  
  "SRTSEQ": "*HEX",  
  "LANGID": "ENG",  
  "CCSID": "*HEX",  
  "CHRIDCTL": "*DEV",  
  "SETJOBATR": ["*CCSID"],  
  "LOCALE": "*C",  
  "USROPT": ["*HLPFULL"],  
  "UID": "*GEN",  
  "HOMEDIR": "*USRPRF",  
  "EIMASSOC": ["*NOCHG"],  
  "USREXPITV": 99,  
  "USREXPDATE": "*USREXPITV",  
  "LMTCPB": "*YES",  
  "CNTRYID": "*SYSVAL",  
  "GRPPRF": "AZURE",  
  "SUPGRPPRF": ["AWS"]  
}" \  
"{secureHostname}/openidm/system/As400/__ACCOUNT__?_action=create&_prettyprint=true"  
{  
  "_id": "BJENSEN",  
  "USROPT": ["*HLPFULL"],  
  "SEV": "50",  
  "USREXPITV": 99,  
  "IsAuthCollectionActive": false,  
  "HOMEDIR": "/home/BJENSEN",  
  "MAXSTG": "10000",  
  "UID": "1277",
```

```

"PTYLMT" : 8,
"__NAME__" : "BJENSEN",
"PRTDEV" : "*SYSVAL",
"__ENABLE__" : true,
"LMTDEVSSN" : "*NO",
"__UID__" : "BJENSEN",
"SRTSEQ" : "*HEX",
"DSPSGNINF" : "*YES",
"PWDCHGBLK" : "93",
"GRPPRF" : "AZURE",
"USREXPDATE" : "12/06/22",
"CURLIB" : "*CRTDFT",
"LMTCPB" : "*YES",
"ASTLVL" : "*BASIC",
"SUPGRPPRF" : [ "AWS" ],
"MSGQ" : "/QSYS.LIB/QUSRSYS.LIB/BJENSEN.MSGQ",
"LANGID" : "ENG",
"CCSID" : "65535",
"PWDEXPITV" : "323",
"IsUserEntitlementRequired" : true,
"TEXT" : "TEXTFILEDVALUE",
"JOBID" : "/QSYS.LIB/QGPL.LIB/QDFTJOBID.JOBID",
"ActionAuditLevel" : "*BASIC",
"ObjectAuditValue" : "*NONE",
>PasswordChangedDate" : "Mon Aug 29 05:15:20 IST 2022",
"ATNPGM" : "/QSYS.LIB/QEZMAIN.PGM",
"LCLPWDMGT" : true,
"INLPGM" : "*NONE",
"USRCLS" : "*USER",
"SPCAUT" : [ "*AUDIT" ],
"SETJOBATR" : [ "*CCSID" ],
"SPCENV" : "*S36",
"ACGCDE" : "",
"IsPasswordNone" : false,
"DLVRY" : "*HOLD",
"IsAuthCollectionRepositoryExist" : false,
"UserExpirationAction" : "*DISABLE",
"INLMNU" : "/QSYS.LIB/%LIB%.LIB/MAIN.MNU",
"LOCALE" : "*C",
"KBDBUF" : "*SYSVAL",
"OWNER" : "*USRPRF",
>PasswordExpireDate" : "Tue Jul 18 00:00:00 IST 2023",
"PWDEXP" : false,
"OUTQ" : "*DEV",
"CNTRYID" : "*SYSVAL",
"CHRIDCTL" : "*DEVD",
"StorageUsed" : "12"
}

```

Note

When you create a new user, you must specify at least the `__NAME__` property. This property can be a maximum of 10 characters. These characters may be:

- Any letter
- Any digits

- The #, \$, _ and @ special characters.

If the `__NAME__` begins with a digit, it must be prefixed with a Q.

+ Query all users

The following example queries all users in the system:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/as400/__ACCOUNT__?_queryId=query-all-ids"
{
  "result": [
    {"_id": "ADAM"},
    {"_id": "BJENSEN"},
    {"_id": "CHERYL"},
    {"_id": "DAVID"},
    {"_id": "EDDIE"}
  ],
  "resultCount":5,
  "pagedResultsCookie":null,
  "totalPagedResultsPolicy":"NONE",
  "totalPagedResults":-1,
  "remainingPagedResults":-1
}
```

+ Query a single user

The following example queries all users in the system:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/as400/__ACCOUNT__/BJENSEN?prettyprint=true"
{
  "_id" : "BJENSEN",
  "USROPT" : [ "*"HLPFULL" ],
  "SEV" : "50",
  "USREXPITV" : 99,
  "IsAuthCollectionActive" : false,
  "HOMEDIR" : "/home/BJENSEN",
  "MAXSTG" : "10000",
  "UID" : "1277",
  "PTYLMT" : 8,
  "__NAME__" : "BJENSEN",
  "PRTDEV" : "*SYSVAL",
  "__ENABLE__" : true,
  "LMTDEVSSN" : "*NO",
}
```

```
{
  "UID" : "BJENSEN",
  "SRTSEQ" : "*HEX",
  "DSPSGNINF" : "*YES",
  "PWDCHGBLK" : "93",
  "GRPPRF" : "AZURE",
  "USREXPDATE" : "12/06/22",
  "CURLIB" : "*CRTDFT",
  "LMTCPB" : "*YES",
  "ASTLVL" : "*BASIC",
  "SUPGRPPRF" : [ "AWS" ],
  "MSGQ" : "/QSYS.LIB/QUSRSYS.LIB/BJENSEN.MSGQ",
  "LANGID" : "ENG",
  "CCSID" : "65535",
  "PWDEXPITV" : "323",
  "IsUserEntitlementRequired" : true,
  "TEXT" : "TEXTFILEVALUE",
  "JOBID" : "/QSYS.LIB/QGPL.LIB/QDFTJOBID.JOBID",
  "ActionAuditLevel" : "*BASIC",
  "ObjectAuditValue" : "*NONE",
  "PasswordChangedDate" : "Mon Aug 29 05:15:20 IST 2022",
  "ATNPGM" : "/QSYS.LIB/QEZMAIN.PGM",
  "LCLPWDGMT" : true,
  "INLPGM" : "*NONE",
  "USRCLS" : "*USER",
  "SPCAUT" : [ "*AUDIT" ],
  "SETJOBATR" : [ "*CCSID" ],
  "SPCENV" : "S36",
  "ACGCDE" : "",
  "IsPasswordNone" : false,
  "DLVRY" : "*HOLD",
  "IsAuthCollectionRepositoryExist" : false,
  "UserExpirationAction" : "*DISABLE",
  "INLMNU" : "/QSYS.LIB/%LIB%.LIB/MAIN.MNU",
  "LOCALE" : "C",
  "KBDBUF" : "*SYSVAL",
  "OWNER" : "*USRPRF",
  "PasswordExpireDate" : "Tue Jul 18 00:00:00 IST 2023",
  "PWDEXP" : false,
  "OUTQ" : "*DEV",
  "CNTRYID" : "*SYSVAL",
  "CHRIDCTL" : "*DEV",
  "StorageUsed" : "12"
}
```

+ Modify a user

You can modify an existing user with a PUT request, including all attributes of the account in the request. You can use the AS400 connector to modify the following attributes:

- **PASSWORD**
- **PWDEXP**
- **STATUS**
- **USRCLS**

- [ASTLVL](#)
- [CURLIB](#)
- [INLPGM](#)
- [INLMNU](#)
- [LMTCPB](#)
- [TEXT](#)
- [SPCAUT](#)
- [SPCENV](#)
- [DSPSGNINF](#)
- [PWDEXPITV](#)
- [PWDCHGBLK](#)
- [LCLPDMGT](#)
- [LMTDEVSSN](#)
- [KBDBUF](#)
- [MAXSTG](#)
- [PTYLMT](#)
- [JOBDB](#)
- [OWNER](#)
- [ACGCDE](#)
- [DOCPWD](#)
- [MSGQ](#)
- [DLVRY](#)
- [SEV](#)
- [PRTDEV](#)
- [OUTQ](#)
- [ATNPGM](#)
- [SRTSEQ](#)
- [LANGID](#)

- CNTRYID
- CCSID
- CHRIDCTL
- SETJOBATR
- LOCALE
- USROPT
- UID
- HOMEDIR
- USREXPDATE
- USREXPITV
- EIMASSOC
- GRPPRF
- SUPGRPPRF

The following request updates a user:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "Accept-API-Version: resource=1.0" \
--header "If-Match: *" \
--request PUT \
--data '{
  "_PASSWORD_": "ASDE1234",
  "PWDEXP": false,
  "_ENABLE_": true,
  "USRCLS": "*USER",
  "ASTLVL": "*BASIC",
  "CURLIB": "*CRTDFT",
  "INLPGM": "*NONE",
  "INLMNU": "MAIN",
  "TEXT": "TEXTFILEDVALUE",
  "SPCAUT": ["*AUDIT"],
  "SPCENV": "*S36",
  "DSPSGNINF": "*YES",
  "PWDEXPITV": "323",
  "PWDCHGBLK": "93",
  "LCLPWDGMT": true,
  "LMTDEVSSN": "*NO",
  "MAXSTG": "10000",
  "PTYLMT": 8,
  "JOBID": "QDFTJOBID",
  "OWNER": "*USRPRF",
  "ACGCDE": "*BLANK",
  "DOCPWD": "W12345",
}
```

```
"MSGQ": "*USRPRF",
"DLVRY": "*HOLD",
"SEV": "50",
"PRTDEV": "*SYSVAL",
"OUTQ": "*DEV",
"ATNPGM": "*ASSIST",
"SRTSEQ": "*HEX",
"LANGID": "ENG",
"CCSID": "*HEX",
"CHRIDCTL": "*DEV",
"SETJOBATR": ["*CCSID"],
"LOCALE": "*C",
"USROPT": ["*HLPFULL"],
"UID": "*GEN",
"HOMEDIR": "*USRPRF",
"EIMASSOC": ["*NOCHG"],
"USREXPITV": 99,
"USREXPDATE": "*USREXPITV",
"LMTCPB": "*YES",
"CNTRYID": "*SYSVAL",
"GRPPRF": "AZURE", "SUPGRPPRF": ["AWS"]
} \
"{secureHostname}/openidm/system/As400/__ACCOUNT__/BJENSEN_prettyprint=true"
{
  "_id": "BJENSEN",
  "USROPT": [ "HLPFULL" ],
  "SEV": "50",
  "USREXPITV": 99,
  "IsAuthCollectionActive": false,
  "HOMEDIR": "/home/BJENSEN",
  "MAXSTG": "10000",
  "UID": "1277",
  "PTYLMT": 8,
  "NAME": "BJENSEN",
  "PRTDEV": "*SYSVAL",
  "ENABLE": true,
  "LMTDEVSSN": "*NO",
  "UID": "BJENSEN",
  "SRTSEQ": "*HEX",
  "DSPSGNINE": "*YES",
  "PWDCHGBLK": "93",
  "GRPPRF": "AZURE",
  "USREXPDATE": "12/06/22",
  "CURLIB": "*CRTDFT",
  "LMTCPB": "*YES",
  "ASTLVL": "*BASIC",
  "SUPGRPPRF": [ "AWS" ],
  "MSGQ": "/QSYS.LIB/QUSRSYS.LIB/BJENSEN.MSGQ",
  "LANGID": "ENG",
  "CCSID": "65535",
  "PWDEXPITV": "323",
  "IsUserEntitlementRequired": true,
  "TEXT": "TEXTFILEDVALUE",
  "JOB": "/QSYS.LIB/QGPL.LIB/QDFTJOB.DJOB",
  "ActionAuditLevel": "*BASIC",
  "ObjectAuditValue": "*NONE",
  "PasswordChangedDate": "Mon Aug 29 05:15:20 IST 2022",
  "ATNPGM": "/QSYS.LIB/QEZMAIN.PGM",
  "LCLPWDGMT": true,
```

```
"INLPGM" : "*NONE",
"USRCLS" : "*USER",
"SPCAUT" : [ "*AUDIT" ],
"SETJOBATR" : [ "*CCSID" ],
"SPCENV" : "*S36",
"ACGCDE" : "",
"IsPasswordNone" : false,
"DLVRY" : "*HOLD",
"IsAuthCollectionRepositoryExist" : false,
"UserExpirationAction" : "*DISABLE",
"INLMNU" : "/QSYS.LIB/%LIBL%.LIB/MAIN.MNU",
"LOCALE" : "*C",
"KBDBUF" : "*SYSVAL",
"OWNER" : "*USRPRF",
"PasswordExpireDate" : "Tue Jul 18 00:00:00 IST 2023",
"PWDEXP" : false,
"OUTQ" : "*DEV",
"CNTRYID" : "*SYSVAL",
"CHRIDCTL" : "*DEV",
"StorageUsed" : "12"
}
```

+ *Reset a user's password*

To reset the password for an AS400 user account, you can use the connector to change the user's password:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "Accept-API-Version: resource=1.0" \
--header "If-Match: *" \
--request PUT \
--data "{
  \"_PASSWORD_\": \"newpassword123\"
}" \
\"{secureHostname}/openidm/system/as400/__ACCOUNT__/BJENSEN_prettyprint=true\"
{
  \"_id\" : \"BJENSEN\",
  \"USROPT\" : [ \"*HLPFULL\" ],
  \"SEV\" : \"50\",
  ...
}
```

+ *Activate a user*

The following example activates a user:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "Accept-API-Version: resource=1.0" \
--header "If-Match: *" \
--request PUT \
--data "{
  \"__ENABLE__\": true
}"
"{secureHostname}/openidm/system/as400/__ACCOUNT__/BJENSEN_prettyprint=true"
{
  "_id" : "BJENSEN",
  ...
  "__ENABLE__": true
  ...
}
```

+ *Deactivate a user*

The following example deactivates a user:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "Accept-API-Version: resource=1.0" \
--header "If-Match: *" \
--request PUT \
--data "{
  \"__ENABLE__\": false
}"
"{secureHostname}/openidm/system/as400/__ACCOUNT__/BJENSEN_prettyprint=true"
{
  "_id" : "BJENSEN",
  ...
  "__ENABLE__": false
  ...
}
```

+ *Delete a user*

The following example deletes a user:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "Accept-API-Version: resource=1.0" \
--request DELETE \
"{secureHostname}/openidm/system/as400/__ACCOUNT__/BJENSEN_prettyprint=true"
{
  "_id" : "BJENSEN",
  ...
}
```

+ Query all groups

The following example queries all AS400 Groups by their IDs:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
"http://localhost:8080/openidm/system/as400/__GROUP__?_queryId=query-all-ids&_prettyprint=true"
{
  {
    "result": [
      {"_id": "AWS"},
      {"_id": "AZURE"},
      {"_id": "CLOUD"}
    ],
    "resultCount" : 3,
    "pagedResultsCookie" : null,
    "totalPagedResultsPolicy" : "NONE",
    "totalPagedResults" : -1,
    "remainingPagedResults" : -1
  }
}
```

+ Query a single group

The following example queries a single AS400 group by its ID:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
"http://localhost:8080/openidm/system/as400/__GROUP__/AWS?prettyprint=true"
{
  "id" : "AWS",
  "GID" : "116",
  "NAME" : "AWS",
  "GRPAUT" : "*NONE",
  "GRPAUTYP" : "*PRIVATE",
  "__UID__" : "AWS"
}
```

Account attributes

The following account attributes are supported by the AS400 connector:

Attribute	Description
USPRF	User Profile Name
PASSWORD	The password used to log in.
PreviousSignOn	The previous sign-on date.
PasswordChangedDate	The last date the password was changed.
IsPasswordNone	Whether or not the password is *NONE.
UserExpirationAction	The user expiration action.
StorageUsed	The storage used.
ObjectAuditValue	A value used for auditing the object.
ActionAuditLevel	The Action Audit Level.
PWDEXP	When the user's password is set to expire.
STATUS	The user's status. Permitted values are enable and disable .
USRCLS	The special access control for the user.
ASTLVL	Specifies which user interface to use.
CURLIB	Specifies the name of the current library associated with the job.
INLPGM	The initial program.
INLMNU	The initial menu.
IsUserEntitlementRequired	Whether or not user entitlement is required.
IsAuthCollectionActive	Whether or not authority collection is active.
MTCPB	Limit capabilities.

Attribute	Description
TEXT	A free-form text field.
SPCAUT	The special access permissions for the user.
SPCENV	The special environment.
DSPSGNINF	The display sign-on information.
PWDEXPITV	The password expiration interval.
PWDCHGBLK	Whether or not to block password change.
LCLPWDMGMT	Local password management.
LMTDEVSSN	Limit device session.
KBDBUF	Keyboard buffering.
MAXSTG	Maximum allowed storage.
PTYLMT	Highest schedule priority.
JOBDESC	Job description.
OWNER	The owner of the user profile.
ACGCDE	The accounting code.
DOCPWD	The document password.
MSGQ	The message queue.
DLVRY	Delivery.
SEV	The severity code.
PRTDEV	The print device.
OUTQ	The output queue.
ATNPGM	The attention program.
SRTSEQ	The sort sequence.
LANGID	The language ID.
CNTRYID	The country or region ID.
CCSID	The Coded Character Set ID.
CHRIDCTL	The character identifier control.
SETJOBATR	The local job attributes.
LOCALE	The locale.
USROPT	The user options.
UID	The user ID number.
HOMEDIR	The home directory.
USREXPDATE	The user's expiration date.
USREXPITV	The user's expiration interval.

Attribute	Description
AUT	Authority.
EIMASSOC	The EIM association.
PasswordExpireDate	The date the password expires.
GRPPRF	Specifies the user's group profile name whose authority is used when there is no job-specific authority given to the user.
SUPGRPPRF	Specifies the user's supplemental group profiles. Used with GRPPRF to determine what authority the user has when there is no job-specific authority given to the user.

OpenICF Interfaces Implemented by the AS400 Connector

The AS400 Connector implements the following OpenICF interfaces.

Create

Creates an object and its `uid`.

Delete

Deletes an object, referenced by its `uid`.

Schema

Describes the object types, operations, and options that the connector supports.

Script on Connector

Enables an application to run a script in the context of the connector. Any script that runs on the connector has the following characteristics:

- The script runs in the same execution environment as the connector and has access to all the classes to which the connector has access.
- The script has access to a `connector` variable that is equivalent to an initialized instance of the connector. At a minimum, the script can access the connector configuration.
- The script has access to any script-arguments passed in by the application.

Search

Searches the target resource for all objects that match the specified object class and filter.

Test

Tests the connector configuration. Testing a configuration checks all elements of the environment that are referred to by the configuration are available. For example, the connector might make a physical connection to a host that is specified in the configuration to verify that it exists and that the credentials that are specified in the configuration are valid.

This operation might need to connect to a resource, and, as such, might take some time. Do not invoke this operation too often, such as before every provisioning operation. The test operation is not intended to check that the connector is alive (that is, that its physical connection to the resource has not timed out).

You can invoke the test operation before a connector configuration has been validated.

Update

Updates (modifies or replaces) objects on a target resource.

AS400 Connector Configuration

The AS400 Connector has the following configurable properties.

Configuration properties

Property	Type	Default	Encrypted ^a	Required ^b
hostName	String	null		Yes
Host name or IP address of As400				
userName	String	null		Yes
The username to login As400				
password	GuardedString	null	Yes	Yes
The password to login As400				
isSecure	boolean	true		Yes
Enable or not secure connection to As400				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

Basic configuration properties

Property	Type	Default	Encrypted ^a	Required ^b
maximumConnections	Integer	10		No
Provide the maximum connections				
connectionTimeout	Integer	300000		No
Provide the maximum connection timeout in milliseconds				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

Cerner Connector

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

Cerner is a healthcare-related service which offers an integrated healthcare IT solution for large healthcare providers. The Cerner connector lets you manage and synchronize accounts between Cerner and IDM managed user objects. A Cerner system account is required for this connector to work.

Before you start

Before you configure the connector, log in to your Cerner system account and note the following:

Bearer token

The bearer token associated with your system account.

Tenant

Your Cerner tenant ID.

Region

The Cerner Cloud region where the tenant resides.

Install the Cerner connector

Download the connector .jar file from the [ForgeRock BackStage download site](#).

- If you are running the connector locally, place it in the `/path/to/openidm/connectors` directory, for example:

```
mv ~/Downloads/cerner-connector-1.5.20.12.jar /path/to/openidm/connectors/
```

- If you are using a remote connector server (RCS), place it in the `/path/to/openicf/connectors` directory on the RCS.

Configure the Cerner connector

Create a connector configuration using the Admin UI:

1. Select **Configure > Connectors** and click **New Connector**.
2. Enter a Connector Name.

3. Select Cerner Connector - 1.5.20.12 as the Connector Type.
4. Provide the Base Connector Details.
5. Click Save.

When your connector is configured correctly, the connector displays as Active in the Admin UI.

Alternatively, test that the configuration is correct by running the following command:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system/Cerner?_action=test"
{
  "name": "Cerner",
  "enabled": true,
  "config": "config/provisioner.openicf/Cerner",
  "connectorRef": {
    "bundleVersion": "${bundleVersion}",
    "bundleName": "org.forgerock.openicf.connectors.cerner-connector",
    "connectorName": "org.forgerock.openicf.connectors.cerner.CernerConnector"
  },
  "displayName": "Cerner Connector",
  "objectTypes": [
    "__ORGANIZATION__",
    "__ACCOUNT__",
    "__ORGANIZATIONGROUP__",
    "__ALL__",
    "__PERSONNELGROUP__"
  ],
  "ok": true
}
```

If the command returns `"ok": true`, your connector was configured correctly, and can authenticate to the Cerner system.

Use the Cerner connector

Supported object types

Connector resource	Cerner resource type
<code>__ACCOUNT__</code>	Personnel
<code>__ORGANIZATION__</code>	Organization
<code>__PERSONNELGROUP__</code>	Personnel Group
<code>__ORGANIZATIONGROUP__</code>	Organization Group

__ACCOUNT__ attributes

Attribute	Notes	
<u>__NAME__</u>	The user's name, in a FAMILY , GIVEN format. Required.	
birthDate	Must be in YYYY-MM-DD format.	
gender	Accepted values are MALE , FEMALE , OTHER , UNKNOWN .	
given	The user's first name. Required.	
family	The user's last name. Required.	
name	given	
	middle	
	family	
	suffix	
	prefix	
addresses	postalCode	
	country	
	use	Accepted values are HOME , WORK .
	city	
	state	
	lines	The street portion of the address.
aliasType	Accepted values are: SPI , TAX , SL , EXTERNAL , UPIN , USER , or UNKNOWN . Required.	
aliasValue		
aliasSystem		
sourceIdentifiers	id	
	dataPartitionId	
qualifications	issuer	
	code	Qualification code such as MD or PhD . Accepted values are: AA , AAS , ABA , AE , AS , BA , BBA , BE , BFA , BN , BS , BSL , BSN , BT , CANP , CER , CMA , CNM , CNP , CNS , CPNP , CRN , CTR , DBA , DED , DIP , DO , EMT , EMTP , FPNP , HS , JD , MA , MBA , MCE , MD , MDA , MDI , ME , MED , MEE , MFA , MME , MS , MSL , MSN , MT , MTH , NG , NP , PA , PHD , PHE ,

Attribute	Notes	
		PNS, PN, PharmD, RMA, RN, RPH, SEC, or TS.
	start	The first date and time that the qualification is valid, in a YYYY-MM-DDThh:mm:ssZ date format.
	end	The date and time that the qualification expires, in a YYYY-MM-DDThh:mm:ssZ date format.
telecoms	system	Accepted values are PHONE, EMAIL, or OTHER.
	value	
languages	For a list of valid language tags, refer to the <i>Internet Assigned Numbers Authority</i> (IANA) language subtag registry.	

__ORGANIZATION__ attributes

Attribute	Notes	
__NAME__	The name of the organization. This corresponds to aliasValue, aliasSystem, comma separated. Required.	
name	The name of the organization. Required.	
aliasType	Alias types related to the organization. DEA, TAX, SOI, and NPI are supported for queries. Organizations with NPI and DEA cannot be created or updated.	
telecoms	system	Accepted values are PHONE, EMAIL, or OTHER.
	value	
addresses	postalCode	
	country	
	text	Formatted display text of the address.
	city	
	state	
	lines	The street portion of the address.
aliases	type	Types of alias for the organization.
	system	
	value	

Attribute	Notes
languages	For a list of valid language tags, refer to the <i>Internet Assigned Numbers Authority</i> (IANA) language subtag registry.
coverageAreaPostalCodes	The postal codes indicating the area of coverage provided by the organization.
sourceIdentifiers	id
	dataPartitionId

PERSONNELGROUP attributes

Attribute	Notes
<u>NAME</u>	A comma-separated name for the personnel group.
mnemonic	The mnemonic determines the function of the personnel group.
mnemonicType	The type of the personnel group mnemonic. Usually either SINGLETON or MULTIVALUED .
name	The name of the personnel group.
aliases	type
	system
	value
aliasType	The type of alias. Requires aliasValue and aliasSystem .
aliasSystem	The source of the alias value. Requires aliasType and aliasValue .
aliasValue	The unique identifier of alias. Requires aliasType and aliasSystem .

ORGANIZATIONGROUP attributes

Attribute	Notes
<u>NAME</u>	A comma-separated name for the organization group.
organizationId	A list of organization IDs that are members of the organization group.
name	The name of the organization group.
aliases	type
	system
	value
aliasType	The type of alias. Requires aliasValue and aliasSystem .
aliasSystem	The source of the alias value. Requires aliasType and aliasValue .
aliasValue	The unique identifier of alias. Requires aliasType and aliasSystem .

You can use the Cerner connector to perform the following actions on a Cerner account:

+ Create a Cerner user

The following example creates a user with the minimum required attributes:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
--data '{
  "given": "Barbara",
  "family": "Jensen",
  "aliasType": "USER",
  "__NAME__": "Jensen, Barbara"
}' \
"http://localhost:8080/openidm/system/Cerner/__ACCOUNT__?_action=create"
{
  "_id": "5170a9cd-e501-4cbf-a1bf-9e6d293362c6",
  "updatedAt": "2022-04-29T22:54:08Z",
  "given": "Barbara",
  "name": {
    "given": "Barbara",
    "family": "Jensen",
    "formatted": "Barbara Jensen"
  },
  "id": "5170a9cd-e501-4cbf-a1bf-9e6d293362c6",
  "languages": [],
  "formattedName": "Barbara Jensen",
  "aliases": {
    "type": "USER",
    "value": "Jensen",
    "system": "Barbara"
  },
  "aliasValue": "Jensen",
  "__NAME__": "Jensen,Barbara",
  "createdAt": "2022-04-29T22:54:08Z",
  "aliasType": "USER",
  "family": "Jensen",
  "isManual": true,
  "aliasSystem": "Barbara"
}
```

Note

When you create a new user, you must specify *at least* `__NAME__`, `aliasType`, `given` and `family`. Refer to the list of available attributes above for more information.

+ Update a Cerner user entry

You can modify an existing user with a PUT request, including all attributes of the account in the request.:

For example, to add the user's middle name:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--header "if-Match:*" \
--request PUT \
--data '{
  "given": "Barbara",
  "family": "Jensen",
  "aliasType": "USER",
  "NAME": "Jensen, Barbara",
  "name": {
    "middle": "Simone"
  }
}' \
"http://localhost:8080/openidm/system/Cerner/_ACCOUNT_/5170a9cd-e501-4cbf-a1bf-9e6d293362c6"

{
  "id": "5170a9cd-e501-4cbf-a1bf-9e6d293362c6",
  "updatedAt": "2022-04-29T23:03:57Z",
  "given": "Barbara",
  "name": {
    "given": "Barbara",
    "middle": "Simone",
    "family": "Jensen",
    "formatted": "Barbara Simone Jensen"
  },
  "id": "5170a9cd-e501-4cbf-a1bf-9e6d293362c6",
  "languages": [],
  "formattedName": "Barbara Simone Jensen",
  "aliases": {
    "type": "USER",
    "value": "Jensen",
    "system": "Barbara"
  },
  "aliasValue": "Jensen",
  "NAME": "Jensen,Barbara",
  "createdAt": "2022-04-29T22:54:08Z",
  "aliasType": "USER",
  "family": "Jensen",
  "isManual": true,
  "aliasSystem": "Barbara"
}
```

+ Query Cerner users

The following example queries all Cerner users:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request GET \
```

```
"http://localhost:8080/openidm/system/Cerner/__ACCOUNT__?_queryId=query-all-ids"
{
  "result": [
    {
      "_id": "7d9538c8-1c2a-4894-a403-129b35308f39"
    },
    {
      "_id": "8f1c2671-9ebb-4105-9537-a3a0fc24afce"
    },
    {
      "_id": "ac944860-705f-4487-99bf-6959c5e6157c"
    },
    {
      "_id": "d308e459-51fa-469a-a07e-72f96906a4b4"
    },
    {
      "_id": "ff9d6902-20be-4c6e-821a-5a0f3ccaebc8"
    },
    {
      "_id": "bf2b9346-715e-4f59-9dc5-2bc89b8216cd"
    },
    {
      "_id": "055def33-a845-4100-bcd1-2b59a3526ec5"
    },
    {
      "_id": "167609b8-dfd0-4302-9022-4a3e8809b166"
    },
    [ ... ]
    {
      "_id": "9f4ea23d-bacc-46ee-b8c9-75916a5f5128"
    },
    {
      "_id": "a4d6be21-a5ce-4a56-91af-94c627701d4f"
    }
  ],
  "resultCount": 1020,
  "pagedResultsCookie": null,
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

Note

Querying all ids can take a significant amount of time to return when the data set is large. Consider using paginated results instead, for example:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/Cerner/___ACCOUNT___?
queryFilter=true&_fields=_id&_pageSize=2&_pagedResultsOffset=50"
{
  "result": [
    {
      "_id": "878c87d4-8322-4908-a858-555a1cb45e36"
    },
    {
      "_id": "9ecaa98b-58df-4dd1-bc99-34341411b151"
    }
  ],
  "resultCount": 2,
  "pagedResultsCookie": null,
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

The following command queries a specific user by their ID:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/Cerner/___ACCOUNT___/5170a9cd-e501-4cbf-a1bf-9e6d293362c6"
{
  "_id": "5170a9cd-e501-4cbf-a1bf-9e6d293362c6",
  "updatedAt": "2022-04-29T23:03:57Z",
  "given": "Barbara",
  "name": {
    "given": "Barbara",
    "middle": "Simone",
    "family": "Jensen",
    "formatted": "Barbara Simone Jensen"
  },
  "id": "5170a9cd-e501-4cbf-a1bf-9e6d293362c6",
  "languages": [],
  "formattedName": "Barbara Simone Jensen",
  "aliases": {
    "type": "USER",
    "value": "Jensen",
    "system": "Barbara"
  },
  "aliasValue": "Jensen",
  "___NAME___": "Jensen,Barbara",
  "createdAt": "2022-04-29T22:54:08Z",
  "aliasType": "USER",
  "family": "Jensen",
  "isManual": true,
}
```

```
{
  "aliasSystem": "Barbara"
}
```

+ Close a Cerner user account

You can use the Cerner connector to delete an account from the Cerner repository.

The following example deletes a Cerner account:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request DELETE \
"http://localhost:8080/openidm/system/Cerner/___ACCOUNT___/5170a9cd-e501-4cbf-a1bf-9e6d293362c6"
{
  "_id": "5170a9cd-e501-4cbf-a1bf-9e6d293362c6",
  "updatedAt": "2022-04-29T23:03:57Z",
  "given": "Barbara",
  "name": {
    "given": "Barbara",
    "middle": "Simone",
    "family": "Jensen",
    "formatted": "Barbara Simone Jensen"
  },
  "id": "5170a9cd-e501-4cbf-a1bf-9e6d293362c6",
  "languages": [],
  "formattedName": "Barbara Simone Jensen",
  "aliases": {
    "type": "USER",
    "value": "Jensen",
    "system": "Barbara"
  },
  "aliasValue": "Jensen",
  "___NAME___": "Jensen,Barbara",
  "createdAt": "2022-04-29T22:54:08Z",
  "aliasType": "USER",
  "family": "Jensen",
  "isManual": true,
  "aliasSystem": "Barbara"
}
```

You can then confirm the account has been deleted by querying the [id](#):

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/Cerner/___ACCOUNT___/5170a9cd-e501-4cbf-a1bf-9e6d293362c6"
{
  "code": 404,
  "reason": "Not Found",
  "message": "Object 5170a9cd-e501-4cbf-a1bf-9e6d293362c6 not found on system/Cerner/___ACCOUNT___"
}
```

All supported resources can be queried. You can update user accounts, organizations, organization groups, and personnel groups, but only user accounts can be created or deleted. Available additional operations include:

+ *Assign personnel groups to a user*

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--header "if-Match:*" \
--request PUT \
--data '{
  "given": "Barbara",
  "family": "Jensen",
  "aliasType": "USER",
  "___NAME___": "Jensen, Barbara",
  "name": {
    "middle": "Simone"
  },
  "personnelGroupId": [
    "8636d4c3-de7c-4f8a-828b-b709d6bfd636"
  ]
}' \
"http://localhost:8080/openidm/system/Cerner/___ACCOUNT___/5170a9cd-e501-4cbf-a1bf-9e6d293362c6"
{
  "_id": "5170a9cd-e501-4cbf-a1bf-9e6d293362c6",
  "formattedName": "Barbara Simone Jensen",
  "___NAME___": "Jensen, Barbara",
  "aliasValue": "Jensen",
  "family": "Jensen",
  "updatedAt": "2022-10-25T23:50:31Z",
  "aliasType": "USER",
  "given": "Barbara",
  "organizationId": [],
  "aliasSystem": "Barbara",
  "name": {
    "given": "Barbara",
    "middle": "Simone",
    "family": "Jensen",
    "formatted": "Barbara Simone Jensen"
  },
}
```

```
{
  "languages": [],
  "id": "5170a9cd-e501-4cbf-a1bf-9e6d293362c6",
  "isManual": true,
  "personnelGroupId": [
    "8636d4c3-de7c-4f8a-828b-b709d6bfd636"
  ],
  "aliases": {
    "type": "USER",
    "value": "Jensen",
    "system": "Barbara"
  },
  "createdAt": "2022-04-29T22:54:08Z"
}
```

+ Remove a user from a personnel group

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--header "if-Match:*" \
--request PUT \
--data '{
  "given": "Barbara",
  "family": "Jensen",
  "aliasType": "USER",
  "__NAME__": "Jensen, Barbara",
  "name": {
    "middle": "Simone"
  },
  "personnelGroupId": []
}' \
"http://localhost:8080/openidm/system/Cerner/___ACCOUNT___/5170a9cd-e501-4cbf-a1bf-9e6d293362c6"
{
  "_id": "5170a9cd-e501-4cbf-a1bf-9e6d293362c6",
  "formattedName": "Barbara Simone Jensen",
  "__NAME__": "Jensen, Barbara",
  "aliasValue": "Jensen",
  "family": "Jensen",
  "updatedAt": "2022-10-26T00:03:40Z",
  "aliasType": "USER",
  "given": "Barbara",
  "organizationId": [],
  "aliasSystem": "Barbara",
  "name": {
    "given": "Barbara",
    "middle": "Simone",
    "family": "Jensen",
    "formatted": "Barbara Simone Jensen"
  },
  "languages": [],
  "id": "5170a9cd-e501-4cbf-a1bf-9e6d293362c6",
  "isManual": true,
  "personnelGroupId": [],
  "aliases": {
    "type": "USER",
```

```
{
  "value": "Jensen",
  "system": "Barbara"
},
"createdAt": "2022-04-29T22:54:08Z"
}
```

+ Assign an organization member

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--header "if-Match:*" \
--request PUT \
--data '{
  "given": "Barbara",
  "family": "Jensen",
  "aliasType": "USER",
  "__NAME__": "Jensen, Barbara",
  "name": {
    "middle": "Simone"
  },
  "organizationId": [
    "c66f037b-50f5-4703-b51f-838f42a49e84"
  ]
}' \
"http://localhost:8080/openidm/system/Cerner/___ACCOUNT___/5170a9cd-e501-4cbf-a1bf-9e6d293362c6"
{
  "id": "5170a9cd-e501-4cbf-a1bf-9e6d293362c6",
  "formattedName": "Barbara Simone Jensen",
  "__NAME__": "Jensen, Barbara",
  "aliasValue": "Jensen",
  "family": "Jensen",
  "updatedAt": "2022-10-26T00:03:40Z",
  "aliasType": "USER",
  "given": "Barbara",
  "organizationId": [
    "c66f037b-50f5-4703-b51f-838f42a49e84"
  ],
  "aliasSystem": "Barbara",
  "name": {
    "given": "Barbara",
    "middle": "Simone",
    "family": "Jensen",
    "formatted": "Barbara Simone Jensen"
  },
  "languages": [],
  "id": "5170a9cd-e501-4cbf-a1bf-9e6d293362c6",
  "isManual": true,
  "personnelGroupId": [],
  "aliases": {
    "type": "USER",
    "value": "Jensen",
    "system": "Barbara"
  },
  "createdAt": "2022-04-29T22:54:08Z"
```

```
}

```

+ Remove an organization member

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--header "if-Match:*" \
--request PUT \
--data '{
  "given": "Barbara",
  "family": "Jensen",
  "aliasType": "USER",
  "NAME": "Jensen, Barbara",
  "name": {
    "middle": "Simone"
  },
  "organizationId": []
}' \
"http://localhost:8080/openidm/system/Cerner/___ACCOUNT___/5170a9cd-e501-4cbf-a1bf-9e6d293362c6"
{
  "id": "5170a9cd-e501-4cbf-a1bf-9e6d293362c6",
  "formattedName": "Barbara Simone Jensen",
  "NAME": "Jensen, Barbara",
  "aliasValue": "Jensen",
  "family": "Jensen",
  "updatedAt": "2022-10-26T00:03:40Z",
  "aliasType": "USER",
  "given": "Barbara",
  "organizationId": [],
  "aliasSystem": "Barbara",
  "name": {
    "given": "Barbara",
    "middle": "Simone",
    "family": "Jensen",
    "formatted": "Barbara Simone Jensen"
  },
  "languages": [],
  "id": "5170a9cd-e501-4cbf-a1bf-9e6d293362c6",
  "isManual": true,
  "personnelGroupId": [],
  "aliases": {
    "type": "USER",
    "value": "Jensen",
    "system": "Barbara"
  },
  "createdAt": "2022-04-29T22:54:08Z"
}
```

+ Assign an organization to an organization group

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
```

```
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--header "if-Match:*" \
--request PUT \
--data '{
  "organizationId": [
    "f90a6224-1880-4935-a838-e19d3079a23c",
    "19b5157e-6fbe-4716-860b-28d6df90f331",
    "c66f037b-50f5-4703-b51f-838f42a49e84"
  ]
}' \
"http://localhost:8080/openidm/system/Cerner/__ORGANIZATIONGROUP__/67203020-aae7-4f44-865f-c8591d618ffc"
{
  "_id": "67203020-aae7-4f44-865f-c8591d618ffc",
  "organizationId": [
    "c66f037b-50f5-4703-b51f-838f42a49e84",
    "f90a6224-1880-4935-a838-e19d3079a23c",
    "19b5157e-6fbe-4716-860b-28d6df90f331"
  ],
  "updatedAt": "2022-05-06T12:56:02Z",
  "aliases": {
    "type": "SOGI",
    "value": "00010RGVALUE",
    "system": "0001System"
  },
  "id": "67203020-aae7-4f44-865f-c8591d618ffc",
  "aliasType": "SOGI",
  "aliasValue": "00010RGVALUE",
  "aliasSystem": "0001System",
  "name": "ABC SK ORG GROUP",
  "createdAt": "2022-05-06T12:56:02Z",
  "__NAME__": "00010RGVALUE,0001System"
}
```

+ Remove an organization from an organization group

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--header "if-Match:*" \
--request PUT \
--data '{
  "organizationId": [
    "f90a6224-1880-4935-a838-e19d3079a23c",
    "19b5157e-6fbe-4716-860b-28d6df90f331"
  ]
}' \
"http://localhost:8080/openidm/system/Cerner/__ORGANIZATIONGROUP__/67203020-aae7-4f44-865f-c8591d618ffc"
{
  "_id": "67203020-aae7-4f44-865f-c8591d618ffc",
  "organizationId": [
    "f90a6224-1880-4935-a838-e19d3079a23c",

```

```

    "19b5157e-6fbe-4716-860b-28d6df90f331"
  ],
  "updatedAt": "2022-05-06T12:56:02Z",
  "aliases": {
    "type": "SOGI",
    "value": "00010RGVALUE",
    "system": "0001System"
  },
  "id": "67203020-aae7-4f44-865f-c8591d618ffc",
  "aliasType": "SOGI",
  "aliasValue": "00010RGVALUE",
  "aliasSystem": "0001System",
  "name": "ABC SK ORG GROUP",
  "createdAt": "2022-05-06T12:56:02Z",
  "__NAME__": "00010RGVALUE,0001System"
}

```

OpenICF Interfaces Implemented by the Cerner Connector

The Cerner Connector implements the following OpenICF interfaces.

Create

Creates an object and its **uid**.

Delete

Deletes an object, referenced by its **uid**.

Schema

Describes the object types, operations, and options that the connector supports.

Script on Connector

Enables an application to run a script in the context of the connector. Any script that runs on the connector has the following characteristics:

- The script runs in the same execution environment as the connector and has access to all the classes to which the connector has access.
- The script has access to a **connector** variable that is equivalent to an initialized instance of the connector. At a minimum, the script can access the connector configuration.
- The script has access to any script-arguments passed in by the application.

Search

Searches the target resource for all objects that match the specified object class and filter.

Test

Tests the connector configuration. Testing a configuration checks all elements of the environment that are referred to by the configuration are available. For example, the connector might make a physical connection to a host that is specified in the configuration to verify that it exists and that the credentials that are specified in the configuration are valid.

This operation might need to connect to a resource, and, as such, might take some time. Do not invoke this operation too often, such as before every provisioning operation. The test operation is not intended to check that the connector is alive (that is, that its physical connection to the resource has not timed out).

You can invoke the test operation before a connector configuration has been validated.

Update

Updates (modifies or replaces) objects on a target resource.

Cerner Connector Configuration

The Cerner Connector has the following configurable properties.

Configuration properties

Property	Type	Default	Encrypted ^a	Required ^b
bearerToken	GuardedString	null	Yes	Yes
Provide the bearer token to authorize Cerner				
tenant	String	playground		No
Provide the tenant to authorize Cerner				
region	String	us-1		No
Provide the region to authorize Cerner				
maximumConnections	Integer	10		No
Provide the maximum connections				
connectionTimeout	Integer	300		No
Provide the maximum connection timeout in seconds				
httpProxyHost	String	null		Yes
Provide the Proxy Host				
httpProxyPort	Integer	null		Yes

Property	Type	Default	Encrypted ^a	Required ^b
Provide the Proxy Port				
<code>httpProxyUsername</code>	String	null		Yes
Provide the Proxy Username				
<code>httpProxyPassword</code>	GuardedString	null	Yes	Yes
Provide the Proxy Password				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

CSV File Connector

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

The CSV file connector is useful when importing users, either for initial provisioning or for ongoing updates. When used continuously in production, a CSV file serves as a change log, often containing only user records that have changed.

Important

This connector does not verify CSV data before attempting a synchronization. You must ensure that your CSV file is complete and properly formed before using the connector.

Configuring the CSV File Connector

A sample CSV file connector configuration is provided in `openidm/samples/example-configurations/provisioners/provisioner.openicf-csvfile.json`.

The following example shows an excerpt of the provisioner configuration. The `connectorHostRef` property is optional and must be provided only if the connector runs remotely.

```
{
  "connectorRef": {
    "connectorHostRef": "#LOCAL",
    "connectorName": "org.forgerock.openicf.csvfile.CSVFileConnector",
    "bundleName": "org.forgerock.openicf.connectors.csvfile-connector",
    "bundleVersion": "[1.5.0.0,1.6.0.0)"
  }
}
```

The only *required* configuration property is the path to the `csvFile`:

```
"configurationProperties" : {  
  "csvFile" : "&{idm.instance.dir}/data/csvConnectorData.csv"  
}
```

For a list of all configuration properties for this connector, see "Configuration properties".

Important

If you change the structure of the CSV file resource, by adding or removing columns, you *must* update the corresponding object **properties** in the provisioner file accordingly.

OpenICF Interfaces Implemented by the CSV File Connector

The CSV File Connector implements the following OpenICF interfaces.

Authenticate

Provides simple authentication with two parameters, presumed to be a user name and password.

Batch

Execute a series of operations in a single request.

Create

Creates an object and its **uid**.

Delete

Deletes an object, referenced by its **uid**.

Resolve Username

Resolves an object by its username and returns the **uid** of the object.

Schema

Describes the object types, operations, and options that the connector supports.

Script on Connector

Enables an application to run a script in the context of the connector. Any script that runs on the connector has the following characteristics:

- The script runs in the same execution environment as the connector and has access to all the classes to which the connector has access.

- The script has access to a `connector` variable that is equivalent to an initialized instance of the connector. At a minimum, the script can access the connector configuration.
- The script has access to any script-arguments passed in by the application.

Search

Searches the target resource for all objects that match the specified object class and filter.

Sync

Polls the target resource for synchronization events, that is, native changes to objects on the target resource.

Test

Tests the connector configuration. Testing a configuration checks all elements of the environment that are referred to by the configuration are available. For example, the connector might make a physical connection to a host that is specified in the configuration to verify that it exists and that the credentials that are specified in the configuration are valid.

This operation might need to connect to a resource, and, as such, might take some time. Do not invoke this operation too often, such as before every provisioning operation. The test operation is not intended to check that the connector is alive (that is, that its physical connection to the resource has not timed out).

You can invoke the test operation before a connector configuration has been validated.

Update

Updates (modifies or replaces) objects on a target resource.

CSV File Connector Configuration

The CSV File Connector has the following configurable properties.

Configuration properties

Property	Type	Default	Encrypted ^a	Required ^b
<code>headerPassword</code>	String	password		No
The CSV header that maps to the password for each row. Use this property when password-based authentication is required.				
<code>spaceReplacementString</code>	String			No
The character(s) used to replace spaces within column names.				

Property	Type	Default	Encrypted ^a	Required ^b
<code>csvFile</code>	File	null		Yes
The full path to the CSV file that is the data source for this connector.				
<code>newlineString</code>	String	\n		No
The character string in the CSV file that is used to terminate each line.				
<code>headerUid</code>	String	uid		No
The CSV header that maps to the uid (or name) for each row.				
<code>quoteCharacter</code>	String	"		No
The character in the CSV file that is used to encapsulate strings.				
<code>escapeCharacter</code>	String	\		No
The character in the CSV file that is used to escape characters.				
<code>fieldDelimiter</code>	String	,		No
The character in the CSV file that is used to separate field values.				
<code>syncFileRetentionCount</code>	int	3		No
The number of historical copies of the CSV file to retain when performing synchronization operations.				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

Database Table Connector

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

The Database Table connector enables provisioning to a single table in a JDBC database.

Configuring the Database Table Connector

A sample connector configuration for the Database Table connector is provided in `samples/example-configurations/provisioners/provisioner.openicf-contractordb.json`. The corresponding data definition language file is provided in `samples/example-configurations/provisioners/provisioner.openicf-contractordb.sql`.

The following excerpt shows the settings for the connector configuration properties in the sample Database Table connector:

```
"configurationProperties" : {
  "url" : "jdbc:mysql://localhost:3306/contractordb?serverTimezone=UTC",
  "driverClassName" : "com.mysql.jdbc.Driver",
  "username" : "root",
  "password" : "password",
  "table" : "people",
  "keyColumn" : "EMAIL",
  "passwordColumn" : "",
  "changeLogColumn" : "CHANGE_TIMESTAMP",
  "disablePaging" : false,
  "enableEmptyString" : false,
  "quoting" : "",
  "rethrowAllSQLExceptions" : true,
  "nativeTimestamps" : false,
  "allNative" : false,
  "suppressPassword" : true,
  "validationQueryTimeout" : -1,
  "validationQuery" : "SELECT 1 FROM DUAL",
  "validationInterval" : 3000,
  "initialSize" : 10,
  "maxIdle" : 100,
  "minIdle" : 10,
  "maxWait" : 30000,
  "maxActive" : 100,
  "maxAge" : 0,
  "minEvictableIdleTimeMillis" : 60000,
  "timeBetweenEvictionRunsMillis" : 5000,
  "testWhileIdle" : false,
  "testOnBorrow" : true
}
```

The mandatory configurable properties are as follows:

url

The JDBC database address that contains the table to which you are provisioning. The format of the url will change depending on the type of database, such as `jdbc:mysql://localhost:3306/contractordb?serverTimezone=UTC`, or `jdbc:oracle:thin:@//localhost:3306/contractordb`. Note that the address includes the name of the database you are connecting to.

driverClassName

The class name of the driver you are using to connect to a database. The name varies depending on the type of database you are using, such as `oracle.jdbc.OracleDriver`, or `com.mysql.jdbc.Driver`.

table

The name of the table in the JDBC database that contains the user accounts.

keyColumn

The column value that is used as the unique identifier for rows in the table.

Note

If you want to map `_NAME` or `UID` to an attribute in IDM, change the `keyColumn` to a column in the SQL schema that does not match any of the target properties mapped in `sync.json`; otherwise, a conflict occurs and IDM does not create the account. Previously, this column was `UNIQUE_ID`.

Unless the database is configured to not need authentication, `username` and `password` are also required.

Tomcat JDBC connection pool

The Database Table connector uses the Apache Tomcat JDBC Connection Pool. Additional configurable properties and information are available in the [Apache Tomcat documentation](#).

Implementation Specifics

- To use this connector for liveSync, add a changelog type column to the database and provide the name of this column in the `changeLogColumn` property. Note that the Database Table connector supports liveSync for create and update operations only. To detect deletes in the database you must run a full reconciliation.
- For PATCH requests, a connector can potentially add, remove, or replace an attribute value. The Database Table connector does not implement the add or remove operations, so a PATCH request always replaces the entire attribute value with the new value.
- The Database Table connector supports paged reconciliation queries *only* for the following databases:
 - MySQL
 - PostgreSQL
 - Oracle Database 12c and later versions
 - Microsoft SQL Server 2012 and later versions

Important

Paging is enabled by default. If you are connecting to a database for which paging is not supported, you must disable it by setting `"disablePaging" : true` in the connector configuration.

For more information about configuring paged reconciliation queries, see "Paging Reconciliation Query Results" in the *Synchronization Guide*.

- If your database does not support precise (nanosecond) timestamps, you can use the `inclusiveSync` configuration property to ensure that modified entries are not missed in liveSync operations. If `inclusiveSync` is set to `true`, the connector synchronizes all entries whose change timestamp is greater than or equal to the `syncToken`. Be aware that if you set this property to `true`, the activity log

creates a new entry *every time* liveSync occurs, even if entries are changed. This can lead to rapid growth of the activity audit log.

OpenICF Interfaces Implemented by the Database Table Connector

The Database Table Connector implements the following OpenICF interfaces.

Authenticate

Provides simple authentication with two parameters, presumed to be a user name and password.

Create

Creates an object and its `uid`.

Delete

Deletes an object, referenced by its `uid`.

Resolve Username

Resolves an object by its username and returns the `uid` of the object.

Schema

Describes the object types, operations, and options that the connector supports.

Script on Connector

Enables an application to run a script in the context of the connector. Any script that runs on the connector has the following characteristics:

- The script runs in the same execution environment as the connector and has access to all the classes to which the connector has access.
- The script has access to a `connector` variable that is equivalent to an initialized instance of the connector. At a minimum, the script can access the connector configuration.
- The script has access to any script-arguments passed in by the application.

Search

Searches the target resource for all objects that match the specified object class and filter.

Sync

Polls the target resource for synchronization events, that is, native changes to objects on the target resource.

Test

Tests the connector configuration. Testing a configuration checks all elements of the environment that are referred to by the configuration are available. For example, the connector might make a

physical connection to a host that is specified in the configuration to verify that it exists and that the credentials that are specified in the configuration are valid.

This operation might need to connect to a resource, and, as such, might take some time. Do not invoke this operation too often, such as before every provisioning operation. The test operation is not intended to check that the connector is alive (that is, that its physical connection to the resource has not timed out).

You can invoke the test operation before a connector configuration has been validated.

Update

Updates (modifies or replaces) objects on a target resource.

Database Table Connector Configuration

The Database Table Connector has the following configurable properties.

Configuration properties

Property	Type	Default	Encrypted ^a	Required ^b
<code>connectionProperties</code>	<code>String</code>	<code>null</code>		No
The connection properties that will be sent to our JDBC driver when establishing new connections. Format of the string must be [propertyName=property;]* NOTE - The "user" and "password" properties will be passed explicitly, so they do not need to be included here. The default value is null.				
<code>propagateInterruptState</code>	<code>boolean</code>	<code>false</code>		No
Set this to true to propagate the interrupt state for a thread that has been interrupted (not clearing the interrupt state). Default value is false for backwards compatibility.				
<code>useDisposableConnectionFacade</code>	<code>boolean</code>	<code>true</code>		No
Set this to true if you wish to put a facade on your connection so that it cannot be reused after it has been closed. This prevents a thread holding on to a reference of a connection it has already called closed on, to execute queries on it.				
<code>defaultCatalog</code>	<code>String</code>	<code>null</code>		No
The default catalog of connections created by this pool.				
<code>validationInterval</code>	<code>long</code>	<code>3000</code>		No
To avoid excess validation, run validation at most at this frequency (in milliseconds). If a connection is due for validation, but was validated within this interval, it will not be validated again. The default value is 3000 (3 seconds).				
<code>ignoreExceptionOnPreLoad</code>	<code>boolean</code>	<code>false</code>		No

Property	Type	Default	Encrypted ^a	Required ^b
Flag whether ignore error of connection creation while initializing the pool. Set to true if you want to ignore error of connection creation while initializing the pool. Set to false if you want to fail the initialization of the pool by throwing exception.				
jmxEnabled	boolean	true		No
Register the pool with JMX or not. The default value is true.				
commitOnReturn	boolean	false		No
If autoCommit==false then the pool can complete the transaction by calling commit on the connection as it is returned to the pool If rollbackOnReturn==true then this attribute is ignored. Default value is false.				
logAbandoned	boolean	false		No
Flag to log stack traces for application code which abandoned a Connection. Logging of abandoned Connections adds overhead for every Connection borrow because a stack trace has to be generated. The default value is false.				
maxIdle	int	100		No
The maximum number of connections that should be kept in the pool at all times. Idle connections are checked periodically (if enabled) and connections that have been idle for longer than minEvictableIdleTimeMillis are released. The default value is derived from maxActive:100. (Also see testWhileIdle.)				
testWhileIdle	boolean	false		No
The indication of whether objects will be validated by the idle object evictor (if any). If an object fails to validate, it will be dropped from the pool. NOTE - for a true value to have any effect, the validationQuery parameter must be set to a non-null string. The default value is false and this property has to be set in order for the pool cleaner/test thread is to run (also see timeBetweenEvictionRunsMillis)				
removeAbandoned	boolean	false		No
Flag to remove abandoned connections if they exceed the removeAbandonedTimeout. If set to true a connection is considered abandoned and eligible for removal if it has been in use longer than the removeAbandonedTimeout Setting this to true can recover db connections from applications that fail to close a connection. See also logAbandoned The default value is false.				
abandonWhenPercentageFull	int	0		No
Connections that have been abandoned (timed out) wont get closed and reported up unless the number of connections in use are above the percentage defined by abandonWhenPercentageFull. The value should be between 0-100. The default value is 0, which implies that connections are eligible for closure as soon as removeAbandonedTimeout has been reached.				
minIdle	int	10		No
The minimum number of established connections that should be kept in the pool at all times. The connection pool can shrink below this number if validation queries fail. The default value is derived from initialSize:10. (Also see testWhileIdle.)				
defaultReadOnly	Boolean	null		No

Property	Type	Default	Encrypted ^a	Required ^b
The default read-only state of connections created by this pool. If not set then the setReadOnly method will not be called. (Some drivers dont support read only mode, ex: Informix)				
maxWait	int	30000		No
The maximum number of milliseconds that the pool will wait (when there are no available connections) for a connection to be returned before throwing an exception. Default value is 30000 (30 seconds)				
logValidationErrors	boolean	false		No
Set this to true to log errors during the validation phase to the log file. If set to true, errors will be logged as SEVERE. Default value is false for backwards compatibility.				
driverClassName	String	null		No
The fully qualified Java class name of the JDBC driver to be used. The driver has to be accessible from the same classloader as tomcat-jdbc.jar				
name	String	Tomcat Connection Pool[1- 153647080]		No
Returns the name of the connection pool. By default a JVM unique random name is assigned.				
useStatementFacade	boolean	true		No
Returns true if this connection pool is configured to wrap statements in order to enable equals() and hashCode() methods to be called on the closed statements if any statement proxy is set.				
initSQL	String	null		No
A custom query to be run when a connection is first created. The default value is null.				
validationQueryTimeout	int	-1		No
The timeout in seconds before a connection validation queries fail. This works by calling java.test_sample.Statement.setQueryTimeout(seconds) on the statement that executes the validationQuery. The pool itself doesnt timeout the query, it is still up to the JDBC driver to enforce query timeouts. A value less than or equal to zero will disable this feature. The default value is -1.				
validationQuery	String	null		No
The SQL query that will be used to validate connections from this pool before returning them to the caller. If specified, this query does not have to return any data, it just cant throw a SQLException. The default value is null. Example values are SELECT 1(mysql), select 1 from dual(oracle), SELECT 1(MS Sql Server)				
rollbackOnReturn	boolean	false		No
If autoCommit==false then the pool can terminate the transaction by calling rollback on the connection as it is returned to the pool Default value is false.				
alternateUsernameAllowed	boolean	false		No

Property	Type	Default	Encrypted ^a	Required ^b
<p>By default, the jdbc-pool will ignore the <code>DataSource.getConnection(username,password)</code> call, and simply return a previously pooled connection under the globally configured properties username and password, for performance reasons. The pool can however be configured to allow use of different credentials each time a connection is requested. To enable the functionality described in the <code>DataSource.getConnection(username,password)</code> call, simply set the property <code>alternateUsernameAllowed</code> to true. Should you request a connection with the credentials <code>user1/password1</code> and the connection was previously connected using different <code>user2/password2</code>, the connection will be closed, and reopened with the requested credentials. This way, the pool size is still managed on a global level, and not on a per schema level.</p>				
<code>validatorClassName</code>	String	null		No
<p>The name of a class which implements the <code>org.apache.tomcat.jdbc.pool.Validator</code> interface and provides a no-arg constructor (may be implicit). If specified, the class will be used to create a <code>Validator</code> instance which is then used instead of any validation query to validate connections. The default value is null. An example value is <code>com.mycompany.project.SimpleValidator</code>.</p>				
<code>suspectTimeout</code>	int	0		No
<p>Timeout value in seconds. Similar to the <code>removeAbandonedTimeout</code> value but instead of treating the connection as abandoned, and potentially closing the connection, this simply logs the warning if <code>logAbandoned</code> is set to true. If this value is equal or less than 0, no suspect checking will be performed. Suspect checking only takes place if the timeout value is larger than 0 and the connection was not abandoned or if abandon check is disabled. If a connection is suspect a WARN message gets logged and a JMX notification gets sent once.</p>				
<code>useEquals</code>	boolean	true		No
<p>Set to true if you wish the <code>ProxyConnection</code> class to use <code>String.equals</code> and set to false when you wish to use <code>==</code> when comparing method names. This property does not apply to added interceptors as those are configured individually. The default value is true.</p>				
<code>removeAbandonedTimeout</code>	int	60		No
<p>Timeout in seconds before an abandoned(in use) connection can be removed. The default value is 60 (60 seconds). The value should be set to the longest running query your applications might have.</p>				
<code>defaultAutoCommit</code>	Boolean	null		No
<p>The default auto-commit state of connections created by this pool. If not set, default is JDBC driver default (If not set then the <code>setAutoCommit</code> method will not be called.)</p>				
<code>testOnConnect</code>	boolean	false		No
<p>Returns true if we should run the validation query when connecting to the database for the first time on a connection. Normally this is always set to false, unless one wants to use the <code>validationQuery</code> as an init query.</p>				
<code>jdbcInterceptors</code>	String	null		No
<p>A semicolon separated list of classnames extending <code>org.apache.tomcat.jdbc.pool.JdbcInterceptor</code> class. See Configuring JDBC interceptors below for more detailed description of syntax and examples. These interceptors will be inserted as an interceptor into the chain of operations on a <code>java.test_sample.Connection</code> object. The default value is null.</p>				

Property	Type	Default	Encrypted ^a	Required ^b
<code>initialSize</code>	<code>int</code>	<code>10</code>		No
The initial number of connections that are created when the pool is started. Default value is 10				
<code>defaultTransactionIsolation</code>	<code>int</code>	<code>-1</code>		No
The default TransactionIsolation state of connections created by this pool. One of the following: NONE, READ_COMMITTED, READ_UNCOMMITTED, REPEATABLE_READ, SERIALIZABLE If not set, the method will not be called and it defaults to the JDBC driver.				
<code>numTestsPerEvictionRun</code>	<code>int</code>	<code>0</code>		No
Property not used in tomcat-jdbc-pool.				
<code>url</code>	<code>String</code>	<code>null</code>		No
The URL used to connect to the database.				
<code>testOnBorrow</code>	<code>boolean</code>	<code>false</code>		No
The indication of whether objects will be validated before being borrowed from the pool. If the object fails to validate, it will be dropped from the pool, and we will attempt to borrow another. NOTE - for a true value to have any effect, the validationQuery parameter must be set to a non-null string. In order to have a more efficient validation, see validationInterval. Default value is false				
<code>fairQueue</code>	<code>boolean</code>	<code>true</code>		No
Set to true if you wish that calls to getConnection should be treated fairly in a true FIFO fashion. This uses the org.apache.tomcat.jdbc.pool.FairBlockingQueue implementation for the list of the idle connections. The default value is true. This flag is required when you want to use asynchronous connection retrieval. Setting this flag ensures that threads receive connections in the order they arrive. During performance tests, there is a very large difference in how locks and lock waiting is implemented. When fairQueue=true there is a decision making process based on what operating system the system is running. If the system is running on Linux (property os.name=Linux. To disable this Linux specific behavior and still use the fair queue, simply add the property org.apache.tomcat.jdbc.pool.FairBlockingQueue.ignoreOS=true to your system properties before the connection pool classes are loaded.				
<code>accessToUnderlyingConnectionAllowed</code>	<code>boolean</code>	<code>true</code>		No
Property not used. Access can be achieved by calling unwrap on the pooled connection. see javax.test_sample.DataSource interface, or call getConnection through reflection or cast the object as javax.test_sample.PooledConnection				
<code>maxAge</code>	<code>long</code>	<code>0</code>		No
Time in milliseconds to keep this connection. When a connection is returned to the pool, the pool will check to see if the now - time-when-connected > maxAge has been reached, and if so, it closes the connection rather than returning it to the pool. The default value is 0, which implies that connections will be left open and no age check will be done upon returning the connection to the pool.				
<code>minEvictableIdleTimeMillis</code>	<code>int</code>	<code>60000</code>		No
The minimum amount of time an object may sit idle in the pool before it is eligible for eviction. The default value is 60000 (60 seconds).				

Property	Type	Default	Encrypted ^a	Required ^b
<code>timeBetweenEvictionRunsMillis</code>	<code>int</code>	<code>5000</code>		No
The number of milliseconds to sleep between runs of the idle connection validation/cleaner thread. This value should not be set under 1 second. It dictates how often we check for idle, abandoned connections, and how often we validate idle connections. The default value is 5000 (5 seconds).				
<code>testOnReturn</code>	<code>boolean</code>	<code>false</code>		No
The indication of whether objects will be validated before being returned to the pool. NOTE - for a true value to have any effect, the <code>validationQuery</code> parameter must be set to a non-null string. The default value is false.				
<code>useLock</code>	<code>boolean</code>	<code>false</code>		No
Return true if a lock should be used when operations are performed on the connection object. Should be set to false unless you plan to have a background thread of your own doing idle and abandon checking such as JMX clients. If the pool sweeper is enabled, then the lock will automatically be used regardless of this setting.				
<code>maxActive</code>	<code>int</code>	<code>100</code>		No
The maximum number of active connections that can be allocated from this pool at the same time. The default value is 100				
<code>username</code>	<code>String</code>	<code>null</code>		No
The connection username to be passed to our JDBC driver to establish a connection. Note that method <code>DataSource.getConnection(username,password)</code> by default will not use credentials passed into the method, but will use the ones configured here. See <code>alternateUsernameAllowed</code> property for more details.				
<code>table</code>	<code>String</code>	<code>TABLE_NAME</code>		Yes
Enter the name of the table in the database that contains the accounts.				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

Basic Configuration

Property	Type	Default	Encrypted ^a	Required ^b
<code>password</code>	<code>String</code>	<code>null</code>	Yes	Yes
The connection password to be passed to the JDBC driver to establish a connection. Note that method <code>DataSource.getConnection(username,password)</code> by default will not use credentials passed into the method, but will use the ones configured here. See <code>alternateUsernameAllowed</code> property for more details.				
<code>quoting</code>	<code>String</code>	<code>NONE</code>		No
Select whether database column names for this resource should be quoted, and the quoting characters. By default, database column names are not quoted (None). For other selections (Single, Double, Back, or Brackets), column names will appear between single quotes, double quotes, back quotes, or brackets in the SQL generated to access the database.				
<code>keyColumn</code>	<code>String</code>	<code>KEY_COLUMN</code>		Yes

Property	Type	Default	Encrypted ^a	Required ^b
This mandatory column value will be used as the unique identifier for rows in the table.				
<code>passwordColumn</code>	String	null		No
Enter the name of the column in the table that will hold the password values. If empty, no validation is done on resources and passwords.				
<code>disablePaging</code>	boolean	false		Yes
If true, optional paging in a query will be ignored by the connector. Defaults to false.				
<code>enableEmptyString</code>	boolean	false		No
Select to enable support for writing an empty string, instead of a NULL value, in character based columns defined as not-null in the table schema. This option does not influence the way strings are written for Oracle based tables. By default empty strings are written as a NULL value.				
<code>rethrowAllSQLExceptions</code>	boolean	true		No
If this is not checked, SQL statements which throw SQLExceptions with a 0 ErrorCode will be have the exception caught and suppressed. Check it to have exceptions with 0 ErrorCodes rethrown.				
<code>nativeTimestamps</code>	boolean	false		No
Select to retrieve Timestamp data type of the columns in java.sql.Timestamp format from the database table.				
<code>allNative</code>	boolean	false		No
Select to retrieve all data types of columns in native format from the database table.				
<code>changeLogColumn</code>	String	null		Sync
The change log column stores the latest change time. Providing this value the Sync capabilities are activated.				
<code>suppressPassword</code>	boolean	true		No
If set to true then the password will not be returned. Never. Even though it is explicitly requested. If set to false then the password will be returned if it is explicitly requested.				
<code>inclusiveSync</code>	boolean	false		No
If true, the SyncOp will query for ChangeLogColumn >= syncToken. One record will always be returned from the database in this case and be handled by the connector. If set to false, the SyncOp will query for ChangeLogColumn > syncToken. Defaults to false.				
<code>returnGeneratedKeys</code>	boolean	true		No
Return Generated Keys				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

DocuSign Connector

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

The DocuSign connector lets you manage DocuSign service accounts and synchronize accounts between DocuSign and the IDM managed user repository.

This chapter describes how to install and configure the DocuSign connector, and how to perform basic tests to ensure that it's running correctly.

For a complete example that includes the configuration required to synchronize users with this connector, see "*Synchronize Data Between IDM and DocuSign*" in the *Samples Guide*.

Before You Start

The instructions in this chapter assume that you have a DocuSign administrator account and that you have added an Integrator Key, as described in the [DocuSign Documentation](#). Before you configure the connector, log in to your administrator account and note the following information:

- API User ID
- API Account ID
- Integration Key

You will also need to set up an RSA Keypair and copy the public and private keys to a location that will be accessible by the connector.

- DocuSign API Hostname
- DocuSign OAuth Hostname

You need these details to configure the connector to interact with your DocuSign environment.

The DocuSign connector uses OAuth to connect to DocuSign. You must grant authorization to the Integration Key by directing your browser to the following URL:

```
https://account-d.docusign.com/oauth/auth?response_type=code&scope=signature%20impersonation&client_id=your-integrator-key&redirect_uri=https://client.example.com/callback
```

In the resulting window, select Accept to grant the required authorization.

The connector requires *signing groups* to be enabled. Depending on your DocuSign plan, you might need to contact the DocuSign Support team to enable signing groups. For more information, see the [DocuSign documentation](#).

Install and Configure the DocuSign Connector

Install the DocuSign Connector

1. Download the connector .jar file from the [ForgeRock BackStage download site](#).
 - If you are running the connector locally, place it in the `/path/to/openidm/connectors` directory, for example:

```
mv ~/Downloads/docusign-connector-1.5.20.11.jar /path/to/openidm/connectors/
```
 - If you are using a remote connector server (RCS), place it in the `/path/to/openicf/connectors` directory on the RCS.
2. Download the connector dependencies. The DocuSign connector has a dependency on the Java JWT library 3.4.0 (`java-jwt-3.4.0.jar`).
 - If you are running the connector locally, place the library in the `/path/to/openidm/lib` directory:

```
mv ~/Downloads/java-jwt-3.4.0.jar /path/to/openidm/lib/
```
 - If you are using a remote connector server (RCS), place the library in the `/path/to/openicf/lib` directory on the RCS.

Configure the DocuSign Connector

Note

If you had already started IDM (or your RCS) before copying the connector .jar file to the `connectors` directory, you must restart the server for the connector to be loaded.

1. Create a connector configuration by using the Admin UI:

Select **Configure > Connectors > New Connector** and select **DocuSign Connector - 1.5.20.11** as the connector type.
2. Alternatively, configure the connector with a configuration file.

IDM provides a sample connector configuration file in the `/path/to/openidm/samples/example-configurations/provisioners` directory.

Copy this sample file (`provisioner.openicf-docusign.json`) to your project's `conf` directory.
3. Provide at least the following configuration properties:

```
"configurationProperties": {  
  "host" : "_CHANGEME_",  
  "oAuthHost" : "_CHANGEME_",  
  "accountId" : "_CHANGEME_",  
  "integratorKey" : "_CHANGEME_",  
  "privateKeyFilePath" : "_CHANGEME_",  
  "publicKeyFilePath" : "_CHANGEME_",  
  "userId" : "_CHANGEME_",  
  ...  
}
```

host

The Docusign API hostname, for example, `demo.docusign.net`.

oAuthHost

The Docusign OAuth hostname, for example, `https://account.docusign.com/oauth`.

userId

The API User ID of the DocuSign user that will authenticate to the REST server. You can locate this ID under Admin > Integrations > API and Keys when you log in to your DocuSign account.

accountId

The API Account ID of the user specified previously. You can locate this account ID under Admin > Integrations > API and Keys when you log in to your DocuSign account.

integratorKey

The DocuSign Integration Key or client ID. You can locate the Integrator Key under Admin > Integrations > API and Keys when you log in to your DocuSign account. For more information, see the corresponding DocuSign documentation.

privateKeyFilePath

The full path to the Private Key of the RSA Keypair. To obtain the Private Key, select Admin > Integrations > API and Keys, then select Add RSA Keypair. Copy the value of the Private Key to a file and specify the file path in this property, for example: `"privateKeyFilePath" : "/path/to/private-key.txt"`.

publicKeyFilePath

The full path to the Public Key of the RSA Keypair. To obtain the Public Key, select Admin > Integrations > API and Keys, then select Add RSA Keypair. Copy the value of the Public Key to a file and specify the file path in this property, for example: `"publicKeyFilePath" : "/path/to/public-key.txt"`.

4. Enable the connector and save the connector configuration.

- When your connector is configured correctly, the connector displays as Active in the UI.

Alternatively, test the configuration over REST by running the following command:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system/docusign?_action=test"
{
  "name": "docusign",
  "enabled": true,
  "config": "config/provisioner.openicf/docusign",
  "connectorRef": {
    "bundleVersion": "${bundleVersion}",
    "bundleName": "org.forgerock.openicf.connectors.docusign-connector",
    "connectorName": "org.forgerock.openicf.connectors.docusign.DocuSignConnector"
  },
  "displayName": "DocuSign Connector",
  "objectTypes": [
    "userSignature",
    "signingGroup",
    "ALL",
    "account",
    "contact"
  ],
  "ok": true
}
```

If the command returns `"ok": true`, your connector has been configured correctly, and can authenticate to the DocuSign server.

Configure Connection Pooling

The DocuSign connector supports connection pooling, which can substantially improve the performance of the connector. The basic connection pooling configuration is described in "[Connection Pooling Configuration](#)".

Use the DocuSign Connector

You can use the DocuSign connector to perform the following actions on a DocuSign account:

+ *Create a DocuSign User*

This example creates a user with the minimum required attributes:

```
curl \
--header "Content-Type: application/json" \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
```

```
--data '{
  "userName": "Carlos Garcia",
  "email": "cgarcia@example.com",
  "password": "Passw0rd"
}' \
"http://localhost:8080/openidm/system/docusign/account?_action=create"
{
  "_id": "dc1c6940-1de7-4434-a91e-1407424cac91",
  "accountManagementGranular": [
    {
      "canManageUsers": "false"
    },
    {
      "canManageAdmins": "false"
    },
    {
      "canManageGroups": "false"
    },
    {
      "canManageSharing": "false"
    },
    {
      "canManageAccountSettings": "false"
    },
    {
      "canManageReporting": "false"
    },
    {
      "canManageAccountSecuritySettings": "false"
    },
    {
      "canManageSigningGroups": "false"
    }
  ],
  "userName": "Carlos Garcia",
  "enableConnectForUser": "false",
  "lastName": "Garcia",
  "createdDateTime": "2018-10-18T07:48:39.3870000Z",
  "userSettings": [
    {
      "name": "expressSendOnly",
      "value": "false"
    }
  ],
  "email": "cgarcia@example.com",
  "sendActivationOnInvalidLogin": "false",
  "userStatus": "ActivationSent",
  "firstName": "Carlos",
  "groupList": [
    {
      "groupName": "Everyone",
      "groupType": "everyoneGroup",
      "groupId": "4428049"
    }
  ],
  "uri": "/users/dc1c6940-1de7-4434-a91e-1407424cac91",
  "isAdmin": "False",
  "userType": "CompanyUser"
}
```

```
}

```

When you create a new user, you must specify *at least* the `userName`, `email`, and `password`. The value of the `userName` attribute determines how the remaining *name* attributes (`firstName`, `lastName`, and so on) are set in the new DocuSign user entry.

If you create the user with a single word as the value of the `userName` attribute, for example, `cgarcia`, the user's `userName` and `lastName` attributes in DocuSign are both set to `cgarcia`.

If you create the user with multiple words as the value of the `userName` attribute, for example, `Carlos Garcia`, the user's `userName` attribute is set to `Carlos Garcia`, their `firstName` attribute is set to `Carlos`, and their `lastName` attribute is set to `Garcia`.

Only the first three words of the `userName` attribute are parsed, into the `firstName`, `middleName`, and `lastName` attributes. Any additional words are ignored.

Important

By default, DocuSign accounts have a strict *password strength* setting. If a create operation fails with a `ConnectorException` and you see the following error in the logs:

```
Caused by: org.identityconnectors.framework.common.exceptions.ConnectorException: Invalid forgotten password challenge.
```

you might need to adjust your Password Rules in DocuSign, as described here.

You can also set a custom `forgottenPasswordQuestion` and `forgottenPasswordAnswer` attribute during the create operation. For example:

```
curl \
--header "Content-Type: application/json" \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
--data '{
  "userName": "Carlos Garcia",
  "email": "cgarcia@example.com",
  "password": "Passw0rd",
  "forgottenPasswordInfo": {
    "forgottenPasswordQuestion1": "my question",
    "forgottenPasswordAnswer1": "my answer"
  }
}' \
"http://localhost:8080/openidm/system/docusign/account?_action=create"
```

+ Query DocuSign User Entries

This example queries all DocuSign users by their IDs:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
"http://localhost:8080/openidm/system/docusign/account?_queryId=query-all-ids"
{
  "result": [
    {
      "_id": "bc9f0464-808a-4703-b4c2-c1e6a77f0c3a",
      "userName": "Babs Jensen"
    },
    {
      "_id": "dc1c6940-1de7-4434-a91e-1407424cac91",
      "userName": "Carlos Garcia"
    },
    {
      "_id": "94be4fed-cfd7-47d5-9fcc-813405084f17",
      "userName": "Olayinka Kuti"
    }
  ],
  "resultCount": 3,
  "pagedResultsCookie": null,
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

The following command queries a specific user by their ID:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
"http://localhost:8080/openidm/system/docusign/account/dc1c6940-1de7-4434-a91e-1407424cac91"
{
  "_id": "dc1c6940-1de7-4434-a91e-1407424cac91",
  "accountManagementGranular": [
    {
      "canManageUsers": "false"
    },
    {
      "canManageAdmins": "false"
    },
    {
      "canManageGroups": "false"
    },
    {
      "canManageSharing": "false"
    },
    {
      "canManageAccountSettings": "false"
    },
    {
      "canManageReporting": "false"
    }
  ],
  {

```

```
    "canManageAccountSecuritySettings": "false"
  },
  {
    "canManageSigningGroups": "false"
  }
],
"userName": "Carlos Garcia",
"enableConnectForUser": "false",
"lastName": "Garcia",
"createdDateTime": "2018-10-18T07:48:39.3870000Z",
"userSettings": [
  {
    "name": "expressSendOnly",
    "value": "false"
  }
],
"email": "cgarcia@example.com",
"sendActivationOnInvalidLogin": "false",
"userStatus": "ActivationSent",
"firstName": "Carlos",
"groupList": [
  {
    "groupName": "Everyone",
    "groupType": "everyoneGroup",
    "groupId": "4428049"
  }
],
"uri": "/users/dcl6940-1de7-4434-a91e-1407424cac91",
"isAdmin": "False",
"userType": "CompanyUser"
}
```

+ *Modify a DocuSign User Entry*

You can modify an existing user with a PATCH request or with a PUT request, including all attributes of the account in the request. You can use the connector to modify the following attributes of a user entry:

- `title`
- `firstName`
- `middleName`
- `lastName`
- `suffix`
- `userName`

After creation, a user's email address is read-only and you cannot modify it using the connector.

If forgotten password recovery has been enabled for the DocuSign user account, (`forgottenPasswordQuestion` and `forgottenPasswordAnswer` have been set) you can use the connector

to change a user's password. You must include the following attributes in a password change request:

- `currentPassword`
- `newPassword`
- `email`
- `forgottenPasswordQuestion`
- `forgottenPasswordAnswer`
- `forgottenPasswordInfo`

This example changes Carlos Garcia's password:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-type: application/json" \
--request PATCH \
--data '[
{
  "operation": "replace",
  "field": "password",
  "value": "MyPassw0rd"
}]' \
"http://localhost:8080/openidm/system/docusign/account/dclc6940-1de7-4434-a91e-1407424cac91"
{
  "_id": "dclc6940-1de7-4434-a91e-1407424cac91",
  "accountManagementGranular": [
    {
      "canManageUsers": "false"
    },
    {
      "canManageAdmins": "false"
    },
    {
      "canManageGroups": "false"
    },
    {
      "canManageSharing": "false"
    },
    {
      "canManageAccountSettings": "false"
    },
    {
      "canManageReporting": "false"
    },
    {
      "canManageAccountSecuritySettings": "false"
    },
    {
      "canManageSigningGroups": "false"
    }
  ]
}
```

```

],
"userName": "Carlos Garcia",
"userProfileLastModifiedDate": "2018-10-18T01:10:59.4230000Z",
"enableConnectForUser": "false",
"lastName": "Garcia",
"createdDateTime": "2018-10-18T07:48:39.3870000Z",
"userSettings": [
  {
    "name": "expressSendOnly",
    "value": "false"
  }
],
"email": "cgarcia@example.com",
"sendActivationOnInvalidLogin": "false",
"userStatus": "ActivationSent",
"firstName": "Carlos",
"groupList": [
  {
    "groupName": "Everyone",
    "groupType": "everyoneGroup",
    "groupId": "4428049"
  }
],
"uri": "/users/dc1c6940-1de7-4434-a91e-1407424cac91",
"isAdmin": "False",
"userType": "CompanyUser"
}

```

If the naming component attributes are sent in an update, these attribute values are set on the DocuSign user. The user's `userName` attribute is re-generated from the individual naming components. If both the `userName` and additional naming component attributes (such as `firstName` or `lastName`) are sent in the update request, the supplied `userName` attribute is ignored and its value is re-generated from the individual naming components.

+ Close a DocuSign User Account

You cannot use the DocuSign connector to delete an account from the DocuSign repository. However, you can use a DELETE request to set the `userStatus` attribute of the account to `Closed`.

This example closes Carlos Garcia's DocuSign account:

```

curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request DELETE \
"http://localhost:8080/openidm/system/docusign/account/dc1c6940-1de7-4434-a91e-1407424cac91"
{
  "id": "dc1c6940-1de7-4434-a91e-1407424cac91",
  "accountManagementGranular": [
    {
      "canManageUsers": "false"
    },
    {
      "canManageAdmins": "false"
    }
  ]
}

```

```

    },
    {
      "canManageGroups": "false"
    },
    {
      "canManageSharing": "false"
    },
    {
      "canManageAccountSettings": "false"
    },
    {
      "canManageReporting": "false"
    },
    {
      "canManageAccountSecuritySettings": "false"
    },
    {
      "canManageSigningGroups": "false"
    }
  ],
  "userName": "Carlos Garcia",
  "userProfileLastModifiedDate": "2018-10-18T01:10:59.4230000Z",
  "enableConnectForUser": "false",
  "lastName": "Garcia",
  "createdDateTime": "2018-10-18T07:48:39.3870000Z",
  "userSettings": [
    {
      "name": "expressSendOnly",
      "value": "false"
    }
  ],
  "email": "cgarcia@example.com",
  "sendActivationOnInvalidLogin": "false",
  "userStatus": "ActivationSent",
  "firstName": "Carlos",
  "groupList": [
    {
      "groupName": "Everyone",
      "groupType": "everyoneGroup",
      "groupId": "4428049"
    }
  ],
  "uri": "/users/dclc6940-1de7-4434-a91e-1407424cac91",
  "isAdmin": "False",
  "userType": "CompanyUser"
}

```

Note

A closed account remains in the DocuSign repository and can still be queried by its ID.

OpenICF Interfaces Implemented by the DocuSign Connector

The DocuSign Connector implements the following OpenICF interfaces.

Create

Creates an object and its `uid`.

Delete

Deletes an object, referenced by its `uid`.

Schema

Describes the object types, operations, and options that the connector supports.

Script on Connector

Enables an application to run a script in the context of the connector. Any script that runs on the connector has the following characteristics:

- The script runs in the same execution environment as the connector and has access to all the classes to which the connector has access.
- The script has access to a `connector` variable that is equivalent to an initialized instance of the connector. At a minimum, the script can access the connector configuration.
- The script has access to any script-arguments passed in by the application.

Search

Searches the target resource for all objects that match the specified object class and filter.

Test

Tests the connector configuration. Testing a configuration checks all elements of the environment that are referred to by the configuration are available. For example, the connector might make a physical connection to a host that is specified in the configuration to verify that it exists and that the credentials that are specified in the configuration are valid.

This operation might need to connect to a resource, and, as such, might take some time. Do not invoke this operation too often, such as before every provisioning operation. The test operation is not intended to check that the connector is alive (that is, that its physical connection to the resource has not timed out).

You can invoke the test operation before a connector configuration has been validated.

Update

Updates (modifies or replaces) objects on a target resource.

DocuSign Connector Configuration

The DocuSign Connector has the following configurable properties.

Basic Configuration Properties

Property	Type	Default	Encrypted ^a	Required ^b
host	String	null		Yes
The DNS name or IP address of the DocuSign REST server				
oAuthHost	String	null		Yes
The OAuth host URL to the DocuSign REST server				
accountId	String	null		Yes
The DocuSign Account ID to manage				
integratorKey	String	null		Yes
The DocuSign integrator key for accessing the REST API				
privateKeyFilePath	String	null		Yes
The path to the private key used to generate a JSON web token (JWT)				
publicKeyFilePath	String	null		Yes
The path to the public key used to generate a JSON web token (JWT)				
userId	String	null		Yes
The user ID of the user creating the JSON web token (JWT)				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

Advanced Configuration Properties

Property	Type	Default	Encrypted ^a	Required ^b
acceptSelfSignedCertificates	boolean	false		Yes
Specifies that the HTTP client accepts self-signed certificates				
disableHostNameVerifier	boolean	false		Yes
Specifies that the HTTP client does not verify the host name				
maximumConnections	Integer	10		No
The maximum number of connections				
httpProxyHost	String	null		Yes
The hostname of the HTTP proxy (if an HTTP proxy is used between the connector and the DocuSign server)				
httpProxyPort	Integer	null		Yes
The proxy port number (if an HTTP proxy is used between the connector and the DocuSign server)				

Property	Type	Default	Encrypted ^a	Required ^b
organizationConsent	Boolean	false		Yes
Specifies that there is consent from the organization				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

Google Cloud Platform Connector

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

Google Cloud Platform (GCP) is a suite of cloud computing services offered by Google. The GCP connector lets you manage and synchronize accounts between GCP and IDM managed user objects. A GCP administrator account is required for this connector to work.

Before you start

Before you configure the connector, log in to your GCP administrator account and note the following:

Domain name

The domain name of the account on GCP — for example, example.com.

Private key

The private key is required to sign the JWT token used to authenticate with GCP.

Service account

The GCP connector uses a service account with [two-legged OAuth](#) to connect to GCP. A service account is identified by its email address, which is unique to the account.

Admin user

The GCP administrator username.

Note

The Admin SDK API must also be enabled to allow viewing and managing users in the Google Cloud Platform.

Install the GCP connector

Download the connector .jar file from the ForgeRock BackStage download site.

- If you are running the connector locally, place it in the `/path/to/openidm/connectors` directory, for example:

```
mv ~/Downloads/gcp-connector-1.5.20.12.jar /path/to/openidm/connectors/
```

- If you are using a remote connector server (RCS), place it in the `/path/to/openicf/connectors` directory on the RCS.

Configure the GCP connector

Create a connector configuration using the Admin UI:

1. Select Configure > Connectors and click New Connector.
2. Enter a Connector Name.
3. Select GCP Connector - 1.5.20.12 as the Connector Type.
4. Provide the Base Connector Details.
5. Click Save.

When your connector is configured correctly, the connector displays as Active in the Admin UI.

Alternatively, test that the configuration is correct by running the following command:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system/gcp?_action=test"
{
  "name": "gcp",
  "enabled": true,
  "config": "config/provisioner.openicf/gcp",
  "connectorRef": {
    "bundleVersion": "${bundleVersion}",
    "bundleName": "org.forgerock.openicf.connectors.gcp-connector",
    "connectorName": "org.forgerock.openicf.connectors.gcp.GcpConnector"
  },
  "displayName": "GCP Connector",
  "objectTypes": [
    "ACCOUNT",
    "ALL"
  ],
  "ok": true
}
```

If the command returns `"ok": true`, your connector was configured correctly, and can authenticate to the Google Cloud Platform system.

Use the GCP connector

The following GCP account attributes are supported by the GCP connector:

Attribute	Description
<code>__NAME__</code>	The username of the user. This maps to a user's <code>primaryEmail</code> property in GCP. Required.
<code>__PASSWORD__</code>	Password for the user account. Required.
<code>givenName</code>	The first name of the user. Required.
<code>familyName</code>	The last name of the user. Required.
<code>__UID__</code>	The user ID for the user account.
<code>emails</code>	A list of emails associated with the user account. For example: <pre> "emails": [{ "address": "liz@example.com", "type": "home", "customType": "", "primary": false }], </pre>
<code>addresses</code>	A list of addresses associated with the user account. For example: <pre> "addresses": [{ "type": "work", "customType": "", "streetAddress": "1234 Example Road", "locality": "Mountain View", "region": "CA", "postalCode": "94043" }], </pre>
<code>organizations</code>	A list of organizations the user account is associated with. For example: <pre> "organizations": [{ "symbol": "Texas", "customType": "te", "costCenter": "Accounting Principles", "domain": "IAM", "name": "cloudauth", "description": "Agreed Accounting Principles", "location": "California", "department": "engineering", "title": "member", "type": "unknown" }], </pre>
<code>phones</code>	A list of phone numbers associated with the user account. For example:

Attribute	Description
	<pre>"phones": [{ "customType": "custom", "type": "custom", "value": "+1 888 555 2312", "primary": false }],</pre>
relations	<p>A list of the user's relationships to other users. For example:</p> <pre>"relations": [{ "customType": "Cousin", "type": "custom", "value": "Bob Jensen" }]</pre>
externalIds	<p>A list of external IDs for the user, such as employee or network IDs. For example:</p> <pre>"externalIds": [{ "customType": "employee", "type": "custom", "value": "12345" }],</pre>

For a full list of attributes on GCP user accounts, refer to the [GCP documentation](#).

You can use the GCP connector to perform the following actions on a GCP account:

+ *Create a GCP user*

The following example creates a user with the minimum required attributes:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
--data '{
  "__NAME__": "bjensen@example.com",
  "__PASSWORD__": "Passw0rd!",
  "givenName": "Barbara",
  "familyName": "Jensen"
}' \
"http://localhost:8080/openidm/system/gcp/__ACCOUNT__?_action=create"
{
  "_id": "115637914640083360831"
}
```

Note

When you create a new user, you must specify *at least* `__NAME__`, `__PASSWORD__`, `givenName` and `familyName`. Refer to the list of available attributes above for more information.

+ Update a GCP user

You can modify an existing user with a PUT request, including all attributes of the account in the request.

For example, to add a new phone to a user:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--header "if-Match:*" \
--request PUT \
--data '{
  "__NAME__": "bjensen@example.com",
  "phones": [{
    "type": "mobile",
    "value": "+1 888 555 2312",
    "primary": true
  }]
}' \
"http://localhost:8080/openidm/system/gcp/__ACCOUNT__/115637914640083360831"
{
  "_id": "115637914640083360831",
  "givenName": "Barbara",
  "__UID__": "115637914640083360831",
  "phones": [
    {
      "value": "+1 888 555 2312",
      "type": "mobile"
    }
  ],
  "__NAME__": "bjensen@example.com",
  "familyName": "Jensen",
  "__ENABLE__": false,
  "emails": [
    {
      "address": "bjensen@example.com",
      "primary": true
    },
    {
      "address": "bjensen@example.com.test-google-a.com"
    }
  ]
}
```

Note

The updated data may not appear in the initial response, but appears on any future queries of that user.

+ Query GCP users

The following example queries all GCP users:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/gcp/__ACCOUNT__?_queryId=query-all-ids"
{
  "result": [
    {
      "_id": "103181194086915091216"
    },
    {
      "_id": "104153234757881174617"
    },
    {
      "_id": "105181741894703739324"
    },
    {
      "_id": "105644268361304742523"
    },
    {
      "_id": "101682225764075422695"
    },
    {
      "_id": "101516788947553424126"
    },
    {
      "_id": "102825554929567443783"
    },
    {
      "_id": "101429904015255587067"
    },
    {
      "_id": "115637914640083360831"
    }
  ],
  "resultCount": 9,
  "pagedResultsCookie": null,
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

The following command queries a specific user by their ID:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/gcp/__ACCOUNT__/115637914640083360831"
{
  "_id": "115637914640083360831",
  "givenName": "Barbara",
  "UID": "115637914640083360831",
  "phones": [
    {
      "value": "+1 888 555 2312",
      "type": "mobile"
    }
  ],
  "NAME": "bjensen@example.com",
  "familyName": "Jensen",
  "ENABLE": false,
  "emails": [
    {
      "address": "bjensen@example.com",
      "primary": true
    },
    {
      "address": "bjensen@example.com.test-google-a.com"
    }
  ]
}
```

+ *Reset a GCP account password*

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--header "if-Match:*" \
--request PATCH \
--data '{
  "operation": "add",
  "field": "PASSWORD",
  "value": "Passw0rd@123!"
}' \
"http://localhost:8080/openidm/system/gcp/__ACCOUNT__/115637914640083360831"
{
  "_id": "115637914640083360831",
  "givenName": "Barbara",
  "UID": "115637914640083360831",
  "phones": [
    {
      "value": "+1 888 555 2312",
      "type": "mobile"
    }
  ],
}
```

```
"__NAME__": "bjensen@example.com",
"familyName": "Jensen",
"__ENABLE__": false,
"emails": [
  {
    "address": "bjensen@example.com",
    "primary": true
  },
  {
    "address": "bjensen@example.com.test-google-a.com"
  }
]
```

Note

While the `__PASSWORD__` field is not returned as part of the response, the user object is updated.

+ Delete a GCP user account

You can use the GCP connector to delete an account from the GCP service.

The following example deletes a GCP account:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request DELETE \
"http://localhost:8080/openidm/system/gcp/___ACCOUNT___/115637914640083360831"
{
  "_id": "115637914640083360831",
  "givenName": "Barbara",
  "___UID___": "115637914640083360831",
  "phones": [
    {
      "value": "+1 888 555 2312",
      "type": "mobile"
    }
  ],
  "___NAME___": "bjensen@example.com",
  "familyName": "Jensen",
  "___ENABLE___": false,
  "emails": [
    {
      "address": "bjensen@example.com",
      "primary": true
    },
    {
      "address": "bjensen@example.com.test-google-a.com"
    }
  ]
}
```

OpenICF Interfaces Implemented by the GCP Connector

The GCP Connector implements the following OpenICF interfaces.

Create

Creates an object and its **uid**.

Delete

Deletes an object, referenced by its **uid**.

Schema

Describes the object types, operations, and options that the connector supports.

Script on Connector

Enables an application to run a script in the context of the connector. Any script that runs on the connector has the following characteristics:

- The script runs in the same execution environment as the connector and has access to all the classes to which the connector has access.

- The script has access to a `connector` variable that is equivalent to an initialized instance of the connector. At a minimum, the script can access the connector configuration.
- The script has access to any script-arguments passed in by the application.

Search

Searches the target resource for all objects that match the specified object class and filter.

Test

Tests the connector configuration. Testing a configuration checks all elements of the environment that are referred to by the configuration are available. For example, the connector might make a physical connection to a host that is specified in the configuration to verify that it exists and that the credentials that are specified in the configuration are valid.

This operation might need to connect to a resource, and, as such, might take some time. Do not invoke this operation too often, such as before every provisioning operation. The test operation is not intended to check that the connector is alive (that is, that its physical connection to the resource has not timed out).

You can invoke the test operation before a connector configuration has been validated.

Update

Updates (modifies or replaces) objects on a target resource.

GCP Connector Configuration

The GCP Connector has the following configurable properties.

Configuration properties

Property	Type	Default	Encrypted ^a	Required ^b
<code>domainName</code>	<code>String</code>	<code>null</code>		Yes
Provide the domain name for GCP				
<code>privateKey</code>	<code>GuardedString</code>	<code>null</code>	Yes	Yes
Provide private key to authenticate GCP				
<code>serviceAccount</code>	<code>String</code>	<code>null</code>		Yes
Provide service account for fetching users from GCP				
<code>adminUser</code>	<code>String</code>	<code>null</code>		Yes
Provide admin user for fetching users from GCP				
<code>maxResults</code>	<code>int</code>	<code>50</code>		No

Property	Type	Default	Encrypted ^a	Required ^b
Provide user max results for fetching users from GCP				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

Basic configuration properties

Property	Type	Default	Encrypted ^a	Required ^b
<code>httpProxyHost</code>	String	null		No
Provide the HTTP proxy host				
<code>httpProxyPort</code>	Integer	null		No
Provide the HTTP proxy port				
<code>httpProxyUsername</code>	String	null		No
Provide the HTTP proxy username				
<code>httpProxyPassword</code>	GuardedString	null	Yes	No
Provide the HTTP Proxy password				
<code>connectionTimeout</code>	Integer	300		No
Provide the maximum connection timeout in seconds				
<code>maximumConnections</code>	Integer	10		No
Provide the maximum connections				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

Google Apps Connector

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

IDM bundles a Google Apps connector, along with a sample connector configuration. The Google Apps connector enables you to interact with Google's web applications.

The Google Apps connector is subject to the [API Limits and Quotas](#) that are imposed by Google. The connector also adheres to the implementation guidelines set out by Google for implementing exponential backoff.

Configuring the Google Apps Connector

The Google Apps connector uses OAuth2 to authorize the connection to the Google service. To use this authorization mechanism, you must supply a `clientId` and `clientSecret` in order to obtain an access token from Google. You can obtain the `clientId` and `clientKey` from the Google Developers Console after you have configured your Web Application.

A sample Google Apps connector configuration file is provided in `samples/example-configurations/provisioners/provisioner.openicf-google.json`

The following is an excerpt of the provisioner configuration file. This example shows an excerpt of the provisioner configuration. The default location of the connector .jar is `openidm/connectors`. Therefore the value of the `connectorHostRef` property must be `"#LOCAL"`:

```
{
  "connectorHostRef": "#LOCAL",
  "connectorName": "org.forgerock.openicf.connectors.googleapps.GoogleAppsConnector",
  "bundleName": "org.forgerock.openicf.connectors.googleapps-connector",
  "bundleVersion": "[1.4.0.0,2.0.0.0)"
},
```

The following excerpt shows the required configuration properties:

```
"configurationProperties": {
  "domain": "",
  "clientId": "",
  "clientSecret": null,
  "refreshToken": null
},
```

These configuration properties are fairly straightforward:

`domain`

Set to the domain name for OAuth 2-based authorization.

`clientId`

A client identifier, as issued by the OAuth 2 authorization server. For more information, see the following section of RFC 6749: *Client Identifier*.

`clientSecret`

Sometimes also known as the client password. OAuth 2 authorization servers can support the use of `clientId` and `clientSecret` credentials, as noted in the following section of RFC 6749: *Client Password*.

`refreshToken`

A client can use an OAuth 2 refresh token to continue accessing resources. For more information, see the following section of RFC 6749: *Refresh Tokens*.

For a sample Google Apps configuration that includes OAuth 2-based entries for `configurationProperties`, see "*Synchronize Accounts With the Google Apps Connector*" in the *Samples Guide*.

OpenICF Interfaces Implemented by the GoogleApps Connector

The GoogleApps Connector implements the following OpenICF interfaces.

Create

Creates an object and its `uid`.

Delete

Deletes an object, referenced by its `uid`.

Schema

Describes the object types, operations, and options that the connector supports.

Script on Connector

Enables an application to run a script in the context of the connector. Any script that runs on the connector has the following characteristics:

- The script runs in the same execution environment as the connector and has access to all the classes to which the connector has access.
- The script has access to a `connector` variable that is equivalent to an initialized instance of the connector. At a minimum, the script can access the connector configuration.
- The script has access to any script-arguments passed in by the application.

Search

Searches the target resource for all objects that match the specified object class and filter.

Test

Tests the connector configuration. Testing a configuration checks all elements of the environment that are referred to by the configuration are available. For example, the connector might make a physical connection to a host that is specified in the configuration to verify that it exists and that the credentials that are specified in the configuration are valid.

This operation might need to connect to a resource, and, as such, might take some time. Do not invoke this operation too often, such as before every provisioning operation. The test operation is not intended to check that the connector is alive (that is, that its physical connection to the resource has not timed out).

You can invoke the test operation before a connector configuration has been validated.

Update

Updates (modifies or replaces) objects on a target resource.

GoogleApps Connector Configuration

The GoogleApps Connector has the following configurable properties.

Basic Configuration Properties

Property	Type	Default	Encrypted ^a	Required ^b
domain	String	null		Yes
Internet domain name. See https://support.google.com/a/answer/177483?hl=en				
clientId	String	null		Yes
Client identifier issued to the client during the registration process.				
clientSecret	GuardedString	null	Yes	Yes
Client secret issued to the client during the registration process.				
refreshToken	GuardedString	null	Yes	Yes
The refresh token allows you to get a new access token that is good for another hour. Refresh tokens never expire, they can only be revoked by the user or programatically by your app.				
proxyHost	String	null		Yes
Defines an HTTP proxy host to use with the connection (example: "myproxy.home.com").				
proxyPort	int	8080		Yes
Defines an HTTP proxy port to use with the connection (defaults to 8080).				
validateCertificate	boolean	true		Yes
Validate the server certificate from the local truststore (defaults to true).				
usersMaxResults	int	100		No
Maximum number of Users to return. Acceptable values are 1 to 500, inclusive.				
groupsMaxResults	int	200		No
Maximum number of Groups to return. Acceptable values are 1 to 200, inclusive.				
membersMaxResults	int	200		No
Maximum number of Members to return. Acceptable values are greater than 1				

Property	Type	Default	Encrypted ^a	Required ^b
<code>listProductMaxResults</code>	<code>long</code>	<code>100</code>		No
Maximum number of Licenses to return. Acceptable values are 1 to 1000, inclusive.				
<code>listProductAndSkuMaxResults</code>	<code>long</code>	<code>100</code>		No
Maximum number of Licenses to return. Acceptable values are 1 to 1000, inclusive.				
<code>availableLicenses</code>	<code>String[]</code>	<code>[]</code>		No
All Google Licenses that will be queried when requesting licenses assigned to a user. The format of the license is ProductId/SkuId (e.g. Google-Apps/101002002)				
<code>roleMaxResults</code>	<code>int</code>	<code>100</code>		No
Maximum number of Licenses to return. Acceptable values are 1 to 100, inclusive.				
<code>roleAssignmentMaxResults</code>	<code>int</code>	<code>100</code>		No
Maximum number of Licenses to return. Acceptable values are 1 to 100, inclusive.				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

Using the Google Apps Connector With a Proxy Server

If the IDM server is hosted behind a firewall and requests to the Google Apps server are routed through a proxy, you must specify the proxy host and port in the connector configuration so that the connector can pass this information to the lower Google API.

To specify the proxy server details, set the `proxyHost`, `proxyPort` and `validateCertificate` properties in the connector configuration. For example:

```
"configurationProperties": {
  ...
  "proxyHost": "myproxy.home.com",
  "proxyPort": 8080,
  "validateCertificate": true,
  ...
}
```

The `validateCertificate` property indicates whether the proxy server should validate the server certificate from the local truststore.

Supported Resource Types

The Google Apps connector uses the Google Enterprise License Manager and Directory APIs to perform CRUD operations against resources within a Google Apps domain.

The following table lists the resource types that are supported by the Google Apps connector:

Supported Resource Types With the Google Apps Connector

ICF Native Type	Google Resource Type	Naming Attribute
__ACCOUNT__	user	primaryEmail
__GROUP__	group	email
Member	member	{groupKey}/email
OrgUnit	orgUnit	{parentOrgUnitPath}/__NAME__
LicenseAssignment	licenseAssignment	{productId}/sku/{skuId}/user/{primaryEmail}

Functional Limitations

The Google Apps connector is subject to the following functional limitations:

- In an UPDATE request, the old object (before the update) is returned in the request result. This behavior differs from that for other connectors, where the updated object is returned.

Although the update is processed correctly, there is a significant delay from Google, and IDM sends its GET request to return the object before the update has taken effect. This behavior has no impact on the success of the update.
- The connector does not implement the ICF Sync operation so you cannot use the connector for liveSync of supported Google Apps resources to IDM managed objects.
- The connector does not implement the Authenticate operation so you cannot use the connector to perform pass-through authentication between IDM and a Google Apps domain. You can also not use this connector to perform password Change operations (as opposed to password Reset) because the connector cannot authenticate on behalf of the end user.
- Support for Filters when performing Search operations is limited to those attributes described in "Supported Search Filters".
- Google Apps creates a new User Alias each time the **primaryEmail** address associated with the User object is modified. You cannot delete User Aliases with the Google Apps connector so you must manage Aliases directly from within the Google Apps console.
- For PATCH requests, a connector can potentially add, remove, or replace an attribute value. The Google Apps connector does not implement the add or remove operations, so a PATCH request always replaces the entire attribute value with the new value.

Supported Search Filters

The Google Apps connector supports filtered searches against Google Apps resources. However, limitations imposed by the APIs provided by the Google Apps Admin SDK prevent filtering of resource types based on arbitrary attributes and values.

The following filter operators and attributes are supported for Search operations with the Google Apps connector:

Supported Operators and Filter Attributes With Google Apps Searches

Object Type	Operators	Attributes
__ACCOUNT__	And, Contains, StartsWith, Equals	primaryEmail
__GROUP__	Contains, Equals	email
Member	Equals	{groupKey}/email
OrgUnit	StartsWith	{parentOrgUnitPath}/__NAME__
LicenseAssignment	Equals	{productId}/sku/{skuId}/user/{primaryEmail}

Groovy Connector Toolkit

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

ICF provides a generic Groovy Connector Toolkit that enables you to run a Groovy script for any ICF operation, such as search, update, create, and others, 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.

Configuring Scripted Groovy Connectors

The Groovy Connector Toolkit is bundled in the JAR [openidm/connectors/groovy-connector-1.5.20.8.jar](#).

The [Samples Guide](#) describes a number of scripted connector implementations. The scripts provided with these samples 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. For information about writing your own scripts, see "[Writing Scripted Connectors With the Groovy Connector Toolkit](#)" in the *Connector Developer's Guide*.

You specify the connector configuration in your project's `conf/provisioner.openicf-connector.json` file. A number of sample configurations for scripted Groovy implementations are provided in `openidm/samples/example-configurations/provisioners/provisioner.openicf-scriptedimpementation.json`. Use these as the basis for configuring your own scripted connector.

+ *Validating Pooled Connections*

The scripted SQL connector uses the [Tomcat JDBC Connection Pool](#) to manage its connections. Occasionally, a JDBC resource accessed by the scripted SQL connector might become unavailable for a period. When the resource comes back online, IDM is able to recover automatically and resume operations. However, the connector might not be able to refresh its connection pool and might then pass a closed connection to its scripts. This can affect operations until IDM is restarted.

To avoid this situation, you can configure *connection validation*, where connections are validated before being borrowed from the connection pool.

To configure connection validation, add the following properties to the `configurationProperties` object in your connector configuration:

`testOnBorrow`

Validates the connection object before it is borrowed from the pool. If the object fails to validate, it is dropped from the pool and the connector attempts to borrow another object.

For this property to have an effect, you must set `validationQuery` to a non-null string.

`validationQuery`

The SQL query used to validate connections from the pool before returning them to the caller.

The precise query will differ, depending on the database that you are accessing. The following list provides sample queries for common databases:

HyperSQL DataBase (HSQLDB)

```
select 1 from INFORMATION_SCHEMA.SYSTEM_USERS
```

Oracle DB

```
select 1 from dual
```

DB2

```
select 1 from sysibm.sysdummy1
```

MySQL

```
select 1
```

Microsoft SQL

```
select 1
```

PostgreSQL

```
select 1
```

Ingres Database

```
select 1
```

Apache Derby

```
values 1
```

H2 Database

```
select 1
```

Firebird SQL

```
select 1 from rdb$database
```

validationInterval

Specifies the maximum frequency (in milliseconds) at which validation is run. If a connection is due for validation but was previously validated within this interval, it is not validated again.

The larger the value, the better the connector performance. However, with a large value you increase the chance of a stale connection being presented to the connector.

Connection validation can have an impact on performance and should not be done too frequently. With the following configuration, connections are validated no more than every 34 seconds:

```
{
  ...
  "configurationProperties" : {
    ...
    "testOnBorrow" : true,
    "validationQuery" : "select 1 from dual",
    "validationInterval" : 34000,
    ...
  },
  ...
}
```

+ *Using Custom Properties*

The **customConfiguration** and **customSensitiveConfiguration** properties enable you to inject custom properties into your scripts. Properties listed in **customSensitiveConfiguration** are encrypted.

For example, the following excerpt of the scripted Kerberos provisioner file shows how these properties inject the Kerberos user and encrypted password into the scripts, using the **kadmin** command.

```
"customConfiguration" : "kadmin { cmd = '/usr/sbin/kadmin.local'; user='<KADMIN USERNAME>';
default_realm='<REALM>' }",
"customSensitiveConfiguration" : "kadmin { password = '<KADMIN PASSWORD>'}",
```

+ Debugging Groovy Scripts

When you call a Groovy script from the Groovy connector, you can use the SLF4J logging facility to obtain debug information.

For instructions on how to use this facility, see the KnowledgeBase article [How do I add logging to Groovy scripts in IDM](#).

+ Script Compilation and Caching

The first time a script is read, it is compiled (from Groovy script to Java bytecode) and cached in memory. Each time the script is called, the Groovy script engine checks the last modified timestamp of the script file to see if it has changed. If it has not changed, the cached bytecode is executed. If it has changed, the script is reloaded, compiled and cached.

Run scripts through the connector

Groovy Toolkit connectors have two operations that allow you to run arbitrary script actions: `runScriptOnConnector` and `runScriptOnResource`. `runScriptOnConnector` is an operation that sends the script action to the connector to be compiled and executed. `runScriptOnResource` is an operation that sends the script to another script to be handled.

`runScriptOnConnector`

The `runScriptOnConnector` script lets you run an arbitrary script action through the connector. This script takes the following variables as input:

`configuration`

A handler to the connector's configuration object.

`options`

A handler to the Operation Options.

`operation`

The operation type that corresponds to the action.

`log`

A handler to the connector's log.

To run an arbitrary script on a Groovy Toolkit connector, define the script in the `systemActions` property of your provisioner file:

```
"systemActions" : [
  {
    "scriptId" : "MyScript",
    "actions" : [
      {
        "systemType" : ".*ScriptedConnector",
        "actionType" : "groovy",
        "actionFile" : "path/to/scriptname.groovy"
      }
    ]
  }
]
```

If you wish to define your script in the provisioner file itself rather than in a separate file, you can use the `actionSource` property instead of the `actionFile` one. A simple example follows:

```
"systemActions" : [
  {
    "scriptId" : "MyScript",
    "actions" : [
      {
        "systemType" : ".*ScriptedConnector",
        "actionType" : "groovy",
        "actionSource" : "2 * 2"
      }
    ]
  }
]
```

Note

It is optional to prepend the last script statement in `actionSource` with `return`.

Running `MyScript` will return:

```
{
  "actions" : [
    {
      "result": 4
    }
  ]
}
```

If your script accepts parameters, you can supply them in the request body or the query string. For example:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
--data-raw '{"param1":"value1"}' \
"http://localhost:8080/openidm/system/groovy?_action=script&scriptId=MyScript&param2=value2"
```

You can also call it through the script engine. The system can accept arbitrary parameters:

```
openidm.action("/system/groovy", "script", {"contentParameter": "value"}, {"scriptId": "MyScript",
"additionalParameter1": "value1", "additionalParameter2": "value2"})
```

runScriptOnResource

To run an arbitrary script using `runScriptOnResource`, you must add some configuration details to your provisioner file. These details include a `scriptOnResourceScriptFileName` that references a script file located in a path contained in the `scriptRoots` array.

Define these properties in your provisioner file:

```
"configurationProperties": {
  "scriptRoots": [
    "path/to/scripts"
  ],
  "scriptOnResourceScriptFileName": "ScriptOnResourceScript.groovy"
},
"systemActions" : [
  {
    "scriptId" : "script-1",
    "actions" : [
      {
        "systemType" : ".*ScriptedConnector",
        "actionType" : "groovy",
        "actionFile" : "path/to/scriptname.groovy"
      }
    ]
  }
]
```

When you have defined the script, call it over REST on the system endpoint:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system/groovy?
_action=script&scriptId=scriptOnResourceScript&scriptExecuteMode=resource"
```

Implemented Interfaces

The following tables list the ICF interfaces that are implemented for non-poolable and poolable connector implementations:

OpenICF Interfaces Implemented by the Scripted Groovy Connector

The Scripted Groovy Connector implements the following OpenICF interfaces.

Authenticate

Provides simple authentication with two parameters, presumed to be a user name and password.

Create

Creates an object and its `uid`.

Delete

Deletes an object, referenced by its `uid`.

Resolve Username

Resolves an object by its username and returns the `uid` of the object.

Schema

Describes the object types, operations, and options that the connector supports.

Script on Connector

Enables an application to run a script in the context of the connector. Any script that runs on the connector has the following characteristics:

- The script runs in the same execution environment as the connector and has access to all the classes to which the connector has access.
- The script has access to a `connector` variable that is equivalent to an initialized instance of the connector. At a minimum, the script can access the connector configuration.
- The script has access to any script-arguments passed in by the application.

Script on Resource

Runs a script on the target resource that is managed by this connector.

Search

Searches the target resource for all objects that match the specified object class and filter.

Sync

Polls the target resource for synchronization events, that is, native changes to objects on the target resource.

Test

Tests the connector configuration. Testing a configuration checks all elements of the environment that are referred to by the configuration are available. For example, the connector might make a

physical connection to a host that is specified in the configuration to verify that it exists and that the credentials that are specified in the configuration are valid.

This operation might need to connect to a resource, and, as such, might take some time. Do not invoke this operation too often, such as before every provisioning operation. The test operation is not intended to check that the connector is alive (that is, that its physical connection to the resource has not timed out).

You can invoke the test operation before a connector configuration has been validated.

Update

Updates (modifies or replaces) objects on a target resource.

OpenICF Interfaces Implemented by the Scripted Poolable Groovy Connector

The Scripted Poolable Groovy Connector implements the following OpenICF interfaces.

Authenticate

Provides simple authentication with two parameters, presumed to be a user name and password.

Create

Creates an object and its `uid`.

Delete

Deletes an object, referenced by its `uid`.

Resolve Username

Resolves an object by its username and returns the `uid` of the object.

Schema

Describes the object types, operations, and options that the connector supports.

Script on Connector

Enables an application to run a script in the context of the connector. Any script that runs on the connector has the following characteristics:

- The script runs in the same execution environment as the connector and has access to all the classes to which the connector has access.
- The script has access to a `connector` variable that is equivalent to an initialized instance of the connector. At a minimum, the script can access the connector configuration.
- The script has access to any script-arguments passed in by the application.

Script on Resource

Runs a script on the target resource that is managed by this connector.

Search

Searches the target resource for all objects that match the specified object class and filter.

Sync

Polls the target resource for synchronization events, that is, native changes to objects on the target resource.

Test

Tests the connector configuration. Testing a configuration checks all elements of the environment that are referred to by the configuration are available. For example, the connector might make a physical connection to a host that is specified in the configuration to verify that it exists and that the credentials that are specified in the configuration are valid.

This operation might need to connect to a resource, and, as such, might take some time. Do not invoke this operation too often, such as before every provisioning operation. The test operation is not intended to check that the connector is alive (that is, that its physical connection to the resource has not timed out).

You can invoke the test operation before a connector configuration has been validated.

Update

Updates (modifies or replaces) objects on a target resource.

Configuration Properties

The following tables list the configuration properties for non-poolable and poolable connector implementations:

Scripted Groovy Connector Configuration

The Scripted Groovy Connector has the following configurable properties.

Configuration properties

Property	Type	Default	Encrypted ^a	Required ^b
<code>customSensitiveConfiguration</code>	<code>GuardedString</code>	<code>null</code>	Yes	No
Custom Sensitive Configuration script for Groovy ConfigSlurper				
<code>customConfiguration</code>	<code>String</code>	<code>null</code>		No

Property	Type	Default	Encrypted ^a	Required ^b
Custom Configuration script for Groovy ConfigSlurper				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

Operation Script Files

Property	Type	Default	Encrypted ^a	Required ^b
<code>createScriptFileName</code>	String	null		Create
The name of the file used to perform the CREATE operation.				
<code>customizerScriptFileName</code>	String	null		No
The script used to customize some function of the connector. Read the documentation for more details.				
<code>authenticateScriptFileName</code>	String	null		Authenticate
The name of the file used to perform the AUTHENTICATE operation.				
<code>scriptOnResourceScriptFileName</code>	String	null		Script On Resource
The name of the file used to perform the RUNSCRIPTONRESOURCE operation.				
<code>deleteScriptFileName</code>	String	null		Delete
The name of the file used to perform the DELETE operation.				
<code>resolveUsernameScriptFileName</code>	String	null		Resolve Username
The name of the file used to perform the RESOLVE_USERNAME operation.				
<code>searchScriptFileName</code>	String	null		Get Search
The name of the file used to perform the SEARCH operation.				
<code>updateScriptFileName</code>	String	null		Update
The name of the file used to perform the UPDATE operation.				
<code>schemaScriptFileName</code>	String	null		Schema
The name of the file used to perform the SCHEMA operation.				
<code>testScriptFileName</code>	String	null		Test
The name of the file used to perform the TEST operation.				
<code>syncScriptFileName</code>	String	null		Sync
The name of the file used to perform the SYNC operation.				

Property	Type	Default	Encrypted ^a	Required ^b
----------	------	---------	------------------------	-----------------------

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

Groovy Engine configuration

Property	Type	Default	Encrypted ^a	Required ^b
<code>targetDirectory</code>	File	null		No
Directory into which to write classes.				
<code>warningLevel</code>	int	1		No
Warning Level of the compiler				
<code>scriptExtensions</code>	String[]	['groovy']		No
Gets the extensions used to find groovy files				
<code>minimumRecompilationInterval</code>	int	100		No
Sets the minimum of time after a script can be recompiled.				
<code>scriptBaseClass</code>	String	null		No
Base class name for scripts (must derive from Script)				
<code>scriptRoots</code>	String[]	null		Yes
The root folder to load the scripts from. If the value is null or empty the classpath value is used.				
<code>tolerance</code>	int	10		No
The error tolerance, which is the number of non-fatal errors (per unit) that should be tolerated before compilation is aborted.				
<code>debug</code>	boolean	false		No
If true, debugging code should be activated				
<code>classpath</code>	String[]	[]		No
Classpath for use during compilation.				
<code>disabledGlobalASTTransformations</code>	String[]	null		No
Sets a list of global AST transformations which should not be loaded even if they are defined in META-INF/org.codehaus.groovy.transform.ASTTransformation files. By default, none is disabled.				
<code>verbose</code>	boolean	false		No
If true, the compiler should produce action information				
<code>sourceEncoding</code>	String	UTF-8		No

Property	Type	Default	Encrypted ^a	Required ^b
Encoding for source files				
<code>recompileGroovySource</code>	<code>boolean</code>	<code>false</code>		No
If set to true recompilation is enabled				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

Scripted Poolable Groovy Connector Configuration

The Scripted Poolable Groovy Connector has the following configurable properties.

Configuration properties

Property	Type	Default	Encrypted ^a	Required ^b
<code>customSensitiveConfiguration</code>	<code>GuardedString</code>	<code>null</code>	Yes	No
Custom Sensitive Configuration script for Groovy ConfigSlurper				
<code>customConfiguration</code>	<code>String</code>	<code>null</code>		No
Custom Configuration script for Groovy ConfigSlurper				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

Operation Script Files

Property	Type	Default	Encrypted ^a	Required ^b
<code>createScriptFileName</code>	<code>String</code>	<code>null</code>		Create
The name of the file used to perform the CREATE operation.				
<code>customizerScriptFileName</code>	<code>String</code>	<code>null</code>		No
The script used to customize some function of the connector. Read the documentation for more details.				
<code>authenticateScriptFileName</code>	<code>String</code>	<code>null</code>		Authenticate
The name of the file used to perform the AUTHENTICATE operation.				
<code>scriptOnResourceScriptFileName</code>	<code>String</code>	<code>null</code>		Script On Resource
The name of the file used to perform the RUNSCRIPTONRESOURCE operation.				
<code>deleteScriptFileName</code>	<code>String</code>	<code>null</code>		Delete
The name of the file used to perform the DELETE operation.				

Property	Type	Default	Encrypted ^a	Required ^b
<code>resolveUsernameScriptFileName</code>	String	null		Resolve Username
The name of the file used to perform the RESOLVE_USERNAME operation.				
<code>searchScriptFileName</code>	String	null		Get Search
The name of the file used to perform the SEARCH operation.				
<code>updateScriptFileName</code>	String	null		Update
The name of the file used to perform the UPDATE operation.				
<code>schemaScriptFileName</code>	String	null		Schema
The name of the file used to perform the SCHEMA operation.				
<code>testScriptFileName</code>	String	null		Test
The name of the file used to perform the TEST operation.				
<code>syncScriptFileName</code>	String	null		Sync
The name of the file used to perform the SYNC operation.				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

Groovy Engine configuration

Property	Type	Default	Encrypted ^a	Required ^b
<code>targetDirectory</code>	File	null		No
Directory into which to write classes.				
<code>warningLevel</code>	int	1		No
Warning Level of the compiler				
<code>scriptExtensions</code>	String[]	['groovy']		No
Gets the extensions used to find groovy files				
<code>minimumRecompilationInterval</code>	int	100		No
Sets the minimum of time after a script can be recompiled.				
<code>scriptBaseClass</code>	String	null		No
Base class name for scripts (must derive from Script)				
<code>scriptRoots</code>	String[]	null		Yes

Property	Type	Default	Encrypted ^a	Required ^b
The root folder to load the scripts from. If the value is null or empty the classpath value is used.				
tolerance	int	10		No
The error tolerance, which is the number of non-fatal errors (per unit) that should be tolerated before compilation is aborted.				
debug	boolean	false		No
If true, debugging code should be activated				
classpath	String[]	[]		No
Classpath for use during compilation.				
disabledGlobalASTTransformations	String[]	null		No
Sets a list of global AST transformations which should not be loaded even if they are defined in META-INF/org.codehaus.groovy.transform.ASTTransformation files. By default, none is disabled.				
verbose	boolean	false		No
If true, the compiler should produce action information				
sourceEncoding	String	UTF-8		No
Encoding for source files				
recompileGroovySource	boolean	false		No
If set to true recompilation is enabled				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

HubSpot Connector

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

The HubSpot connector lets you synchronize HubSpot contacts and companies with managed objects in an IDM repository.

This chapter describes how to install and configure the HubSpot connector and how to perform basic tests to ensure that it's running correctly.

For a complete example that includes the configuration required to synchronize users with this connector, see "*Synchronize Data Between IDM and HubSpot*" in the *Samples Guide*.

Before you configure the HubSpot connector, you must have a client app in HubSpot, with the corresponding `clientId`, `clientSecret` and `refreshToken`.

Install and Configure the HubSpot Connector

Install the HubSpot Connector

- Download the connector .jar file from the ForgeRock BackStage download site.
 - If you are running the connector locally, place it in the `/path/to/openidm/connectors` directory, for example:

```
mv ~/Downloads/hubspot-connector-1.5.20.11.jar /path/to/openidm/connectors/
```
 - If you are using a remote connector server (RCS), place it in the `/path/to/openicf/connectors` directory on the RCS.

Configure the HubSpot Connector

Note

If you had already started IDM (or your RCS) before copying the connector .jar file to the `connectors` directory, you must restart the server for the connector to be loaded.

- Create a connector configuration by using the Admin UI:

Select **Configure > Connectors > New Connector** and select **HubSpot Connector - 1.5.20.11** as the connector type.

- Alternatively, configure the connector with a configuration file.

IDM provides a sample connector configuration file in the `/path/to/openidm/samples/example-configurations/provisioners` directory.

Copy this sample file (`provisioner.openicf-hubspot.json`) to your project's `conf` directory.

- Adjust the `configurationProperties` to match your HubSpot application details. You *must* provide a `clientId`, `clientSecret`, and `refreshToken`. Other properties are optional:

```
"configurationProperties" : {  
  "clientId" : "daa533ae-xxxx-xxxx-xxxx-6e66d84e6448",  
  "clientSecret" : "c598a365-xxxx-xxxx-xxxx-24b32b6ae04d",  
  "refreshToken" : "f37e1132-xxxx-xxxx-xxxx-4b9e724ce4a0",  
  "acceptSelfSignedCertificates" : true,  
  "readSchema" : "true",  
  "disableHostNameVerifier" : false,  
  "maximumConnections" : "10",  
  "permitsPerSecond" : "10",  
  "httpProxyHost" : null,  
  "httpProxyPort" : null  
}
```

IDM encrypts the `clientSecret` and `refreshToken` as soon as the connector is enabled.

4. Enable the connector and save the connector configuration.
5. When your connector is configured correctly, the connector displays as Active in the UI.

Alternatively, test the configuration over REST by running the following command:

```
curl \  
--header "X-OpenIDM-Username: openidm-admin" \  
--header "X-OpenIDM-Password: openidm-admin" \  
--header "Accept-API-Version: resource=1.0" \  
--request POST \  
"http://localhost:8080/openidm/system?_action=test"  
[  
  {  
    "name": "hubspot",  
    "enabled": true,  
    "config": "config/provisioner.openicf/hubspot",  
    "connectorRef": {  
      "bundleVersion": "${bundleVersion}",  
      "bundleName": "org.forgerock.openicf.connectors.hubspot-connector",  
      "connectorName": "org.forgerock.openicf.connectors.hubspot.HubspotConnector"  
    },  
    "displayName": "Hubspot Connector",  
    "objectTypes": [  
      "company",  
      "contactProperties",  
      "__ALL__",  
      "companyProperties",  
      "contact"  
    ],  
    "ok": true  
  }  
]
```

A status of `"ok": true` indicates that the connector can connect to HubSpot.

Implementation Specifics

For PATCH requests, a connector can potentially add, remove, or replace an attribute value. The HubSpot connector does not implement the add or remove operations, so a PATCH request always replaces the entire attribute value with the new value.

Using the HubSpot Connector With a Proxy Server

If the IDM server is hosted behind a firewall and requests to the resource provider are routed through a proxy, you must specify the proxy host and port in the connector configuration.

To specify the proxy server details, set the `httpProxyHost`, and `httpProxyPort` properties in the connector configuration. For example:

```
"configurationProperties": {  
  ...  
  "httpProxyHost": "myproxy.home.com",  
  "httpProxyPort": 8080,  
  ...  
}
```

OpenICF Interfaces Implemented by the Hubspot Connector

The Hubspot Connector implements the following OpenICF interfaces.

Create

Creates an object and its `uid`.

Delete

Deletes an object, referenced by its `uid`.

Schema

Describes the object types, operations, and options that the connector supports.

Script on Connector

Enables an application to run a script in the context of the connector. Any script that runs on the connector has the following characteristics:

- The script runs in the same execution environment as the connector and has access to all the classes to which the connector has access.
- The script has access to a `connector` variable that is equivalent to an initialized instance of the connector. At a minimum, the script can access the connector configuration.
- The script has access to any script-arguments passed in by the application.

Search

Searches the target resource for all objects that match the specified object class and filter.

Test

Tests the connector configuration. Testing a configuration checks all elements of the environment that are referred to by the configuration are available. For example, the connector might make a physical connection to a host that is specified in the configuration to verify that it exists and that the credentials that are specified in the configuration are valid.

This operation might need to connect to a resource, and, as such, might take some time. Do not invoke this operation too often, such as before every provisioning operation. The test operation is not intended to check that the connector is alive (that is, that its physical connection to the resource has not timed out).

You can invoke the test operation before a connector configuration has been validated.

Update

Updates (modifies or replaces) objects on a target resource.

Hubspot Connector Configuration

The Hubspot Connector has the following configurable properties.

Basic Configuration Properties

Property	Type	Default	Encrypted ^a	Required ^b
<code>clientId</code>	<code>String</code>	<code>null</code>		Yes
Client ID of the OAuth application in Hubspot				
<code>clientSecret</code>	<code>GuardedString</code>	<code>null</code>	Yes	Yes
Client Secret for the preceding Client ID				
<code>refreshToken</code>	<code>GuardedString</code>	<code>null</code>	Yes	Yes
Refresh token for application in Hubspot				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

Advanced Connection Properties

Property	Type	Default	Encrypted ^a	Required ^b
<code>acceptSelfSignedCertificates</code>	<code>boolean</code>	<code>false</code>		Yes
Specifies whether the HubSpot server should accept self-signed certificates. Defaults to false.				

Property	Type	Default	Encrypted ^a	Required ^b
<code>readSchema</code>	Boolean	false		Yes
If false, the Hubspot connector provides a default schema for Hubspot contacts and companies				
<code>disableHostNameVerifier</code>	boolean	false		Yes
If hostname verification is disabled, the HubSpot server accepts connections from any host. Defaults to false.				
<code>maximumConnections</code>	Integer	10		Yes
Maximum number of simultaneous connections to HubSpot.				
<code>permitsPerSecond</code>	Integer	10		Yes
Number of Api calls to be made per second				
<code>httpProxyHost</code>	String	null		Yes
Specifies the Hostname if an HTTP proxy is used between the connector and HubSpot. Defaults to null.				
<code>httpProxyPort</code>	Integer	null		Yes
Specifies the Port number if an HTTP proxy is used between the connector and HubSpot . Defaults to null.				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

Kerberos Connector

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

The Kerberos connector is an implementation of the SSH connector, and is based on Java Secure Channel (JSch) and the Java implementation of the Expect library (Expect4j). The connector depends on the following files, provided with IDM:

- `/path/to/openidm/lib/ssh-connector-1.5.20.8.jar`
- `/path/to/openidm/lib/expect4j-<version>.jar`
- `/path/to/openidm/lib/jsch-<version>.jar`

The Kerberos connector enables you to manage Kerberos user principals from IDM. The connector is provided in `/path/to/openidm/connectors/kerberos-connector-1.5.20.8.jar` and bundles a number of Groovy scripts to interact with a Kerberos admin server. Users of the Kerberos connector are not expected to edit the bundled Groovy scripts. The bundled scripts use the `kadmin` utility to communicate with the Kerberos server.

The Kerberos connector enables you to perform the following operations on Kerberos user principals:

- List the existing principals.
- Display the details of a principal.
- Add a user principal.
- Change the password of a user principal and unlock the principal.
- Delete a user principal.

Kerberos Connector Schema

The Kerberos connector can only be used to manage the Kerberos `principal` object type (which maps to the ICF `__ACCOUNT__` object). The following attributes are supported in the schema:

- `principal` - (maps to `__NAME__` and `__UID__`)
- `__PASSWORD__` - updatable, required when an object is created
- `__LOCK_OUT__` - updatable only; unlock an account by setting this attribute to `false`
- `policy` - the password policy used by the principal
- `expirationDate` - the date that the user principal expires
- `passwordExpiration` - the date that the password expires
- `maximumTicketLife` - the maximum ticket life for the principal. At the end of the ticket lifetime, the ticket can no longer be used. However, if the renewable lifetime (`maximumRenewableLife`) is longer than the ticket lifetime, the ticket holder can present the ticket to the KDC and request a new ticket.
- `maximumRenewableLife` - the period during which the ticket can be renewed. A renewed ticket usually has a new ticket lifetime, dating from the time that it was renewed, that is constrained by the renewable ticket lifetime.

In addition, the following read-only attributes are supported:

- `lastPasswordChange`
- `lastModified`
- `lastSuccessfulAuthentication`
- `lastFailedAuthentication`
- `failedPasswordAttempts`

Configuring the Kerberos Connector

A sample connector configuration (`provisioner.openicf-kerberos.json`) is provided in the `/path/to/openidm/samples/sync-with-kerberos/conf/` directory. You can copy the sample connector configuration to your project's `conf/` directory, and adjust it to match your Kerberos environment.

Set the authentication properties, as described in "Configuring Authentication to the SSH Server". In addition, set at least the following properties:

customConfiguration

Specify the details of the user principal and the default realm here. The sample provisioner file has the following custom configuration:

```
"customConfiguration" : "kadmin {
  cmd = '/usr/sbin/kadmin.local';
  user = '<KADMIN USERNAME>';
  default_realm = '<REALM, e.g. EXAMPLE.COM>'
}"
```

A complete custom configuration will look something like this:

```
"customConfiguration" : "kadmin {
  cmd = '/usr/sbin/kadmin.local';
  user = 'openidm/admin';
  default_realm = 'EXAMPLE.COM'
}"
```

customSensitiveConfiguration

Set the password for the user principal here. The sample provisioner has the following configuration:

```
"customSensitiveConfiguration" : "kadmin {password = '<KADMIN PASSWORD>'}"
```

Change this to reflect your user principal password, for example:

```
"customSensitiveConfiguration" : "kadmin {password = 'Passw0rd'}"
```

The following section describes the configuration parameters in the sample Kerberos connector configuration. For a complete list of the configuration properties for the Kerberos connector, see "Configuration properties":

host

The host name or IP address of the SSH server on which the **kadmin** command is run.

port

The port number on which the SSH server listens.

Default: 22 (the default SSH port)

user

The username of the account that is used to connect to the SSH server.

Note

This is *not* the same as your Kerberos user principal. This account must be able to **ssh** into the server on which Kerberos is running, with the password provided in the next parameter.

If you use the `root` user, the `sudo` command in the Test script will never get the `'pass: '` prompt. Instead of using the `root` user, create a regular user and add that user to the group that has `sudo` privileges. Alternatively, modify the Test script so that it does not use `sudo`.

password

The password of the account that is used to connect to the SSH server.

prompt

A string representing the remote SSH session prompt. This must be the exact prompt string, in the format `username@target:`, for example `root@localhost:~$`.

If the prompt includes a trailing space, you must include the space in the value of this property.

Consider customizing your Linux prompt with the `PS1` and `PS2` variables, to set a *safe* prompt. For information about customizing prompts, see [this article](#).

sudoCommand

A string that shows the full path to the `sudo` command, for example `/usr/bin/sudo`.

echoOff

If set to `true` (the default), the input command echo is disabled. If set to `false`, every character that is sent to the server is sent back to the client in the `expect()` call.

terminalType

Sets the terminal type to use for the session. The list of supported types is determined by your Linux/UNIX system. For more information, see the `terminfo` manual page (**man terminfo**).

Default: `vt102`

setLocale

If set to `true`, indicates that the default environment locale should be changed to the value of the `locale` property.

Default: `false`

locale

Sets the locale for `LC_ALL`, `LANG` and `LANGUAGE` environment variables, if `setLocale` is set to `true`.

Default: `en_US.utf8`

connectionTimeout

Specifies the connection timeout to the remote server, in milliseconds.

Default: `5000`

expectTimeout

Specifies the timeout used by the `expect()` calls in scripts, in milliseconds.

Default: `5000`

authenticationType

Sets the authentication type, either `PASSWORD` or `PUBKEY`. For more information, see "Configuring Authentication to the SSH Server".

Default: `PASSWORD`

throwOperationTimeoutException

If `true`, the connector throws an exception when the timeout is reached for an operation. Otherwise, the operation fails silently.

Default: `true`

scriptRoots

The path to the Groovy scripts that will perform the ICF operations, relative to your installation directory. For the Kerberos connector, the scripts are bundled up in the connector JAR file, so this path is set to `jar:file:connectors/kerberos-connector-1.5.20.8.jar!/script/kerberos/` in the sample connector configuration.

classpath

The directory in which the compiler should look for compiled classes. The default classpath, if not is specified, is `install-dir/lib`.

***ScriptFileName**

The script that is used for each ICF operation. Do not change these script names in the bundled Kerberos connector.

OpenICF Interfaces Implemented by the Kerberos Connector

The Kerberos Connector implements the following OpenICF interfaces.

Authenticate

Provides simple authentication with two parameters, presumed to be a user name and password.

Create

Creates an object and its `uid`.

Delete

Deletes an object, referenced by its `uid`.

Resolve Username

Resolves an object by its username and returns the `uid` of the object.

Schema

Describes the object types, operations, and options that the connector supports.

Script on Connector

Enables an application to run a script in the context of the connector. Any script that runs on the connector has the following characteristics:

- The script runs in the same execution environment as the connector and has access to all the classes to which the connector has access.
- The script has access to a `connector` variable that is equivalent to an initialized instance of the connector. At a minimum, the script can access the connector configuration.
- The script has access to any script-arguments passed in by the application.

Script on Resource

Runs a script on the target resource that is managed by this connector.

Search

Searches the target resource for all objects that match the specified object class and filter.

Sync

Polls the target resource for synchronization events, that is, native changes to objects on the target resource.

Test

Tests the connector configuration. Testing a configuration checks all elements of the environment that are referred to by the configuration are available. For example, the connector might make a physical connection to a host that is specified in the configuration to verify that it exists and that the credentials that are specified in the configuration are valid.

This operation might need to connect to a resource, and, as such, might take some time. Do not invoke this operation too often, such as before every provisioning operation. The test operation is not intended to check that the connector is alive (that is, that its physical connection to the resource has not timed out).

You can invoke the test operation before a connector configuration has been validated.

Update

Updates (modifies or replaces) objects on a target resource.

Kerberos Connector Configuration

The Kerberos Connector has the following configurable properties.

Configuration properties

Property	Type	Default	Encrypted ^a	Required ^b
<code>customSensitiveConfiguration</code>	<code>GuardedString</code>	<code>null</code>	Yes	No
Description is not available				
<code>createScriptFileName</code>	<code>String</code>	<code>null</code>		Create
Description is not available				
<code>targetDirectory</code>	<code>File</code>	<code>null</code>		No
Description is not available				
<code>customizerScriptFileName</code>	<code>String</code>	<code>null</code>		No
Description is not available				
<code>warningLevel</code>	<code>int</code>	<code>1</code>		No
Description is not available				
<code>authenticateScriptFileName</code>	<code>String</code>	<code>null</code>		Authenticate
Description is not available				
<code>scriptExtensions</code>	<code>String[]</code>	<code>['groovy']</code>		No
Description is not available				
<code>scriptOnResourceScriptFileName</code>	<code>String</code>	<code>null</code>		Script On Resource
Description is not available				
<code>minimumRecompilationInterval</code>	<code>int</code>	<code>100</code>		No
Description is not available				
<code>deleteScriptFileName</code>	<code>String</code>	<code>null</code>		Delete
Description is not available				
<code>scriptBaseClass</code>	<code>String</code>	<code>null</code>		No
Description is not available				
<code>scriptRoots</code>	<code>String[]</code>	<code>null</code>		Yes
Description is not available				

Property	Type	Default	Encrypted ^a	Required ^b
<code>customConfiguration</code>	String	null		No
Description is not available				
<code>resolveUsernameScriptFileName</code>	String	null		Resolve Username
Description is not available				
<code>searchScriptFileName</code>	String	null		Get Search
Description is not available				
<code>tolerance</code>	int	10		No
Description is not available				
<code>updateScriptFileName</code>	String	null		Update
Description is not available				
<code>debug</code>	boolean	false		No
Description is not available				
<code>classpath</code>	String[]	[]		No
Description is not available				
<code>disabledGlobalASTTransformations</code>	String[]	null		No
Description is not available				
<code>schemaScriptFileName</code>	String	null		Schema
Description is not available				
<code>verbose</code>	boolean	false		No
Description is not available				
<code>testScriptFileName</code>	String	null		Test
Description is not available				
<code>sourceEncoding</code>	String	UTF-8		No
Description is not available				
<code>syncScriptFileName</code>	String	null		Sync
Description is not available				
<code>recompileGroovySource</code>	boolean	false		No

Property	Type	Default	Encrypted ^a	Required ^b
Description is not available				
host	String	null		Yes
Description is not available				
port	int	22		Yes
Description is not available				
user	String	null		Yes
Description is not available				
password	GuardedString	null	Yes	No
Description is not available				
passphrase	GuardedString	null	Yes	No
Description is not available				
privateKey	String[]	[]	Yes	No
Description is not available				
authenticationType	String	PASSWORD		Yes
Description is not available				
prompt	String	root@localhost:#		Yes
Description is not available				
sudoCommand	String	/usr/bin/sudo		Yes
Description is not available				
echoOff	boolean	true		Yes
Description is not available				
terminalType	String	vt102		Yes
Description is not available				
locale	String	en_US.utf8		Yes
Description is not available				
setLocale	boolean	false		Yes
Description is not available				
connectionTimeout	int	5000		Yes

Property	Type	Default	Encrypted ^a	Required ^b
Description is not available				
<code>expectTimeout</code>	<code>long</code>	<code>5000</code>		Yes
Description is not available				
<code>throwOperationTimeoutException</code>	<code>boolean</code>	<code>true</code>		Yes
Description is not available				
<code>promptReadyTimeout</code>	<code>long</code>	<code>20</code>		No
Description is not available				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

LDAP Connector

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

The generic LDAP connector is based on the Java Naming and Directory Interface (JNDI), and can be used to connect to any LDAPv3-compliant directory server, such as ForgeRock Directory Services (DS), Active Directory, SunDS, Oracle Directory Server Enterprise Edition, IBM Security Directory Server, and OpenLDAP.

Because it is based on JNDI, the LDAP connector is restricted to the attribute types that are supported by JNDI. JNDI supports only strings and an array of bytes. If you attempt to use different attribute value types, the connector throws a malformed attribute value exception. For more information, see the corresponding [JNDI documentation](#).

Setting Up the Generic LDAP Connector

IDM bundles version 1.5.20.12 of the LDAP connector. Two sample LDAP connector configurations are provided in the `path/to/openidm/samples/example-configurations/provisioners/` directory:

- `provisioner.openicf-dslldap.json` provides a sample LDAP connector configuration for a ForgeRock Directory Services (DS) server.
- `provisioner.openicf-adldap.json` provides a sample LDAP connector configuration for an Active Directory server.

You should be able to adapt one of these sample configurations for any LDAPv3-compliant server.

The `connectorRef` configuration property provides information about the LDAP connector bundle, and is the same in all three sample LDAP connector configurations:

```
{
  "connectorRef": {
    "connectorHostRef": "#LOCAL",
    "connectorName": "org.identityconnectors.ldap.LdapConnector",
    "bundleName": "org.forgerock.openicf.connectors.ldap-connector",
    "bundleVersion": "[1.4.0.0,2.0.0.0)"
  }
}
```

The `connectorHostRef` property is optional, if you use the connector .jar provided in [openidm/connectors](#), and you use a local connector server.

The following excerpt shows the configuration properties in the sample LDAP connector for DS. These properties are described in detail later in this section. For additional information on the properties that affect synchronization, see "Controlling What the LDAP Connector Synchronizes". For a complete list of the configuration properties for the LDAP connector, see "LDAP Connector Configuration":

```
"configurationProperties" : {
  "host" : "localhost",
  "port" : 1389,
  "ssl" : false,
  "startTLS" : false,
  "privateKeyAlias" : null,
  "alternateKeyStore" : null,
  "alternateKeyStoreType" : null,
  "alternateKeyStorePassword" : null,
  "principal" : "uid=admin",
  "credentials" : "password",
  "baseContexts" : [
    "dc=example,dc=com"
  ],
  "baseContextsToSynchronize" : [
    "dc=example,dc=com"
  ],
  "accountSearchFilter" : null,
  "accountSynchronizationFilter" : null,
  "groupSearchFilter" : null,
  "groupSynchronizationFilter" : null,
  "removeLogEntryObjectClassFromFilter" : true,
  "modifiersNamesToFilterOut" : [ ],
  "changeLogBlockSize" : 100,
  "attributesToSynchronize" : [ ],
  "changeNumberAttribute" : "changeNumber",
  "filterWithOrInsteadOfAnd" : false,
  "objectClassesToSynchronize" : [
    "inetOrgPerson"
  ],
  "vlvSortAttribute" : "uid",
  "passwordAttribute" : "userPassword",
  "useBlocks" : false,
  "maintainPosixGroupMembership" : false,
  "failover" : [ ],
  "readSchema" : true,
```

```
"accountObjectClasses" : [
  "top",
  "person",
  "organizationalPerson",
  "inetOrgPerson"
],
"accountUserNameAttributes" : [
  "uid"
],
"groupMemberAttribute" : "uniqueMember",
"passwordHashAlgorithm" : null,
"usePagedResultControl" : true,
"blockSize" : 100,
"uidAttribute" : "entryUUID",
"maintainLdapGroupMembership" : false,
"respectResourcePasswordPolicyChangeAfterReset" : false
},
```

host

The host name or IP address of the server on which the LDAP instance is running.

port

The port on which the LDAP server listens for LDAP requests. The sample configuration specifies a default port of 1389.

ssl

If `true`, the specified port listens for LDAPS connections.

For instructions on using the LDAP connector over SSL, see "Configuring the LDAP Connector to Use SSL and StartTLS".

startTLS

Specifies whether to use the startTLS operation to initiate a TLS/SSL session. To use startTLS, set `"startTLS":true`, and `"ssl":false`. Your connection should use the insecure LDAP port (typically 389 or 1389 for a DS server).

Specify the certificates that should be used for authentication, as described in "Configuring the LDAP Connector to Use SSL and StartTLS".

principal

The bind DN that is used to connect to the LDAP server.

credentials

The password of the `principal` that is used to connect to the LDAP server.

baseContexts

One or more starting points in the LDAP tree that will be used when searching the tree. Searches are performed when discovering users from the LDAP server or when looking for the groups

of which a user is a member. During reconciliation operations, IDM searches through the base contexts listed in this property for changes. (See also "Controlling What the LDAP Connector Synchronizes").

baseContextsToSynchronize

One or more starting points in the LDAP tree that will be used to determine if a change should be synchronized. During liveSync operations, IDM searches through the base contexts listed in this property for changes. If no value is specified here, the values in listed in the **baseContexts** property are used. (See also "Controlling What the LDAP Connector Synchronizes").

accountSynchronizationFilter

Used during synchronization actions to filter out LDAP accounts. (See also "Controlling What the LDAP Connector Synchronizes").

accountObjectClasses

This property lists all the object classes that represent an account. If this property has multiple values, an **AND** filter is used to determine the affected entries. For example, if the value of this property is `["organizationalPerson", "inetOrgPerson"]`, any entry with the object class **organizationalPerson** AND the object class **inetOrgPerson** is considered as an account entry. You can override the value of this property by specifying the user object classes during the create operation.

If no object class is specified when you create a user, this property is used as the default list of object classes for the new entry.

accountSearchFilter

Search filter that user accounts must match. (See also "Controlling What the LDAP Connector Synchronizes").

accountUserNameAttributes

Attributes holding the account's user name. Used during authentication to find the LDAP entry matching the user name.

attributesToSynchronize

List of attributes used during object synchronization. IDM ignores change log updates that do not include any of the specified attributes. If empty, IDM considers all changes. (See also "Controlling What the LDAP Connector Synchronizes").

blockSize

Block size for simple paged results and VLV index searches, reflecting the maximum number of entries retrieved at any one time.

changeLogBlockSize

Block size used when fetching change log entries.

changeNumberAttribute

Change log attribute containing the last change number.

failover

LDAP URLs specifying alternative LDAP servers to connect to if IDM cannot connect to the primary LDAP server specified in the `host` and `port` properties.

filterWithOrInsteadOfAnd

In most cases, the filter to fetch change log entries is AND-based. If this property is set, the filter ORs the required change numbers instead.

groupMemberAttribute

LDAP attribute holding members for non-POSIX static groups.

groupSearchFilter

Search filter that group entries must match.

maintainLdapGroupMembership

If `true`, IDM modifies group membership when entries are renamed or deleted.

Does not apply to Active Directory.

In the sample LDAP connector configuration file provided with IDM, this property is set to `false`. This means that LDAP group membership is not modified when entries are renamed or deleted in IDM. To ensure that entries are removed from LDAP groups when the entries are deleted, set this property to `true` or enable referential integrity on the LDAP server. For information about configuring referential integrity in DS, see *Referential Integrity* in the *Configuration Guide* for ForgeRock Directory Services.

maintainPosixGroupMembership

If `true`, IDM modifies POSIX group membership when entries are renamed or deleted.

modifiersNamesToFilterOut

Use this property to avoid loops caused by changes made to managed user objects being synchronized. For more information, see "Controlling What the LDAP Connector Synchronizes".

objectClassesToSynchronize

IDM synchronizes only entries that have these object classes. See also "Controlling What the LDAP Connector Synchronizes".

passwordAttribute

Attribute to which IDM writes the predefined `PASSWORD` attribute.

passwordHashAlgorithm

Hash password values with the specified algorithm, if the LDAP server stores them in clear text. The hash algorithm can be one of the following:

- **NONE** - Clear text
- **WIN-AD** - Used for password changes to Active Directory
- **SHA** - Secure Hash Algorithm
- **SHA-1** - A 160-bit hash algorithm that resembles the MD5 algorithm
- **SSHA** - Salted SHA
- **MD5** - A 128-bit message-digest algorithm
- **SMD5** - Salted MD5

readSchema

If **true**, read the schema from the LDAP server.

This property is used only during the connector setup, to generate the object types.

If this property is **false**, the LDAP connector provides a basic default schema that can manage LDAP users and groups. The default schema maps **inetOrgPerson** to the OpenICF **__ACCOUNT__** property, and **groupOfUniqueNames** to the OpenICF **__GROUP__** property. The following LDAP object classes are also included in the default schema:

```
organization
organizationalUnit
person
organizationalPerson
account
groupOfNames
```

removeLogEntryObjectClassFromFilter

If **true**, the filter to fetch change log entries does not contain the **changeLogEntry** object class, and IDM expects no entries with other object types in the change log. The default setting is **true**.

respectResourcePasswordPolicyChangeAfterReset

If **true**, bind with the Password Expired and Password Policy controls, and throw **PasswordExpiredException** and other exceptions appropriately.

uidAttribute

Specifies the LDAP attribute that should be used as the immutable ID for the entry. You can use a DN (or any unique attribute) for the **__id**. As a best practice, you *should* use an attribute that is both unique and immutable, such as the **entryUUID**. For a DS resource, you must use the

`entryUUID` as the `uidAttribute`, otherwise you might encounter problems with synchronizing delete operations.

`useBlocks`

If `useBlocks` is `false`, no pagination is used. If `useBlocks` is `true`, the connector uses block-based LDAP controls, either the simple paged results control, or the virtual list view control, depending on the setting of the `usePagedResultControl` property.

`usePagedResultControl`

Taken into account only if `useBlocks` is `true`. If `usePagedResultControl` is `false`, the connector uses the virtual list view (VLV) control, if it is available. If `usePagedResultControl` is `true`, the connector uses the simple paged results control for search operations.

`useTimestampsForSync`

If `true`, use timestamps for liveSync operations, instead of the change log.

By default, the LDAP connector has a change log strategy for LDAP servers that support a change log, such as ForgeRock Directory Services (DS) and Oracle Directory Server Enterprise Edition. If the LDAP server does not support a change log, or if the change log is disabled, liveSync for create and modify operations can still occur, based on the timestamps of modifications.

Regardless of the value of `useTimestampsForSync`, the connector uses a timestamp strategy for liveSync for the following LDAP server types:

- MS Active Directory Global Catalog
- OpenLDAP
- *Unknown*

An LDAP server type is marked *unknown* if it is anything other than IBM, Novell, UnboundID, RedHat/Fedora 389, CA LDAP, OpenDS, ForgeRock OpenDJ / DS, Sun DSEE Directory, MS Active Directory, MS Active Directory Lightweight Directory Services (LDS), MS Active Directory Global Catalog, or OpenLDAP.

`vlvSortAttribute`

Attribute used as the sort key for virtual list view.

`sendCAUDTxId`

If `true`, propagate the Common Audit Transaction ID to a DS server.

Configuring the LDAP Connector to Use SSL and StartTLS

To use the LDAP connector over SSL, update your connector configuration file as follows:

1. For a connection over SSL, set the `ssl` property to `true` and set the `port` to a secure port, for example, `636`.

To initiate a connection using startTLS, set `"startTLS":true`, and `"ssl":false`. Set the `port` to an insecure LDAP port, for example, `389`.

2. If you are using a CA-signed server certificate, add that certificate to the IDM truststore, for example:

```
keytool \
  -importcert \
  -alias server-cert \
  -keystore /path/to/openidm/security/truststore \
  -storepass changeit \
  -file /path/to/server-cert.crt
```

3. Specify the certificate that the LDAP connector will use to authenticate to the remote LDAP server.

By default, the LDAP connector uses the self-signed certificate that is generated in the IDM keystore when IDM first starts up. You have two options to change this default behavior:

- a. Set the `privateKeyAlias` to the alias of a certificate in the IDM keystore. The alias name is case-sensitive.

If you set `privateKeyAlias` to `null`, no private key is sent during the SSL handshake, so only the server certificate is used. You must import the server certificate into the IDM truststore, as shown in the previous step.

If `privateKeyAlias` is set to an alias within the IDM keystore, the connector uses that private key for SSL mutual authentication.

- b. Specify a different keystore for the connector.

If you do not want to use the default IDM keystore, set the following properties:

- `alternateKeyStore` - specifies the full path to an alternate keystore.
- `alternateKeyStoreType` - specifies alternate keystore type. Valid values are `JKS`, `JCEKS` and `PKCS12`.
- `alternateKeyStorePassword` - specifies password for the alternate keystore.

4. (Optional) Enable hostname verification to prevent a third party from manipulating DNS entries or spoofing the LDAP Server IP.

When hostname verification is enabled, the connector compares the hostname in the certificate `subject` and `subjectAltName` with a simple hostname pattern defined in the `hostNameVerification` property.

To enable hostname verification, set `"hostNameVerification": true` and set the `hostNameVerification` property to the hostname you want to match. If the pattern matches, the connector is initialized successfully. If the pattern does not match, connector initialization throws an error. The `hostNameVerification` property supports wild card matching.

Assume, for example, a server certificate principal hostname of `server1.example.com`. With the following connector configuration, IDM starts up and the connector is initialized:

```
"configurationProperties" : {
  ...
  "hostNameVerification" : true,
  "hostNameVerifierPattern" : "server1.example.com",
  ...
}
```

Similarly, with the following connector configuration, IDM starts up and the connector is initialized:

```
"configurationProperties" : {
  ...
  "hostNameVerification" : true,
  "hostNameVerifierPattern" : "*.example.com",
  ...
}
```

With the following connector configuration, IDM starts up but connector initialization throws an error:

```
"configurationProperties" : {
  ...
  "hostNameVerification" : true,
  "hostNameVerifierPattern" : "server2.example.com",
  ...
}
```

The error returned is similar to the following:

```
The host name from the server certificate'CN=server1.example.com' does not match the provided pattern
'server2.example.com'
```

Controlling What the LDAP Connector Synchronizes

To control the set of LDAP entries that are affected by reconciliation and automatic synchronization operations, set the following properties in the provisioner configuration. *Automatic synchronization* includes liveSync (synchronization of changes from the LDAP server to IDM) and implicit sync (synchronization from IDM to the LDAP server). For more information, see "Types of Synchronization" in the *Synchronization Guide*.

`accountSearchFilter`

Only user accounts that match this filter are searched, and therefore affected by reconciliation and synchronization operations. If you do not set this property, all accounts within the base contexts specified previously are searched.

accountSynchronizationFilter

This property is used during reconciliation and automatic synchronization operations, and filters out any LDAP accounts that you specifically want to exclude from these operations.

attributesToSynchronize

During automatic synchronization operations, *only* the attributes listed here are considered for changes. Objects that include these attributes are synchronized. Objects that do not include these attributes are ignored. If this property is not set, IDM considers changes to all attributes specified in the mapping.

This attribute works only with LDAP servers that log changes in a change log, not with servers (such as Active Directory) that use other mechanisms to track changes.

baseContexts

The starting points in the LDAP tree that are used when searching the directory tree; for example, `dc=example,dc=com`. These base contexts must include the set of users *and the set of groups* that must be searched during reconciliation operations.

baseContextsToSynchronize

The starting points in the LDAP tree that are used to determine if a change should be synchronized. This property is used only for automatic synchronization operations. Only entries that fall under these base contexts are considered during synchronization operations.

modifiersNamesToFilterOut

This property lets you define a list of DNs. During synchronization operations, the connector ignores changes made by these DNs.

When a managed user object is updated, and that change is synchronized to the LDAP server, the change made on the LDAP server is recorded in the change log. A liveSync operation picks up the change, and attempts to replay the change on the managed user object, effectively resulting in a loop of updates.

To avoid this situation, you can specify a unique user in your LDAP directory, that will be used *only* for the LDAP connector. The unique user must be something other than `uid=admin`; for example `cn=idmuser`. You can then include that user DN as the value of `modifiersNamesToFilterOut`. When a change is made through the LDAP connector, and that change is recorded in the change log, the modifier's name (`cn=idmuser`) is flagged, and IDM does not attempt to replay the change back to the managed user repository. So, you are effectively indicating that IDM should not synchronize changes back to managed user that originated from managed user, thus preventing the update loop.

This attribute works only with LDAP servers that log changes in a change log, not with servers (such as Active Directory) that use other mechanisms to track changes.

objectClassesToSynchronize

During automatic synchronization operations, only the object classes listed here are considered for changes. IDM ignores change log updates (or changes to managed objects) which do not have any of the object classes listed here.

Using the Generic LDAP Connector With Active Directory

The LDAP connector provides functionality specifically for managing Active Directory users and groups. The connector can handle the following operational attributes to manage Active Directory accounts:

__ENABLE__

Uses the `userAccountControl` attribute to get or set the account status of an object.

The LDAP connector reads the `userAccountControl` to determine if an account is enabled or disabled. The connector modifies the value of the `userAccountControl` attribute if IDM changes the value of `__ENABLE__`.

__ACCOUNT_EXPIRES__

Gets or sets the `accountExpires` attribute of an Active Directory object.

__LOCK_OUT__

Uses the `msDS-User-Account-Control-Computed` system attribute to check if a user account has been locked.

If IDM sets `__LOCK_OUT__` to `FALSE`, the LDAP connector sets the Active Directory `lockoutTime` to `0` to unlock the account.

If IDM sets `__LOCK_OUT__` to `TRUE`, the LDAP connector ignores the change and logs a message.

__PASSWORD_EXPIRED__

Uses the `msDS-User-Account-Control-Computed` system attribute to check if a user password has expired.

To force password expiration (that is, to force a user to change their password when they next log in), set `pwdLastSet` to `0`. The LDAP connector sets `pwdLastSet` to `0`, if IDM sets `__PASSWORD_EXPIRED__` to `TRUE`.

To remove password expiration, set `pwdLastSet` to `0` and then to `-1`. This sets the value of `pwdLastSet` to the current time. The LDAP connector sets `pwdLastSet` to `-1` if IDM sets `__PASSWORD_EXPIRED__` to `FALSE`.

Note

Active Directory does not allow you to create an enabled account with an expired password. If you are using `__PASSWORD_EXPIRED__` to force a new user to change their password when they next log in, you can

create the user account as disabled initially (`__ENABLE__=false`). You can then patch the new user account to enable it. You can use the same workaround for synchronization operations, creating new user accounts as disabled, then issuing an `openidm.patch` call in a `postCreate` script to enable the account.

`__CURRENT_PASSWORD__`

For a password change request, the connector supplies the `__CURRENT_PASSWORD__`, along with the new password. The connector can also do a password *reset* where only the new password is supplied.

The sample connector configuration file (`openidm/samples/example-configurations/provisioners/provisioner.openicf-adldap.json`) includes these operational attributes.

Note that the `passwordAttribute` property in this provisioner file is set to `unicodePwd`. This property specifies the attribute in Active Directory that holds the user password. When a user's password is changed, the new value is set in this attribute.

Managing Active Directory Users With the LDAP Connector

If you create or update users in Active Directory, and those user entries include passwords, you *must* use the LDAP connector over SSL. You cannot create or update an Active Directory user password in clear text. To use the connector over SSL, follow the instructions in "Configuring the LDAP Connector to Use SSL and StartTLS".

The following command adds an Active Directory user. The output shows the operational attributes described in the previous section:

```
curl \
--header "Content-Type: application/json" \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
--data '{
  "dn": "CN=Brian Smith,CN=Users,DC=example,DC=com",
  "cn": "Brian Smith",
  "sAMAccountName": "bsmith",
  "userPrincipalName": "bsmith@example.com",
  "userAccountControl": "512",
  "givenName": "Brian",
  "mail": "bsmith@example.com",
  "__PASSWORD__": "Passw0rd"
}' \
http://localhost:8080/openidm/system/ad/account?_action=create
{
  "_id": "e1418d64-096c-4cb0-b903-ebb66562d99d",
  "mobile": null,
  "postalCode": null,
  "st": null,
  "employeeType": [],
  "objectGUID": "e1418d64-096c-4cb0-b903-ebb66562d99d",
  "cn": "Brian Smith",
```

```
{
  "department": null,
  "l": null,
  "description": null,
  "info": null,
  "manager": null,
  "sAMAccountName": "bsmith",
  "sn": null,
  "whenChanged": "20151217131254.0Z",
  "userPrincipalName": "bsmith@example.com",
  "userAccountControl": "512",
  "___ENABLE___": true,
  "displayName": null,
  "givenName": "Brian",
  "middleName": null,
  "facsimileTelephoneNumber": null,
  "lastLogon": "0",
  "countryCode": "0",
  "employeeID": null,
  "co": null,
  "physicalDeliveryOfficeName": null,
  "pwdLastSet": "2015-12-17T13:12:54Z",
  "streetAddress": null,
  "homePhone": null,
  "___PASSWORD_NOTREQD___": false,
  "telephoneNumber": null,
  "dn": "CN=Brian Smith,CN=Users,DC=example,DC=com",
  "title": null,
  "mail": "bsmith@example.com",
  "postOfficeBox": null,
  "___SMARTCARD_REQUIRED___": false,
  "uSNChanged": "86144",
  "___PASSWORD_EXPIRED___": false,
  "initials": null,
  "___LOCK_OUT___": false,
  "company": null,
  "employeeNumber": null,
  "accountExpires": "0",
  "c": null,
  "whenCreated": "20151217131254.0Z",
  "uSNCreated": "86142",
  "division": null,
  "groups": [],
  "___DONT_EXPIRE_PASSWORD___": false,
  "otherHomePhone": []
}
```

Important

- Previous versions of the LDAP connector appended `<GUID=` to the GUID for Active Directory objects. This behavior ensured compatibility with the legacy .NET connector.

The LDAP connector no longer appends `<GUID=` to the object GUID. The new GUID format is compatible with objects created using the AD Powershell connector, for example `e1418d64-096c-4cb0-b903-ebb66562d99d`. In existing deployments, this might mean that your links are incompatible with the new GUID format. To update links to the new format, run a reconciliation operation. To retain the legacy behavior, set `"useOldADGUIDFormat" : true` in your provisioner file.

- You cannot sort by `_id` when you return results from an Active Directory (or Active Directory LDS) server. The `_id` attribute used by default is the `objectGUID`, which is a binary attribute, and cannot be used for sorting.

Note that the command sets the `userAccountControl` to `512`, which is an `enabled` account. The value of the `userAccountControl` determines the account policy. The following list describes the common values for the `userAccountControl`.

512

Enabled account.

514

Disabled account.

544

Enabled account, password not required.

546

Disabled account, password not required.

66048

Enabled account, password does not expire.

66050

Disabled account, password does not expire.

66080

Enabled account, password does not expire and is not required.

66082

Disabled account, password does not expire and is not required.

262656

Enabled account, smartcard required.

262658

Disabled account, smartcard required.

262688

Enabled account, smartcard required, password not required.

262690

Disabled account, smartcard required, password not required.

328192

Enabled account, smartcard required, password does not expire.

328192

Enabled account, smartcard required, password does not expire.

328194

Disabled account, smartcard required, password does not expire.

328224

Enabled account, smartcard required, password does not expire and is not required.

328226

Disabled account, smartcard required, password does not expire and is not required.

Managing Active Directory Groups With the LDAP Connector

The following command creates a basic Active Directory group with the LDAP connector:

```
curl \
  --header "Content-Type: application/json" \
  --header "X-OpenIDM-Username: openidm-admin" \
  --header "X-OpenIDM-Password: openidm-admin" \
  --header "Accept-API-Version: resource=1.0" \
  --request POST \
  --data '{
    "dn": "CN=Employees,DC=example,DC=com"
  }' \
  http://localhost:8080/openidm/system/ad/group?_action=create
{
  "_id": "240da4e9-59d8-1547-ad86-29f5b2b5114d"
}
```

The LDAP connector exposes two special attributes to handle Active Directory group scope and type:

GROUP_SCOPE and **GROUP_TYPE**.

The **GROUP_SCOPE** attribute is defined in the provisioner configuration as follows:

```
{
  ...
  "__GROUP_SCOPE__" : {
    "type" : "string",
    "nativeName" : "__GROUP_SCOPE__",
    "nativeType" : "string"
  },
  ...
}
```

The value of the **GROUP_SCOPE** attribute can be **global**, **domain**, or **universal**. If no group scope is set when the group is created, the scope is **global** by default. For more information about the different group scopes, see the corresponding [Microsoft documentation](#).

The `GROUP_TYPE` attribute is defined in the provisioner configuration as follows:

```
...
"__GROUP_TYPE__" : {
  "type" : "string",
  "nativeName" : "__GROUP_TYPE__",
  "nativeType" : "string"
},
```

The value of the `GROUP_TYPE` attribute can be `security` or `distribution`. If no group type is set when the group is created, the type is `security` by default. For more information about the different group types, see the corresponding [Microsoft documentation](#).

The following example creates a new distribution group, with universal scope:

```
curl \
--header "Content-Type: application/json" \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
--data '{
  "dn": "CN=NewGroup,DC=example,DC=com",
  "__GROUP_SCOPE__": "universal",
  "__GROUP_TYPE__": "distribution"
}' \
http://localhost:8080/openidm/system/ad/group?_action=create
{
  "_id": "f189df8a-276f-9147-8ad5-055b1580cbcb"
}
```

Handling Active Directory Dates

Most dates in Active Directory are represented as the number of 100-nanosecond intervals since January 1, 1601 (UTC). For example:

```
pwdLastSet: 130698687542272930
```

IDM generally represents dates as an ISO 8601-compliant string with `yyyy-MM-dd'T'HH:mm:ssZ` format. For example:

```
2015-03-02T20:17:48Z
```

The generic LDAP connector therefore converts any dates from Active Directory to ISO 8601 format, for fields such as `pwdLastSet`, `accountExpires`, `lockoutTime`, and `lastLogon`.

Working with Multiple Active Directory Domains

In a multi-domain Active Directory Domain Services (AD DS) forest, the global catalog (GC) provides a read-only (searchable) representation of every object in the forest. Each domain controller (DC) in the forest stores a writable replica of the objects *in its domain*. Therefore, a DC can locate only the objects in its domain.

If your Active Directory deployment has only one domain controller, you can configure the connector to connect to that single domain controller. If your deployment spans multiple domains, you must configure the connector to connect to the Global Catalog (GC) to have a comprehensive view of all the domains.

Using a GC as the authoritative data source has the following limitations:

- Only a subset of attributes is replicated from other domains to the GC.

Certain attributes required by the LDAP connector might be missing. To avoid this problem, modify the Active Directory schema to ensure that the required attributes are replicated to the GC.

- Delete operations are not detected immediately.

A liveSync operation will therefore not update IDM with the result of a delete operation. Delete operations are detected by a reconciliation operation, so data stores are only temporarily "out of sync" with regard to deletes.

- Not all group types are supported.

Group membership information is replicated to the GC for universal groups only. You must therefore use universal groups if your directory service has more than one domain.

Note

You can use the **USN** value for liveSync but *must* connect to the GC in this case, and ensure that you never failover to a different GC or to a DC. Using the USN for liveSync instead of the timestamp mechanism is generally preferred, because of the issue with detecting delete operations.

Constructing the LDAP Search Filter

The LDAP connector constructs an LDAP search filter using a combination of filters, in the following order:

```
(& (native filter) (user filter) (object class filter) )
```

The filter components are as follows:

Native Filter

The native filter is the query filter that has been translated to an LDAP query. For example, **uid+eq+user123** is translated to **uid=user123**.

This part of the filter is processed first.

User Filter

You can define a user filter with the properties **accountSearchFilter** and **groupSearchFilter** in the connector configuration.

These properties enable you to construct a more granular or specific search filter. If a user filter is specified, the connector does not use the object class filter. If no user filter is specified, (`accountSearchFilter` and `groupSearchFilter` set to `null` or absent from the connector configuration), the connector uses the object class filter.

Object Class Filter

This part of the filter includes the object classes that the entry must have in order to be returned by the search.

The `__ACCOUNT__` and `__GROUPS__` object classes are defined by the properties `accountObjectClasses` and `groupObjectClasses` in the connector configuration. For example, the following excerpt of a sample `provisioner.openicf-ldap.json` file indicates that the `accountObjectClasses` include the LDAP object classes `top`, `person`, `organizationalPerson`, and `inetOrgPerson`:

```
"configurationProperties" : {
  ...
  "accountObjectClasses" : [
    "top",
    "person",
    "organizationalPerson",
    "inetOrgPerson"
  ],
  ...
}
```

With this configuration, the search filter for accounts is constructed as follows:

```
(&(objectClass=top)(objectClass=person)(objectClass=organizationalPerson)(objectClass=inetOrgPerson))
```

If no `accountObjectClasses` or `groupObjectClasses` are defined in the connector configuration, the connector uses the name of the ICF ObjectClass in the filter. For example, an object of type `organizationUnit` will result in:

```
(&(objectClass=organizationUnit))
```

OpenICF Interfaces Implemented by the LDAP Connector

The LDAP Connector implements the following OpenICF interfaces.

Authenticate

Provides simple authentication with two parameters, presumed to be a user name and password.

Create

Creates an object and its `uid`.

Delete

Deletes an object, referenced by its `uid`.

Resolve Username

Resolves an object by its username and returns the **uid** of the object.

Schema

Describes the object types, operations, and options that the connector supports.

Script on Connector

Enables an application to run a script in the context of the connector. Any script that runs on the connector has the following characteristics:

- The script runs in the same execution environment as the connector and has access to all the classes to which the connector has access.
- The script has access to a **connector** variable that is equivalent to an initialized instance of the connector. At a minimum, the script can access the connector configuration.
- The script has access to any script-arguments passed in by the application.

Search

Searches the target resource for all objects that match the specified object class and filter.

Sync

Polls the target resource for synchronization events, that is, native changes to objects on the target resource.

Test

Tests the connector configuration. Testing a configuration checks all elements of the environment that are referred to by the configuration are available. For example, the connector might make a physical connection to a host that is specified in the configuration to verify that it exists and that the credentials that are specified in the configuration are valid.

This operation might need to connect to a resource, and, as such, might take some time. Do not invoke this operation too often, such as before every provisioning operation. The test operation is not intended to check that the connector is alive (that is, that its physical connection to the resource has not timed out).

You can invoke the test operation before a connector configuration has been validated.

Update

Updates (modifies or replaces) objects on a target resource.

LDAP Connector Configuration

The LDAP Connector has the following configurable properties.

Configuration properties

Property	Type	Default	Encrypted ^a	Required ^b
<code>filterWithOrInsteadOfAnd</code>	<code>boolean</code>	<code>false</code>		Sync
Normally the filter used to fetch change log entries is an and-based filter retrieving an interval of change entries. If this property is set, the filter will or together the required change numbers instead.				
<code>objectClassesToSynchronize</code>	<code>String[]</code>	<code>['inetOrgPerson']</code>		Sync
The object classes to synchronize. The change log is for all objects; this filters updates to just the listed object classes. You should not list the superclasses of an object class unless you intend to synchronize objects with any of the superclass values. For example, if only "inetOrgPerson" objects should be synchronized, but the superclasses of "inetOrgPerson" ("person", "organizationalperson" and "top") should be filtered out, then list only "inetOrgPerson" here. All objects in LDAP are subclassed from "top". For this reason, you should never list "top", otherwise no object would be filtered.				
<code>baseContextsToSynchronize</code>	<code>String[]</code>	<code>[]</code>		Sync
One or more starting points in the LDAP tree that will be used to determine if a change should be synchronized. The base contexts attribute will be used to synchronize a change if this property is not set.				
<code>attributesToSynchronize</code>	<code>String[]</code>	<code>[]</code>		Sync
The names of the attributes to synchronize. This ignores updates from the change log if they do not update any of the named attributes. For example, if only "department" is listed, then only changes that affect "department" will be processed. All other updates are ignored. If blank (the default), then all changes are processed.				
<code>changeNumberAttribute</code>	<code>String</code>	<code>changeNumber</code>		Sync
The name of the change number attribute in the change log entry.				
<code>modifiersNamesToFilterOut</code>	<code>String[]</code>	<code>[]</code>		Sync
The list of names (DNs) to filter from the changes. Changes with the attribute "modifiersName" that match entries in this list will be filtered out. The standard value is the administrator name used by this adapter, to prevent loops. Entries should be of the format "cn=Directory Manager".				
<code>credentials</code>	<code>GuardedString</code>	<code>null</code>	Yes	No
Password for the principal.				
<code>changeLogBlockSize</code>	<code>int</code>	<code>100</code>		Sync
The number of change log entries to fetch per query.				
<code>useTimestampsForSync</code>	<code>boolean</code>	<code>false</code>		Sync
If true, the connector will use the createTimestamp and modifyTimestamp system attributes to detect changes (Create/Update) on the directory instead of native change detection mechanism (cn=changelog on OpenDJ or Update Sequence Number -USN- on Active Directory for instance). Default value is false.				
<code>accountSynchronizationFilter</code>	<code>String</code>	<code>null</code>		Sync

Property	Type	Default	Encrypted ^a	Required ^b
An optional LDAP filter for the objects to synchronize. Because the change log is for all objects, this filter updates only objects that match the specified filter. If you specify a filter, an object will be synchronized only if it matches the filter and includes a synchronized object class.				
<code>removeLogEntryObjectClassFromFilter</code>	<code>boolean</code>	<code>true</code>		Sync
If this property is set (the default), the filter used to fetch change log entries does not contain the "changeLogEntry" object class, expecting that there are no entries of other object types in the change log.				
<code>alternateKeyStorePassword</code>	<code>GuardedString</code>	<code>null</code>	Yes	No
Password to use for the alternate keystore				
<code>groupSynchronizationFilter</code>	<code>String</code>	<code>null</code>		Sync
An optional LDAP filter for the objects to synchronize. Because the change log is for all objects, this filter updates only objects that match the specified filter. If you specify a filter, an object will be synchronized only if it matches the filter and includes a synchronized object class.				
<code>groupMemberAttribute</code>	<code>String</code>	<code>uniqueMember</code>		No
The name of the group attribute that will be updated with the distinguished name of the user when the user is added to the group.				
<code>accountSearchFilter</code>	<code>String</code>	<code>null</code>		No
An optional LDAP filter to control which accounts are returned from the LDAP resource. If no filter is specified, only accounts that include all specified object classes are returned.				
<code>privateKeyAlias</code>	<code>String</code>	<code>null</code>		No
Specifies the name of a private key alias from the keystore that should be used for SSL mutual authentication. If null, no private key is sent during SSL handshake so only server cert is used. This alias name is case sensitive.				
<code>ssl</code>	<code>boolean</code>	<code>false</code>		No
Select the check box to connect to the LDAP server using SSL.				
<code>maintainPosixGroupMembership</code>	<code>boolean</code>	<code>false</code>		No
When enabled and a user is renamed or deleted, update any POSIX groups to which the user belongs to reflect the new name. Otherwise, the LDAP resource must maintain referential integrity with respect to group membership.				
<code>checkAliveMinInterval</code>	<code>long</code>	<code>60</code>		No
The minimum interval (seconds) at which the target directory is polled when a connection is reused from the pool. Defaults to 60 seconds.				
<code>groupSearchFilter</code>	<code>String</code>	<code>null</code>		No
An optional LDAP filter to control which groups are returned from the LDAP resource. If no filter is specified, only groups that include all specified object classes are returned.				

Property	Type	Default	Encrypted ^a	Required ^b
<code>referralsHandling</code>	String	<code>follow</code>		No
Defines how to handle LDAP referrals. Possible values can be follow, ignore or throw.				
<code>host</code>	String	<code>null</code>		No
The name or IP address of the host where the LDAP server is running.				
<code>maintainLdapGroupMembership</code>	boolean	<code>false</code>		No
When enabled and a user is renamed or deleted, update any LDAP groups to which the user belongs to reflect the new name. Otherwise, the LDAP resource must maintain referential integrity with respect to group membership.				
<code>resetSyncToken</code>	String	<code>never</code>		No
Connector can reset the sync token if ever the value of the sync token is greater than the last change number in the directory changelog. Defaults to "never" (no reset). If set to "first" it will reset the sync token to the value of the firstChangeNumber changelog attribute. If set to "last" it will reset the sync token to the value of the lastChangeNumber changelog attribute.				
<code>vlvSortAttribute</code>	String	<code>uid</code>		No
Specify the sort attribute to use for VLV indexes on the resource.				
<code>convertGTTToISO8601</code>	String[]	<code>['whenCreated', 'whenChanged']</code>		No
Converts the Greenwich Time to ISO8601 format				
<code>baseContexts</code>	String[]	<code>[]</code>		No
One or more starting points in the LDAP tree that will be used when searching the tree. Searches are performed when discovering users from the LDAP server or when looking for the groups of which a user is a member.				
<code>hostNameVerification</code>	boolean	<code>false</code>		No
If true, the connector will verify the hostname in the certificate (subject + alternative subject) against the defined hostNameVerifierPattern.				
<code>blockSize</code>	int	<code>100</code>		No
The maximum number of entries that can be in a block when retrieving entries in blocks.				
<code>groupObjectClasses</code>	String[]	<code>['top', 'groupOfUniqueN</code>		No
The default list of object classes that will be used when creating new group objects in the LDAP tree. This can be overridden by specifying the group object classes during the Create operation.				
<code>accountUserNameAttributes</code>	String[]	<code>['uid', 'cn']</code>		No
Attribute or attributes which holds the account's user name. They will be used when authenticating to find the LDAP entry for the user name to authenticate.				

Property	Type	Default	Encrypted ^a	Required ^b
failover	String[]	[]		No
List all servers that should be used for failover in case the preferred server fails. If the preferred server fails, JNDI will connect to the next available server in the list. List all servers in the form of "ldap://ldap.example.com:389/", which follows the standard LDAP v3 URLs described in RFC 2255. Only the host and port parts of the URL are relevant in this setting.				
port	int	389		No
TCP/IP port number used to communicate with the LDAP server.				
convertADIntervalToISO8601	String[]	['pwdLastSet', 'accountExpires', 'lockoutTime', 'lastLogon']		No
Converts the AD Interval to ISO8601				
hostNameVerifierPattern	String	null		No
A simple pattern used to match the hostname from the certificate. It can contains * character (server1.example.com, *.example.com)				
passwordAttribute	String	userPassword		No
The name of the LDAP attribute that holds the password. When changing a users password, the new password is set to this attribute.				
useDNSSRVRecord	boolean	false		No
If true, the connector will do a DNS query to find SRV records associated with the value set for host property ("_ldap._tcp.example.com" for example). Defaults to false.				
getGroupMemberId	boolean	false		No
Specifies whether to add an extra _memberId attribute to get the group members __UID__. CAUTION: Setting this property to true can incur a large performance cost on group handling.				
lastCheckAlive	long	1670009912790		No
The last time the connector was checked to see if it was alive				
ldapGroupsUseStaticGroups	boolean	false		No
When set to true, The ldapGroups attribute will search group membership through static groups only. If false, it will leverage the "memberOf" attribute of an object (defaults to true).				
startTLS	boolean	false		No
Specifies whether to use the startTLS operation to initiate a TLS/SSL session.				
allowTreeDelete	boolean	false		No
Connector can delete an entry (node) with leaf entry if this value is set to true (defaults to false). The LDAP control LDAP_SERVER_TREE_DELETE_OID (1.2.840.113556.1.4.805) is used.				

Property	Type	Default	Encrypted ^a	Required ^b
<code>respectResourcePasswordPolicyChange</code>	<code>boolean</code>	<code>false</code>		No
When this resource is specified in a Login Module (i.e., this resource is a pass-through authentication target) and the resource's password policy is configured for change-after-reset, a user whose resource account password has been administratively reset will be required to change that password after successfully authenticating.				
<code>uidAttribute</code>	<code>String</code>	<code>entryUUID</code>		No
The name of the LDAP attribute that is mapped to the OpenICF UID attribute.				
<code>principal</code>	<code>String</code>	<code>null</code>		No
The distinguished name with which to authenticate to the LDAP server.				
<code>accountObjectClasses</code>	<code>String[]</code>	<code>['top', 'person', 'organizational', 'inetOrgPerson']</code>		No
The default list of object classes that will be used when creating new user objects in the LDAP tree. This can be overridden by specifying the user object classes during the Create operation.				
<code>alternateKeyType</code>	<code>String</code>	<code>null</code>		No
Defines the type of the alternate key store. Valid values are JKS, JCEKS and PKCS12				
<code>passwordHashAlgorithm</code>	<code>String</code>	<code>null</code>		No
Indicates the algorithm that the Identity system should use to hash the password. Currently supported values are SSHA, SHA, SMD5, MD5 and WIN-AD (when AD is the target). A blank value indicates that the system will not hash passwords. This will cause clear text passwords to be stored in LDAP unless the LDAP server performs the hash (as ForgeRock's OpenDJ does, for example).				
<code>alternateKeyStore</code>	<code>String</code>	<code>null</code>		No
Defines the filename of an alternate keystore. If specified, the connector will not use the default keystore specified by the <code>javax.net.ssl.keyStore</code> property.				
<code>authType</code>	<code>String</code>	<code>simple</code>		No
The authentication mechanism to use: Simple or SASL-GSSAPI. Defaults to "simple".				
<code>connectionTimeout</code>	<code>int</code>	<code>30000</code>		No
The timeout (in ms) before the connection attempt is aborted.				
<code>useBlocks</code>	<code>boolean</code>	<code>false</code>		No
Specifies whether to use block-based LDAP controls, like the simple paged results or VLV control. When performing search operations on large numbers of entries, the entries are returned in blocks to reduce the amount of memory used by the operation.				
<code>readSchema</code>	<code>boolean</code>	<code>true</code>		No

Property	Type	Default	Encrypted ^a	Required ^b
If true, the connector will read the schema from the server. If false, the connector will provide a default schema based on the object classes in the configuration. This property must be true in order to use extended object classes.				
<code>usePagedResultControl</code>	boolean	false		No
When enabled, the LDAP Paged Results control is preferred over the VLV control when retrieving entries. If disabled, paged queries will be ignored.				
<code>useOldADGUIDFormat</code>	boolean	false		No
The connector used to transform the AD ObjectGUID in the form <GUID=xxxxxx>. It now used dashed notation (xxxx-xx-xx-xxxx-xxxxxx) by default. Set to true to keep the old format.				
<code>sendCAUDTxId</code>	boolean	false		No
Connector can send the Common Audit Transaction Id (if present) to the target OpenDJ server when this value is set to true (defaults to false). The LDAP control TransactionIdControl (1.3.6.1.4.1.36733.2.1.5.1) is used.				
<code>gssapiLoginContext</code>	String	null		No
Defines the name used in the JAAS configuration file to define the JAAS login configuration. If null, it defaults to "org.identityconnectors.ldap.LdapConnector".				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

Marketo Connector

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

The Marketo connector enables synchronization between IDM managed users and a Marketo leads database.

This connector forms part of ForgeRock's support for customer data management (CDM). You can synchronize any managed user to Marketo—those who have been added directly to the IDM repository, and those who have registered themselves through one of the Social Identity Providers described in "*Social Registration*" in the *Self-Service Reference*.

The Marketo connector is an implementation of the Scripted Groovy Connector Toolkit, and enables you to interact with leads in a Marketo database, using Groovy scripts for the ICF operations.

To use the Marketo connector, you need the following:

- A Marketo account
- A client ID and client secret

- The REST API URL for your IDM service
- A custom list created in your Marketo leads database

To obtain these details from Marketo, see the [Marketo documentation](#).

A sample connector configuration file is available, at `/path/to/openidm/samples/example-configurations/provisioners/provisioner.openicf-marketo.json`. To test the Marketo connector, copy that file to your project's `conf/` directory, and edit at least the `configurationProperties` to provide the REST API URL, client ID and client secret.

Set the `enabled` property in the connector configuration to `true`. IDM encrypts the client secret on startup. Optionally, you can specify the `listName` to which leads should be added when they are synchronized from IDM. The following excerpt from the sample connector configuration file shows the properties that you must set:

```
{
  "displayName" : "MarketoConnector",
  "description" : "Connector used to sync users to Marketo leads",
  "author" : "ForgeRock",
  "enabled" : true,
  "connectorRef" : {
    "bundleName" : "org.forgerock.openicf.connectors.marketo-connector",
    "bundleVersion" : "1.5.3.0",
    "connectorName" : "org.forgerock.openicf.connectors.marketo.MarketoConnector"
  },
  ...
  "configurationProperties" : {
    "instance" : "<INSTANCE_FQDN>",
    "clientId" : "<CLIENT_ID>",
    "clientSecret" : "<CLIENT_SECRET>",
    "leadFields" : null,
    "partitionName" : null,
    "listName" : "<LEAD_LIST_NAME>",
    ...
  },
  ...
}
```

instance

To locate the REST API endpoint URL in Marketo, select Admin > Web Services, scroll down to REST API, and find the endpoint. Use that REST endpoint as the value of the `instance` property in your connector configuration. Remove the protocol and `/rest` from the URL. For example, if the endpoint is `https://some-number.mktorest.com/rest`, the value of the `instance` property must be `some-number.mktorest.com`.

clientId

Locate the client ID in the details of your Marketo service [LaunchPoint](#).

clientSecret

Locate the client secret in the details of your Marketo service [LaunchPoint](#).

listName

The name of the custom list created in your Marketo Leads database.

You can also configure the Marketo connector through the Admin UI. Select Configure > Connectors > New Connector and select Marketo Connector - 1.5.20.11 as the Connector Type. Configuration properties correspond to those described in the previous list. For details of all the configuration properties, see "[Marketo Connector Configuration](#)".

When your connector is configured correctly, you can test its status by running the following command:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system?_action=test"
[
  {
    "name": "marketo",
    "enabled": true,
    "config": "config/provisioner.openicf/marketo",
    "objectTypes": [
      "ALL",
      "account"
    ],
    "connectorRef": {
      "bundleName": "org.forgerock.openicf.connectors.marketo-connector",
      "connectorName": "org.forgerock.openicf.connectors.marketo.MarketoConnector",
      "bundleVersion": "1.5.20.11"
    },
    "displayName": "Marketo Connector",
    "ok": true
  }
]
```

A status of `"ok": true` indicates that the connector can reach your Marketo database.

Reconciling Users With a Marketo Leads Database

The Marketo connector enables you to reconcile IDM users (including managed users and users who have registered through a social identity provider) with a Marketo leads database. To set up reconciliation to a Marketo database, copy the following sample mapping file to your project's `conf` directory:

`/path/to/openidm/samples/example-configurations/marketo/sync.json`

This file sets up a mapping from the managed user repository to Marketo user accounts. The file includes transformations for user accounts registered through Facebook and LinkedIn. You can use these transformations as a basis for transformations from other social identity providers.

If you have an existing mapping configuration, add the content of this sample `sync.json` to your existing mapping.

The sample mapping restricts reconciliation to users who have accepted the marketing preferences with the following `validSource` script:

```
"validSource" : {
  "type" : "text/javascript",
  "globals" : {
    "preferences" : [
      "marketing"
    ]
  },
  "file" : "ui/preferenceCheck.js"
}
```

When a user registers with IDM, they can choose to accept this condition. As a regular user, they can also select (or deselect) the condition in the End User UI by logging into IDM at <http://localhost:8080/>, and selecting Preferences.

If a user deselects the marketing preference after their account has been reconciled to Marketo, the next reconciliation run will remove the account from the Marketo database.

For more information on how preferences work in a mapping, see "Configure User Preferences" in the *Self-Service Reference*.

Implementation Specifics

For PATCH requests, a connector can potentially add, remove, or replace an attribute value. The Marketo connector does not implement the add or remove operations, so a PATCH request always replaces the entire attribute value with the new value.

OpenICF Interfaces Implemented by the Marketo Connector

The Marketo Connector implements the following OpenICF interfaces.

Authenticate

Provides simple authentication with two parameters, presumed to be a user name and password.

Create

Creates an object and its `uid`.

Delete

Deletes an object, referenced by its `uid`.

Resolve Username

Resolves an object by its username and returns the `uid` of the object.

Schema

Describes the object types, operations, and options that the connector supports.

Script on Connector

Enables an application to run a script in the context of the connector. Any script that runs on the connector has the following characteristics:

- The script runs in the same execution environment as the connector and has access to all the classes to which the connector has access.
- The script has access to a **connector** variable that is equivalent to an initialized instance of the connector. At a minimum, the script can access the connector configuration.
- The script has access to any script-arguments passed in by the application.

Script on Resource

Runs a script on the target resource that is managed by this connector.

Search

Searches the target resource for all objects that match the specified object class and filter.

Sync

Polls the target resource for synchronization events, that is, native changes to objects on the target resource.

Test

Tests the connector configuration. Testing a configuration checks all elements of the environment that are referred to by the configuration are available. For example, the connector might make a physical connection to a host that is specified in the configuration to verify that it exists and that the credentials that are specified in the configuration are valid.

This operation might need to connect to a resource, and, as such, might take some time. Do not invoke this operation too often, such as before every provisioning operation. The test operation is not intended to check that the connector is alive (that is, that its physical connection to the resource has not timed out).

You can invoke the test operation before a connector configuration has been validated.

Update

Updates (modifies or replaces) objects on a target resource.

Marketo Connector Configuration

The Marketo Connector has the following configurable properties.

Configuration properties

Property	Type	Default	Encrypted ^a	Required ^b
<code>customSensitiveConfiguration</code>	<code>GuardedString</code>	<code>null</code>	Yes	No
Custom Sensitive Configuration script for Groovy ConfigSlurper				
<code>customConfiguration</code>	<code>String</code>	<code>null</code>		No
Custom Configuration script for Groovy ConfigSlurper				
<code>instance</code>	<code>String</code>	<code>null</code>		Yes
The Marketo-assigned FQDN for your instance				
<code>clientId</code>	<code>String</code>	<code>null</code>		Yes
Your OAuth2 client ID				
<code>clientSecret</code>	<code>GuardedString</code>	<code>null</code>	Yes	Yes
Your OAuth2 client secret				
<code>leadFields</code>	<code>String</code>	<code>null</code>		No
Comma-delimited list of lead fields to fetch; Leave empty for default set				
<code>partitionName</code>	<code>String</code>	<code>null</code>		No
Name of the partition in which to create and update leads; May be left empty				
<code>listName</code>	<code>String</code>	<code>null</code>		Yes
Name of the Marketo static list the connector will use to manage leads				
<code>accessToken</code>	<code>String</code>	<code>null</code>		Yes
The access token for the application				
<code>tokenExpiration</code>	<code>Long</code>	<code>null</code>		Yes
The expiration token for the application				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

Operation Script Files

Property	Type	Default	Encrypted ^a	Required ^b
<code>createScriptFileName</code>	<code>String</code>	<code>CreateMarketo.groovy</code>		Create
The name of the file used to perform the CREATE operation.				
<code>customizerScriptFileName</code>	<code>String</code>	<code>null</code>		No

Property	Type	Default	Encrypted ^a	Required ^b
The script used to customize some function of the connector. Read the documentation for more details.				
<code>authenticateScriptFileName</code>	String	null		Authenticate
The name of the file used to perform the AUTHENTICATE operation.				
<code>scriptOnResourceScriptFileName</code>	String	null		Script On Resource
The name of the file used to perform the RUNSCRIPTONRESOURCE operation.				
<code>deleteScriptFileName</code>	String	DeleteMarketo.groovy		Delete
The name of the file used to perform the DELETE operation.				
<code>resolveUsernameScriptFileName</code>	String	null		Resolve Username
The name of the file used to perform the RESOLVE_USERNAME operation.				
<code>searchScriptFileName</code>	String	SearchMarketo.groovy		Get Search
The name of the file used to perform the SEARCH operation.				
<code>updateScriptFileName</code>	String	UpdateMarketo.groovy		Update
The name of the file used to perform the UPDATE operation.				
<code>schemaScriptFileName</code>	String	SchemaMarketo.groovy		Schema
The name of the file used to perform the SCHEMA operation.				
<code>testScriptFileName</code>	String	TestMarketo.groovy		Test
The name of the file used to perform the TEST operation.				
<code>syncScriptFileName</code>	String	null		Sync
The name of the file used to perform the SYNC operation.				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

Groovy Engine configuration

Property	Type	Default	Encrypted ^a	Required ^b
<code>targetDirectory</code>	File	null		No

Property	Type	Default	Encrypted ^a	Required ^b
Directory into which to write classes.				
<code>warningLevel</code>	<code>int</code>	<code>1</code>		No
Warning Level of the compiler				
<code>scriptExtensions</code>	<code>String[]</code>	<code>['groovy']</code>		No
Gets the extensions used to find groovy files				
<code>minimumRecompilationInterval</code>	<code>int</code>	<code>100</code>		No
Sets the minimum of time after a script can be recompiled.				
<code>scriptBaseClass</code>	<code>String</code>	<code>null</code>		No
Base class name for scripts (must derive from Script)				
<code>scriptRoots</code>	<code>String[]</code>	<code>null</code>		Yes
The root folder to load the scripts from. If the value is null or empty the classpath value is used.				
<code>tolerance</code>	<code>int</code>	<code>10</code>		No
The error tolerance, which is the number of non-fatal errors (per unit) that should be tolerated before compilation is aborted.				
<code>debug</code>	<code>boolean</code>	<code>false</code>		No
If true, debugging code should be activated				
<code>classpath</code>	<code>String[]</code>	<code>[]</code>		No
Classpath for use during compilation.				
<code>disabledGlobalASTTransformations</code>	<code>String[]</code>	<code>null</code>		No
Sets a list of global AST transformations which should not be loaded even if they are defined in META-INF/org.codehaus.groovy.transform.ASTTransformation files. By default, none is disabled.				
<code>verbose</code>	<code>boolean</code>	<code>false</code>		No
If true, the compiler should produce action information				
<code>sourceEncoding</code>	<code>String</code>	<code>UTF-8</code>		No
Encoding for source files				
<code>recompileGroovySource</code>	<code>boolean</code>	<code>false</code>		No
If set to true recompilation is enabled				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

MongoDB Connector

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

The MongoDB connector is an implementation of the Scripted Groovy Connector Toolkit. This connector enables you to interact with a MongoDB document database, using Groovy scripts for the ICF operations.

The connector is bundled with IDM in the `connectors/` directory (`mongodb-connector-1.5.20.8.jar`).

Note

Version 1.5.20.8 of the connector is supported only with MongoDB version 3.6.x.

Before You Start

In a production environment, enable access control on your MongoDB database. If your connector will manage MongoDB users and roles, you must create an administrative user in the `admin` database. If your connector will manage collections in a database, this administrative user must create a specific user and role for the connector for the target database.

For information about enabling access control in MongoDB, see the [MongoDB documentation](#).

The commands in this chapter assume an administrative user named `myUserAdmin` with password `Passw0rd` who has the `readWrite` role on the `test` database.

Configuring the MongoDB Connector

The easiest way to configure the MongoDB connector is through the Admin UI:

1. Select **Configure > Connectors > New Connector**.
2. Enter a name for the connector configuration, for example, `mongodb`.
3. Select **MongoDB Connector - 1.5.20.8** as the Connector Type.
4. Enable the connector, and set the Base Configuration Properties. For information about the configurable properties, see "[Basic Configuration Properties](#)".

Alternatively, configure the connector with a configuration file.

A sample connector configuration file (`provisioner.openicf-mongodb.json`) is provided in the `/path/to/openidm/samples/example-configurations/provisioners` directory. Copy the sample connector configuration

to your project's `conf/` directory, and adjust the `configurationProperties` to match your MongoDB instance:

```
"configurationProperties" : {
  "connectionURI" : "mongodb://localhost:27017",
  "host" : "localhost",
  "port" : "27017",
  "user" : "myUserAdmin",
  "password" : "Passw0rd",
  "userDatabase" : "admin",
  "database" : "test",
  ...
}
```

Set `"enabled" : true` to enable the connector.

When your connector is configured correctly, you can test its status by running the following command:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system?_action=test"
[
  {
    "name": "mongodb",
    "enabled": true,
    "config": "config/provisioner.openicf/mongodb",
    "connectorRef": {
      "bundleVersion": "1.5.20.8",
      "bundleName": "org.forgerock.openicf.connectors.mongodb-connector",
      "connectorName": "org.forgerock.openicf.connectors.mongodb.MongoDBConnector"
    },
    "displayName": "MongoDB Connector",
    "objectTypes": [
      "__ALL__",
      "account",
      "role"
    ],
    "ok": true
  }
]
```

A status of `"ok": true` indicates that the MongoDB connector can connect to the database.

OpenICF Interfaces Implemented by the MongoDB Connector

The MongoDB Connector implements the following OpenICF interfaces.

Authenticate

Provides simple authentication with two parameters, presumed to be a user name and password.

Create

Creates an object and its **uid**.

Delete

Deletes an object, referenced by its **uid**.

Resolve Username

Resolves an object by its username and returns the **uid** of the object.

Schema

Describes the object types, operations, and options that the connector supports.

Script on Connector

Enables an application to run a script in the context of the connector. Any script that runs on the connector has the following characteristics:

- The script runs in the same execution environment as the connector and has access to all the classes to which the connector has access.
- The script has access to a **connector** variable that is equivalent to an initialized instance of the connector. At a minimum, the script can access the connector configuration.
- The script has access to any script-arguments passed in by the application.

Script on Resource

Runs a script on the target resource that is managed by this connector.

Search

Searches the target resource for all objects that match the specified object class and filter.

Sync

Polls the target resource for synchronization events, that is, native changes to objects on the target resource.

Test

Tests the connector configuration. Testing a configuration checks all elements of the environment that are referred to by the configuration are available. For example, the connector might make a physical connection to a host that is specified in the configuration to verify that it exists and that the credentials that are specified in the configuration are valid.

This operation might need to connect to a resource, and, as such, might take some time. Do not invoke this operation too often, such as before every provisioning operation. The test operation

is not intended to check that the connector is alive (that is, that its physical connection to the resource has not timed out).

You can invoke the test operation before a connector configuration has been validated.

Update

Updates (modifies or replaces) objects on a target resource.

MongoDB Connector Configuration

The MongoDB Connector has the following configurable properties.

Configuration properties

Property	Type	Default	Encrypted ^a	Required ^b
<code>customSensitiveConfiguration</code>	<code>GuardedString</code>	<code>null</code>	Yes	No
Custom Sensitive Configuration script for Groovy ConfigSlurper				
<code>customConfiguration</code>	<code>String</code>	<code>null</code>		No
Custom Configuration script for Groovy ConfigSlurper				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

Operation Script Files

Property	Type	Default	Encrypted ^a	Required ^b
<code>createScriptFileName</code>	<code>String</code>	<code>null</code>		Create
The name of the file used to perform the CREATE operation.				
<code>customizerScriptFileName</code>	<code>String</code>	<code>null</code>		No
The script used to customize some function of the connector. Read the documentation for more details.				
<code>authenticateScriptFileName</code>	<code>String</code>	<code>null</code>		Authenticate
The name of the file used to perform the AUTHENTICATE operation.				
<code>scriptOnResourceScriptFileName</code>	<code>String</code>	<code>null</code>		Script On Resource
The name of the file used to perform the RUNSCRIPTONRESOURCE operation.				
<code>deleteScriptFileName</code>	<code>String</code>	<code>null</code>		Delete
The name of the file used to perform the DELETE operation.				

Property	Type	Default	Encrypted ^a	Required ^b
<code>resolveUsernameScriptFileName</code>	String	null		Resolve Username
The name of the file used to perform the RESOLVE_USERNAME operation.				
<code>searchScriptFileName</code>	String	null		Get Search
The name of the file used to perform the SEARCH operation.				
<code>updateScriptFileName</code>	String	null		Update
The name of the file used to perform the UPDATE operation.				
<code>schemaScriptFileName</code>	String	null		Schema
The name of the file used to perform the SCHEMA operation.				
<code>testScriptFileName</code>	String	null		Test
The name of the file used to perform the TEST operation.				
<code>syncScriptFileName</code>	String	null		Sync
The name of the file used to perform the SYNC operation.				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

Groovy Engine configuration

Property	Type	Default	Encrypted ^a	Required ^b
<code>targetDirectory</code>	File	null		No
Directory into which to write classes.				
<code>warningLevel</code>	int	1		No
Warning Level of the compiler				
<code>scriptExtensions</code>	String[]	['groovy']		No
Gets the extensions used to find groovy files				
<code>minimumRecompilationInterval</code>	int	100		No
Sets the minimum of time after a script can be recompiled.				
<code>scriptBaseClass</code>	String	null		No
Base class name for scripts (must derive from Script)				
<code>scriptRoots</code>	String[]	null		Yes

Property	Type	Default	Encrypted ^a	Required ^b
The root folder to load the scripts from. If the value is null or empty the classpath value is used.				
tolerance	int	10		No
The error tolerance, which is the number of non-fatal errors (per unit) that should be tolerated before compilation is aborted.				
debug	boolean	false		No
If true, debugging code should be activated				
classpath	String[]	[]		No
Classpath for use during compilation.				
disabledGlobalASTTransformations	String[]	null		No
Sets a list of global AST transformations which should not be loaded even if they are defined in META-INF/org.codehaus.groovy.transform.ASTTransformation files. By default, none is disabled.				
verbose	boolean	false		No
If true, the compiler should produce action information				
sourceEncoding	String	UTF-8		No
Encoding for source files				
recompileGroovySource	boolean	false		No
If set to true recompilation is enabled				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

Basic Configuration Properties

Property	Type	Default	Encrypted ^a	Required ^b
connectionURI	String	null		No
The MongoDB client connection URI, for example "mongodb://localhost:27017". Overrides other connection parameters				
host	String	localhost		No
The MongoDB server host name (localhost by default).				
port	int	27017		No
The MongoDB server port number (27017 by default).				
user	String	null		No
The MongoDB username				

Property	Type	Default	Encrypted ^a	Required ^b
password	GuardedString	null	Yes	No
The password used to connect to MongoDB				
userDatabase	String	null		No
The name of the database in which the MongoDB user is defined				
clusterAddresses	String[]	null		No
A list of additional mongodbDB servers when connecting to a MongoDB cluster (["host1:27017", "host2:27017", "..."])				
dateAttributes	String[]	[]		No
Defines the list of attributes to convert to MongoDB BSON Date type on create/update.				
database	String	null		No
The database to use				
arrayAttributes	String[]	[]		No
Defines the list of attributes that should be considered as BSON Arrays.				
includeNullValue	boolean	false		No
If set to true, retains null values in the target MongoDB document (false by default).				
includeEmptyList	boolean	false		No
If set to true, retains null values in the target MongoDB document (false by default).				
dateFormat	String	yyyy-MM-dd'T'HH:mm:ss'Z'		No
Defines the date format to use for MongoDB Date attributes (defaults to ISO 8601 "yyyy-MM-ddTHH:mm:ssZ").				
timeZone	String	UTC		No
Defines the timezone to use for MongoDB Date attributes.				
ICFName	String	name		No
Defines the name to use in the target MongoDB document for the ICF __NAME__ attribute.				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

Connection Configuration Properties

Property	Type	Default	Encrypted ^a	Required ^b
sslEnabled	boolean	true		No

Property	Type	Default	Encrypted ^a	Required ^b
Use secure socket layer to connect to MongoDB (true by default)				
<code>sslHostNameValidation</code>	<code>boolean</code>	<code>true</code>		No
Defines if host name should be validated when SSL is enabled				
<code>maxConnectionIdleTime</code>	<code>int</code>	<code>0</code>		No
The maximum idle time for a pooled connection in ms (0 means no limit)				
<code>maxConnectionLifeTime</code>	<code>int</code>	<code>0</code>		No
The maximum life time for a pooled connection in ms (0 means no limit)				
<code>minConnectionsPerHost</code>	<code>int</code>	<code>0</code>		No
The minimum number of connections per host (must be ≥ 0)				
<code>maxConnectionsPerHost</code>	<code>int</code>	<code>5</code>		No
The maximum number of connections per host (must be > 0)				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

MS Graph API Java Connector

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

The MS Graph API Java connector uses the MS Graph SDK for Java and the Authentication Providers for the MS Graph Java SDK. Unlike the PowerShell connector for Azure, the MS Graph API connector is a Java connector, and does not need a .NET RCS to run. As a Java connector, the MS Graph API connector functions like any standard IDM connector.

The MS Graph API connector can read, search, and fetch data from Microsoft Azure, when Azure is the authoritative data source, and can provision to Azure, when IDM is the authoritative data source.

The MS Graph API connector is available from the [ForgeRock Download Center](#). The connector bundles all its dependencies.

Before You Start

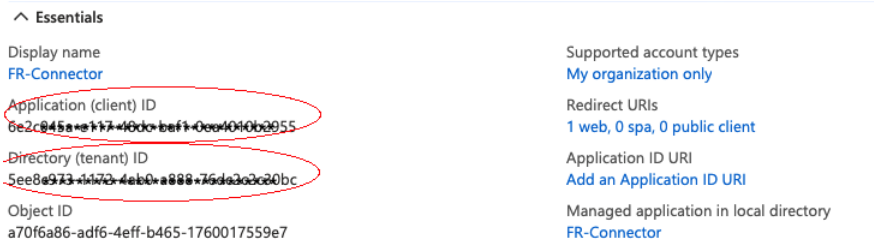
Before you can use the connector, you must register an application with Azure. You need a Microsoft Azure subscription to complete this procedure:

1. Log in to the [MS Azure portal](#) as an administrative user.

2. Under Azure services, select App registrations.
3. On the Register an application page, enter a name for the application; for example, FR-Connector.

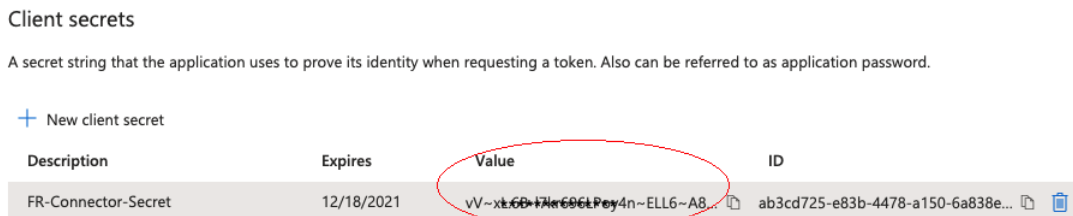
Select the supported account types, and enter a Redirect URI. The redirect URI is the IDM URI that Azure should redirect to after successful authentication; for example, <https://idm.example.com:8443/>.

4. On the new registration page for your application, make a note of the Application (client) ID and the Directory (tenant) ID. You will need these to configure the connector:



^ Essentials	
Display name FR-Connector	Supported account types My organization only
Application (client) ID 6e2c845a-117-40dc-bef1-0ee4010b2955	Redirect URIs 1 web, 0 spa, 0 public client
Directory (tenant) ID 5ee8973-1173-4eb0-a888-76da30cc20bc	Application ID URI Add an Application ID URI
Object ID a70f6a86-adf6-4eff-b465-1760017559e7	Managed application in local directory FR-Connector

5. Generate a client secret:
 - a. Select Certificates & secrets > New client secret.
 - b. Enter a description, select an expiry date, and click Add.
 - c. Copy the client secret Value:



Client secrets

A secret string that the application uses to prove its identity when requesting a token. Also can be referred to as application password.

+ New client secret

Description	Expires	Value	ID
FR-Connector-Secret	12/18/2021	vV~xL6P7wC6CLPey4n~ELL6~A8...	ab3cd725-e83b-4478-a150-6a838e...

Important

You will not be able to retrieve the client secret in cleartext after you exit this screen.

6. Set the API permissions:
 - a. Select API permissions, click Microsoft Graph, then click Application permissions.

Request API permissions



Microsoft Graph

<https://graph.microsoft.com/> [Docs](#) [🔗](#)

What type of permissions does your application require?

Delegated permissions

Your application needs to access the API as the signed-in user.

Application permissions

Your application runs as a background service or daemon without a signed-in user.

b. From the User item, select the following permissions:

-
-
-
-

c. From the Group item, select the following permissions:

-
-
-

d. From the Directory item, select the following permissions:

-
-

e. Click Add permissions.

7. Grant admin consent for the API permissions:

On the Configured permissions page, Grant admin consent for *org-name*, then click Yes.

Configured permissions

Applications are authorized to call APIs when they are granted permissions by users/admins as part of the consent process. The list of configured permissions should include all the permissions the application needs. [Learn more about permissions and consent](#)

+ Add a permission ✔ Grant admin consent for ForgeRock

API / Permissions name	Type	Description	Admin consent req...	Status
▼ Microsoft Graph (10)				...
Directory.Read.All	Application	Read directory data	Yes	⚠ Not granted for ForgeR... ...
Directory.ReadWrite.All	Application	Read and write directory data	Yes	⚠ Not granted for ForgeR... ...
Group.Create	Application	Create groups	Yes	⚠ Not granted for ForgeR... ...
Group.Read.All	Application	Read all groups	Yes	⚠ Not granted for ForgeR... ...
Group.ReadWrite.All	Application	Read and write all groups	Yes	⚠ Not granted for ForgeR... ...
User.Export.All	Application	Export user's data	Yes	⚠ Not granted for ForgeR... ...
User.ManageIdentities	Application	Manage all users' identities	Yes	⚠ Not granted for ForgeR... ...
User.Read	Delegated	Sign in and read user profile	-	...
User.Read.All	Application	Read all users' full profiles	Yes	⚠ Not granted for ForgeR... ...
User.ReadWrite.All	Application	Read and write all users' full profiles	Yes	⚠ Not granted for ForgeR... ...

Install and Configure the MS Graph API Connector

1. Download the MS Graph API connector .jar file from the [ForgeRock Download Center](#), and move it to the `openidm/connectors` directory:

```
mv ~/Downloads/msgraphapi-connector-1.5.20.11.jar /path/to/openidm/connectors
```

2. Create a configuration for the connector.

Currently, you cannot configure the MS Graph API connector through the UI. Configure the connector over REST, as described in ["Configure Connectors Over REST"](#).

Alternatively, copy this sample `connector configuration file` to your project's `conf` directory.

Set at least the Azure `tenant`, `clientId` and `clientSecret` in the `configurationProperties`. For example:

```
{
  "configurationProperties" : {
    "tenant" : "your tenant ID",
    "clientId" : "your client ID",
    "clientSecret" : "your client secret"
  }
}
```

Test the MS Graph API Connector

Start IDM, if it is not running. Then use these examples to test that the connector is configured correctly and operating as expected:

+ *Check the Connector Configuration*

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system?_action=test"
[
  {
    "name": "msgraph",
    "enabled": true,
    "config": "config/provisioner.openicf/msgraph",
    "connectorRef": {
      "bundleVersion": "[1.5.0.0,1.6.0.0]",
      "bundleName": "org.forgerock.openicf.connectors.msgraphapi-connector",
      "connectorName": "org.forgerock.openicf.connectors.msgraphapi.MSGraphAPIConnector"
    },
    "displayName": "MSGraphAPI Connector",
    "objectTypes": [
      "user",
      "ALL",
      "group"
    ],
    "ok": true
  }
]
```

A status of `"ok": true` indicates that the connector is configured correctly.

+ *List User Entries*

This command retrieves a list of users in your Azure tenant. You can also use any system-enabled filter, such as those described in "Construct Queries" in the *Object Modeling Guide*:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
"http://localhost:8080/openidm/system/msgraph/user?_queryId=query-all-ids"
{
  "result": [
    {
      "_id": "c48be8cc-5846-4059-95e8-a7acbf6aec31"
    },
    {
      "_id": "c7fe57e2-3159-45e1-b67a-435232fd88d9"
    },
    {
      "_id": "9e714b5c-345a-430c-93f5-d8c6f9a2f225"
    },
    ...
  ],
  ...
}
```

+ Return a User Entry

This command retrieves a specific user entry from your Azure tenant:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
"http://localhost:8080/openidm/system/msgraph/user/c48be8cc-5846-4059-95e8-a7acbf6aec31"
{
  "_id": "c48be8cc-5846-4059-95e8-a7acbf6aec31",
  "surname": "Jensen",
  "displayName": "Babs Jensen",
  "memberOf": [
    "036f288c-6f71-41ae-9d09-6a68c8ba315b"
  ],
  "mail": "babs.jensen@example.onmicrosoft.com",
  "onPremisesExtensionAttributes": {
    ...
  },
  "usageLocation": "FR",
  "userType": "Member",
  "identities": [
    {
      "signInType": "userPrincipalName",
      "issuerAssignedId": "00991235@example.onmicrosoft.com",
      "issuer": "example.onmicrosoft.com"
    }
  ],
  "businessPhones": [],
  "createdDateTime": "2020-11-20T11:09:15Z",
  "accountEnabled": true,
  "userPrincipalName": "00991235@example.onmicrosoft.com",
}
```

```
"proxyAddresses": [
  "smtp:00991235@example.onmicrosoft.com",
  "SMTP:babs.jensen@example.onmicrosoft.com"
],
"imAddresses": [],
"passwordPolicies": "None",
"mailNickname": "00991235",
"givenName": "Babs",
"__NAME__": "00991235@example.onmicrosoft.com"
}
```

+ Create Users or Groups

This command creates a new user in your Azure tenant:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
--header "content-type: application/json" \
--data '{
  "surname": "Carter",
  "displayName": "Steve Carter",
  "givenName": "Steve",
  "userType": "Member",
  "accountEnabled": true,
  "mailNickname": "00654321",
  "userPrincipalName": "00654321@forgedemo.onmicrosoft.com",
  "PASSWORD__": "MyPassw0rd"
}' \
"http://localhost:8080/openidm/system/msgraph/user?_action=create"
{
  "_id": "9fa6c765-0872-45f6-8714-1dcd1ed94859",
  "surname": "Carter",
  "displayName": "Steve Carter",
  "memberOf": [],
  "onPremisesExtensionAttributes": {
    "extensionAttribute14": null,
    ...
  },
  "userType": "Member",
  "identities": [
    {
      "signInType": "userPrincipalName",
      "issuerAssignedId": "00654321@example.onmicrosoft.com",
      "issuer": "example.onmicrosoft.com"
    }
  ],
  "businessPhones": [],
  "createdDateTime": "2020-12-18T13:23:58Z",
  "accountEnabled": true,
  "userPrincipalName": "00654321@example.onmicrosoft.com",
  "proxyAddresses": [],
  "imAddresses": [],
  "mailNickname": "00654321",
  "givenName": "Steve",
}
```

```
{
  "__NAME__": "00654321@example.onmicrosoft.com"
}
```

+ Update Entries

This command changes the password for the user created previously:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request PATCH \
--header "content-type: application/json" \
--data '[ {
  "operation": "replace",
  "field": "__PASSWORD__",
  "value": "MyNewPassword"
} ]' \
"http://localhost:8080/openidm/system/msgraph/user/9fa6c765-0872-45f6-8714-1dcd1ed94859"
{
  "_id": "9fa6c765-0872-45f6-8714-1dcd1ed94859",
  "surname": "Carter",
  "displayName": "Steve Carter",
  "memberOf": [],
  "onPremisesExtensionAttributes": {
    "extensionAttribute14": null,
    ...
  },
  "userType": "Member",
  "identities": [
    {
      "signInType": "userPrincipalName",
      "issuerAssignedId": "00654321@forgedemo.onmicrosoft.com",
      "issuer": "forgedemo.onmicrosoft.com"
    }
  ],
  "businessPhones": [],
  "createdDateTime": "2020-12-18T13:23:58Z",
  "accountEnabled": true,
  "userPrincipalName": "00654321@forgedemo.onmicrosoft.com",
  "proxyAddresses": [],
  "imAddresses": [],
  "mailNickname": "00654321",
  "givenName": "Steve",
  "__NAME__": "00654321@forgedemo.onmicrosoft.com"
}
```

+ Delete Users and Groups

This command deletes the user created previously:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
```

```
--header "Accept-API-Version: resource=1.0" \
--request DELETE \
"http://localhost:8080/openidm/msgraph/user/9fa6c765-0872-45f6-8714-1dcd1ed94859"
{
  "_id": "9fa6c765-0872-45f6-8714-1dcd1ed94859",
  "surname": "Carter",
  "displayName": "Steve Carter",
  "memberOf": [],
  "onPremisesExtensionAttributes": {
    "extensionAttribute14": null,
    ...
  },
  "userType": "Member",
  "identities": [
    {
      "signInType": "userPrincipalName",
      "issuerAssignedId": "00654321@forgedemo.onmicrosoft.com",
      "issuer": "forgedemo.onmicrosoft.com"
    }
  ],
  "businessPhones": [],
  "createdDateTime": "2020-12-18T13:23:58Z",
  "accountEnabled": true,
  "userPrincipalName": "00654321@forgedemo.onmicrosoft.com",
  "proxyAddresses": [],
  "imAddresses": [],
  "mailNickname": "00654321",
  "givenName": "Steve",
  "__NAME__": "00654321@forgedemo.onmicrosoft.com"
}
```

Synchronize Accounts Between IDM and Azure

To use the MS Graph API connector to synchronize accounts between IDM and Azure, set up a [mapping](#) between the two data stores.

You can use this sample [mapping](#) as a starting point.

OpenICF Interfaces Implemented by the MSGraphAPI Connector

The MSGraphAPI Connector implements the following OpenICF interfaces.

Authenticate

Provides simple authentication with two parameters, presumed to be a user name and password.

Create

Creates an object and its [uid](#).

Delete

Deletes an object, referenced by its [uid](#).

Schema

Describes the object types, operations, and options that the connector supports.

Script on Connector

Enables an application to run a script in the context of the connector. Any script that runs on the connector has the following characteristics:

- The script runs in the same execution environment as the connector and has access to all the classes to which the connector has access.
- The script has access to a **connector** variable that is equivalent to an initialized instance of the connector. At a minimum, the script can access the connector configuration.
- The script has access to any script-arguments passed in by the application.

Search

Searches the target resource for all objects that match the specified object class and filter.

Sync

Polls the target resource for synchronization events, that is, native changes to objects on the target resource.

Test

Tests the connector configuration. Testing a configuration checks all elements of the environment that are referred to by the configuration are available. For example, the connector might make a physical connection to a host that is specified in the configuration to verify that it exists and that the credentials that are specified in the configuration are valid.

This operation might need to connect to a resource, and, as such, might take some time. Do not invoke this operation too often, such as before every provisioning operation. The test operation is not intended to check that the connector is alive (that is, that its physical connection to the resource has not timed out).

You can invoke the test operation before a connector configuration has been validated.

Update

Updates (modifies or replaces) objects on a target resource.

MSGraphAPI Connector Configuration

The MSGraphAPI Connector has the following configurable properties.

Basic Configuration Properties

Property	Type	Default	Encrypted ^a	Required ^b
tenant	String	null		Yes
The Azure AD tenant name or id				
clientId	String	null		Yes
The clientId used by the connector during the OAuth flow				
clientSecret	GuardedString	null	Yes	No
The client secret used by the connector during the OAuth flow				
httpProxyHost	String	null		No
The Http proxy host				
httpProxyPort	Integer	null		No
The Http proxy port				
httpProxyUsername	String	null		No
The Http proxy user name				
httpProxyPassword	GuardedString	null	Yes	No
The Http proxy user password				
performHardDelete	boolean	false		No
If set to true, the Azure object will be deleted permanently on delete operation.				
readRateLimit	String	null		No
Define throttling for read operations either per seconds ("30/sec") or per minute ("100/min").				
writeRateLimit	String	null		No
Define throttling for write operations (create/update/delete) either per second ("30/sec") or per minute ("100/min").				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

PeopleSoft Connector

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

The PeopleSoft connector lets you manage and synchronize accounts between Oracle PeopleSoft and IDM managed user objects. A PeopleSoft administrator account is required for this connector to work.

Before you start

Before you configure the connector, log in to your PeopleSoft administrator account and note the following:

Host

The host address of the PeopleSoft instance.

Port

The port for the PeopleSoft instance.

UserID

The username to log into the PeopleSoft instance.

Password

The password to log into the PeopleSoft instance.

Domain Connect Password

The domain connection password for the PeopleSoft WebLogic application server.

Install the PeopleSoft connector

1. Download the connector .jar file from the [ForgeRock BackStage download site](#).
2. If you are running the connector locally, place it in the `/path/to/openidm/connectors` directory, for example:

```
mv ~/Downloads/peoplesoft-connector-1.5.20.12.jar /path/to/openidm/connectors/
```

3. If you are using a remote connector server (RCS), place it in the `/path/to/openicf/connectors` directory on the RCS.

4. Download the connector dependencies.

- **psjoe.jar** is a file unique to each installation of PeopleSoft. It is compiled and provided by your PeopleSoft administrator. If it is not provided to you, see [Generate **psjoe.jar**](#).
- **psft.jar** is created by the following commands:

```
set CLASSPATH=%JAVA_HOME%\lib\tools.jar;%CLASSPATH%
jar cvf psft.jar .\PeopleSoft\Generated\CompIntfc*.class
```

Generate **psjoe.jar**

Note

This procedure is only required if your PeopleSoft Administrator did not provide **psjoe.jar**.

1. Start PeopleSoft Application Designer, and open any Component Interface definition.
2. Select Build > PeopleSoft APIs. The Build PeopleSoft API Binding window displays.
3. Under the Java Classes group box, select Build, and specify a target directory.
4. To build the selected bindings, click OK. The app builds the selected bindings in the target directory. If the operation is successful, a Done message appears in the PeopleSoft Application Designer Build window.
5. Compile the generated APIs:

Windows

```
cd %PS_HOME%\class\PeopleSoft\Generated\CompIntfc
javac -classpath %PS_HOME%\class\psjoe.jar *.java
cd c:\pt8\class\PeopleSoft\Generated\PeopleSoft
javac -classpath %PS_HOME%\class\psjoe.jar *.java
```

Linux

```
cd $PS_HOME/class/PeopleSoft/Generated/CompIntfc
javac classpath $PS_HOME/class/psjoe.jar *.java
cd $PS_HOME/class/PeopleSoft/Generated/PeopleSoft
javac classpath $PS_HOME/class/psjoe.jar *.java
```

6. Copy **psjoe.jar** and generated jar into **/path/to/openicf/lib**.

Configure the PeopleSoft connector

Create a connector configuration using the Admin UI:

1. Select Configure > Connectors and click New Connector.
2. Enter a Connector Name.

3. Select PeopleSoft Connector - 1.5.20.12 as the Connector Type.
4. Provide the Base Connector Details.
5. Click Save.

When your connector is configured correctly, the connector displays as Active in the admin UI.

Alternatively, test that the configuration is correct by running the following command:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system/peoplesoft?_action=test"
{
  "name": "peoplesoft",
  "enabled": true,
  "config": "config/provisioner.openicf/peoplesoft",
  "connectorRef": {
    "bundleVersion": "${bundleVersion}",
    "bundleName": "org.forgerock.openicf.connectors.peoplesoft-connector",
    "connectorName": "org.forgerock.openicf.connectors.peoplesoft.PeopleSoftConnector"
  },
  "displayName": "PeopleSoft Connector",
  "objectTypes": [
    "_ACCOUNT_",
    "_ALL_"
  ],
  "ok": true
}
```

If the command returns **"ok": true**, your connector has been configured correctly, and can authenticate to the PeopleSoft server.

Use the PeopleSoft connector

The following PeopleSoft account attributes are supported by the PeopleSoft connector:

Attribute	Description
__NAME__	The name of the user. Required.
UserID	ID of the user. Required.
IDTypes	<p>The type of ID and ID value for the user. Required. This is an object, containing IDType and AttributeValue as sub-attributes. For example:</p> <pre>"IDTypes": [{ "IDType": "EMP", "AttributeValue": "0001" }]</pre>

Attribute	Description																						
	<p><i>Supported ID types</i></p> <table> <tr> <th>ID Type</th><th>Name</th></tr> <tr> <td>BID</td><td>Bidder</td></tr> <tr> <td>CNT</td><td>Customer Contact</td></tr> <tr> <td>CST</td><td>Customer</td></tr> <tr> <td>EJA</td><td>External Job Applicant</td></tr> <tr> <td>EMP</td><td>Employee</td></tr> <tr> <td>NON</td><td>None</td></tr> <tr> <td>ORG</td><td>Organization ID</td></tr> <tr> <td>PER</td><td>Person (CRM)</td></tr> <tr> <td>VND</td><td>Vendor</td></tr> <tr> <td>PTN</td><td>Partner</td></tr> </table>	ID Type	Name	BID	Bidder	CNT	Customer Contact	CST	Customer	EJA	External Job Applicant	EMP	Employee	NON	None	ORG	Organization ID	PER	Person (CRM)	VND	Vendor	PTN	Partner
ID Type	Name																						
BID	Bidder																						
CNT	Customer Contact																						
CST	Customer																						
EJA	External Job Applicant																						
EMP	Employee																						
NON	None																						
ORG	Organization ID																						
PER	Person (CRM)																						
VND	Vendor																						
PTN	Partner																						
UserIDAlias	Alias ID of the user. This should be a fully qualified email address.																						
UserDescription	A description of the user.																						
PrimaryPermissionList	Primary permission list for the user. Displays which permissions the user is granted in the primary permission list.																						
RowSecurityPermissionList	Row security permission list for the user. Displays which permissions the user is granted in the row security permission list.																						
ProcessProfilePermissionList	Process profile permission list for the user. Displays which permissions the user is granted in the process profile permission list.																						
NavigatorHomePermissionList	Navigator home permission list for the user. Displays which permissions the user is granted in the navigator home permission list.																						
SymbolicID	The symbolic ID of the user.																						
LanguageCode	<p>The user's language preference.</p> <p><i>Supported Languages</i></p> <table> <tr> <th>Language</th><th>Code</th></tr> <tr> <td>Arabic</td><td>ARA</td></tr> <tr> <td>Afrikaans</td><td>AFR</td></tr> <tr> <td>Bulgarian</td><td>BUL</td></tr> </table>	Language	Code	Arabic	ARA	Afrikaans	AFR	Bulgarian	BUL														
Language	Code																						
Arabic	ARA																						
Afrikaans	AFR																						
Bulgarian	BUL																						

Attribute	Description	
	Language	Code
	Simplified Chinese	ZHS
	Traditional Chinese	ZHT
	Croatian	CRO
	Czech	CZE
	Danish	DAN
	Dutch	DUT
	English	ENG
	UK English	UKE
	French	FRA
	Canadian French	CFR
	German	GER
	Greek	GRK
	Finnish	FIN
	Hebrew	HEB
	Hungarian	HUN
	Italian	ITA
	Japanese	JPN
	Korean	KOR
	Bahasa Malay	MAY
	Norwegian	NOR
	Polish	POL
	Portuguese	POR
	Romanian	ROM
	Russian	RUS
	Serbian	SER
	Slovak	SLK
	Slovenian	SLV
	Spanish	ESP
	Swedish	SVE
	Thai	THA
	Turkish	TUR
	Vietnamese	VIE

Attribute	Description								
	<p>Note</p> <p>The list of supported languages can vary depending on your Oracle PeopleSoft version.</p>								
MultiLanguageEnabled	Enable support for multiple languages for the user.								
AccountLocked	Whether the user account is locked.								
CurrencyCode	Three letter code for the user's preferred currency.								
FailedLogins	The number of failed logins for the user.								
ExpertEntry	Whether the user is marked as an expert.								
Opertype	The type of operation.								
AllowSwitchUser	Determines whether the user has access to user switching.								
WorklistEntriesCount	Number of worklist entries associated with the user.								
WorklistUser	Whether there is a worklist associated with the user. Must be either Y (Yes) or N (No).								
EmailUser	Email preference of the user. Must be either Y (Yes) or N (No).								
AlternateUserID	Fallback user to route to if the user is unavailable. This must be filled out if you specify EffectiveDateFrom or EffectiveDateTo .								
EffectiveDateFrom	Effective start date that a user will be unavailable. Must be in MM/DD/YYYY format.								
EffectiveDateTo	Effective end date, marking when a user will become available again. Must be in MM/DD/YYYY format.								
EmailAddresses	<p>List of email addresses associated with the user. This is an object, with EmailType, EmailAddress, and PrimaryEmail as sub-attributes. For example:</p> <pre>"EmailAddresses": [{ "EmailType": "BUS", "EmailAddress": "test@example.com", "PrimaryEmail": "Y" }]</pre> <p><i>Supported email types</i></p> <table> <tr> <th>Email Code</th><th>Email Type</th></tr> <tr> <td>BB</td><td>Blackberry</td></tr> <tr> <td>HOME</td><td>Home</td></tr> <tr> <td>WORK</td><td>Work</td></tr> </table>	Email Code	Email Type	BB	Blackberry	HOME	Home	WORK	Work
Email Code	Email Type								
BB	Blackberry								
HOME	Home								
WORK	Work								

Attribute	Description	
	Email Code	Email Type
	BUS	Business
	OTH	Other
	EMPTY	Empty field
Roles	List of roles the user has. Users inherit permissions based on the roles the user has. This is an object, with RoleName and Dynamic as sub-attributes. For example: <pre>"Roles": [{ "RoleName": "PeopleSoft User" }]</pre>	
__PASSWORD__	The password for the user.	
ConfirmPassword	Used to confirm the password of the user. This needs to match the user's password.	
Encrypted	Status showing whether the user profile is encrypted.	

Operations on PeopleSoft accounts

You can use the PeopleSoft connector to perform the following actions on a PeopleSoft account:

+ *Create a PeopleSoft user*

The following example creates a user with the minimum required attributes:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
--data '{
  "__NAME__": "Barbara Jensen",
  "UserID": "BJENSEN",
  "IDTypes": [{
    "IDType": "EMP",
    "AttributeValue": "0001"
  }]
}' \
"http://localhost:8080/openidm/system/peoplesoft/__ACCOUNT__?_action=create"
{
  "_id": "BJENSEN",
  "ExpertEntry": 0,
  "LanguageCode": "ENG",
  "EmailUser": "Y",
  "__ENABLE__": 0,
  "__NAME__": "Barbara Jensen",
  "IDTypes": [
    {
      "IDType": "EMP",
```

```

    "AttributeValue": "0001"
  }
],
"Encrypted": 1,
"UserID": "BJENSEN",
"Opertype": 0,
"MultiLanguageEnabled": 0,
"WorklistUser": "Y",
"WorklistEntriesCount": 0,
"AllowSwitchUser": 0,
"FailedLogins": 0
}

```

Note

When you create a new user, you must specify at least `__NAME__`, `UserID`, and `IDTypes`. See the list of available attributes for more information.

+ *Modify a PeopleSoft user entry*

You can modify an existing user with a PUT request, including all attributes of the account in the request. The following attributes can be modified on a user:

- `UserIDAlias`
- `UserDescription`
- `PrimaryPermissionList`
- `RowSecurityPermissionList`
- `ProcessProfilePermissionList`
- `NavigatorHomePermissionList`
- `SymbolicID`
- `LanguageCode`
- `MultiLanguageEnabled`
- `AccountLocked`
- `CurrencyCode`
- `FailedLogins`
- `ExpertEntry`
- `Opertype`

- AllowSwitchUser
- WorklistUser
- EmailUser
- AlternateUserID
- EffectiveDateFrom
- EffectiveDateTo
- EmailAddresses
- Roles
- IDTypes
- Password
- ConfirmPassword
- Encrypted

For example, to add an email address to a user:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--header "if-Match:*" \
--request PUT \
--data '{
  "__NAME__": "Barbara Jensen",
  "UserID": "BJENSEN",
  "IDTypes": [{
    "IDType": "EMP",
    "AttributeValue": "0001"
  }],
  "EmailAddresses": [{
    "EmailType": "BUS",
    "EmailAddress": "test@example.com",
    "PrimaryEmail": "Y"
  }]
}' \
"http://localhost:8080/openidm/system/peoplesoft/__ACCOUNT__/BJENSEN"
{
  "id": "BJENSEN",
  "ExpertEntry": 0,
  "LanguageCode": "ENG",
  "EmailUser": "Y",
  "__ENABLE__": 0,
  "__NAME__": "Barbara Jensen",
  "IDTypes": [
    {
      "IDType": "EMP",
```

```

    "AttributeValue": "0001"
  },
  ],
  "Encrypted": 1,
  "EmailAddresses": [
    {
      "EmailType": "BUS",
      "EmailAddress": "test@example.com",
      "PrimaryEmail": "Y"
    }
  ],
  "UserID": "BJENSEN",
  "Opertype": 0,
  "MultiLanguageEnabled": 0,
  "WorklistUser": "Y",
  "WorklistEntriesCount": 0,
  "AllowSwitchUser": 0,
  "FailedLogins": 0
}

```

+ Query PeopleSoft user entries

The following example queries all PeopleSoft users:

```

curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/peoplesoft/__ACCOUNT__?_queryId=query-all-ids"
{
  "result": [
    {
      "_id": "AZIGLAR"
    },
    {
      "_id": "BCHALMERS"
    },
    {
      "_id": "BDAVIS"
    },
    {
      "_id": "BFRANCISCO"
    },
    {
      "_id": "BGONZALES"
    },
    {
      "_id": "BJENSEN"
    },
    {
      "_id": "BLOCHERTY"
    },
    [ ... ]
    {
      "_id": "SUNDERWOOD"
    }
  ]
}

```

```

    },
    {
      "_id": "SVANDERSTEEN"
    },
    {
      "_id": "SWALTERS"
    },
    {
      "_id": "TCORY"
    },
    {
      "_id": "TELLIS"
    }
  ],
  "resultCount": 300,
  "pagedResultsCookie": null,
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}

```

The following command queries a specific user by their ID:

```

curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/peoplesoft/___ACCOUNT___/BJENSEN"
{
  "_id": "BJENSEN",
  "ExpertEntry": 0,
  "LanguageCode": "ENG",
  "EmailUser": "Y",
  "___ENABLE___": 0,
  "___NAME___": "Barbara Jensen",
  "IDTypes": [
    {
      "IDType": "EMP",
      "AttributeValue": "0001"
    }
  ],
  "Encrypted": 1,
  "EmailAddresses": [
    {
      "EmailType": "BUS",
      "EmailAddress": "test@example.com",
      "PrimaryEmail": "Y"
    }
  ],
  "UserID": "BJENSEN",
  "Opertype": 0,
  "MultiLanguageEnabled": 0,
  "WorklistUser": "Y",
  "WorklistEntriesCount": 0,
  "AllowSwitchUser": 0,
  "FailedLogins": 0
}

```

```
}

```

+ *Reset a PeopleSoft user account password*

To reset the password for PeopleSoft user account, you can use the connector to change a user's password.

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--header "if-Match:*" \
--request PUT \
--data '{
  "_PASSWORD_": "Passw0rd",
  "_CURRENT_PASSWORD_": "Passw0rd"
}' \
"http://localhost:8080/openidm/system/peoplesoft/__ACCOUNT__/BJENSEN"
{
  "_id": "BJENSEN",
  "ExpertEntry": 0,
  "LanguageCode": "ENG",
  "EmailUser": "Y",
  "_ENABLE_": 0,
  "_NAME_": "Barbara Jensen",
  "IDTypes": [
    {
      "IDType": "EMP",
      "AttributeValue": "0001"
    }
  ],
  "Encrypted": 1,
  "EmailAddresses": [
    {
      "EmailType": "BUS",
      "EmailAddress": "test@example.com",
      "PrimaryEmail": "Y"
    }
  ],
  "UserID": "BJENSEN",
  "Opertype": 0,
  "MultiLanguageEnabled": 0,
  "WorklistUser": "Y",
  "WorklistEntriesCount": 0,
  "AllowSwitchUser": 0,
  "FailedLogins": 0
}
```

Note

While the `PASSWORD` field is not returned as part of the response, the user object is updated.

+ Enable a PeopleSoft user

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--header "if-Match:*" \
--request PUT \
--data '{
  "__NAME__": "Barbara Jensen",
  "__ENABLE__": 1
}' \
"http://localhost:8080/openidm/system/peoplesoft/__ACCOUNT__/BJENSEN"
{
  "id": "BJENSEN",
  "ExpertEntry": 0,
  "LanguageCode": "ENG",
  "EmailUser": "N",
  "__ENABLE__": 1,
  "__NAME__": "Barbara Jensen",
  "IDTypes": [
    {
      "IDType": "EMP",
      "AttributeValue": "0001"
    }
  ],
  "Encrypted": 1,
  "EmailAddresses": [
    {
      "EmailType": "BUS",
      "EmailAddress": "test@example.com",
      "PrimaryEmail": "Y"
    }
  ],
  "UserID": "BJENSEN",
  "Opertype": 0,
  "MultiLanguageEnabled": 0,
  "WorklistUser": "N",
  "WorklistEntriesCount": 0,
  "AllowSwitchUser": 0,
  "FailedLogins": 0
}
```

+ Disable a PeopleSoft user

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
```

```
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--header "if-Match:*" \
--request PUT \
--data '{
  "__NAME__": "Barbara Jensen",
  "__ENABLE__": 0
}' \
"http://localhost:8080/openidm/system/peoplesoft/__ACCOUNT__/BJENSEN"
{
  "id": "BJENSEN",
  "ExpertEntry": 0,
  "LanguageCode": "ENG",
  "EmailUser": "N",
  "__ENABLE__": 0,
  "__NAME__": "Barbara Jensen",
  "IDTypes": [
    {
      "IDType": "EMP",
      "AttributeValue": "0001"
    }
  ],
  "Encrypted": 1,
  "EmailAddresses": [
    {
      "EmailType": "BUS",
      "EmailAddress": "test@example.com",
      "PrimaryEmail": "Y"
    }
  ],
  "UserID": "BJENSEN",
  "Opertype": 0,
  "MultiLanguageEnabled": 0,
  "WorklistUser": "N",
  "WorklistEntriesCount": 0,
  "AllowSwitchUser": 0,
  "FailedLogins": 0
}
```

+ Delete a PeopleSoft user

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request DELETE \
"http://localhost:8080/openidm/system/peoplesoft/__ACCOUNT__/Barbara_Jensen"
{
  "id": "BJENSEN",
  "ExpertEntry": 0,
  "LanguageCode": "ENG",
  "EmailUser": "N",
  "__ENABLE__": 0,
  "__NAME__": "Barbara Jensen",
  "IDTypes": [
    {

```

```

    "IDType": "EMP",
    "AttributeValue": "0001"
  }
],
"Encrypted": 1,
"EmailAddresses": [
  {
    "EmailType": "BUS",
    "EmailAddress": "test@example.com",
    "PrimaryEmail": "Y"
  }
],
"UserID": "BJENSEN",
"Opertype": 0,
"MultiLanguageEnabled": 0,
"WorklistUser": "N",
"WorklistEntriesCount": 0,
"AllowSwitchUser": 0,
"FailedLogins": 0
}

```

Operations on other objects

The following operations are supported for other objects; including Employee, Permission, External Job Applicant, and Role:

+ *Query all employees*

The following example queries all employees' details:

```

curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/peoplesoft/__EMPLOYEE__?_queryId=query-all-ids"
{
  "result": [
    { "_id": "0001" },
    { "_id": "21" },
    { "_id": "22" },
    { "_id": "25" },
    { "_id": "AA0001" }
  ],
  "resultCount": 5,
  "pagedResultsCookie": null,
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}

```

+ *Query a single employee*

The following example queries a single employee's details:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/peoplesoft/__EMPLOYEE__/BJENSEN"
{
  "id" : "BJENSEN",
  "NAME" : "BJENSEN",
  "LAST_NAME" : "Jensen",
  "PROP_DERIVED_EMP" : "N",
  "COLL_NAME_TYPE_VW" : [ {
    "KEYPROP_NAME_TYPE" : "PRI",
    "FIRST_NAME" : "Barbara",
    "LAST_NAME" : "Jensen"
  }, {
    "KEYPROP_NAME_TYPE" : "PRF",
    "FIRST_NAME" : "Barbara",
    "LAST_NAME" : "Jensen"
  } ],
  "PROP_NAME" : "Barbara Jensen",
  "UID" : "BJENSEN",
  "COLL_ADDRESS_TYPE_VW" : [ {
    "KEYPROP_ADDRESS_TYPE" : "",
    "KEYPROP_EFFDT" : "11/14/2022",
    "PROP_EFF_STATUS" : "A",
    "PROP_COUNTRY" : "",
    "PROP_ADDRESS1" : "",
    "PROP_ADDRESS2" : "",
    "PROP_ADDRESS3" : "",
    "PROP_ADDRESS4" : "",
    "PROP_CITY" : "",
    "PROP_NUM1" : "",
    "PROP_NUM2" : "",
    "PROP_HOUSE_TYPE" : "",
    "PROP_ADDR_FIELD1" : "",
    "PROP_ADDR_FIELD2" : "",
    "PROP_ADDR_FIELD3" : "",
    "PROP_COUNTY" : "",
    "PROP_STATE" : "",
    "PROP_POSTAL" : "",
    "PROP_GEO_CODE" : "",
    "PROP_IN_CITY_LIMIT" : "",
    "PROP_ADDRESS1_AC" : "",
    "PROP_ADDRESS2_AC" : "",
    "PROP_ADDRESS3_AC" : "",
    "PROP_CITY_AC" : "",
    "PROP_REG_REGION" : ""
  } ],
  "COLL_PERSONAL_PHONE" : [ {
    "KEYPROP_PHONE_TYPE" : "",
    "PROP_COUNTRY_CODE" : "",
    "PROP_PHONE" : "",
    "PROP_EXTENSION" : "",
    "PROP_PREF_PHONE_FLAG" : "N"
  } ]
}
```

```

    } ],
    "COLL_EMAIL_ADDRESSES" : [ {
      "KEYPROP_E_ADDR_TYPE" : "",
      "PROP_EMAIL_ADDR" : "",
      "PROP_PREF_EMAIL_FLAG" : "N"
    } ]
  } ]
}

```

+ Query all permissions

The following example queries all employee permissions:

```

curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/peoplesoft/__PERMISSION__?_queryId=query-all-ids"
{
  "result": [
    { "_id": "11" },
    { "_id": "CI_PERSONAL_DATA" },
    { "_id": "CRM8000" },
    { "_id": "CRRW1000" },
    { "_id": "EOCB_CLIENT_USER" }
  ],
  "resultCount": 5,
  "pagedResultsCookie": null,
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}

```

+ Query a single permission

The following example queries a single permission's details:

```

curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/peoplesoft/__PERMISSION__/HCCPCSALL"
{
  "_id" : "HCCPCSALL",
  "_UID_" : "HCCPCSALL",
  "_NAME_" : "Campus - Hidden Objects",
  "KEYPROP_CLASSID" : "HCCPCSALL"
}

```

+ Query all external job applicants

The following example queries all external job applicants:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/peoplesoft/__EXTERNAL_JOB_APPLICANT__?_queryId=query-all-ids"
{
  "result": [
    {"_id": "500000"},
    {"_id": "500001"},
    {"_id": "500002"},
    {"_id": "500003"},
    {"_id": "500004"}
  ],
  "resultCount": 5,
  "pagedResultsCookie": null,
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

+ Query a single external job applicant

The following example queries a single external job applicant's details:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/peoplesoft/__EXTERNAL_JOB_APPLICANT__/500258"
{
  "_id" : "500258",
  "__NAME__" : "500258",
  "__UID__" : "500258"
}
```

+ Query all roles

The following example queries all employee roles:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/peoplesoft/__ROLE__?_queryId=query-all-ids"
{
  "result": [
    { "_id": "ACM Administrator"},
    { "_id": "ADS Designer"},
    { "_id": "AG Composer Administrator"},
    { "_id": "AG Composer User"},
    { "_id": "AM Administrator"}
  ],
  "resultCount": 5,
  "pagedResultsCookie": null,
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

+ Query a single role

The following example queries a single role's details:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/peoplesoft/__ROLE__/HR%20Matrix%20Manager"
{
  "_id" : "HR Matrix Manager",
  "PSROLEGRANTORVW" : [ {
    "GRANTROLENAME" : "",
    "ROLENAME" : "HR Matrix Manager"
  } ],
  "PC_FUNCTION_NAME" : "HR Matrix Manager",
  " _UID_ " : "HR Matrix Manager",
  "DESCRLONG" : "HR Matrix Manager",
  "ALLOWNOTIFY" : "HR Matrix Manager",
  "ROLE_PCODE_RULE_ON" : "HR Matrix Manager",
  " _NAME_ " : "HR Matrix Manager",
  "PSROLECANGRANT" : [ {
    "GRANTROLENAME" : "",
    "ROLENAME" : "HR Matrix Manager"
  } ],
  "DESCR" : "HR Matrix Manager",
  "QRYNAME" : "HR Matrix Manager",
  "ROLE_QUERY_RULE_ON" : "HR Matrix Manager",
  "RECNAME" : "HR Matrix Manager",
  "FIELDNAME" : "HR Matrix Manager",
  "PSROLEMEMBER" : [ {
    "ROLEUSER" : "",
    "ROLENAME" : "HR Matrix Manager"
  } ],
}
```

```
"PSROLEDYNMEMBER" : [ {
  "ROLEUSER" : "",
  "ROLENAME" : "HR Matrix Manager"
} ],
"ALLOWLOOKUP" : "HR Matrix Manager",
"PSROLECLASS" : [ {
  "CLASSID" : "HCCPHR9435"
} ],
"LDAP_RULE_ON" : "HR Matrix Manager"
}
```

OpenICF Interfaces Implemented by the PeopleSoft Connector

The PeopleSoft Connector implements the following OpenICF interfaces.

Create

Creates an object and its `uid`.

Delete

Deletes an object, referenced by its `uid`.

Schema

Describes the object types, operations, and options that the connector supports.

Script on Connector

Enables an application to run a script in the context of the connector. Any script that runs on the connector has the following characteristics:

- The script runs in the same execution environment as the connector and has access to all the classes to which the connector has access.
- The script has access to a `connector` variable that is equivalent to an initialized instance of the connector. At a minimum, the script can access the connector configuration.
- The script has access to any script-arguments passed in by the application.

Search

Searches the target resource for all objects that match the specified object class and filter.

Test

Tests the connector configuration. Testing a configuration checks all elements of the environment that are referred to by the configuration are available. For example, the connector might make a physical connection to a host that is specified in the configuration to verify that it exists and that the credentials that are specified in the configuration are valid.

This operation might need to connect to a resource, and, as such, might take some time. Do not invoke this operation too often, such as before every provisioning operation. The test operation

is not intended to check that the connector is alive (that is, that its physical connection to the resource has not timed out).

You can invoke the test operation before a connector configuration has been validated.

Update

Updates (modifies or replaces) objects on a target resource.

PeopleSoft Connector Configuration

The PeopleSoft Connector has the following configurable properties.

Configuration properties

Property	Type	Default	Encrypted ^a	Required ^b
host	String	null		Yes
Host name or IP address to connect to PeopleSoft server				
port	int	0		Yes
Port to connect to PeopleSoft server				
userId	String	null		Yes
The userid used to login to PeopleSoft server				
password	GuardedString	null	Yes	Yes
The password used to login to PeopleSoft server				
domainConnectPassword	GuardedString	null	Yes	Yes
The password for PeopleSoft app server domain				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

PowerShell Connector Toolkit

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

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, including, but not limited to, Active Directory, Microsoft SQL, MS Exchange, SharePoint, Azure, and Office365. Essentially, any task that can be performed with PowerShell can be executed through connectors based on this toolkit.

The PowerShell Connector Toolkit is available from the [ForgeRock BackStage download site](#).

IDM includes sample scripts for synchronization of users between Windows Active Directory and IDM. These sample scripts can help you get started with the PowerShell Connector toolkit. For more information, see "[Connect to Active Directory With the PowerShell Connector](#)" in the *Samples Guide*.

Before You Start

To implement a scripted PowerShell connector, you must install the following:

- Microsoft .NET Framework 4.5 or later. Connectors created with the PowerShell Connector Toolkit run on the .NET platform and require the installation of a .NET connector server on the Windows system. To install the .NET connector server, follow the instructions in "[Set Up a .NET Connector Server](#)".
- PowerShell version 4.0 or above.
- The PowerShell Connector Toolkit.

Setting Up the PowerShell Connector

To run the commands in this procedure, start with the PowerShell command line. Some of the commands in this procedure require administrative privileges.

1. [Install, configure, and start the .NET connector server on a Windows host](#). If you are running an Active Directory Domain Controller, install the .NET connector server on the same host on which the Windows PowerShell module is installed.
2. [Configure IDM to connect to the .NET connector server](#).
3. Download the PowerShell Connector Toolkit archive ([mspowershell-connector-1.4.7.0.zip](#)) from the [ForgeRock BackStage download site](#).

Extract the archive and move the `MsPowerShell.Connector.dll` to the folder in which the connector server application executable file (`ConnectorServerService.exe`) is located.

4. The `openidm\samples\scripted-powershell-with-ad` directory contains sample scripts for a connection to Active Directory. Copy these scripts to the host on which the .NET connector server is installed.

The full path to the scripts must be referenced in your connector configuration file (`provisioner.openicf-*.json`), for example:

```
"CreateScriptFileName" : "C:/openidm/samples/scripted-powershell-with-ad/tools/ADCreate.ps1",  
...
```

- Copy the sample connector configuration file (`provisioner.openicf-adpowershell.json`) from the `samples\example-configurations\provisioners` directory to your project's `conf` directory.

Verify that at least the path to the scripts and the connection and authentication details are correct for your deployment. The following section describes the configurable properties in the sample connector configuration files.

Note

Paths in these files must use forward slash characters and not the backslash characters that you would expect in a Windows path.

Configuring the PowerShell Connector

Your PowerShell connector configuration file should include the following properties:

Property	Type	Example	Encrypted ^a	Required ^b
<code>operationScriptFileName</code>	String	<code>C:/openidm/AD/ADCreate.ps1</code> ,	No	Yes
The full path to the script that implements the corresponding OpenICF operation.				
<code>VariablesPrefix</code>	String	<code>Connector</code>	No	No
To avoid variable namespace conflicts, you can define a prefix for the connector variables. All variables are injected into the script under that prefix and can be used with the dotted notation.				
<code>QueryFilterType</code>	String	<code>AdPsModule</code> (for Active Directory)	No	Yes
A configurable query filter visitor property that defines the format in which the query will be injected into the connector. Possible values are:				
<ul style="list-style-type: none"> <code>Map</code> - the query filter is a map <code>Ldap</code> - the query filter is in LDAP search format, for example, <code>"(cn=Joe)"</code> <code>Native</code> - the query filter is a native OpenICF query filter <code>AdPsModule</code> - the query filter is compatible with the Active Directory PowerShell module, <code>Get-ADUser Filter</code> 				
<code>ReloadScriptOnExecution</code>	Boolean	<code>true</code>	No	No
When <code>true</code> , the connector reloads the script from disk every time it is executed. This can be useful for debugging purposes. Set to <code>false</code> in production.				
<code>UseInterpretersPool</code>	Boolean	<code>true</code>	No	No
If <code>true</code> , the connector leverages the PowerShell RunSpace Pool.				
<code>MaxInterpretersPoolSize</code>	Integer	<code>5</code>	No	No
The maximum size of the interpreter pool.				

Property	Type	Example	Encrypted ^a	Required ^b
MinInterpretersPoolSize	Integer	1	No	No
The minimum size of the interpreter pool.				
PoolCleanupInterval	Double	60	No	No
Specifies the interval (in minutes) at which unused interpreter instances are discarded. To avoid cleaning up unused interpreter instances, set this property to 0.				
SubstituteUidAndNameInQueryFilter	Boolean	true	No	No
Specifies whether the <code>__UID__</code> and <code>__NAME__</code> should be replaced by the value defined in the <code>NameAttributeName</code> and <code>UidAttributeName</code> in the query filter.				
UidAttributeName	String	ObjectGUID	No	No
The attribute on the resource that contains the object <code>__UID__</code>				
NameAttributeName	String	DistinguishedName	No	No
The attribute on the resource that contains the object <code>__NAME__</code>				
PsModulesToImport	Array	["ActiveDirectory", "C:/openidm/samples/scripted-powershell-with-ad/tools/ADSIsearch.psm1"]	No	No
An array of additional PowerShell modules that the connector must import				
Host	String	ad.example.com	No	Yes
The host name or IP address of the Active Directory server				
Port	Integer	null	No	Yes
The port number on which the remote resource listens for connections				
Login	String	""	No	Yes
The user account in the remote resource that is used for the connection				
Password	String	null	Encrypted	Yes
The password of the user account that is used for the connection				
CustomProperties	Array	[]	No	No
An array of Strings to define custom configuration properties. Each property takes the format <code>"name=value"</code> . For example:				
<pre> "configurationProperties" : { ... "CustomProperties" : ["baseContext = CN=Users,DC=example,DC=com"], ... } </pre>				

Property	Type	Example	Encrypted ^a	Required ^b
The custom property can then be read from the PowerShell scripts as follows: <code>\$base = \$Connector.Configuration.PropertyBag.baseContext</code>				

^a Indicates whether the property value is considered confidential, and therefore encrypted in IDM.

^b A list of operations in this column indicates that the property is required for those operations.

Testing the PowerShell Connector

Start IDM with the configuration for your PowerShell connector project.

The following tests assume that the configuration is in the default `path/to/openidm` directory. If your PowerShell project is in a different directory, use the `startup` command with the `-p` option to point to that directory.

```
/path/to/openidm/startup.sh
```

Confirming the Connector Configuration

To test that the PowerShell connector has been configured correctly, run the following REST call:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system?_action=test"
{
  "name" : "adpowershell",
  "enabled" : true,
  "config" : "config/provisioner.openicf/adpowershell",
  "objectTypes" : [ "__ALL__", "group", "account" ],
  "connectorRef" : {
    "connectorName" : "Org.ForgeRock.OpenICF.Connectors.MsPowerShell.MsPowerShellConnector",
    "bundleName" : "MsPowerShell.Connector",
    "bundleVersion" : "[1.4.3.0,1.5.0.0]"
  },
  "displayName" : "PowerShell Connector",
  "ok" : true
}
```

When you run this test, you should also see a log entry associated with the .NET connector server, in the `logs/` directory of that server.

Searching With the Connector

You can use the connector, with a PowerShell search script, to retrieve information from a target system. The PowerShell search script accepts IDM queries, including `query-all-ids` and `_queryFilter`.

With the following command, you can retrieve a list of existing users in an Active Directory server. You can also use any system-enabled filter, such as those described in "Presence Expressions" in the *Object Modeling Guide*:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
"http://localhost:8080/openidm/system/adpowershell/account?_queryId=query-all-ids"
```

Creating With the Connector

You can use the connector to create new users or groups on the target system, based on options listed in the relevant `provisioner.openicf-*` configuration file.

For example, the following command creates a new user in Active Directory:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
--header "content-type: application/json" \
--data '{
  "distinguishedName" : "CN=Robert Smith,CN=Users,DC=EXAMPLE,DC=COM",
  "sAMAccountName" : "robert.smith",
  "sn" : "Smith",
  "cn" : "Robert Smith",
  "userPrincipalName": "Robert.Smith@example.com",
  "enabled" : true,
  "password" : "Passw0rd",
  "telephoneNumber" : "0052-611-091"
}' \
"http://localhost:8080/openidm/system/adpowershell/account?_action=create"
```

Updating With the Connector

The PowerShell scripts associated with update functionality support changes to the following properties:

- Password
- Principal Name
- License
- Common user attributes

As an example, you could use the following command to change the password for the user with the noted `_id`:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request PATCH \
--header "content-type: application/json" \
--data '{
  "operation": "replace",
  "Field": "password",
  "value": "Passw1rd"
}' \
"http://localhost:8080/openidm/system/adpowershell/account/1d4c9276-6937-4d9e-9c60-67e8b4207f4e"
```

Deleting With the Connector

You can use the PowerShell connector to delete user and group objects. The following command deletes a user in Active Directory, based on their id:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request DELETE \
"http://localhost:8080/openidm/system/adpowershell/account/1d4c9276-6937-4d9e-9c60-67e8b4207f4e"
```

Running a Script on the Connector

The `runScriptOnConnector` script enables you to run an arbitrary script action through the connector. This script takes the following variables as input:

Configuration

A handler to the connector's configuration object.

Options

A handler to the Operation Options.

Operation

The operation type that corresponds to the action (`RUNSCRIPTONCONNECTOR` in this case).

Arguments

A map of script arguments (this can be `null`).

The script can return any object that can be serialized by OpenICF, such as `Boolean`, `String`, `Array`, or `Dictionary`. If the object type cannot be serialized, such as `Hashtable`, the script fails with the error:

```
"error": "No serializer for class: System.Collections.Hashtable"
```

To run an arbitrary script on the PowerShell connector, define the script in the `systemActions` property of your provisioner file:

```
"systemActions" : [
  {
    "scriptId" : "MyScript",
    "actions" : [
      {
        "systemType" : ".*PowerShellConnector",
        "actionType" : "PowerShell",
        "actionFile" : "scripts/Myactionscript.ps1"
      }
    ]
  }
]
```

When you have defined the script, you can call it over REST on the system endpoint, as follows:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system/adpowershell?
_action=script&scriptId=MyScript&param1=value1&param2=value2"
```

You can also call it through the IDM script engine, as follows:

```
openidm.action("/system/adpowershell", "script", {}, {"scriptId": "MyScript", "param1": "value1",
"param2": "value2"})
```

Important

Because the action script is stored locally with IDM, it must be transmitted across the network every time it is called. An alternative approach is to write a PowerShell module and to load it using the `PsModulesToImport` option of the PowerShell connector. In this case, the action script is limited to a function call and you do not need a script file on the IDM side.

The following example uses the `actionSource` property in the provisioner, instead of the `actionFile` property, to call the action. The example calls a custom `Set-Exchange` function from a module loaded on the .Net connector server by the PowerShell connector:

```
"systemActions" : [
  {
    "scriptId" : "SetExchange",
    "actions" : [
      {
        "systemType" : ".*PowerShellConnector",
        "actionType" : "PowerShell",
        "actionSource" : "Set-Exchange $Connector.Arguments.dn"
      }
    ]
  }
]
```

Manage Azure AD Objects With the PowerShell Connector

ForgeRock provides two sets of sample scripts to let you manage objects in Azure AD with the PowerShell connector:

- **Version 1:** These scripts are based on the older Microsoft Online (MSOL) V1 PowerShell module. For information on connecting to your Azure AD with this module, see the corresponding [Microsoft documentation](#). Microsoft has expressed its intention to deprecate this module when its functionality has been completely migrated to the newer Azure Active Directory PowerShell for Graph Module. These scripts are supported only up to Windows 2012 R2.

The Version 1 scripts can manage security groups but not dynamic groups.

- **Version 2:** These scripts are based on the [Azure Active Directory PowerShell for Graph Module](#). For information on connecting to your Azure AD with this module, see the corresponding [Microsoft documentation](#). The cmdlets in this module let you perform CRUD operations on an Azure AD instance, and configure the directory and its features.

The Version 2 scripts can manage user password policies, security and mail groups, dynamic groups, and devices.

Follow these procedures to use the sample Azure AD scripts with the PowerShell connector:

Set Up Your Systems

1. Set up IDM.

These steps assume that IDM 7 is running locally on a UNIX/Linux system.

2. Install a .NET connector server on your windows host. These steps assume a Windows hostname of `windows-host.example.com`.
3. On `windows-host.example.com`, install the PowerShell connector.

When you have installed the PowerShell connector, make sure that the ICF .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 files. 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:

This file came from another computer and might be blocked to help protect this computer.

Click the Unblock button to unblock the connector .dll file. Then restart the connector server.

4. On `windows-host.example.com`, install the Windows Azure AD Module that corresponds to the version of the scripts you are using.
 - For Version 1 scripts, install the MSOnline module.
 - For Version 2 scripts, install the Azure AD module.
5. These instructions assume that you have an existing Azure AD instance.
Create a specific administrative account in Azure AD, to run the PowerShell connector scripts.
6. In a PowerShell window on `windows-host.example.com`, verify that your Windows host can connect to your Azure AD tenant:
 - For Version 1 scripts, run `Connect-MsolService`.
 - For Version 2 scripts, run `Connect-AzureAD`.

Set Up the PowerShell Azure AD Scripts

When all your systems are installed and running, and you have verified that your Windows host can connect to your Azure AD, set up the sample scripts as follows:

1. On `windows-host.example.com`, create a directory for the PowerShell scripts, for example:

```
PS C:\> mkdir -Path openid\scripted-powershell-with-azure-ad\scripts
```

Whatever location you choose for the scripts will be referenced in your connector configuration (provisioner file).

2. Download the Azure AD scripts from the [ForgeRock stash repository](#).

Download either the V1 or V2 scripts, depending on your Azure AD module, and place them in the scripts directory you created in the previous step:

```
ls C:\openid\scripted-powershell-with-azure-ad\scripts
Directory: C:\openid\scripted-powershell-with-azure-ad\scripts
```

Mode	LastWriteTime	Length	Name
-a---	7/21/2020 4:00 AM	10965	AzureADCreate.ps1
-a---	7/21/2020 4:00 AM	3547	AzureADDelete.ps1
-a---	7/21/2020 4:00 AM	6952	AzureADSchema.ps1
-a---	7/21/2020 4:00 AM	8149	AzureADSearch.ps1
-a---	7/21/2020 4:00 AM	2465	AzureADTest.ps1
-a---	7/21/2020 4:00 AM	10840	AzureADUpdate.ps1

Note

By default, Windows does not trust downloaded scripts. To be able to run the scripts, you might need to do the following:

- Run the Unblock-File cmdlet. This cmdlet unblocks PowerShell script files that were downloaded from the Internet so that you can run them, regardless of the PowerShell execution policy.
- Change the PowerShell execution policy to let you run the scripts.

3. On the IDM host, configure IDM to connect to the .NET connector server.
4. On the IDM host, create a connector configuration file for the PowerShell connector in your project's `conf` directory.

The ForgeRock stash repository includes a sample provisioner file for both versions of the scripts. Use those files as a starting point.

Save the file as `conf/provisioner.openicf-azureadpowershell.json` and edit it to match your deployment. Set at least the following properties:

- `connectorHostRef`: The name of the connector server referenced in the previous step.
- `*ScriptFileName`: Set the path to the script directory that you created on `windows-host.example.com`.

Test the PowerShell Connector With Azure AD

1. Start IDM.
2. Test that the connector has been configured correctly and can reach the Azure AD:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--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": "[1.4.2.0,1.5.0.0]"
  },
  "displayName": "PowerShell Connector ",
  "ok": true
}
```

If you see no response from this connector test, check your connector configuration and the connection to the .NET connector server.

IBM RACF Connector

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

IBM Resource Access Control Facility (RACF) is an access control system for IBM mainframes running z/OS. The RACF connector lets you manage and synchronize accounts between RACF and IDM managed user objects. A RACF administrator account is required for this connector to work.

Before you start

Before you configure the connector, log in to your RACF administrator account and note the following:

Host name

The domain name or IP address of the host where RACF is running.

Port

The port RACF is configured to use.

User ID

The RACF administrator user ID.

Password

The password for the RACF administrator account.

Segments

A list of RACF user profile segments that are supported. Refer to ??? for a list of available segments.

Accept self-signed certificates

A boolean determining whether RACF is configured to allow self-signed certificates. This should usually be `false` in production environments, but may be `true` during development.

Client certificate alias

Alias name for the client certificate.

Client certificate password

Password for the client certificate.

Install the RACF connector

Download the connector .jar file from the [ForgeRock BackStage download site](#).

- If you are running the connector locally, place it in the `/path/to/openidm/connectors` directory, for example:

```
mv ~/Downloads/racf-connector-1.5.20.12.jar /path/to/openidm/connectors/
```

- If you are using a remote connector server (RCS), place it in the `/path/to/openicf/connectors` directory on the RCS.

Configure the RACF connector

Create a connector configuration using the Admin UI:

1. Select Configure > Connectors and click New Connector.
2. Enter a Connector Name.
3. Select RACF Connector - 1.5.20.12 as the Connector Type.
4. Provide the Base Connector Details.
5. Click Save.

When your connector is configured correctly, the connector displays as Active in the Admin UI.

Alternatively, test that the configuration is correct by running the following command:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system/racf?_action=test"
{
  "name": "racf",
  "enabled": true,
  "config": "config/provisioner.openicf/racf",
  "connectorRef": {
    "bundleVersion": "${bundleVersion}",
    "bundleName": "org.forgerock.openicf.connectors.racf-connector",
    "connectorName": "org.forgerock.openicf.connectors.racf.RacfConnector"
  },
  "displayName": "RACF Connector",
  "objectTypes": [
    "ACCOUNT",
    "ALL",
    "GROUP"
  ],
  "ok": true
}
```

If the command returns `"ok": true`, your connector was configured correctly, and can authenticate to the RACF system.

RACF segments and attributes

The following tables list available attributes by segment. Attributes listed in the Base segment are available by default. To use any other attributes, include the segment name in the list of segments in the RACF connector configuration.

User accounts support create, update, query, and delete actions. Groups only support query actions.

Base segment

Attribute	Description
<code>userId</code>	The user's ID. Required.
<code>__NAME__</code>	The user's system name. Must match <code>`userId`</code> . Required.
<code>NAME</code>	The user's name.
<code>OWNER</code>	Owner of the user's profile.
<code>DFLTGRP</code>	Default group of the user.
<code>AUTHORITY</code>	User's authority in the default group.
<code>__PASSWORD__</code>	The user's password.

Attribute	Description
PHRASE	Optional password phrase.
REVOKE	Expiration date for the user's system access.
RESUME	Date a user's system access is restored.
WHEN	Days of the week and hours of the day the user has access to the system.
CLAUTH	Classes in which the user can define profiles.
MODEL	Name of the data model profile used when creating new data profiles (either generic or discrete).
GROUP	The group the user belongs to.
SECLABEL	The user's default security label.
GRPACC	Whether other group members have access to any other group set the user protects.
RESTRICTED	Indicates that when checking global access, the account will not be used to allow access to a resource.
AUDITOR	Gives the user the system-wide auditor attribute.
OPERATIONS	Gives the user the system-wide operations attribute.
SPECIAL	Gives the user the system-wide special attribute.
ADSP	Indicates all permanent data sets this user creates should be discrete profiles in RACF.

CICS segment

Attribute	Description
CICS_OPCLASS	The classes the user is assigned in CICS. Determines which basic mapping support (BMS) messages are routed to the user. Represented as a number ranging from 01 to 24.
CICS_OPIDENT	A 1-3 character identification of the user for use by BMS.
CICS_OPPTY	The number (0 to 255) that represents the priority of the user.
CICS_RSLKEY	The resource security level (RSL) keys assigned to the user.
CICS_TIMEOUT	The time in hours and minutes (either HMM or HHMM format) that the operator is allowed to be idle before being signed out.
CICS_TSLKEY	The transaction security level (TLS) keys assigned to the user.
CICS_XRFSOFF	Indicates whether the user should be signed out when an XRF takeover occurs.

DCE segment

Attribute	Description
DCE_AUTOLOGIN	Single Sign On (SSO) processing. Either YES or NO.

Attribute	Description
DCE_DCENAME	The user's DCE principal name.
DCE_HOMECELL	The user's DCE home cell.
DCE_HOMEUUID	The user's DCE UUID.
DCE_UUID	The user's principal DCE UUID.

DFP segment

Attribute	Description
DFP_DATAAPPL	The user's DFP data application identifier.
DFP_DATACLAS	The user's default data class for attributes used during allocation of any new data sets.
DFP_MGMTCLAS	The user's default management class for attributes used in managing a data set after it is allocated.
DFP_STORCLAS	The user's default storage class for logical storage attributes.

KERB segment

Attribute	Description
KERB_ENCRYPT	The user's encryption key types. Available values include: DES , DES3 , DESD , AES128 , and AES256 .
KERB_KERBNAME	The user's local principal name. The value specified must be unique.
KERB_MAXTKTLFE	The maximum Kerberos ticket life specified in seconds. Note that 0 is not a valid value.

LANGUAGE segment

Attribute	Description
LANGUAGE_PRIMARY	The user's primary language.
LANGUAGE_SECONDARY	The user's secondary language.

LNOTES segment

Attribute	Description
LNOTES_SNAME	The user's short name for use with Lotus Notes in z/OS.

NDS segment

Attribute	Description
NDS_UNAME	The user's name for use with Novell Directory Services.

NETVIEW segment

Attribute	Description
NETVIEW_CONSNAME	Master Console Station (MCS) console identifier.
NETVIEW_CTL	Specifies whether a security check is performed for this user. Either GLOBAL , GENERAL , or SPECIFIC .
NETVIEW_DOMAINS	The domain identifier for any domains where the user can start a cross-domain session.
NETVIEW_IC	The initial command or list of commands to be executed by NetView when the user logs in.
NETVIEW_MSGRECVR	Indicates whether the user can receive unsolicited messages.
NETVIEW_NGMFADMN	Indicates whether the user can use the NetView graphic monitor facility.
NETVIEW_OPCLASS	NetView scope classes the user has authority with. The class value is a number from 1 to 2040 .

OMVS segment

Attribute	Description
OMVS_ASSIZEMAX	The user's z/OS maximum address space size.
OMVS_CPUTIMEMAX	The user's z/OS maximum CPU time allowed.
OMVS_FILEPROCMAX	The user's z/OS maximum number of files allowed per process.
OMVS_HOME	The user's z/OS home directory path.
OMVS_MEMLIMIT	The user's z/OS non-shared memory size limit.
OMVS_MMAPAREAMAX	The user's z/OS maximum memory map size.
OMVS_PROCSERMAX	The user's maximum number of processes per UID in z/OS.
OMVS_PROGRAM	The user's z/OS path name, such as a default shell program.
OMVS_SHMEMMAX	The user's z/OS maximum shared memory size.
OMVS_THREADSMAX	The user's z/OS maximum number of threads per process.
OMVS_UID	The user's z/OS user ID.

OPERPARM segment

Attribute	Description
OPERPARM_ALTGRP	Alternative console group used for recovery.
OPERPARM_AUTH	The user's command authority.
OPERPARM_CMDSYS	Name of the system to which the user is connected for command processing.

Attribute	Description
OPERPARM_DOM	Indicates whether the console can receive delete operator message (DOM) requests.
OPERPARM_HC	Indicates whether this console should receive all messages that are directed to hardcopy.
OPERPARM_INTIDS	Indicates whether or not a console should receive messages directed to the internal console.
OPERPARM_KEY	Indicates a data retrieval key used to search for user consoles using the DISPLAY CONSOLES command.
OPERPARM_LEVEL	Message level the user should receive. Available values include R , I , CE , E , IN , NB , or ALL . If you specify ALL , you cannot specify R , I , CE , E , or IN .
OPERPARM_LOGCMDRESP	Indicates whether command responses received by the user are logged.
OPERPARM_MFORM	Specifies the format messages are displayed in. Available values include J , M , S , T , and X .
OPERPARM_MIGID	Indicates whether the user should receive a migration console ID.
OPERPARM_MONITOR	List of events the user can monitor.
OPERPARM_MSCOPE	List of the systems this console can receive unsolicited messages from.
OPERPARM_ROUTECD	Routing codes for messages this console receives.
OPERPARM_STORAGE	The amount of virtual storage (in megabytes) the console is allowed for message queuing.
OPERPARM_UD	Specifies whether this console should receive undelivered messages.
OPERPARM_UNKNIDS	Indicates whether a console should receive messages directed to unknown console IDs.

OVM segment

Attribute	Description
OVM_UID	The user's OpenExtensions for z/VM user ID.
OVM_FSR00T	The user's OpenExtensions for z/VM file system root directory path.
OVM_HOME	The user's OpenExtensions for z/VM home directory path.
OVM_PROGRAM	The user's OpenExtensions for z/VM program path, such as a default shell program.

PROXY segment

Attribute	Description
PROXY_LDAPHOST	The URL of the LDAP server which the z/OS LDAP server contacts when acting as a proxy.
PROXY_BINDDN	The distinguished name (DN) which the z/OS LDAP server uses when acting as a proxy.

TSO segment

Attribute	Description
TSO_ACCTNUM	The user's default TSO account number.
TSO_HOLDCLASS	The user's default hold class.
TSO_JOBCLASS	The user's default job class.
TSO_MAXSIZE	The user's maximum region size.
TSO_MSGCLASS	The user's default message class.
TSO_PROC	The name of the user's default login procedure.
TSO_SIZE	The user's default region size.

WORKATTR segment

Attribute	Description
WORKATTR_WANAME	User name on SYSOUT .
WORKATTR_WABLDG	Building on SYSOUT .
WORKATTR_WADEPT	Department on SYSOUT .
WORKATTR_WAROOM	Room on SYSOUT .
WORKATTR_WAADDR1	SYSOUT address line 1.
WORKATTR_WAADDR2	SYSOUT address line 2.
WORKATTR_WAADDR3	SYSOUT address line 3.
WORKATTR_WAADDR4	SYSOUT address line 4.
WORKATTR_WAACCNT	Account number.
WORKATTR_WAEMAIL	User email address.

Group attributes

The following attributes are available to the **__GROUP__** resource object:

Attribute	Description
UID	ID of the group.
__NAME__	Name of the group.
OWNER	Owner of the group.
SUBGROUP	List of subgroups part of this group.
SUPGROUP	List of groups this group is part of.

Attribute	Description
USERS	List of users part of this group.

Use the RACF connector

You can use the RACF connector to perform the following actions on a RACF account:

+ Create a RACF user

The following example creates a user with the minimum required attributes:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
--data '{
  "NAME": "BJENSEN",
  "userId": "BJENSEN"
}' \
"http://localhost:8080/openidm/system/racf/__ACCOUNT__?_action=create"
{
  "_id": "BJENSEN",
  "NAME": "UNKNOWN",
  "LAST-ACCESS": "UNKNOWN",
  "DFLTGRP": "SYS1",
  "WHEN": {
    "DAYS": "ANYDAY",
    "TIME": "ANYTIME"
  },
  "PASS-INTERVAL": "N/A",
  "PHRASEDATE": "N/A",
  "NAME": "BJENSEN",
  "ENABLE": true,
  "SECLABEL": "NONE SPECIFIED",
  "userId": "BJENSEN",
  "ATTRIBUTES": [
    "PROTECTED"
  ],
  "PASSDATE": "N/A",
  "SECLEVEL": "NONE SPECIFIED",
  "GROUP": [
    {
      "GROUP": "SYS1",
      "OWNER": "IBMUSER",
      "AUTH": "USE",
      "UACC": "NONE"
    }
  ],
  "OWNER": "IBMUSER"
}
```

Note

When you create a new user, you must specify *at least* `__NAME__`, `userId`. Refer to the list of available attributes above for more information.

+ Update a RACF user

You can modify an existing user with a PUT request, including all attributes of the account in the request.

For example, to add a work email and update the name of the user:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--header "if-Match:*" \
--request PUT \
--data '{
  "__NAME__": "BJENSEN",
  "userId": "BJENSEN",
  "WORKATTR_WAEMAIL": "bjensen@example.com",
  "NAME": "Barbara Jensen"
}' \
"http://localhost:8080/openidm/system/racf/__ACCOUNT__/BJENSEN"
{
  "_id": "BJENSEN",
  "NAME": "BARBARA JENSEN",
  "LAST-ACCESS": "UNKNOWN",
  "DFLTGRP": "SYS1",
  "WORKATTR_WAEMAIL": "bjensen@example.com",
  "WHEN": {
    "DAYS": "ANYDAY",
    "TIME": "ANYTIME"
  },
  "PASS-INTERVAL": "N/A",
  "PHRASEDATE": "N/A",
  "__NAME__": "BJENSEN",
  "__ENABLE__": true,
  "SECLABEL": "NONE SPECIFIED",
  "userId": "BJENSEN",
  "ATTRIBUTES": [
    "PROTECTED"
  ],
  "PASSDATE": "N/A",
  "SECLEVEL": "NONE SPECIFIED",
  "__GROUP__": [
    {
      "GROUP": "SYS1",
      "OWNER": "IBMUSER",
      "AUTH": "USE",
      "UACC": "NONE"
    }
  ]
}
```

```
  ],  
  "OWNER": "IBMUSER"  
}
```

+ Query RACF users

The following example queries all RACF users:

```
curl \  
--header "X-OpenIDM-Username: openidm-admin" \\  
--header "X-OpenIDM-Password: openidm-admin" \\  
--header "Accept-API-Version: resource=1.0" \\  
--header "Content-Type: application/json" \\  
--request GET \\  
"http://localhost:8080/openidm/system/racf/__ACCOUNT__?_queryId=query-all-ids"  
{  
  "result": [  
    {  
      "_id": "ADCDY"  
    },  
    {  
      "_id": "ADCDZ"  
    },  
    {  
      "_id": "BJENSEN"  
    },  
    {  
      "_id": "BPX0INIT"  
    },  
    {  
      "_id": "CEA"  
    },  
    {  
      "_id": "CFZSRV"  
    },  
    {  
      "_id": "CICSUSER"  
    },  
    {  
      "_id": "DANY101"  
    },  
    {  
      "_id": "DANY102"  
    },  
    [ ... ]  
    {  
      "_id": "ZOSCAGL"  
    },  
    {  
      "_id": "ZOSCSRV"  
    },  
    {  
      "_id": "ZOSMFAD"  
    },  
    {  
      "_id": "ZOSUGST"  
    },  
  ],  
}
```

```
{
  "_id": "ZWESIUSR"
},
{
  "_id": "ZWESVUSR"
}
],
"resultCount": 162,
"pagedResultsCookie": null,
"totalPagedResultsPolicy": "NONE",
"totalPagedResults": -1,
"remainingPagedResults": -1
}
```

The following command queries a specific user by their ID:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/racf/__ACCOUNT__/BJENSEN"
{
  "_id": "BJENSEN",
  "NAME": "BARBARA JENSEN",
  "LAST-ACCESS": "UNKNOWN",
  "DFLTGRP": "SYS1",
  "WORKATTR_WAEMAIL": "bjensen@example.com",
  "WHEN": {
    "DAYS": "ANYDAY",
    "TIME": "ANYTIME"
  },
  "PASS-INTERVAL": "N/A",
  "PHRASEDATE": "N/A",
  "__NAME__": "BJENSEN",
  "__ENABLE__": true,
  "SECLABEL": "NONE SPECIFIED",
  "userId": "BJENSEN",
  "ATTRIBUTES": [
    "PROTECTED"
  ],
  "PASSDATE": "N/A",
  "SECLEVEL": "NONE SPECIFIED",
  "__GROUP__": [
    {
      "GROUP": "SYS1",
      "OWNER": "IBMUSER",
      "AUTH": "USE",
      "UACC": "NONE"
    }
  ],
  "OWNER": "IBMUSER"
}
```

+ *Reset a RACF account password*

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--header "if-Match:*" \
--request PATCH \
--data '{
  "operation": "add",
  "field": "__PASSWORD__",
  "value": "Passw0rd@123!"
}' \
"http://localhost:8080/openidm/system/racf/__ACCOUNT__/BJENSEN"
{
  "_id": "BJENSEN",
  "NAME": "BARBARA JENSEN",
  "LAST-ACCESS": "22.304/12:17:39",
  "DFLTGRP": "SYS1",
  "WORKATTR_WAEMAIL": "bjensen@example.com",
  "WHEN": {
    "DAYS": "ANYDAY",
    "TIME": "ANYTIME"
  },
  "PASS-INTERVAL": "180",
  "PHRASEDATE": "00.000",
  "__NAME__": "BJENSEN",
  "__ENABLE__": true,
  "SECLABEL": "NONE SPECIFIED",
  "userId": "BJENSEN",
  "ATTRIBUTES": [
    "NOPASSWORD",
    "PASSPHRASE"
  ],
  "PASSDATE": "N/A",
  "SECLEVEL": "NONE SPECIFIED",
  "__GROUP__": [
    {
      "GROUP": "SYS1",
      "OWNER": "IBMUSER",
      "AUTH": "USE",
      "UACC": "NONE"
    }
  ],
  "OWNER": "IBMUSER"
}
```

Note

While the `__PASSWORD__` field is not returned as part of the response, the user object is updated.

+ Delete a RACF user account

You can use the RACF connector to delete an account from the RACF service.

The following example deletes a RACF account:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request DELETE \
"http://localhost:8080/openidm/system/racf/__ACCOUNT__/BJENSEN"
{
  "_id": "BJENSEN",
  "NAME": "BARBARA JENSEN",
  "LAST-ACCESS": "22.304/12:17:39",
  "DFLTGRP": "SYS1",
  "WORKATTR_WAEMAIL": "bjensen@example.com",
  "WHEN": {
    "DAYS": "ANYDAY",
    "TIME": "ANYTIME"
  },
  "PASS-INTERVAL": "180",
  "PHRASEDATE": "00.000",
  "__NAME__": "BJENSEN",
  "__ENABLE__": true,
  "SECLABEL": "NONE SPECIFIED",
  "userId": "BJENSEN",
  "ATTRIBUTES": [
    "NOPASSWORD",
    "PASSPHRASE"
  ],
  "PASSDATE": "N/A",
  "SECLEVEL": "NONE SPECIFIED",
  "__GROUP__": [
    {
      "GROUP": "SYS1",
      "OWNER": "IBMUSER",
      "AUTH": "USE",
      "UACC": "NONE"
    }
  ],
  "OWNER": "IBMUSER"
}
```

OpenICF Interfaces Implemented by the RACF Connector

The RACF Connector implements the following OpenICF interfaces.

Create

Creates an object and its `uid`.

Delete

Deletes an object, referenced by its `uid`.

Schema

Describes the object types, operations, and options that the connector supports.

Script on Connector

Enables an application to run a script in the context of the connector. Any script that runs on the connector has the following characteristics:

- The script runs in the same execution environment as the connector and has access to all the classes to which the connector has access.
- The script has access to a `connector` variable that is equivalent to an initialized instance of the connector. At a minimum, the script can access the connector configuration.
- The script has access to any script-arguments passed in by the application.

Search

Searches the target resource for all objects that match the specified object class and filter.

Test

Tests the connector configuration. Testing a configuration checks all elements of the environment that are referred to by the configuration are available. For example, the connector might make a physical connection to a host that is specified in the configuration to verify that it exists and that the credentials that are specified in the configuration are valid.

This operation might need to connect to a resource, and, as such, might take some time. Do not invoke this operation too often, such as before every provisioning operation. The test operation is not intended to check that the connector is alive (that is, that its physical connection to the resource has not timed out).

You can invoke the test operation before a connector configuration has been validated.

Update

Updates (modifies or replaces) objects on a target resource.

RACF Connector Configuration

The RACF Connector has the following configurable properties.

Configuration properties

Property	Type	Default	Encrypted ^a	Required ^b
<code>hostName</code>	<code>String</code>	<code>null</code>		Yes
Host name or IP address of RACF				
<code>port</code>	<code>Integer</code>	<code>null</code>		Yes

Property	Type	Default	Encrypted ^a	Required ^b
TCP/IP port number used to communicate with the RACF				
<code>userId</code>	String	null		Yes
The user id used to login to RACF				
<code>password</code>	GuardedString	null	Yes	Yes
The password used to login to RACF				
<code>segments</code>	String	null		No
To retrieve data based on RACF segments				
<code>acceptSelfSignedCertificates</code>	boolean	false		Yes
Accept or not self-signed certificates				
<code>clientCertAlias</code>	String	null		No
Alias for the client certificate				
<code>clientCertPassword</code>	GuardedString	null	Yes	No
Password for the client certificate				
<code>maximumConnections</code>	Integer	10		No
Provide the maximum connections				
<code>connectionTimeout</code>	Integer	300		No
Provide the maximum connection timeout in seconds				
<code>httpProxyHost</code>	String	null		No
Provide the Proxy Host				
<code>httpProxyPort</code>	Integer	null		No
Provide the Proxy Port				
<code>httpProxyUsername</code>	String	null		No
Provide the Proxy Username				
<code>httpProxyPassword</code>	GuardedString	null	Yes	No
Provide the Proxy Password				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

Salesforce Connector

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

The Salesforce connector enables provisioning, reconciliation, and synchronization between Salesforce accounts and the IDM managed user repository.

This chapter describes how to install and configure the Salesforce connector, and how to perform basic tests to ensure that it's running correctly. For a complete example that includes the configuration required to synchronize users with this connector, see "*Synchronize Users Between Salesforce and IDM*" in the *Samples Guide*.

Before You Configure the Salesforce Connector

The instructions in this chapter assume that you have an existing Salesforce organization, a Salesforce administrative account, and a Connected App with OAuth enabled.

For instructions on setting up a Connected App, see the corresponding [Salesforce documentation](#). When you have set up the Connected App, locate the *Consumer Key* and *Consumer Secret*. You will need these details to configure the connector.

The Salesforce connector is bundled with IDM and has no specific installation requirements.

Configuring the Salesforce Connector

You can configure the Salesforce connector using the Admin UI, or by setting up a provisioner file in your project's `conf` directory. Using the Admin UI is recommended.

To Configure the Salesforce Connector Through the UI

1. To configure the connector using the Admin UI, start IDM:

```
/path/to/openidm/startup.sh
```
2. Log in to the Admin UI at <https://localhost:8443/admin> (substitute `localhost` for the host on which your IDM instance is running).
3. Select Configure > Connectors, and click New Connector.
4. Enter a Connector Name (for example, Salesforce) and select Salesforce Connector - 1.5.20.11 as the Connector Type.
5. Supply the Login URL, Consumer Key, Consumer Secret and click Save.

The Login URL is the OAuth endpoint that will be used to make the OAuth authentication request to Salesforce.

Note

When you create your connected app, you are instructed to wait 2-10 minutes for the settings to propagate across all the Salesforce data centers. If you are using a Salesforce test tenant, such as <https://eu26.lightning.force.com>, you can specify a custom URL here and enter the FQDN of the test tenant. This will enable you to test the connector without waiting for the new app settings to be propagated.

6. Select Save to update the connector configuration.

The connector now attempts to access your Salesforce organization.

Enter your Salesforce login credentials.

On the permission request screen click Allow, to enable IDM to access your Salesforce Connected App.

7. When your connector is configured correctly, the connector displays as Active in the UI.

To Configure the Salesforce Connector With a Configuration File

1. IDM provides a sample connector configuration file in the `/path/to/openidm/samples/example-configurations/provisioners` directory.

Copy this sample file (`provisioner.openicf-salesforce.json`) to your project's `conf` directory, and set at least the following properties:

```
"configurationProperties" : {
  "loginUrl" : "loginURL",
  "clientSecret" : "clientSecret",
  "clientId" : "clientId",
  "refreshToken" : "refreshToken"
  "instanceUrl" : "instanceURL",
}
```

loginUrl

The OAuth endpoint that will be used to make the OAuth authentication request to Salesforce.

The default endpoint for a production system is <https://login.salesforce.com/services/oauth2/token>. The default endpoint for a sandbox (test) system is <https://test.salesforce.com/services/oauth2/token>.

clientSecret

The Consumer Secret associated with your Connected App.

clientId

The Consumer Key associated with your Connected App.

refreshToken and instanceURL

The Admin UI obtains these properties on your behalf. If you are configuring the connector manually, obtain the refresh token and instance URL from salesforce.com as follows:

1. Point your browser to the following URL:

```
SALESFORCE_URL/services/oauth2/authorize?
response_type=code&client_id=CONSUMER_KEY&redirect_uri=REDIRECT_URI&scope=id+api+refresh_token
```

Where:

- *SALESFORCE_URL* is one of the following:
 - A production URL (<https://login.salesforce.com>)
 - A sandbox URL (<https://test.salesforce.com>)
 - A custom Salesforce MyDomain URL, such as:

```
https://ic-example-com--SUP1.cs21.my.salesforce.com
```

- *CONSUMER_KEY* is the Consumer Key associated with the Connected App that you created within your Salesforce organization.
- *REDIRECT_URI* is the IDM URI Salesforce should redirect to during authentication. It must match the Redirect URI specified within your Salesforce Connect App configuration, for example:

```
https://localhost:8443/
```

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

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.

- Copy the authorization code from the response URL and use it as the value of the `code` parameter in the following REST call. The `consumer-key`, `redirect-uri`, and `SALESFORCE_URL` must match what you used in the first step of this procedure:

```
curl \
--verbose \
--data "grant_type=authorization_code" \
--data "client_id=consumer-key" \
--data "client_secret=consumer-secret" \
--data "redirect_uri=https://localhost:8443/" \
--data "code=access-token-code" \
--data "SALESFORCE_URL/services/oauth2/token"
{
  "access_token": "00DS0000003K4fU!AQMAQ0zEU.8tCjg8Wk79yKPKCtrtaszX5jrHtoT4NBpJ8x...",
  "signature": "2uREX1lseXdg3Vng/2+Hrlo/KHOWYoim+poj74wKFtw=",
  "refresh_token": "5Aep861KIwKdekr90I4iHdtDgWwRoG70_6uHrgJ.yVtMS0UaGxRqE6WFM77W7...",
  "token_type": "Bearer",
  "instance_url": "https://example-com.csl.my.salesforce.com",
  "scope": "id api refresh_token",
  "issued_at": "1417182949781",
  "id": "https://login.salesforce.com/id/00DS0000003K4fUMAS/00530000009hWLcAAM"
}
```

The output includes the `refresh_token` and the `instance_url` that you need to configure the connector.

- Set `"enabled" : true` to enable the connector.
- Save the connector configuration.
- Test that the configuration is correct by running the following command:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system/salesforce?_action=test"
{
  "name": "salesforce",
  "enabled": true,
  "config": "config/provisioner.openicf/salesforce",
  "connectorRef": {
    "bundleVersion": "1.5.20.11",
    "bundleName": "org.forgerock.openicf.connectors.salesforce-connector",
    "connectorName": "org.forgerock.openicf.connectors.salesforce.SalesforceConnector"
  },
  "displayName": "Salesforce Connector",
  "objectTypes": [
    "ALL",
    "User"
  ],
  "ok": true
}
```

If the command returns `"ok": true`, your connector has been configured correctly, and can authenticate to Salesforce.

Implementation Specifics

For PATCH requests, a connector can potentially add, remove, or replace an attribute value. The Salesforce connector does not implement the add or remove operations, so a PATCH request always *replaces* the entire attribute value with the new value. Salesforce does not support multi-valued attributes.

Attributes themselves cannot be removed from Salesforce. The connector therefore performs an update with `"` as the value of the attribute being removed. This sets the value of the removed attribute to `null`.

Note

Salesforce does not support application user DELETE requests.

OpenICF Interfaces Implemented by the Salesforce Connector

The Salesforce Connector implements the following OpenICF interfaces.

Create

Creates an object and its `uid`.

Delete

Deletes an object, referenced by its `uid`.

Schema

Describes the object types, operations, and options that the connector supports.

Script on Connector

Enables an application to run a script in the context of the connector. Any script that runs on the connector has the following characteristics:

- The script runs in the same execution environment as the connector and has access to all the classes to which the connector has access.
- The script has access to a `connector` variable that is equivalent to an initialized instance of the connector. At a minimum, the script can access the connector configuration.
- The script has access to any script-arguments passed in by the application.

Search

Searches the target resource for all objects that match the specified object class and filter.

Sync

Polls the target resource for synchronization events, that is, native changes to objects on the target resource.

Test

Tests the connector configuration. Testing a configuration checks all elements of the environment that are referred to by the configuration are available. For example, the connector might make a physical connection to a host that is specified in the configuration to verify that it exists and that the credentials that are specified in the configuration are valid.

This operation might need to connect to a resource, and, as such, might take some time. Do not invoke this operation too often, such as before every provisioning operation. The test operation is not intended to check that the connector is alive (that is, that its physical connection to the resource has not timed out).

You can invoke the test operation before a connector configuration has been validated.

Update

Updates (modifies or replaces) objects on a target resource.

Salesforce Connector Configuration

The Salesforce Connector has the following configurable properties.

Basic Configuration Properties

Property	Type	Default	Encrypted ^a	Required ^b
<code>clientId</code>	String	null		Yes
The client identifier				
<code>clientSecret</code>	GuardedString	null		Yes
The secure client secret for OAUTH				
<code>refreshToken</code>	GuardedString	null		Yes
The refresh token for the application				
<code>loginUrl</code>	String	<code>https://login.salesforce.com/services/oauth2/token</code>		Yes
The endpoint from which a new access token should be queried (<code>https://login.salesforce.com/services/oauth2/token</code>)				
<code>instanceUrl</code>	String	null		Yes
The URL of the Salesforce instance (such as <code>https://example-com.cs1.my.salesforce.com</code>)				
<code>version</code>	double	48.0		No
The Salesforce API version				
<code>connectTimeout</code>	long	120000		No
The maximum connection timeout				
<code>proxyHost</code>	String	null		No
The hostname of an http proxy, used between the connector and the Salesforce service provider				
<code>proxyPort</code>	Integer	3128		No
The proxy port number, if an HTTP proxy is used between the connector and the Salesforce service provider				
<code>maximumConnections</code>	int	10		No
The maximum size of the HTTP connection pool				
<code>supportedObjectTypes</code>	String[]	[]		No
Defines a list of Salesforce objects that will be used to dynamically build the provisioner schema				
<code>proxyUri</code>	String	null		No
The URI of an HTTP proxy that contains the scheme, host, and port number for that proxy				
<code>proxyUsername</code>	String	null		No

Property	Type	Default	Encrypted ^a	Required ^b
The proxy username to use with a proxy that requires authentication				
<code>proxyPassword</code>	<code>GuardedString</code>	<code>null</code>		No
The proxy user password to use with a proxy that requires authentication				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

SAP Connector

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

The SAP connector is an implementation of the Scripted Groovy Connector Toolkit that connects to any SAP system using the SAP JCo Java libraries. This chapter describes how to install and configure the scripted SAP connector, and how to test the sample scripts that are bundled with the connector.

The sample scripts illustrate the following scenarios:

- Synchronization of users between an SAP HR module and IDM
- Synchronization of users between IDM and an SAP (R/3) system

Before You Start

1. Download the SAP connector from the [ForgeRock BackStage download site](#).
2. Copy the SAP connector JAR file (`sap-connector-1.5.20.12.jar`) to the `openidm/connectors` directory:

```
cp ~/Downloads/sap-connector-1.5.20.12.jar /path/to/openidm/connectors
```

3. The SAP connector requires the SAP Java Connector (JCo) libraries, version 3.0.12 or later. ForgeRock distributes the SAP connector without these JCo libraries. Before you can use the SAP connector, you must obtain the JCo libraries that correspond to your architecture.

Copy the required SAP JCo libraries to the `/path/to/openidm/lib` directory. For example:

```
cp sapjco3.jar /path/to/openidm/lib
cp libsapjco3.so /path/to/openidm/lib
```

4. Change your IDM logging configuration to log messages from the SAP connector.

By default, IDM logs nothing for the SAP connector. To troubleshoot any issues with the connector, set the following properties in your project's `conf/logging.properties` file:

```
# SAP Connector Logging
org.forgerock.openicf.connectors.sap.level=FINER
scripts.sap.r3.level=FINER
scripts.sap.hr.level=FINER
scripts.sap.level=FINER
```

Using the SAP Connector With an SAP HR System

The SAP HR sample scripts enable you to manage the email address and global employee UID of records in an SAP HR system.

The following sections explain how to configure IDM to use these sample scripts, how to test the connection to the SAP HR system, and how to update user records.

Setting up IDM for the SAP HR Samples

1. Create a connector configuration file for the SAP connector and place it in your project's `conf/` directory.

You can use this sample `provisioner.openicf-saphr.json` as a guide.

Edit that file with the connection details for your SAP HR system. Specifically, set at least the following properties:

`destination`

An alias to the SAP system to which you are connecting, for example, `SAP1`. If you are connecting to more than one SAP system, the `destination` property for each system must be unique.

The sample connector configuration assumes a connection to a single SAP system, so the value for this property in the sample configuration is `OPENIDM`.

`asHost`

The FQDN of your SAP Application Server, for example `sap.example.com`.

`user`

Your SAP user account.

`password`

The password of this SAP user account.

`client`

The SAP Client number that will be used to connect to the SAP system.

systemNumber

The SAP system number.

directConnection

A boolean (true/false). If **true**, the connection goes directly to an SAP ABAP Application server or SAP router. If **false**, the connection goes to a group of SAP instances, through an SAP message server.

sapRouter

The IP address and port of the SAP router, if applicable. The syntax is `/H/host[/S/port]`, for example `/H/203.0.113.0/S/3299`.

poolCapacity

The maximum number of idle connections kept open by the destination. If there is no connection pooling, set this to **0**. The default value is **1**.

For optimum performance, set this value to an integer between **5** and **10**.

- The connector bundles a number of SAP-certified sample Groovy scripts:

TestSAP.groovy
SearchSAPHR.groovy
UpdateSAPHR.groovy
SchemaSAPHR.groovy
EmplComm.groovy

If necessary, you can customize these scripts to suit your deployment by extracting them from the connector JAR and updating the connector configuration to point to the new file path.

The sample connector configuration assumes the following locations for the scripts (relative to the value of the `scriptRoots` property):

```
"testScriptFileName" : "scripts/sap/TestSAP.groovy",
"searchScriptFileName" : "scripts/sap/hr/SearchSAPHR.groovy",
"updateScriptFileName" : "scripts/sap/hr/UpdateSAPHR.groovy",
"schemaScriptFileName" : "scripts/sap/hr/SchemaSAPHR.groovy",
```

The `EmplComm.groovy` must be placed in the same location as the Search, Update, and Schema scripts.

Important

The Groovy scripts belong to a specific package. The parent directory where the scripts are located must be the same as the package name. So the `TestSAP.groovy` script must be under a `scripts/sap` directory

(because it belongs to the `scripts/sap` package) and the remaining HR scripts must be under a `scripts/sap/hr` directory (because they belong to the `hr` package).

Testing the Connection to the SAP HR System

1. Start IDM with the configuration for your SAP connector project.

This procedure assumes that the configuration is in the default `path/to/openidm` directory. If your SAP project is in a different directory, use the `-p` option with the startup command to point to that directory:

```
path/to/openidm/startup.sh
```

2. Test that the connector has been configured correctly and that the SAP HR system can be reached:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system/saphr/?_action=test"
{
  "name" : "saphr",
  "enabled" : true,
  "config" : "config/provisioner.openicf/saphr2",
  "objectTypes" : [ "__ALL__", "employee" ],
  "connectorRef" : {
    "connectorName" : "org.forgerock.openicf.connectors.sap.SapConnector",
    "bundleName" : "org.forgerock.openicf.connectors.sap-connector",
    "bundleVersion" : "1.5.20.12"
  },
  "displayName" : "Sap Connector",
  "ok" : true
}
```

3. Retrieve a list of the existing users (with their employee number) in the SAP HR system:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
"http://localhost:8080/openidm/system/saphr/employee?_queryId=query-all-ids"
{
  "result" : [ {
    "_id" : "000000010",
    "_NAME" : "000000010"
  }, {
    "_id" : "000000069",
    "_NAME" : "000000069"
  }, {
    "_id" : "000000070",
    "_NAME" : "000000070"
  },
  ...
}
```

4. Retrieve the complete record of an employee in the SAP HR system by including the employee's ID in the URL.

The following command retrieves the record for employee Maria Gonzales:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
"http://localhost:8080/openidm/system/saphr/employee/55099307"
{
  "_id" : "55099307",
  "PERSONAL_DATA" : {
    "PERNO" : "55099307",
    "INFOTYPE" : "0002",
    "TO_DATE" : "Fri Dec 31 00:00:00 CET 9999",
    "FROM_DATE" : "Tue Mar 30 00:00:00 CET 1954",
    "SEQNO" : "000",
    "CH_ON" : "Thu Mar 27 00:00:00 CET 2003",
    "CHANGED_BY" : "MAYROCK",
    "LAST_NAME" : "Gonzales",
    "FIRSTNAME" : "Maria",
    "NAME_FORM" : "00",
    "FORMOFADR" : "2",
    "GENDER" : "2",
    "BIRTHDATE" : "Tue Mar 30 00:00:00 CET 1954",
    "LANGU" : "D",
    "NO_O_CHLDR" : "0",
    "BIRTHYEAR" : "1954",
    "BIRTHMONTH" : "03",
    "BIRTHDAY" : "30",
    "LASTNAME_M" : "GONZALES",
    "FSTNAME_M" : "MARIA"
  },
  ...
}
```

Using the SAP Connector to Manage Employee Information (SAP HR)

The following sample commands show how the SAP connector is used to manage the email account of user Maria Gonzales, retrieved in the previous step. Management of the global UID (**SYS-UNAME**) works in the same way.

1. Check if Maria Gonzales already has an email account on the SAP HR system by filtering a query on her user account for the **EMAIL** field:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
"http://localhost:8080/openidm/system/saphr/employee/55099307?_fields=EMAIL"
{
  "_id" : "55099307",
}
```

No email account is found for Maria Gonzales.

2. Add an email account by sending a PUT request. The JSON payload should include the email address as the value of the **ID** property:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request PUT \
--data '{
  "EMAIL": { "ID": "maria.gonzales@example.com" }
}' \
"http://localhost:8080/openidm/system/saphr/employee/55099307"
{
  "_id" : "55099307",
  "EMAIL" : [ {
    "EMPLOYEEID" : "55099307",
    "SUBTYPE" : "0010",
    "VALIDEND" : "Fri Dec 31 00:00:00 CET 9999",
    "VALIDBEGIN" : "Fri March 18 00:00:00 CET 2016",
    "RECORDNR" : "000",
    "COMMTYPE" : "0010",
    "NAMEOFCOMMTYPE" : "E-mail",
    "ID" : "Maria.Gonzales@example.com"
  } ],
  ...
}
```

By default, the connector sets the **VALIDBEGIN** date to the current date, and the **VALIDEND** date to the SAP "END" date (12/31/9999). You can specify different temporal constraints by including these properties in the JSON payload, with the format **YYYYMMDD**. For example:

```
{
  "EMAIL": {
    "ID": "maria.gonzales@example.com"
    "VALIDBEGIN": "20160401",
    "VALIDEND": "20161231"
  }
}
```

3. To change the value of an existing email account, provide a new value for the **ID**.

The JSON payload of the change request must also include the **RECORDNR** attribute, as well as the **VALIDBEGIN** and **VALIDEND** dates, in SAP format (YYYYMMDD).

The following example changes Maria Gonzales' email address to **maria.gonzales-admin@example.com**:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request PUT \
--data '{
  "EMAIL": {
    "ID": "maria.gonzales-admin@example.com",
    "RECORDNR" : "000",
    "VALIDEND" : "99991231",
    "VALIDBEGIN" : "20000101"
  }
}' \
"http://localhost:8080/openidm/system/saphr/employee/55099307"
```

4. To change the temporal constraint (**VALIDEND** date) of the record, include the existing **VALIDEND** data in the JSON payload, and specify the new end date as a value of the **DELIMIT_DATE** attribute.

The following example changes the end date of Maria Gonzale's new mail address to December 31st, 2016:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request PUT \
--data '{
  "EMAIL": {
    "ID": "maria.gonzales-admin@example.com",
    "RECORDNR" : "000",
    "VALIDEND" : "99991231",
    "VALIDBEGIN" : "20000101",
    "DELIMIT_DATE": "20161231"
  }
}' \
"http://localhost:8080/openidm/system/saphr/employee/55099307"
```

5. To delete the email address of the record, send a PUT request with the current **RECORDNR**, **VALIDBEGIN**, and **VALIDEND** attributes, but without the **ID**.

The following request removes the email address from Maria Gonzales' record:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request PUT \
--data '{
  "EMAIL": {
    "RECORDNR" : "000",
    "VALIDEND" : "99991231",
    "VALIDBEGIN" : "20000101"
  }
}' \
"http://localhost:8080/openidm/system/saphr/employee/55099307"
```

Using the SAP Connector to Manage SAP Basis System (R/3) Users

The SAP Connector enables you to perform the following operations on SAP system user accounts:

- List all users
- List all activity groups (roles)
- Manage user profiles
- List all user companies
- Obtain a user's details
- Create a user
- Update a user
- Assign roles to a user
- Lock a user account
- Unlock a user account
- Delete a user account

Currently, the SAP connector cannot detect changes on the SAP system in real time. You must run a reconciliation operation to detect changes on the SAP system.

Setting up IDM for the SAP R/3 Samples

1. Create a connector configuration file for the SAP connector and place it in your project's `conf/` directory.

You can use this sample `provisioner.openicf-sapr3.json` as a guide.

Edit that file with the connection details for your SAP R/3 system. Specifically, set at least the following properties:

`destination`

An alias to the SAP system to which you are connecting, for example, `SAP1`. If you are connecting to more than one SAP system, the `destination` property for each system must be unique.

The sample connector configuration assumes a connection to a single SAP system, `MYSAP`.

`asHost`

The FQDN of your SAP Application Server, for example `sap.example.com`.

`user`

Your SAP user account.

`password`

The password of this SAP user account.

`client`

The SAP Client number that will be used to connect to the SAP system.

`systemNumber`

The SAP system number.

`directConnection`

A boolean (true/false). If `true`, the connection goes directly to an SAP ABAP Application server or SAP router. If `false`, the connection goes to a group of SAP instances, through an SAP message server.

`sapRouter`

The IP address and port of the SAP router, if applicable. The syntax is `/H/host[/S/port]`, for example `/H/203.0.113.0/S/3299`.

`poolCapacity`

The maximum number of idle connections kept open by the destination. If there is no connection pooling, set this to `0`. The default value is `1`.

For optimum performance, set this value to an integer between `5` and `10`.

2. The connector bundles a number of SAP-certified sample Groovy scripts:

```
TestSAP.groovy
SearchSAPR3.groovy
CreateSAPR3.groovy
UpdateSAPR3.groovy
DeleteSAPR3.groovy
SchemaSAPR3.groovy
```

If necessary, you can customize these scripts to suit your deployment by extracting them from the connector JAR and updating the connector configuration to point to the new file path.

The sample connector configuration assumes the following locations for the scripts (relative to the value of the `scriptRoots` property):

```
"testScriptFileName" : "scripts/sap/TestSAP.groovy",
"searchScriptFileName" : "scripts/sap/r3/SearchSAPR3.groovy",
"createScriptFileName" : "scripts/sap/r3/CreateSAPR3.groovy",
"updateScriptFileName" : "scripts/sap/r3/UpdateSAPR3.groovy",
"deleteScriptFileName" : "scripts/sap/r3/DeleteSAPR3.groovy",
"schemaScriptFileName" : "scripts/sap/r3/SchemaSAPR3.groovy",
```

Important

The Groovy scripts belong to a specific package. The parent directory where the scripts are located must be the same as the package name. So the `TestSAP.groovy` script must be under a `scripts/sap` directory (because it belongs to the `scripts/sap` package) and the R/3 scripts must be under a `scripts/sap/r3` directory (because they belong to the `r3` package).

Testing the Connection to the SAP R/3 System

1. Start IDM with the configuration for your SAP R/3 project.

This procedure assumes that the configuration is in the default `path/to/openidm` directory. If your SAP project is in a different directory, use the `-p` option with the startup command to point to that directory:

```
/path/to/openidm/startup.sh
```

2. Test that the connector has been configured correctly and that the SAP R/3 system can be reached:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system/mysap/?_action=test"
{
  "name": "mysap",
  "enabled": true,
  "config": "config/provisioner.openicf/mysap",
  "objectTypes": [
    "__ALL__",
    "user",
    "activity_group",
    "company",
    "profile"
  ],
  "connectorRef": {
    "connectorName": "org.forgerock.openicf.connectors.sap.SapConnector",
    "bundleName": "org.forgerock.openicf.connectors.sap-connector",
    "bundleVersion": "1.5.20.12"
  },
  "displayName": "Sap Connector",
  "ok": true
}
```

Using the SAP Connector to Manage SAP R/3 Users

This section provides sample commands for managing users in an SAP system.

Listing the Users in the SAP System

The following command returns a list of the existing users in the SAP system, with their IDs:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
"http://localhost:8080/openidm/system/mysap/user?_queryId=query-all-ids"
{
  "result": [
    {
      "_id": "BJENSEN",
      "__NAME__": "BJENSEN"
    },
    {
      "_id": "DDIC",
      "__NAME__": "DDIC"
    },
    ...
    {
      "_id": "USER4",
      "__NAME__": "USER4"
    }
  ],
}
```

```
{
  "_id": "USER6",
  "__NAME__": "USER6"
},
{
  "_id": "USER7",
  "__NAME__": "USER7"
}
],
"resultCount": 9,
"pagedResultsCookie": null,
"totalPagedResultsPolicy": "NONE",
"totalPagedResults": -1,
"remainingPagedResults": -1
}
```

Obtaining the Details of an SAP User

The following command uses the SAP connector to obtain a user's details from a target SAP system:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
"http://localhost:8080/openidm/system/mysap/user/BJENSEN"
{
  " __NAME__ ": "BJENSEN",
  " __ENABLE__ ": true,
  " __ENABLE_DATE__ ": "2015-09-01",
  " __DISABLE_DATE__ ": "2016-09-01",
  " __LOCK_OUT__ ": false,
  "ADDTel": [
    {
      "COUNTRY": "DE",
      "TELEPHONE": "19851444",
      ...
    },
    ...
  ],
  "PROFILES": [
    {
      "BAPIPROF": "T_ALM_CONF",
      ...
    }
  ],
  "ISLOCKED": {
    "WRNG_LOGON": "U",
    ...
  },
  "ACTIVITYGROUPS": [
    {
      "AGR_NAME": "MW_ADMIN",
      "FROM_DAT": "2015-07-15",
      "TO_DAT": "9999-12-31",
      "AGR_TEXT": "Middleware Administrator"
    },
    ...
  ]
}
```

```

    },
    "DEFAULTS": {
        ...
    },
    "COMPANY": {
        "COMPANY": "SAP AG"
    },
    "ADDRESS": {
        ...
    },
    "UCLASS": {
        ...
    },
    "LASTMODIFIED": {
        "MODDATE": "2015-07-15",
        "MODTIME": "14:22:57"
    },
    "LOGONDATA": {
        "GLTGV": "2015-09-01",
        "GLTGB": "2016-09-01",
        ...
    },
    "_id": "BJENSEN"
}

```

In addition to the standard user attributes, the GET request returns the following ICF operational attributes:

- `__ENABLE__` - indicates whether the account is enabled, based on the value of the `LOGONDATA` attribute
- `__ENABLE_DATE__` - set to the value of `LOGONDATA/GLTGV` (date from which the user account is valid)
- `__DISABLE_DATE__` - set to the value of `LOGONDATA/GLTGB` (date to which the user account is valid)
- `__LOCK_OUT__` - indicates whether the account is locked

Creating SAP User Accounts

To create a user, you must supply *at least* a username and password. If you do not provide a lastname, the connector uses the value of the username.

The following command creates a new SAP user, `SCARTER`:

```

curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request POST \
--data '{
  " __NAME__ " : "SCARTER",
  " __PASSWORD__ ": "Passw0rd"
}' \
"http://localhost:8080/openidm/system/mysap/user/?_action=create"
{
  "_id": "SCARTER",

```

```
"COMPANY": {
  "COMPANY": "SAP AG"
},
"_LOCK_OUT_": false,
"ADDRESS": {
  ...
},
"_NAME_": "SCARTER",
"LASTMODIFIED": {
  "MODDATE": "2016-04-20",
  "MODTIME": "04:14:29"
},
"UCLASS": {
  "COUNTRY_SURCHARGE": "0",
  "SUBSTITUTE_FROM": "0000-00-00",
  "SUBSTITUTE_UNTIL": "0000-00-00"
},
"_ENABLE_": true,
"DEFAULTS": {
  "SPDB": "H",
  "SPDA": "K",
  "DATFM": "1",
  "TIMEFM": "0"
},
"LOGONDATA": {
  ...
},
"ISLOCKED": {
  "WRNG_LOGON": "U",
  "LOCAL_LOCK": "U",
  "GLOB_LOCK": "U",
  "NO_USER_PW": "U"
}
}
```

The SAP account that is created is valid and enabled, but the password is expired by default. To log in to the SAP system, the newly created user must first provide a new password.

To create a user with a valid (non-expired) password, include the `__PASSWORD_EXPIRED__` attribute in the JSON payload, with a value of `false`. For example:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request POST \
--data '{
  "_NAME_": "SCARTER",
  "_PASSWORD_": "Passw0rd",
  "_PASSWORD_EXPIRED_": false
}' \
"http://localhost:8080/openidm/system/mysap/user/?_action=create"
```

To create an account that is locked by default, include the `__LOCK_OUT__` attribute in the JSON payload, with a value of `true`. For example:

```
curl \
```

```
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request POST \
--data '{
  "_NAME_" : "SCARTER",
  "_PASSWORD_" : "Passw0rd",
  "_LOCK_OUT_" : true
},' \
"http://localhost:8080/openidm/system/mysap/user/?_action=create"
{
  "_NAME_" : "SCARTER",
  "_ENABLE_" : false,
  "_LOCK_OUT_" : true,
  "LOGONDATA": {
    "GLTGV": "0000-00-00",
    "GLTGB": "0000-00-00",
    "USTYP": "A",
    "LTIME": "00:00:00",
    "BCODE": "2FC0D86C99AA5862",
    "CODVN": "B",
    "PASSCODE": "1DBBD983287D7CB4D8177B4333F439F808A395FA",
    "CODVC": "F",
    "PWDSALTEDHASH": "{x-issaha, 1024}zrs3Zm/fX/l/KFGATp3kv0Glis3zLLiPmPVCDpJ9XF0=",
    "CODVS": "I"
  },
  "LASTMODIFIED": {
    "MODDATE": "2015-10-01",
    "MODTIME": "15:25:18"
  },
  "ISLOCKED": {
    "WRNG_LOGON": "U",
    "LOCAL_LOCK": "L",      // "L" indicates that the user is locked on the local system
    "GLOB_LOCK": "U",
    "NO_USER_PW": "U"
  },
  ...
}
```

Schema Used by the SAP Connector For User Accounts

For the most part, the SAP connector uses the standard SAP schema to create a user account. The most common attributes in an SAP user account are as follows:

- **ADDRESS** - user address data
- **LOGONDATA** - user logon data
- **DEFAULTS** - user account defaults
- **COMPANY** - the company to which the user is assigned
- **REF_USER** - the usernames of the Reference User
- **ALIAS** - an alias for the username

- **UCLASS** - license-related user classification
- **LASTMODIFIED** - read-only attribute that indicates the date and time that the account was last changed
- **ISLOCKED** - read-only attribute that indicates the lockout status of the account
- **IDENTITY** - assignment of a personal identity to the user account
- **PROFILES** - any profiles assigned to the user account (see "Managing User Profiles").
- **ACTIVITYGROUPS** - activity groups assigned to the user
- **ADDTEL** - telephone numbers assigned to the user

In addition, the SAP connector supports the following ICF operational attributes for CREATE requests:

- **LOCK_OUT**
- **PASSWORD**
- **PASSWORD_EXPIRED**

The following example creates a user, KVAUGHAN, with all of the standard attributes:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request POST \
--data '{
  "_NAME_": "KVAUGHAN",
  "_PASSWORD_": "Passw0rd",
  "_PASSWORD_EXPIRED_": false,
  "LOGONDATA": {
    "GLTGV": "2016-04-01",
    "GLTGB": "2016-12-01",
    "USTYP": "A"
  },
  "ADDRESS": {
    "FIRSTNAME": "Katie",
    "LASTNAME": "Vaughan",
    "TEL1_NUMBR": "33297603177",
    "E_MAIL": "katie.vaughan@example.com",
    "FUNCTION": "Test User"
  },
  "COMPANY": {
    "COMPANY": "EXAMPLE.COM"
  },
  "ALIAS": {
    "USERALIAS": "KVAUGHAN"
  }
}' \
"http://localhost:8080/openidm/system/mysap/user/?_action=create"
{
  "_id": "KVAUGHAN",
  "ADDRESS": {
```

```
"PERS_NO": "0000010923",
"ADDR_NO": "0000010765",
"FIRSTNAME": "Katie",
"LASTNAME": "Vaughan",
"FULLNAME": "Katie Vaughan",
...
"E_MAIL": "katie.vaughan@example.com",
"LANGU_CR_P": "E",
"LANGU_CPI50": "EN"
},
"LOGONDATA": {
  "GLTGV": "2016-04-01",
  "GLTGB": "2016-12-01",
  ...
},
"COMPANY": {
  "COMPANY": "SAP AG"
},
"__ENABLE__": true,
"ADDTEL": [
  {
    ...
  }
],
"ISLOCKED": {
  "WRNG_LOGON": "U",
  "LOCAL_LOCK": "U",
  "GLOB_LOCK": "U",
  "NO_USER_PW": "U"
},
"UCLASS": {
  "COUNTRY_SURCHARGE": "0",
  "SUBSTITUTE_FROM": "0000-00-00",
  "SUBSTITUTE_UNTIL": "0000-00-00"
},
"ALIAS": {
  "USERALIAS": "KVAUGHAN"
},
"__NAME__": "KVAUGHAN",
"__LOCK_OUT__": false,
"LASTMODIFIED": {
  "MODDATE": "2016-04-20",
  "MODTIME": "04:55:08"
},
"__ENABLE_DATE__": "2016-04-01",          // (Value of LOGONDATA/GLTGV)
"DEFAULTS": {
  "SPDB": "H",
  "SPDA": "K",
  "DATFM": "1",
  "TIMEFM": "0"
},
"__DISABLE_DATE__": "2016-12-01"        // (Value of LOGONDATA/GLTGB)
}
```

Updating SAP User Accounts

The following sections provide sample commands for updating an existing user account.

Locking and Unlocking an Account

To lock or unlock a user's account, send a PUT request, and set the value of the user's `__LOCK_OUT__` attribute to `true`.

The following example locks user KVAUGHAN's account:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--header "If-Match: *" \
--request PUT \
--data '{
  "__LOCK_OUT__": true
}' \
"http://localhost:8080/openidm/system/mysap/user/KVAUGHAN"
```

The following example unlocks KVAUGHAN's account:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--header "If-Match: *" \
--request PUT \
--data '{
  "__LOCK_OUT__": false
}' \
"http://localhost:8080/openidm/system/mysap/user/KVAUGHAN"
```

Updating the Standard Attributes of a User's Account

To update a user's standard attributes, send a PUT request to the user ID. The JSON payload must respect the structure for each attribute, as indicated in "Schema Used by the SAP Connector For User Accounts".

The following command updates the `ADDRESS` attribute of user KVAUGHAN:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--header "If-Match: *" \
--request PUT \
--data '{
  "ADDRESS": {
    "FIRSTNAME": "Katie",
    "LASTNAME": "Vaughan",
    "FULLNAME": "Katie Vaughan",
    "FUNCTION": "Administrator",
    "TITLE": "Company",
    "NAME": "EXAMPLE.COM",
    "CITY": "San Francisco",
```

```

    "POSTL_COD1": "94105",
    "STREET": "Sacramento St",
    "HOUSE_NO": "2912",
    "COUNTRY": "US",
    "COUNTRYISO": "US",
    "LANGU": "E",
    "LANGU_ISO": "EN",
    "REGION": "CA",
    "TIME_ZONE": "PST",
    "TEL1_NUMBR": "33297603177",
    "E_MAIL": "katie.vaughan@example.com",
    "LANGU_CR_P": "E",
    "LANGUCPISO": "EN"
  }
}' \
"http://localhost:8080/openidm/system/mysap/user/KVAUGHAN"

```

Resetting a User's Password

To reset the user's password, provide the new password as the value of the `__PASSWORD__` attribute, in a PUT request. The following command resets KVAUGHAN's password to `MyPassw0rd`:

```

curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--header "If-Match: *" \
--request PUT \
--data '{
  "__PASSWORD__": "MyPassw0rd"
}' \
"http://localhost:8080/openidm/system/mysap/user/KVAUGHAN"

```

Note that unless you set the `__PASSWORD_EXPIRED__` attribute to `false`, the user will be required to reset her password the next time she logs into the SAP system.

The following command resets KVAUGHAN's password to `MyPassw0rd`, and ensures that she does not have to reset her password the next time she logs in:

```

curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request PUT \
--data '{
  "__PASSWORD__": "MyPassw0rd",
  "__PASSWORD_EXPIRED__": false
}' \
"http://localhost:8080/openidm/system/mysap/user/KVAUGHAN"

```

Deleting User Accounts

To delete a user account, send a DELETE request to the user ID. The following example deletes KVAUGHAN:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request DELETE \
"http://localhost:8080/openidm/system/mysap/user/KVAUGHAN"
```

The command returns the complete user object that was deleted.

Managing User Profiles

An SAP system uses *profiles* to manage authorization. The following examples demonstrate how to add, change, and remove a user's profiles.

Creating a User With One or More Profiles

Profiles are added as an array of one or more objects.

The following command creates a user BJENSEN, with the system administrator profile (**S_A.SYSTEM**):

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request POST \
--data '{
  "__NAME__": "BJENSEN",
  "PASSWORD": "Passw0rd",
  "PASSWORD_EXPIRED": false,
  "PROFILES": [
    {"BAPIPROF": "S_A.SYSTEM"}
  ]
}' \
"http://localhost:8080/openidm/system/mysap/user/?_action=create"
{
  "id": "BJENSEN",
  "COMPANY": {
    "COMPANY": "SAP AG"
  },
  "PROFILES": [
    {
      "BAPIPROF": "S_A.SYSTEM",
      "BAPIPTXT": "System administrator (Superuser)",
      "BAPITYPE": "S",
      "BAPIAKTPS": "A"
    }
  ],
  ...
  "__NAME__": "BJENSEN"
}
```

Note that the additional information regarding that profile is added to the user account automatically.

Updating a User's Profiles

To update a user's profiles, send a PUT request to the user's ID, specifying the new profiles as an array of values for the **PROFILES** attribute. The values provided in the PUT request will replace the current profiles, so you must include the existing profiles in the request.

The following example adds the **SAP_ALL** profile to user BJENSEN's account:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--header "If-Match: *" \
--request PUT \
--data '{
  "PROFILES": [
    {"BAPIPROF": "S_A.SYSTEM"},
    {"BAPIPROF": "SAP_ALL"}
  ]
}' \
"http://localhost:8080/openidm/system/mysap/user/BJENSEN"
{
  "_id": "BJENSEN",
  "COMPANY": {
    "COMPANY": "SAP AG"
  },
  "PROFILES": [
    {
      "BAPIPROF": "SAP_ALL",
      "BAPIPTXT": "All SAP System authorizations",
      "BAPITYPE": "C",
      "BAPIAKTPS": "A"
    },
    {
      "BAPIPROF": "S_A.SYSTEM",
      "BAPIPTXT": "System administrator (Superuser)",
      "BAPITYPE": "S",
      "BAPIAKTPS": "A"
    }
  ],
  ...
  "__NAME__": "BJENSEN"
}
```

Removing All Profiles From a User Account

To remove all profiles from a user's account, update the account with an empty array. The following example removes all profiles from BJENSEN's account:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--header "If-Match: *" \
--request PUT \
--data '{
  "PROFILES": []
}' \
"http://localhost:8080/openidm/system/mysap/user/BJENSEN"

{"_id": "BJENSEN",
 "COMPANY": {
  "COMPANY": "SAP AG"
 },
 ...
 "__NAME__": "BJENSEN"
}
```

The output shows no **PROFILES** attribute, as this attribute is now empty for this user.

Managing User Roles

SAP user roles (or *activity groups*) are an alternative mechanism to grant authorization to an SAP system. Essentially, a role encapsulates a set of one or more profiles.

Roles can be granted with *temporal constraints*, that is, a period during which the role is valid. If no temporal constraints are specified, the SAP connector sets the FROM date to the current date and the TO date to 9999-12-31.

Creating a User With One or More Profiles

Roles are added as an array of one or more objects.

The following command creates a user SCARTER, with two roles: **SAP_AUDITOR_SA_CCM_USR** and **SAP_ALM_ADMINISTRATOR**. The auditor role has a temporal constraint, and is valid only from May 1st, 2016 to April 30th, 2017. The format of the temporal constraint is **YYYY-mm-dd**:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request POST \
--data '{
  "__NAME__": "SCARTER",
  "PASSWORD": "Passw0rd",
  "PASSWORD_EXPIRED": false,
  "ACTIVITYGROUPS": [
    {
      "AGR_NAME": "SAP_AUDITOR_SA_CCM_USR",
      "FROM_DAT": "2016-05-01",
```

```

        "TO_DAT": "2017-04-30"
      },
      {
        "AGR_NAME": "SAP_ALM_ADMINISTRATOR"
      }
    ]
  },
  \
  "http://localhost:8080/openidm/system/mysap/user/?_action=create"
{
  "id": "SCARTER",
  "COMPANY": {
    "COMPANY": "SAP AG"
  },
  "PROFILES": [
    {
      "BAPIPROF": "T_ALM_CONF",
      "BAPIPTXT": "Profile for the Role SAP_ALM_ADMINISTRATOR",
      "BAPITYPE": "G",
      "BAPIAKTPS": "A"
    }
  ],
  ...
  "ACTIVITYGROUPS": [
    {
      "AGR_NAME": "SAP_ALM_ADMINISTRATOR",
      "FROM_DAT": "2016-04-20",
      "TO_DAT": "9999-12-31",
      "AGR_TEXT": "Alert Management Administrator"
    },
    {
      "AGR_NAME": "SAP_AUDITOR_SA_CCM_USR",
      "FROM_DAT": "2016-05-01",
      "TO_DAT": "2017-04-30",
      "AGR_TEXT": "AIS - System Audit - Users and Authorizations"
    }
  ],
  "__NAME__": "SCARTER"
}

```

When a role is granted, the corresponding profiles are attached to the user account automatically.

Updating a User's Roles

To update a user's roles, send a PUT request to the user's ID, specifying the new roles as an array of values of the **ACTIVITYGROUPS** attribute. The values provided in the PUT request will replace the current **ACTIVITYGROUPS**.

The following example removes the **SAP_AUDITOR_SA_CCM_USR** role and changes the temporal constraints on the **SAP_ALM_ADMINISTRATOR** role for SCARTER's account:

```

curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--header "If-Match: *" \

```

```
--request PUT \
--data '{
  "ACTIVITYGROUPS": [
    {
      "AGR_NAME": "SAP_ALM_ADMINISTRATOR",
      "FROM_DAT": "2015-06-02",
      "TO_DAT": "2016-06-02"
    }
  ]
}' \
"http://localhost:8080/openidm/system/mysap/user/SCARTER"
{
  "_id": "SCARTER",
  "COMPANY": {
    "COMPANY": "SAP AG"
  },
  "PROFILES": [
    {
      "BAPIPROF": "T_ALM_CONF",
      "BAPIPTXT": "Profile for the Role SAP_ALM_ADMINISTRATOR",
      "BAPITYPE": "G",
      "BAPIAKTPS": "A"
    }
  ],
  ...
  "ACTIVITYGROUPS": [
    {
      "AGR_NAME": "SAP_ALM_ADMINISTRATOR",
      "FROM_DAT": "2015-06-02",
      "TO_DAT": "2016-06-02",
      "AGR_TEXT": "Alert Management Administrator"
    }
  ],
  "__NAME__": "SCARTER"
}
```

Removing All Roles From a User Account

To remove all roles from a user's account, update the value of the **ACTIVITYGROUPS** attribute with an empty array. The following example removes all roles from SCARTER's account:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--header "If-Match: *" \
--request PUT \
--data '{
  "ACTIVITYGROUPS": []
}' \
"http://localhost:8080/openidm/system/mysap/user/SCARTER"
{
  "_id": "SCARTER",
  "COMPANY": {
    "COMPANY": "SAP AG"
  },
  ...
  "LASTMODIFIED": {
    "MODDATE": "2016-04-21",
    "MODTIME": "04:27:00"
  },
  "__NAME__": "SCARTER"
}
```

The output shows no **ACTIVITYGROUPS** attribute, as this attribute is now empty.

Configuring the SAP Connector For SNC

The SAP connector supports an SNC (Secure Network Connection) configuration. SNC is a software layer in the SAP System architecture that provides an interface to an external security product.

For a list of the configuration properties specific to SNC, see "SAP Secure Network Connection Configuration".

Implementation Specifics

For PATCH requests, a connector can potentially add, remove, or replace an attribute value. The SAP connector implements the add, remove, and replace operations but the sample scripts provided with the connector implement only the replace operation. If you use these sample scripts, a PATCH request will therefore always replace the entire attribute value with the new value.

Setting Productive Passwords on the SAP System

Synchronization of passwords to the SAP system *requires* that you configure SNC and SSO. If you do not configure these two elements correctly, passwords that are updated by IDM are set as *initial* passwords rather than *productive* passwords, and users are forced to change their passwords on login.

1. To configure the SAP connector to use SNC, set the **sncMode** property to **"1"**.

To configure the connector to use SSO with SNC, set the `sncSSO` property to `"1"`.

2. The logon session during which a productive password is set must be secured using the authentication method Single Sign-On (SSO) using Secure Network Communications (SNC). IDM must request and receive an SSO logon ticket from the SAP system to allow the `BAPI_USER_CHANGE` process to set a productive password. For more information, see [SAP Note 1287410](#).

To configure the connector to request this logon ticket, set the value of the `x509Cert` property as follows:

- If you are using an X509 certificate to negotiate with the SAP server, set the `x509Cert` property to the base 64-encoded certificate.

Note that the certificate must be a valid, CA-signed certificate. You cannot use a self-signed certificate here.

- If you are not using an X509 certificate to negotiate with the SAP server, set the `x509Cert` property to `null`.

In this case, the connector will use the `user` and `password` specified in the connector configuration to request the SSO logon ticket.

OpenICF Interfaces Implemented by the SAP Connector

The SAP Connector implements the following OpenICF interfaces.

Authenticate

Provides simple authentication with two parameters, presumed to be a user name and password.

Create

Creates an object and its `uid`.

Delete

Deletes an object, referenced by its `uid`.

Resolve Username

Resolves an object by its username and returns the `uid` of the object.

Schema

Describes the object types, operations, and options that the connector supports.

Script on Connector

Enables an application to run a script in the context of the connector. Any script that runs on the connector has the following characteristics:

- The script runs in the same execution environment as the connector and has access to all the classes to which the connector has access.
- The script has access to a `connector` variable that is equivalent to an initialized instance of the connector. At a minimum, the script can access the connector configuration.
- The script has access to any script-arguments passed in by the application.

Script on Resource

Runs a script on the target resource that is managed by this connector.

Search

Searches the target resource for all objects that match the specified object class and filter.

Sync

Polls the target resource for synchronization events, that is, native changes to objects on the target resource.

Test

Tests the connector configuration. Testing a configuration checks all elements of the environment that are referred to by the configuration are available. For example, the connector might make a physical connection to a host that is specified in the configuration to verify that it exists and that the credentials that are specified in the configuration are valid.

This operation might need to connect to a resource, and, as such, might take some time. Do not invoke this operation too often, such as before every provisioning operation. The test operation is not intended to check that the connector is alive (that is, that its physical connection to the resource has not timed out).

You can invoke the test operation before a connector configuration has been validated.

Update

Updates (modifies or replaces) objects on a target resource.

SAP Connector Configuration

The SAP Connector has the following configurable properties.

Operation Script Files

Property	Type	Default	Encrypted ^a	Required ^b
<code>createScriptFileName</code>	String	null		Create
The name of the file used to perform the CREATE operation.				
<code>customizerScriptFileName</code>	String	null		No

Property	Type	Default	Encrypted ^a	Required ^b
The script used to customize some function of the connector. Read the documentation for more details.				
resolveUsernameScriptFileName	String	null		Resolve Username
The name of the file used to perform the RESOLVE_USERNAME operation.				
updateScriptFileName	String	null		Update
The name of the file used to perform the UPDATE operation.				
schemaScriptFileName	String	null		Schema
The name of the file used to perform the SCHEMA operation.				
authenticateScriptFileName	String	null		Authenticate
The name of the file used to perform the AUTHENTICATE operation.				
scriptOnResourceScriptFileName	String	null		Script On Resource
The name of the file used to perform the RUNSCRIPTONRESOURCE operation.				
deleteScriptFileName	String	null		Delete
The name of the file used to perform the DELETE operation.				
searchScriptFileName	String	null		Get Search
The name of the file used to perform the SEARCH operation.				
testScriptFileName	String	null		Test
The name of the file used to perform the TEST operation.				
syncScriptFileName	String	null		Sync
The name of the file used to perform the SYNC operation.				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

Groovy Engine configuration

Property	Type	Default	Encrypted ^a	Required ^b
targetDirectory	File	null		No
Directory into which to write classes.				
warningLevel	int	1		No
Warning Level of the compiler				

Property	Type	Default	Encrypted ^a	Required ^b
<code>scriptExtensions</code>	<code>String[]</code>	<code>['groovy']</code>		No
Gets the extensions used to find groovy files				
<code>scriptBaseClass</code>	<code>String</code>	<code>null</code>		No
Base class name for scripts (must derive from Script)				
<code>scriptRoots</code>	<code>String[]</code>	<code>null</code>		Yes
The root folder to load the scripts from. If the value is null or empty the classpath value is used.				
<code>tolerance</code>	<code>int</code>	<code>10</code>		No
The error tolerance, which is the number of non-fatal errors (per unit) that should be tolerated before compilation is aborted.				
<code>disabledGlobalASTTransformations</code>	<code>String[]</code>	<code>null</code>		No
Sets a list of global AST transformations which should not be loaded even if they are defined in META-INF/org.codehaus.groovy.transform.ASTTransformation files. By default, none is disabled.				
<code>sourceEncoding</code>	<code>String</code>	<code>UTF-8</code>		No
Encoding for source files				
<code>recompileGroovySource</code>	<code>boolean</code>	<code>false</code>		No
If set to true recompilation is enabled				
<code>minimumRecompilationInterval</code>	<code>int</code>	<code>100</code>		No
Sets the minimum of time after a script can be recompiled.				
<code>debug</code>	<code>boolean</code>	<code>false</code>		No
If true, debugging code should be activated				
<code>classpath</code>	<code>String[]</code>	<code>[]</code>		No
Classpath for use during compilation.				
<code>verbose</code>	<code>boolean</code>	<code>false</code>		No
If true, the compiler should produce action information				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

Configuration properties

Property	Type	Default	Encrypted ^a	Required ^b
<code>customSensitiveConfiguration</code>	<code>GuardedString</code>	<code>null</code>	Yes	No

Property	Type	Default	Encrypted ^a	Required ^b
Custom Sensitive Configuration script for Groovy ConfigSlurper				
<code>customConfiguration</code>	String	null		No
Custom Configuration script for Groovy ConfigSlurper				
<code>x509Cert</code>	String	null	Yes	No
The X509 certificate supplied for authentication.				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

Basic Configuration

Property	Type	Default	Encrypted ^a	Required ^b
<code>asHost</code>	String	null		Yes
The FQDN of your SAP Application Server, for example sap.example.com				
<code>gwHost</code>	String	null		Yes
SAP gateway host name				
<code>gwServ</code>	String	null		Yes
SAP gateway service				
<code>user</code>	String	null		Yes
SAP Logon user				
<code>password</code>	GuardedString	null	Yes	Yes
SAP Logon password				
<code>client</code>	String	000		Yes
SAP client				
<code>systemNumber</code>	String	00		Yes
SAP system number				
<code>language</code>	String	EN		Yes
SAP Logon language				
<code>destination</code>	String	OPENIDM		Yes
SAP JCo destination name				
<code>directConnection</code>	boolean	true		Yes

Property	Type	Default	Encrypted ^a	Required ^b
If true, direct connection to an SAP ABAP Application server or SAP router. If false connection to a group of SAP instances through an SAP message server				
sapRouter	String	null		Yes
SAP router string to use for a system protected by a firewall. (/H/host[/S/port])				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

SAP Jco Logs Configuration

Property	Type	Default	Encrypted ^a	Required ^b
trace	String	0		No
Enable/disable RFC trace (0 or 1)				
cpicTrace	String	0		No
Enable/disable CPIC trace [0..3]				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

Advanced Configuration

Property	Type	Default	Encrypted ^a	Required ^b
msHost	String	null		No
Specifies the host that the message server is running on				
group	String	null		No
Specifies the group name of the application servers, used when you log in to a logon group that uses load balancing				
msServ	String	null		No
Name of the service where the message server can be reached				
r3Name	String	null		No
Specifies the name of the SAP system, used when you log in to a logon group that uses load balancing				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

SAP Secure Network Connection Configuration

Property	Type	Default	Encrypted ^a	Required ^b
sncMode	String	0		Yes
Flag used to activate SNC. Possible values are 0 (OFF) and 1 (ON).				
sncQoP	String	3		No
Specifies the security level to use for the connection. Possible values are 1 - Authentication only, 2 - Integrity protection, 3 - Privacy protection, 8 - Use the value from snc/data_protection/use on the application server, 9 - Use the value from snc/data_protection/max on the application server				
sncLibrary	String	null		No
Specifies the path to the external library that provides Secure Network Connection service. The default is the system-defined library as defined in the environment variable SNC_LIB.				
sncPartnerName	String	null		No
Specifies the AS ABAP SNC name, for example, "p:CN=ABC, O=MyCompany, C=US". You can find the application server SNC name in the profile parameter snc/identity/as on the AS ABAP.				
sncMyName	String	null		No
Specifies the connector SNC name, for example, "p:CN=OpenIDM, O=MyCompany, C=US". This parameter is optional, but you should set it to make sure that the correct SNC name is used for the connection.				
sncSSO	String	0		No
Specifies whether the connection should be configured for single sign-on (SSO). Possible values are 0 (OFF) and 1 (ON).				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

JCo Connection Pool Configuration

Property	Type	Default	Encrypted ^a	Required ^b
poolCapacity	String	1		No
Maximum number of idle connections kept open by the destination. 0 = no connection pooling. Default is 1.				
expirationTime	String	60000		No
Time in ms after that a free connection can be closed. Default is one minute.				
maxGetTime	String	30000		No
Maximum time in ms to wait for a connection, if the maximum allowed number of connections is allocated by the pool. Default is 30 seconds.				
peakLimit	String	0		No

Property	Type	Default	Encrypted ^a	Required ^b
Maximum number of active connections that can be created for a destination simultaneously. The default is 0 (unlimited).				
<code>expirationPeriod</code>	String	60000		No
Period in ms after that the destination checks the released connections for expiration. Default is one minute				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

SCIM Connector

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

The SCIM connector is based on the Simple Cloud Identity Management (SCIM) protocol and enables you to manage user and group accounts on any SCIM-compliant resource provider, such as Slack or Facebook. The SCIM connector implements both 1.1 and 2.0 endpoints. The SCIM connector is bundled with IDM in the `connectors/` directory.

The SCIM connector uses the Apache HTTP client, which leverages the HTTP client connection pool, not the ICF connector pool.

+ To Configure the SCIM Connector Using the Filesystem

1. Copy `/path/to/openidm/samples/example-configurations/provisioners/provisioner.openicf-scim.json` to your project's `conf/` directory.
2. Edit `conf/provisioner.openicf-scim.json`, as necessary. The following changes are required:

- `"enabled" : true`
- To specify the connection details to the SCIM resource provider, set the `configurationProperties`. The required properties vary, based on the `authenticationMethod`:

OAuth

The minimum required properties are `grantType`, `SCIMEndpoint`, `tokenEndpoint`, `clientId`, and `clientSecret`.

BASIC

The minimum required properties are `user` and `password`.

TOKEN

The minimum required property is **authToken**.

+ Sample Configuration Using OAUTH

```
"configurationProperties" : {  
  "SCIMEndpoint" : "https://example.com/scim",  
  "SCIMVersion" : 1,  
  "authenticationMethod" : "OAUTH",  
  "user" : null,  
  "password" : null,  
  "tokenEndpoint" : "https://example.com/oauth2/token",  
  "clientId" : "Kdvl.....j3fka",  
  "clientSecret" : "xxxxxxxxxxxxxxxxxx",  
  "acceptSelfSignedCertificates" : true,  
  "grantType" : "client_credentials",  
  "disableHostNameVerifier" : true,  
  "maximumConnections" : 10,  
  "httpProxyHost" : null,  
  "httpProxyPort" : null  
}
```

Note

On startup, IDM encrypts the value of the **clientSecret**.

After the connector is properly configured, you can test its status:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system?_action=test"
[
  {
    "name": "SCIM",
    "enabled": true,
    "config": "config/provisioner.openicf/SCIM",
    "connectorRef": {
      "bundleName": "org.forgerock.openicf.connectors.scim-connector",
      "connectorName": "org.forgerock.openicf.connectors.scim.ScimConnector",
      "bundleVersion": "1.5.20.12"
    },
    "displayName": "Scim Connector",
    "objectTypes": [
      "ACCOUNT",
      "ALL",
      "GROUP"
    ],
    "ok": true
  }
]
```

A status of `"ok": true` indicates that the SCIM connector can reach the configured resource provider.

Implementation Specifics

For PATCH requests, a connector can potentially add, remove, or replace an attribute value. The SCIM connector does not implement the add or remove operations, so a PATCH request always replaces the entire attribute value with the new value.

Using the SCIM Connector With a Proxy Server

If the IDM server is hosted behind a firewall and requests to the resource provider are routed through a proxy, you must specify the proxy host and port in the connector configuration.

To specify the proxy server details, set the `httpProxyHost`, and `httpProxyPort` properties in the connector configuration. For example:

```
"configurationProperties": {
  ...
  "httpProxyHost": "myproxy.home.com",
  "httpProxyPort": 8080,
  ...
},
```

OpenICF Interfaces Implemented by the Scim Connector

The Scim Connector implements the following OpenICF interfaces.

Create

Creates an object and its `uid`.

Delete

Deletes an object, referenced by its `uid`.

Schema

Describes the object types, operations, and options that the connector supports.

Script on Connector

Enables an application to run a script in the context of the connector. Any script that runs on the connector has the following characteristics:

- The script runs in the same execution environment as the connector and has access to all the classes to which the connector has access.
- The script has access to a `connector` variable that is equivalent to an initialized instance of the connector. At a minimum, the script can access the connector configuration.
- The script has access to any script-arguments passed in by the application.

Search

Searches the target resource for all objects that match the specified object class and filter.

Sync

Polls the target resource for synchronization events, that is, native changes to objects on the target resource.

Test

Tests the connector configuration. Testing a configuration checks all elements of the environment that are referred to by the configuration are available. For example, the connector might make a physical connection to a host that is specified in the configuration to verify that it exists and that the credentials that are specified in the configuration are valid.

This operation might need to connect to a resource, and, as such, might take some time. Do not invoke this operation too often, such as before every provisioning operation. The test operation is not intended to check that the connector is alive (that is, that its physical connection to the resource has not timed out).

You can invoke the test operation before a connector configuration has been validated.

Update

Updates (modifies or replaces) objects on a target resource.

Scim Connector Configuration

The Scim Connector has the following configurable properties.

Basic Configuration Properties

Property	Type	Default	Encrypted ^a	Required ^b
SCIMEndpoint	String	null		Yes
The HTTP URL defining the root for the SCIM endpoint (https://myserver.com/service/scim)				
SCIMVersion	Integer	1		Yes
Defines the SCIM protocol version. Values can be either 1 or 2. Default is 1				
authenticationMethod	String	OAuth		Yes
Defines which method is to be used to authenticate on the remote server. Options are BASIC (username/password), OAuth (Client id/secret) or TOKEN (static token). Defaults to OAuth				
user	String	null		Yes
In case of BASIC authentication type, this property defines the remote user.				
password	GuardedString	null	Yes	No
In case of BASIC authentication type, this property defines the remote password.				
tokenEndpoint	String	null		No
When using OAuth, this property defines the endpoint where a new access token should be requested (https://myserver.com/oauth2/token)				
clientId	String	null		Yes
Secure client identifier for OAuth2				
clientSecret	GuardedString	null	Yes	No
Secure client secret for OAuth2				
authToken	GuardedString	null	Yes	No
Some service providers (Slack for instance) use static authentication tokens.				
refreshToken	GuardedString	null		Yes
Used by the refresh_token grant type				
grantType	String	null		No
The OAuth2 grant type to use (client_credentials or refresh_token)				
scope	String	null		No
The OAuth2 scope to use.				

Property	Type	Default	Encrypted ^a	Required ^b
<code>acceptSelfSignedCertificates</code>	boolean	false		Yes
To be used for debug/test purposes. To be avoided in production. Defaults to false.				
<code>disableHostNameVerifier</code>	boolean	false		Yes
To be used for debug/test purposes. To be avoided in production. Defaults to false.				
<code>disableHttpCompression</code>	boolean	false		Yes
Content compression is enabled by default. Set this property to true to disable it. Defaults to false.				
<code>clientCertAlias</code>	String	null		Yes
If TLS Mutual Auth is needed, set this to the certificate alias from the keystore.				
<code>clientCertPassword</code>	GuardedString	null	Yes	Yes
If TLS Mutual Auth is needed and the client certificate (private key) password is different than the keystore password, set this to the client private key password.				
<code>maximumConnections</code>	Integer	10		Yes
Defines the max size of the http connection pool used. Defaults to 10.				
<code>httpProxyHost</code>	String	null		Yes
Defines the Hostname if an HTTP proxy is used between the connector and the SCIM service provider. Defaults to null.				
<code>httpProxyPort</code>	Integer	null		Yes
Defines the Port if an HTTP proxy is used between the connector and the SCIM service provider. Defaults to null.				
<code>httpProxyUsername</code>	String	null		Yes
Defines Proxy Username if an HTTP proxy is used between the connector and the SCIM service provider. Defaults to null.				
<code>httpProxyPassword</code>	GuardedString	null	Yes	Yes
Defines Proxy Password if an HTTP proxy is used between the connector and the SCIM service provider. Defaults to null.				
<code>connectionTimeout</code>	int	30		No
Defines a timeout for the underlying http connection in seconds. Defaults to 30.				
<code>authorizationTokenPrefix</code>	String	Bearer		No
The prefix to be used in the Authorization HTTP header for Token authentication. Defaults to "Bearer".				
<code>useBasicAuthForOAuthTokenNeg</code>	boolean	true		Yes
Description is not available				

Property	Type	Default	Encrypted ^a	Required ^b
<code>readRateLimit</code>	String	null		No
Define throttling for read operations either per seconds ("30/sec") or per minute ("100/min").				
<code>acceptHeader</code>	String	null		No
The connector is using "application/json" by default. SCIM V2 Service Provider may require "application/scim+json". It can be overwritten with this property				
<code>contentTypeHeader</code>	String	null		No
The connector is using "application/json" by default. SCIM V2 Service Provider may require "application/scim+json". It can be overwritten with this property				
<code>writeRateLimit</code>	String	null		No
Define throttling for write operations (create/update/delete) either per second ("30/sec") or per minute ("100/min").				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

Scripted REST Connector

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

The Scripted REST connector is an implementation of the Scripted Groovy Connector Toolkit. This connector enables you to interact with any REST API, using Groovy scripts for the ICF operations.

Note

The Scripted REST connector is not a *poolable* connector.

Configuring the Scripted REST Connector

The Scripted REST Connector is bundled in the JAR `openidm/connectors/scriptedrest-connector-1.5.20.11.jar`.

A sample connector configuration and scripts are provided in the `/path/to/openidm/samples/scripted-rest-with-dj/` directory and described in "*Connect to DS With ScriptedREST*" in the *Samples Guide*. The scripts provided with this sample demonstrate how the connector can be used but most likely cannot be used as is in your deployment. They are a good starting point on which to base your customization. For information about writing your own scripts, see "*Writing Scripted Connectors With the Groovy Connector Toolkit*" in the *Connector Developer's Guide*.

Using the Scripted REST Connector With a Proxy Server

If the IDM server is hosted behind a firewall and requests to the resource are routed through a proxy, you must specify the proxy host and port in the connector configuration.

To specify the proxy server details, set the `proxyAddress` property in the connector configuration. For example:

```
"configurationProperties": {  
  ...  
  "proxyAddress": "http://myproxy:8080",  
  ...  
}
```

Run scripts through the connector

Groovy Toolkit connectors have two operations that allow you to run arbitrary script actions: `runScriptOnConnector` and `runScriptOnResource`. `runScriptOnConnector` is an operation that sends the script action to the connector to be compiled and executed. `runScriptOnResource` is an operation that sends the script to another script to be handled.

`runScriptOnConnector`

The `runScriptOnConnector` script lets you run an arbitrary script action through the connector. This script takes the following variables as input:

`configuration`

A handler to the connector's configuration object.

`options`

A handler to the Operation Options.

`operation`

The operation type that corresponds to the action.

`log`

A handler to the connector's log.

To run an arbitrary script on a Groovy Toolkit connector, define the script in the `systemActions` property of your provisioner file:

```
"systemActions" : [
  {
    "scriptId" : "MyScript",
    "actions" : [
      {
        "systemType" : ".*ScriptedConnector",
        "actionType" : "groovy",
        "actionFile" : "path/to/scriptname.groovy"
      }
    ]
  }
]
```

If you wish to define your script in the provisioner file itself rather than in a separate file, you can use the `actionSource` property instead of the `actionFile` one. A simple example follows:

```
"systemActions" : [
  {
    "scriptId" : "MyScript",
    "actions" : [
      {
        "systemType" : ".*ScriptedConnector",
        "actionType" : "groovy",
        "actionSource" : "2 * 2"
      }
    ]
  }
]
```

Note

It is optional to prepend the last script statement in `actionSource` with `return`.

Running `MyScript` will return:

```
{
  "actions" : [
    {
      "result": 4
    }
  ]
}
```

If your script accepts parameters, you can supply them in the request body or the query string. For example:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
--data-raw '{"param1":"value1"}' \
"http://localhost:8080/openidm/system/groovy?_action=script&scriptId=MyScript&param2=value2"
```

You can also call it through the script engine. The system can accept arbitrary parameters:

```
openidm.action("/system/groovy", "script", {"contentParameter": "value"}, {"scriptId": "MyScript",
"additionalParameter1": "value1", "additionalParameter2": "value2"})
```

runScriptOnResource

To run an arbitrary script using `runScriptOnResource`, you must add some configuration details to your provisioner file. These details include a `scriptOnResourceScriptFileName` that references a script file located in a path contained in the `scriptRoots` array.

Define these properties in your provisioner file:

```
"configurationProperties": {
  "scriptRoots": [
    "path/to/scripts"
  ],
  "scriptOnResourceScriptFileName": "ScriptOnResourceScript.groovy"
},
"systemActions" : [
  {
    "scriptId" : "script-1",
    "actions" : [
      {
        "systemType" : ".*ScriptedConnector",
        "actionType" : "groovy",
        "actionFile" : "path/to/scriptname.groovy"
      }
    ]
  }
]
```

When you have defined the script, call it over REST on the system endpoint:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system/groovy?
_action=script&scriptId=scriptOnResourceScript&scriptExecuteMode=resource"
```

Implemented Interfaces

The following table lists the ICF interfaces that are implemented for the scripted REST connector:

OpenICF Interfaces Implemented by the Scripted REST Connector

The Scripted REST Connector implements the following OpenICF interfaces.

Authenticate

Provides simple authentication with two parameters, presumed to be a user name and password.

Create

Creates an object and its `uid`.

Delete

Deletes an object, referenced by its `uid`.

Resolve Username

Resolves an object by its username and returns the `uid` of the object.

Schema

Describes the object types, operations, and options that the connector supports.

Script on Connector

Enables an application to run a script in the context of the connector. Any script that runs on the connector has the following characteristics:

- The script runs in the same execution environment as the connector and has access to all the classes to which the connector has access.
- The script has access to a `connector` variable that is equivalent to an initialized instance of the connector. At a minimum, the script can access the connector configuration.
- The script has access to any script-arguments passed in by the application.

Script on Resource

Runs a script on the target resource that is managed by this connector.

Search

Searches the target resource for all objects that match the specified object class and filter.

Sync

Polls the target resource for synchronization events, that is, native changes to objects on the target resource.

Test

Tests the connector configuration. Testing a configuration checks all elements of the environment that are referred to by the configuration are available. For example, the connector might make a physical connection to a host that is specified in the configuration to verify that it exists and that the credentials that are specified in the configuration are valid.

This operation might need to connect to a resource, and, as such, might take some time. Do not invoke this operation too often, such as before every provisioning operation. The test operation is not intended to check that the connector is alive (that is, that its physical connection to the resource has not timed out).

You can invoke the test operation before a connector configuration has been validated.

Update

Updates (modifies or replaces) objects on a target resource.

Configuration Properties

The following table lists the configuration properties for the scripted REST connector:

Scripted REST Connector Configuration

The Scripted REST Connector has the following configurable properties.

Configuration properties

Property	Type	Default	Encrypted ^a	Required ^b
<code>customSensitiveConfiguration</code>	<code>GuardedString</code>	<code>null</code>	Yes	No
Custom Sensitive Configuration script for Groovy ConfigSlurper				
<code>customConfiguration</code>	<code>String</code>	<code>null</code>		No
Custom Configuration script for Groovy ConfigSlurper				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

Operation Script Files

Property	Type	Default	Encrypted ^a	Required ^b
<code>createScriptFileName</code>	<code>String</code>	<code>null</code>		Create
The name of the file used to perform the CREATE operation.				
<code>customizerScriptFileName</code>	<code>String</code>	<code>null</code>		No
The script used to customize some function of the connector. Read the documentation for more details.				
<code>authenticateScriptFileName</code>	<code>String</code>	<code>null</code>		Authenticate
The name of the file used to perform the AUTHENTICATE operation.				
<code>scriptOnResourceScriptFileName</code>	<code>String</code>	<code>null</code>		Script On Resource
The name of the file used to perform the RUNSCRIPTONRESOURCE operation.				
<code>deleteScriptFileName</code>	<code>String</code>	<code>null</code>		Delete
The name of the file used to perform the DELETE operation.				

Property	Type	Default	Encrypted ^a	Required ^b
<code>resolveUsernameScriptFileName</code>	String	null		Resolve Username
The name of the file used to perform the RESOLVE_USERNAME operation.				
<code>searchScriptFileName</code>	String	null		Get Search
The name of the file used to perform the SEARCH operation.				
<code>updateScriptFileName</code>	String	null		Update
The name of the file used to perform the UPDATE operation.				
<code>schemaScriptFileName</code>	String	null		Schema
The name of the file used to perform the SCHEMA operation.				
<code>testScriptFileName</code>	String	null		Test
The name of the file used to perform the TEST operation.				
<code>syncScriptFileName</code>	String	null		Sync
The name of the file used to perform the SYNC operation.				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

Groovy Engine configuration

Property	Type	Default	Encrypted ^a	Required ^b
<code>targetDirectory</code>	File	null		No
Directory into which to write classes.				
<code>warningLevel</code>	int	1		No
Warning Level of the compiler				
<code>scriptExtensions</code>	String[]	['groovy']		No
Gets the extensions used to find groovy files				
<code>minimumRecompilationInterval</code>	int	100		No
Sets the minimum of time after a script can be recompiled.				
<code>scriptBaseClass</code>	String	null		No
Base class name for scripts (must derive from Script)				
<code>scriptRoots</code>	String[]	null		Yes
The root folder to load the scripts from. If the value is null or empty the classpath value is used.				

Property	Type	Default	Encrypted ^a	Required ^b
<code>tolerance</code>	<code>int</code>	<code>10</code>		No
The error tolerance, which is the number of non-fatal errors (per unit) that should be tolerated before compilation is aborted.				
<code>debug</code>	<code>boolean</code>	<code>false</code>		No
If true, debugging code should be activated				
<code>classpath</code>	<code>String[]</code>	<code>[]</code>		No
Classpath for use during compilation.				
<code>disabledGlobalASTTransformations</code>	<code>String[]</code>	<code>null</code>		No
Sets a list of global AST transformations which should not be loaded even if they are defined in META-INF/org.codehaus.groovy.transform.ASTTransformation files. By default, none is disabled.				
<code>verbose</code>	<code>boolean</code>	<code>false</code>		No
If true, the compiler should produce action information				
<code>sourceEncoding</code>	<code>String</code>	<code>UTF-8</code>		No
Encoding for source files				
<code>recompileGroovySource</code>	<code>boolean</code>	<code>false</code>		No
If set to true recompilation is enabled				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

Basic Configuration Properties

Property	Type	Default	Encrypted ^a	Required ^b
<code>username</code>	<code>String</code>	<code>null</code>		No
The Remote user to authenticate with				
<code>password</code>	<code>GuardedString</code>	<code>null</code>	Yes	No
The Password to authenticate with				
<code>serviceAddress</code>	<code>URI</code>	<code>null</code>		Yes
The service URI (example: <code>http://myservice.com/api</code>)				
<code>proxyAddress</code>	<code>URI</code>	<code>null</code>		No
The optional Proxy server URI (example: <code>http://myproxy:8080</code>)				
<code>proxyUsername</code>	<code>String</code>	<code>null</code>		No

Property	Type	Default	Encrypted ^a	Required ^b
The username to authenticate with the proxy server				
proxyPassword	GuardedString	null	Yes	No
The password to authenticate with the proxy server				
defaultAuthMethod	String	BASIC		No
Authentication method used. Defaults to BASIC.				
defaultContentType	String	application/ json		No
Default HTTP request content type. Defaults to JSON. Can be: TEXT, XML, HTML, URLENC, BINARY				
defaultRequestHeaders	String[]	null		No
Placeholder for default HTTP request headers.				
OAuthTokenEndpoint	URI	null		No
When using OAUTH, this property defines the endpoint where a new access token should be queried for (https://myserver.com/oauth2/token)				
OAuthClientId	String	null		No
The client identifier				
OAuthClientSecret	GuardedString	null	Yes	No
Secure client secret for OAUTH				
OAuthRefreshToken	GuardedString	null	Yes	No
The refresh token used to renew the access token for the refresh_token grant type				
OAuthScope	String	null		No
The optional scope				
OAuthGrantType	String	CLIENT_ CREDENTIALS		No
The grant type to use. Can be CLIENT_CREDENTIALS (default) REFRESH_TOKEN AUTHORIZATION_CODE				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

Scripted SQL Connector

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

The Scripted SQL connector is an implementation of the Scripted Groovy Connector Toolkit. This connector enables you to interact with any SQL database, using Groovy scripts for the ICF operations.

Configuring the Scripted SQL Connector

The Scripted SQL Connector is bundled in the JAR `openidm/connectors/scriptedsql-connector-1.5.20.8.jar`.

A sample connector configuration and scripts are provided in the `/path/to/openidm/samples/scripted-sql-with-mysql/` directory and described in "*Connect to a MySQL Database With ScriptedSQL*" in the *Samples Guide*. The scripts provided with this sample demonstrate how the connector can be used but most likely cannot be used as is in your deployment. They are a good starting point on which to base your customization. For information about writing your own scripts, see "*Writing Scripted Connectors With the Groovy Connector Toolkit*" in the *Connector Developer's Guide*.

Run scripts through the connector

Groovy Toolkit connectors have two operations that allow you to run arbitrary script actions: `runScriptOnConnector` and `runScriptOnResource`. `runScriptOnConnector` is an operation that sends the script action to the connector to be compiled and executed. `runScriptOnResource` is an operation that sends the script to another script to be handled.

`runScriptOnConnector`

The `runScriptOnConnector` script lets you run an arbitrary script action through the connector. This script takes the following variables as input:

configuration

A handler to the connector's configuration object.

options

A handler to the Operation Options.

operation

The operation type that corresponds to the action.

Log

A handler to the connector's log.

To run an arbitrary script on a Groovy Toolkit connector, define the script in the `systemActions` property of your provisioner file:

```
"systemActions" : [
  {
    "scriptId" : "MyScript",
    "actions" : [
      {
        "systemType" : ".*ScriptedConnector",
        "actionType" : "groovy",
        "actionFile" : "path/to/scriptname.groovy"
      }
    ]
  }
]
```

If you wish to define your script in the provisioner file itself rather than in a separate file, you can use the `actionSource` property instead of the `actionFile` one. A simple example follows:

```
"systemActions" : [
  {
    "scriptId" : "MyScript",
    "actions" : [
      {
        "systemType" : ".*ScriptedConnector",
        "actionType" : "groovy",
        "actionSource" : "2 * 2"
      }
    ]
  }
]
```

Note

It is optional to prepend the last script statement in `actionSource` with `return`.

Running `MyScript` will return:

```
{
  "actions" : [
    {
      "result": 4
    }
  ]
}
```

If your script accepts parameters, you can supply them in the request body or the query string. For example:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
--data-raw '{"param1":"value1"}' \
"http://localhost:8080/openidm/system/groovy?_action=script&scriptId=MyScript&param2=value2"
```

You can also call it through the script engine. The system can accept arbitrary parameters:

```
openidm.action("/system/groovy", "script", {"contentParameter": "value"}, {"scriptId": "MyScript",
"additionalParameter1": "value1", "additionalParameter2": "value2"})
```

runScriptOnResource

To run an arbitrary script using `runScriptOnResource`, you must add some configuration details to your provisioner file. These details include a `scriptOnResourceScriptFileName` that references a script file located in a path contained in the `scriptRoots` array.

Define these properties in your provisioner file:

```
"configurationProperties": {
  "scriptRoots": [
    "path/to/scripts"
  ],
  "scriptOnResourceScriptFileName": "ScriptOnResourceScript.groovy"
},
"systemActions" : [
  {
    "scriptId" : "script-1",
    "actions" : [
      {
        "systemType" : ".*ScriptedConnector",
        "actionType" : "groovy",
        "actionFile" : "path/to/scriptname.groovy"
      }
    ]
  }
]
```

When you have defined the script, call it over REST on the system endpoint:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system/groovy?_action=script&scriptId=scriptOnResourceScript&scriptExecuteMode=resource"
```

Implemented Interfaces

The following table lists the ICF interfaces that are implemented for the scripted SQL connector:

OpenICF Interfaces Implemented by the Scripted SQL Connector

The Scripted SQL Connector implements the following OpenICF interfaces.

Authenticate

Provides simple authentication with two parameters, presumed to be a user name and password.

Create

Creates an object and its `uid`.

Delete

Deletes an object, referenced by its `uid`.

Resolve Username

Resolves an object by its username and returns the `uid` of the object.

Schema

Describes the object types, operations, and options that the connector supports.

Script on Connector

Enables an application to run a script in the context of the connector. Any script that runs on the connector has the following characteristics:

- The script runs in the same execution environment as the connector and has access to all the classes to which the connector has access.
- The script has access to a `connector` variable that is equivalent to an initialized instance of the connector. At a minimum, the script can access the connector configuration.
- The script has access to any script-arguments passed in by the application.

Script on Resource

Runs a script on the target resource that is managed by this connector.

Search

Searches the target resource for all objects that match the specified object class and filter.

Sync

Polls the target resource for synchronization events, that is, native changes to objects on the target resource.

Test

Tests the connector configuration. Testing a configuration checks all elements of the environment that are referred to by the configuration are available. For example, the connector might make a

physical connection to a host that is specified in the configuration to verify that it exists and that the credentials that are specified in the configuration are valid.

This operation might need to connect to a resource, and, as such, might take some time. Do not invoke this operation too often, such as before every provisioning operation. The test operation is not intended to check that the connector is alive (that is, that its physical connection to the resource has not timed out).

You can invoke the test operation before a connector configuration has been validated.

Update

Updates (modifies or replaces) objects on a target resource.

Configuration Properties

The following table lists the configuration properties for the scripted SQL connector:

Scripted SQL Connector Configuration

The Scripted SQL Connector has the following configurable properties.

Operation Script Files

Property	Type	Default	Encrypted ^a	Required ^b
<code>createScriptFileName</code>	String	null		Create
The name of the file used to perform the CREATE operation.				
<code>customizerScriptFileName</code>	String	null		No
The script used to customize some function of the connector. Read the documentation for more details.				
<code>resolveUsernameScriptFileName</code>	String	null		Resolve Username
The name of the file used to perform the RESOLVE_USERNAME operation.				
<code>updateScriptFileName</code>	String	null		Update
The name of the file used to perform the UPDATE operation.				
<code>schemaScriptFileName</code>	String	null		Schema
The name of the file used to perform the SCHEMA operation.				
<code>authenticateScriptFileName</code>	String	null		Authenticate
The name of the file used to perform the AUTHENTICATE operation.				
<code>scriptOnResourceScriptFileName</code>	String	null		Script On Resource

Property	Type	Default	Encrypted ^a	Required ^b
The name of the file used to perform the RUNSCRIPTONRESOURCE operation.				
<code>deleteScriptFileName</code>	String	null		Delete
The name of the file used to perform the DELETE operation.				
<code>searchScriptFileName</code>	String	null		Get Search
The name of the file used to perform the SEARCH operation.				
<code>testScriptFileName</code>	String	null		Test
The name of the file used to perform the TEST operation.				
<code>syncScriptFileName</code>	String	null		Sync
The name of the file used to perform the SYNC operation.				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

Groovy Engine configuration

Property	Type	Default	Encrypted ^a	Required ^b
<code>targetDirectory</code>	File	null		No
Directory into which to write classes.				
<code>warningLevel</code>	int	1		No
Warning Level of the compiler				
<code>scriptExtensions</code>	String[]	['groovy']		No
Gets the extensions used to find groovy files				
<code>scriptBaseClass</code>	String	null		No
Base class name for scripts (must derive from Script)				
<code>scriptRoots</code>	String[]	null		Yes
The root folder to load the scripts from. If the value is null or empty the classpath value is used.				
<code>tolerance</code>	int	10		No
The error tolerance, which is the number of non-fatal errors (per unit) that should be tolerated before compilation is aborted.				
<code>disabledGlobalASTTransformations</code>	String[]	null		No
Sets a list of global AST transformations which should not be loaded even if they are defined in META-INF/org.codehaus.groovy.transform.ASTTransformation files. By default, none is disabled.				

Property	Type	Default	Encrypted ^a	Required ^b
<code>sourceEncoding</code>	<code>String</code>	<code>UTF-8</code>		No
Encoding for source files				
<code>recompileGroovySource</code>	<code>boolean</code>	<code>false</code>		No
If set to true recompilation is enabled				
<code>minimumRecompilationInterval</code>	<code>int</code>	<code>100</code>		No
Sets the minimum of time after a script can be recompiled.				
<code>debug</code>	<code>boolean</code>	<code>false</code>		No
If true, debugging code should be activated				
<code>classpath</code>	<code>String[]</code>	<code>[]</code>		No
Classpath for use during compilation.				
<code>verbose</code>	<code>boolean</code>	<code>false</code>		No
If true, the compiler should produce action information				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

Configuration properties

Property	Type	Default	Encrypted ^a	Required ^b
<code>password</code>	<code>String</code>	<code>null</code>	Yes	No
The connection password to be passed to our JDBC driver to establish a connection. Note that method <code>DataSource.getConnection(username,password)</code> by default will not use credentials passed into the method, but will use the ones configured here. See <code>alternateUsernameAllowed</code> property for more details.				
<code>customSensitiveConfiguration</code>	<code>GuardedString</code>	<code>null</code>	Yes	No
Custom Sensitive Configuration script for Groovy ConfigSlurper				
<code>customConfiguration</code>	<code>String</code>	<code>null</code>		No
Custom Configuration script for Groovy ConfigSlurper				
<code>connectionProperties</code>	<code>String</code>	<code>null</code>		No
The connection properties that will be sent to our JDBC driver when establishing new connections. Format of the string must be <code>[propertyName=property;]*</code> NOTE - The "user" and "password" properties will be passed explicitly, so they do not need to be included here. The default value is null.				
<code>propagateInterruptState</code>	<code>boolean</code>	<code>false</code>		No
Set this to true to propagate the interrupt state for a thread that has been interrupted (not clearing the interrupt state). Default value is false for backwards compatibility.				

Property	Type	Default	Encrypted ^a	Required ^b
<code>useDisposableConnectionFacade</code>	<code>boolean</code>	<code>true</code>		No
Set this to true if you wish to put a facade on your connection so that it cannot be reused after it has been closed. This prevents a thread holding on to a reference of a connection it has already called closed on, to execute queries on it.				
<code>defaultCatalog</code>	<code>String</code>	<code>null</code>		No
The default catalog of connections created by this pool.				
<code>validationInterval</code>	<code>long</code>	<code>30000</code>		No
avoid excess validation, only run validation at most at this frequency - time in milliseconds. If a connection is due for validation, but has been validated previously within this interval, it will not be validated again. The default value is 30000 (30 seconds).				
<code>ignoreExceptionOnPreLoad</code>	<code>boolean</code>	<code>false</code>		No
Flag whether ignore error of connection creation while initializing the pool. Set to true if you want to ignore error of connection creation while initializing the pool. Set to false if you want to fail the initialization of the pool by throwing exception.				
<code>jmxEnabled</code>	<code>boolean</code>	<code>true</code>		No
Register the pool with JMX or not. The default value is true.				
<code>commitOnReturn</code>	<code>boolean</code>	<code>false</code>		No
If autoCommit==false then the pool can complete the transaction by calling commit on the connection as it is returned to the pool If rollbackOnReturn==true then this attribute is ignored. Default value is false.				
<code>logAbandoned</code>	<code>boolean</code>	<code>false</code>		No
Flag to log stack traces for application code which abandoned a Connection. Logging of abandoned Connections adds overhead for every Connection borrow because a stack trace has to be generated. The default value is false.				
<code>maxIdle</code>	<code>int</code>	<code>100</code>		No
The maximum number of connections that should be kept in the pool at all times. Default value is maxActive:100 Idle connections are checked periodically (if enabled) and connections that been idle for longer than minEvictableIdleTimeMillis will be released. (also see testWhileIdle)				
<code>testWhileIdle</code>	<code>boolean</code>	<code>false</code>		No
The indication of whether objects will be validated by the idle object evictor (if any). If an object fails to validate, it will be dropped from the pool. NOTE - for a true value to have any effect, the validationQuery parameter must be set to a non-null string. The default value is false and this property has to be set in order for the pool cleaner/test thread is to run (also see timeBetweenEvictionRunsMillis)				
<code>removeAbandoned</code>	<code>boolean</code>	<code>false</code>		No
Flag to remove abandoned connections if they exceed the removeAbandonedTimeout. If set to true a connection is considered abandoned and eligible for removal if it has been in use longer than the				

Property	Type	Default	Encrypted ^a	Required ^b
removeAbandonedTimeout Setting this to true can recover db connections from applications that fail to close a connection. See also logAbandoned The default value is false.				
abandonWhenPercentageFull	int	0		No
Connections that have been abandoned (timed out) wont get closed and reported up unless the number of connections in use are above the percentage defined by abandonWhenPercentageFull. The value should be between 0-100. The default value is 0, which implies that connections are eligible for closure as soon as removeAbandonedTimeout has been reached.				
minIdle	int	10		No
The minimum number of established connections that should be kept in the pool at all times. The connection pool can shrink below this number if validation queries fail. Default value is derived from initialSize:10 (also see testWhileIdle)				
defaultReadOnly	Boolean	null		No
The default read-only state of connections created by this pool. If not set then the setReadOnly method will not be called. (Some drivers dont support read only mode, ex: Informix)				
maxWait	int	30000		No
The maximum number of milliseconds that the pool will wait (when there are no available connections) for a connection to be returned before throwing an exception. Default value is 30000 (30 seconds)				
logValidationErrors	boolean	false		No
Set this to true to log errors during the validation phase to the log file. If set to true, errors will be logged as SEVERE. Default value is false for backwards compatibility.				
driverClassName	String	null		No
The fully qualified Java class name of the JDBC driver to be used. The driver has to be accessible from the same classloader as tomcat-jdbc.jar				
name	String	Tomcat Connection Pool[4- 153647080]		No
Returns the name of the connection pool. By default a JVM unique random name is assigned.				
useStatementFacade	boolean	true		No
If a statement proxy is set, wrap statements so that equals() and hashCode() methods can be called on closed statements.				
initSQL	String	null		No
A custom query to be run when a connection is first created. The default value is null.				
validationQueryTimeout	int	-1		No

Property	Type	Default	Encrypted ^a	Required ^b
<p>The timeout in seconds before a connection validation queries fail. This works by calling <code>java.test_sample.Statement.setQueryTimeout(seconds)</code> on the statement that executes the <code>validationQuery</code>. The pool itself doesn't timeout the query, it is still up to the JDBC driver to enforce query timeouts. A value less than or equal to zero will disable this feature. The default value is -1.</p>				
<code>validationQuery</code>	<code>String</code>	<code>null</code>		No
<p>The SQL query that will be used to validate connections from this pool before returning them to the caller. If specified, this query does not have to return any data, it just can't throw a <code>SQLException</code>. The default value is <code>null</code>. Example values are <code>SELECT 1(mysql)</code>, <code>select 1 from dual(oracle)</code>, <code>SELECT 1(MS Sql Server)</code></p>				
<code>rollbackOnReturn</code>	<code>boolean</code>	<code>false</code>		No
<p>If <code>autoCommit==false</code> then the pool can terminate the transaction by calling <code>rollback</code> on the connection as it is returned to the pool. Default value is <code>false</code>.</p>				
<code>alternateUsernameAllowed</code>	<code>boolean</code>	<code>false</code>		No
<p>By default, the <code>jdbc-pool</code> will ignore the <code>DataSource.getConnection(username,password)</code> call, and simply return a previously pooled connection under the globally configured properties <code>username</code> and <code>password</code>, for performance reasons. The pool can however be configured to allow use of different credentials each time a connection is requested. To enable the functionality described in the <code>DataSource.getConnection(username,password)</code> call, simply set the property <code>alternateUsernameAllowed</code> to <code>true</code>. Should you request a connection with the credentials <code>user1/password1</code> and the connection was previously connected using <code>different user2/password2</code>, the connection will be closed, and reopened with the requested credentials. This way, the pool size is still managed on a global level, and not on a per schema level.</p>				
<code>dataSourceJNDI</code>	<code>String</code>	<code>null</code>		No
<p>The JNDI name for a data source to be looked up in JNDI and then used to establish connections to the database. See the <code>dataSource</code> attribute. Default value is <code>null</code></p>				
<code>validatorClassName</code>	<code>String</code>	<code>null</code>		No
<p>The name of a class which implements the <code>org.apache.tomcat.jdbc.pool.Validator</code> interface and provides a no-arg constructor (may be implicit). If specified, the class will be used to create a <code>Validator</code> instance which is then used instead of any validation query to validate connections. The default value is <code>null</code>. An example value is <code>com.mycompany.project.SimpleValidator</code>.</p>				
<code>suspectTimeout</code>	<code>int</code>	<code>0</code>		No
<p>Timeout value in seconds. Similar to the <code>removeAbandonedTimeout</code> value but instead of treating the connection as abandoned, and potentially closing the connection, this simply logs the warning if <code>logAbandoned</code> is set to <code>true</code>. If this value is equal or less than 0, no suspect checking will be performed. Suspect checking only takes place if the timeout value is larger than 0 and the connection was not abandoned or if abandon check is disabled. If a connection is suspect a <code>WARN</code> message gets logged and a JMX notification gets sent once.</p>				
<code>useEquals</code>	<code>boolean</code>	<code>true</code>		No
<p>Set to <code>true</code> if you wish the <code>ProxyConnection</code> class to use <code>String.equals</code> and set to <code>false</code> when you wish to use <code>==</code> when comparing method names. This property does not apply to added interceptors as those are configured individually. The default value is <code>true</code>.</p>				

Property	Type	Default	Encrypted ^a	Required ^b
<code>removeAbandonedTimeout</code>	<code>int</code>	<code>60</code>		No
Timeout in seconds before an abandoned(in use) connection can be removed. The default value is 60 (60 seconds). The value should be set to the longest running query your applications might have.				
<code>defaultAutoCommit</code>	<code>Boolean</code>	<code>null</code>		No
The default auto-commit state of connections created by this pool. If not set, default is JDBC driver default (If not set then the <code>setAutoCommit</code> method will not be called.)				
<code>testOnConnect</code>	<code>boolean</code>	<code>false</code>		No
Validate the connection when connecting to the database for the first time. False by default. Set to true if you want to use the <code>validationQuery</code> as an init query.				
<code>jdbcInterceptors</code>	<code>String</code>	<code>null</code>		No
A semicolon separated list of classnames extending <code>org.apache.tomcat.jdbc.pool.JdbcInterceptor</code> class. See Configuring JDBC interceptors below for more detailed description of syntaz and examples. These interceptors will be inserted as an interceptor into the chain of operations on a <code>java.test_sample.Connection</code> object. The default value is null.				
<code>initialSize</code>	<code>int</code>	<code>10</code>		No
The initial number of connections that are created when the pool is started. Default value is 10				
<code>defaultTransactionIsolation</code>	<code>int</code>	<code>-1</code>		No
The default TransactionIsolation state of connections created by this pool. One of the following: NONE, READ_COMMITTED, READ_UNCOMMITTED, REPEATABLE_READ, SERIALIZABLE If not set, the method will not be called and it defaults to the JDBC driver.				
<code>numTestsPerEvictionRun</code>	<code>int</code>	<code>0</code>		No
Property not used in tomcat-jdbc-pool.				
<code>url</code>	<code>String</code>	<code>null</code>		No
The URL used to connect to the database.				
<code>testOnBorrow</code>	<code>boolean</code>	<code>false</code>		No
The indication of whether objects will be validated before being borrowed from the pool. If the object fails to validate, it will be dropped from the pool, and we will attempt to borrow another. NOTE - for a true value to have any effect, the <code>validationQuery</code> parameter must be set to a non-null string. In order to have a more efficient validation, see <code>validationInterval</code> . Default value is false				
<code>fairQueue</code>	<code>boolean</code>	<code>true</code>		No
Set to true if you wish that calls to <code>getConnection</code> should be treated fairly in a true FIFO fashion. This uses the <code>org.apache.tomcat.jdbc.pool.FairBlockingQueue</code> implementation for the list of the idle connections. The default value is true. This flag is required when you want to use asynchronous connection retrieval. Setting this flag ensures that threads receive connections in the order they arrive. During performance tests, there is a very large difference in how locks and lock waiting is implemented. When <code>fairQueue=true</code> there is a decision making process based on what operating system the system is running. If the system is running on Linux				

Property	Type	Default	Encrypted ^a	Required ^b
(property os.name=Linux. To disable this Linux specific behavior and still use the fair queue, simply add the property org.apache.tomcat.jdbc.pool.FairBlockingQueue.ignoreOS=true to your system properties before the connection pool classes are loaded.				
accessToUnderlyingConnectionAllowed	boolean	true		No
Property not used. Access can be achieved by calling unwrap on the pooled connection. see javax.test_sample.DataSource interface, or call getConnection through reflection or cast the object as javax.test_sample.PooledConnection				
maxAge	long	0		No
Time in milliseconds to keep this connection. When a connection is returned to the pool, the pool will check to see if the now - time-when-connected > maxAge has been reached, and if so, it closes the connection rather than returning it to the pool. The default value is 0, which implies that connections will be left open and no age check will be done upon returning the connection to the pool.				
minEvictableIdleTimeMillis	int	60000		No
The minimum amount of time an object may sit idle in the pool before it is eligible for eviction. The default value is 60000 (60 seconds).				
timeBetweenEvictionRunsMillis	int	5000		No
The number of milliseconds to sleep between runs of the idle connection validation/cleaner thread. This value should not be set under 1 second. It dictates how often we check for idle, abandoned connections, and how often we validate idle connections. The default value is 5000 (5 seconds).				
testOnReturn	boolean	false		No
The indication of whether objects will be validated before being returned to the pool. NOTE - for a true value to have any effect, the validationQuery parameter must be set to a non-null string. The default value is false.				
useLock	boolean	false		No
Use a lock when performing operations on the connection object. False by default. Set to true if you will use a separate background thread for idle and abandon checking (e.g. JMX clients). If the pool sweeper is enabled, a lock is used, regardless of this setting.				
maxActive	int	100		No
The maximum number of active connections that can be allocated from this pool at the same time. The default value is 100				
username	String	null		No
The connection username to be passed to our JDBC driver to establish a connection. Note that method DataSource.getConnection(username,password) by default will not use credentials passed into the method, but will use the ones configured here. See alternateUsernameAllowed property for more details.				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

ServiceNow Connector

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

This connector enables you to manage objects in the ServiceNow platform, integrating with ServiceNow's REST API. The connector is bundled with IDM in the `connectors/` directory (`servicenow-connector-1.5.20.11.jar`).

Before You Start

The connector requires a ServiceNow instance with OAuth enabled. You might need to activate the OAuth plugin and set the OAuth activation property if OAuth is not yet enabled on your ServiceNow instance. For more information, see the [ServiceNow documentation](#) that corresponds to your ServiceNow version.

When OAuth is enabled, register an OAuth client application for the connection to IDM. Take note of the `client_id` and `client_secret` of the application, as you need these values when you configure the connector.

The connector configuration must include a ServiceNow user who has the following roles:

- `admin`
- `rest_api_explorer`

If you do not want to give complete `admin` rights to this user, you can create a new role that provides access to the following tables:

- `sys_user_has_role`
- `sys_user_grmember`
- `sys_user_delegate`
- `sys_user_role`
- `sys_user_group`
- `core_company`
- `cmn_department`
- `cmn_cost_center`

- `cmn_location`

Configuring the Connector

The easiest way to configure the ServiceNow connector is through the Admin UI:

1. Select Configure > Connectors > New Connector.
2. Enter a name for the connector configuration, for example, `serviceNow`.
3. Select ServiceNow Connector - 1.5.20.11 as the Connector Type.
4. Enable the connector, and set the properties that specify the connection to your ServiceNow instance:

`instance` (string)

The ServiceNow instance URL, for example `example.service-now.com/`.

`username` (string)

The name of a ServiceNow user with the `admin` and `rest_api_explorer` roles.

`password` (string)

The password of the ServiceNow user.

`clientID` (string)

The ID of your OAuth application.

`clientSecret` (string)

The client secret of your OAuth application.

The UI creates the corresponding provisioner file for the connector in your project's `conf/` directory. The following excerpt of a sample provisioner file shows the required `configurationProperties`:

```
"configurationProperties" : {
  "instance" : "example.service-now.com/",
  "username" : "admin",
  "password" : {encrypted-password},
  "clientID" : "4xxxxxxxxxxxxxxxxxxxxxxxxxxxxee",
  "clientSecret" : {encrypted-client-secret},
  "readSchema" : false
}
```

IDM encrypts the value of the `password` and `clientSecret` on startup.

When your connector is configured correctly, you can test its status by running the following command:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system?_action=test"
[
  {
    "name": "serviceNow",
    "enabled": true,
    "config": "config/provisioner.openicf/serviceNow",
    "connectorRef": {
      "bundleVersion": "1.5.20.11",
      "bundleName": "org.forgerock.openicf.connectors.servicenow-connector",
      "connectorName": "org.forgerock.openicf.connectors.servicenow.ServiceNowConnector"
    },
    "displayName": "ServiceNow Connector",
    "objectTypes": [
      "delegate",
      "role",
      "__ALL__",
      "costCenter",
      "location",
      "company",
      "userHasGroup",
      "department",
      "user",
      "userHasRole",
      "group"
    ],
    "ok": true
  }
]
```

A status of `"ok": true` indicates that the ServiceNow connector can reach the configured resource provider.

Managing Users With the ServiceNow Connector

The following sample queries demonstrate the basic CRUD operations using the ServiceNow connector.

Querying All ServiceNow Users

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
"http://localhost:8080/openidm/system/serviceNow/user?_queryId=query-all-ids"
{
  "result": [
    {
      "_id": "02826bf03710200044e0bfc8bcbe5d3f",
      "__NAME__": "lucius.bagnoli@example.com"
    },
    {
      "_id": "02826bf03710200044e0bfc8bcbe5d55",
      "__NAME__": "jimmie.barninger@example.com"
    },
    {
      "_id": "02826bf03710200044e0bfc8bcbe5d5e",
      "__NAME__": "melinda.carleton@example.com"
    },
    ...
  ],
  "resultCount": 578,
  "pagedResultsCookie": null,
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

Querying a Single ServiceNow User

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
"http://localhost:8080/openidm/system/serviceNow/user/02826bf03710200044e0bfc8bcbe5d3f"
{
  "_id": "02826bf03710200044e0bfc8bcbe5d3f",
  "internal_integration_user": false,
  "department": "5d7f17f03710200044e0bfc8bcbe5d43",
  "sys_mod_count": "5",
  "location": "0002c0a93790200044e0bfc8bcbe5df5",
  "web_service_access_only": false,
  "sys_updated_on": "2018-02-25 16:42:47",
  "sys_domain": "global",
  "notification": "2",
  "sys_created_by": "admin",
  "locked_out": "false",
  "__NAME__": "lucius.bagnoli@example.com",
  "company": "81fd65ecac1d55eb42a426568fc87a63",
  "sys_domain_path": "/",
  "password_needs_reset": "false",
}
```

```
{
  "active": "true",
  "gender": "Male",
  "sys_created_on": "2012-02-18 03:04:49",
  "sys_class_name": "sys_user",
  "calendar_integration": "1",
  "email": "lucius.bagnoli@example.com",
  "sys_id": "02826bf0371020044e0bfc8bcbe5d3f",
  "user_password": "md5230ls7L",
  "user_name": "lucius.bagnoli",
  "sys_updated_by": "developer.program@snc",
  "vip": "false",
  "last_name": "Bagnoli",
  "first_name": "Lucius"
}
```

Creating a ServiceNow User

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request POST \
--data '{
  "__NAME__": "bjensen@example.com",
  "first_name": "Barbara",
  "last_name": "Jensen",
  "email": "bjensen@example.com",
  "phone": "555-123-1234"
}' \
"http://localhost:8080/openidm/system/serviceNow/user?_action=create"
{
  "id": "4116e0690fa01300f6af65ba32050e7a",
  "sys_mod_count": "0",
  "password_needs_reset": "false",
  "notification": "2",
  "locked_out": "false",
  "phone": "555-123-1234",
  "sys_created_on": "2018-02-27 13:33:38",
  "first_name": "Barbara",
  "email": "bjensen@example.com",
  "active": "true",
  "sys_domain": "global",
  "calendar_integration": "1",
  "web_service_access_only": false,
  "vip": "false",
  "sys_id": "4116e0690fa01300f6af65ba32050e7a",
  "sys_updated_on": "2018-02-27 13:33:38",
  "sys_domain_path": "/",
  "sys_created_by": "admin",
  "sys_class_name": "sys_user",
  "last_name": "Jensen",
  "__NAME__": "bjensen@example.com",
  "sys_updated_by": "admin",
  "internal_integration_user": false
}
```

Updating a ServiceNow User

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--header "If-Match:*" \
--request PUT \
--data '{
  "__NAME__": "bjensen@example.com",
  "first_name": "Barbara",
  "last_name": "Jensen",
  "email": "bjensen@example.com",
  "phone": "555-000-0000"
}' \
"http://localhost:8080/openidm/system/serviceNow/user/4116e0690fa01300f6af65ba32050e7a"
{
  "_id": "4116e0690fa01300f6af65ba32050e7a",
  "sys_mod_count": "1",
  "password_needs_reset": "false",
  "notification": "2",
  "locked_out": "false",
  "phone": "555-000-0000",
  "sys_created_on": "2018-02-27 13:33:38",
  "first_name": "Barbara",
  "email": "bjensen@example.com",
  "active": "true",
  "sys_domain": "global",
  "calendar_integration": "1",
  "web_service_access_only": false,
  "vip": "false",
  "sys_id": "4116e0690fa01300f6af65ba32050e7a",
  "sys_updated_on": "2018-02-27 13:35:32",
  "sys_domain_path": "/",
  "sys_created_by": "admin",
  "sys_class_name": "sys_user",
  "last_name": "Jensen",
  "__NAME__": "bjensen@example.com",
  "sys_updated_by": "admin",
  "internal_integration_user": false
}
```

Deleting a ServiceNow User

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "If-Match:*" \
--request DELETE \
"http://localhost:8080/openidm/system/serviceNow/user/4116e0690fa01300f6af65ba32050e7a"
{
  "_id": "4116e0690fa01300f6af65ba32050e7a",
  "sys_mod_count": "1",
}
```

```
"password_needs_reset": "false",
"notification": "2",
"locked_out": "false",
"phone": "555-000-0000",
"sys_created_on": "2018-02-27 13:33:38",
"first_name": "Barbara",
"email": "bjensen@example.com",
"active": "true",
"sys_domain": "global",
"calendar_integration": "1",
"web_service_access_only": false,
"vip": "false",
"sys_id": "4116e0690fa01300f6af65ba32050e7a",
"sys_updated_on": "2018-02-27 13:35:32",
"sys_domain_path": "/",
"sys_created_by": "admin",
"sys_class_name": "sys_user",
"last_name": "Jensen",
"__NAME__": "bjensen@example.com",
"sys_updated_by": "admin",
"internal_integration_user": false
}
```

Synchronizing ServiceNow Users

The ServiceNow connector supports bidirectional reconciliation and liveSync. To set up user synchronization, specify a mapping between managed users and ServiceNow users. For more information, see *"Mapping Data Between Resources"* in the *Synchronization Guide*.

The following example assumes that you have configured a mapping. The example runs a reconciliation operation from ServiceNow to the managed user repository:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/recon?_action=recon&mapping=systemServicenowUser_managedUser"
{
  "_id": "19755e51-5c3b-4362-b316-601856cb282c-13624",
  "state": "ACTIVE"
}
```

The following example runs a liveSync operation from ServiceNow to the managed user repository:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system/serviceNow/user?_action=liveSync"
{
  "connectorData": {
    "nativeType": "string",
    "syncToken": "2018-02-27 11:29:15"
  },
  "_rev": "0000000031285d9b",
  "_id": "SYSTEMSERVICENOWUSER"
}
```

Note

The ServiceNow connector does not support the `__ALL__` object type so you must specify the object type (for example, `User`) in your liveSync operation.

Implementation Specifics

For PATCH requests, a connector can potentially add, remove, or replace an attribute value. The ServiceNow connector does not implement the add or remove operations, so a PATCH request always replaces the entire attribute value with the new value.

OpenICF Interfaces Implemented by the ServiceNow Connector

The ServiceNow Connector implements the following OpenICF interfaces.

Create

Creates an object and its `uid`.

Delete

Deletes an object, referenced by its `uid`.

Schema

Describes the object types, operations, and options that the connector supports.

Script on Connector

Enables an application to run a script in the context of the connector. Any script that runs on the connector has the following characteristics:

- The script runs in the same execution environment as the connector and has access to all the classes to which the connector has access.

- The script has access to a **connector** variable that is equivalent to an initialized instance of the connector. At a minimum, the script can access the connector configuration.
- The script has access to any script-arguments passed in by the application.

Search

Searches the target resource for all objects that match the specified object class and filter.

Sync

Polls the target resource for synchronization events, that is, native changes to objects on the target resource.

Test

Tests the connector configuration. Testing a configuration checks all elements of the environment that are referred to by the configuration are available. For example, the connector might make a physical connection to a host that is specified in the configuration to verify that it exists and that the credentials that are specified in the configuration are valid.

This operation might need to connect to a resource, and, as such, might take some time. Do not invoke this operation too often, such as before every provisioning operation. The test operation is not intended to check that the connector is alive (that is, that its physical connection to the resource has not timed out).

You can invoke the test operation before a connector configuration has been validated.

Update

Updates (modifies or replaces) objects on a target resource.

ServiceNow Connector Configuration

The ServiceNow Connector has the following configurable properties.

Basic configuration properties

Property	Type	Default	Encrypted ^a	Required ^b
instance	String	null		Yes
URL of the ServiceNow instance, for example: dev00000.service-now.com				
username	String	null		Yes
An API user in ServiceNow that can consume the REST API				

Property	Type	Default	Encrypted ^a	Required ^b
password	GuardedString	null	Yes	Yes
Password for the user				
clientId	String	null		Yes
Client ID of the OAuth application in ServiceNow				
clientSecret	GuardedString	null	Yes	Yes
Client Secret for the preceding Client ID				
pageSize	int	100		No
Default page size				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

SSH Connector

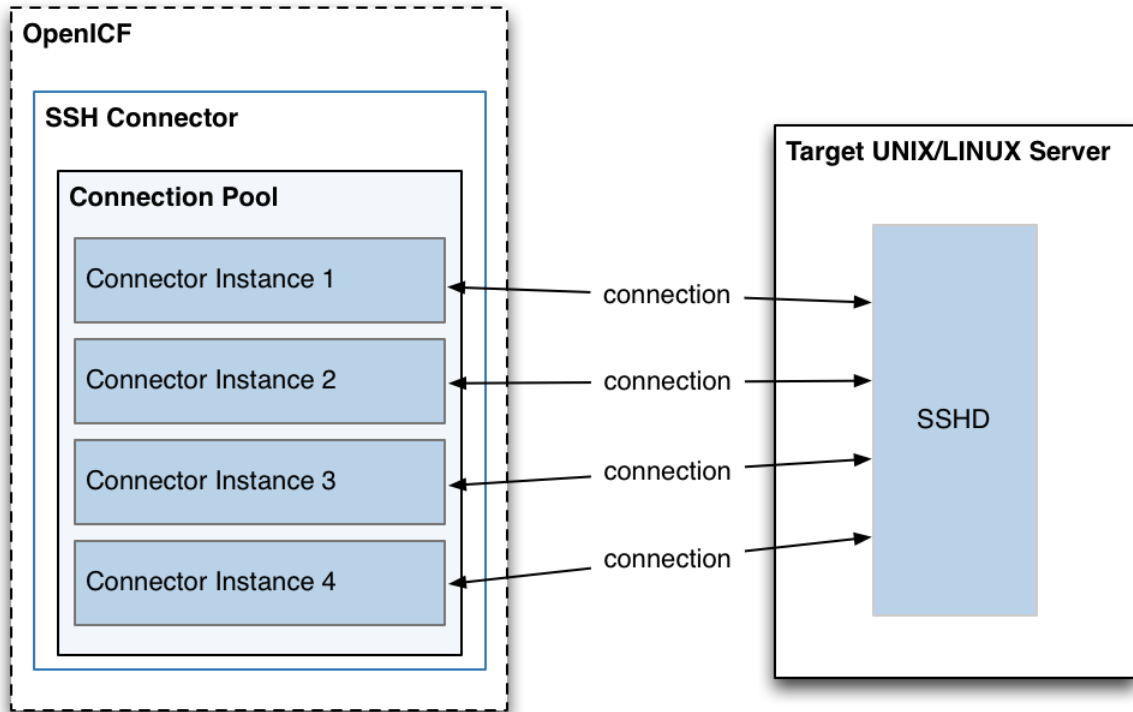
Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

The SSH connector is an implementation of the Scripted Groovy Connector Toolkit, and is based on Java Secure Channel (JSch) and the Java implementation of the Expect library (Expect4j). This connector enables you to interact with any SSH server, using Groovy scripts for the ICF operations.

The SSH connector is a *poolable connector*. This means that each connector instance is placed into a connection pool every time an action is completed. Subsequent actions can re-use connector instances from the connector pool. When a new connector instance is created, a new SSH client connection is created against the target SSH server. This SSH connection remains open as long as the connector instance is in the connection pool. Note that when a new action is performed, it finds the SSH connection in the exact state that it was left by the previous action.

The following image shows the relationship between SSH connector instances and SSH connections to the target server:



Configuring Authentication to the SSH Server

The SSH connector authenticates to the SSH server using either a login/password or a public/private key. The authentication method is specified in the `authenticationType` property in the connector configuration file (`conf/provisioner.openicf-ssh.json`).

Authenticating with a login and password

To authenticate with a login and password, set the `authenticationType` to `PASSWORD` in the connector configuration file, and set a `user` and `password`. For example:

```
"configurationProperties" : {
  ...
  "authenticationType" : "PASSWORD",
  "user" : "<USERNAME>",
  "password" : "<PASSWORD>",
  ...
}
```

The password is encrypted when IDM loads the provisioner file.

Authenticating with a passphrase and private key

To authenticate with a secure certificate, generate a pair of public/private keys. Install the public key on the server side and the private key on the IDM host (where the connector is located).

Set the `authenticationType` to `PUBKEY` in the connector configuration file and set the `user`, `password`, `passphrase` and `privateKey` properties. For example:

```
"configurationProperties" : {
  ...
  "authenticationType" : "PUBKEY",
  "user" : "<USERNAME>",
  "password" : "<PASSWORD>",
  "passphrase" : "secret",
  "privateKey" : [ "-----BEGIN DSA PRIVATE KEY-----",
    "MIIBugIBAAKBgQDcB0ztVMCFptpJhqLLNZSdN/5cDL3S7a0Vy52Ae7vwCqQPCQr",
    "6NyUk+wtkDr07NLYd3sg7a9hbsEnLYChsuX+/WUIvb0KdMfeqcQ+jKK26YdkTCGj",
    "g86dBj9JYhobSHDoQ9ov31pYN/cfw5BAZwkm9TdpEjHPvMIA0xx7GPGKwWIVALbD",
    "CEuflyJk9UB7v0dmJS7bKkbxAoGARcbAuDP4rB6MsgAAkVwf+1sHXEiGpShYwRVV",
    "qBgCZ/S45ELqUuaN/1N/nip/Cc/0SBPKqwl7o50CUg9GH9kTAjmXiwmkbkwtUv+",
    "Xjn5vCHS0w18yc3rGwyr2wj+D9KtDLFJ8+T5HmsbPoDQ3mIZ9xPmRQuRFfVMd9wr",
    "DY0Rs7cCgYAxjGjWDSKThowsv0UCiE0yS6tWggHH3LTrS4Mfh2t0tnbUfrXq2cw",
    "3CN+T6brgnpYbyX5XI17p859C+cw90MD8N6vvBxaN8QMDRFk+hHNUeSy8gXeem9x",
    "00vdIxGgKvA4dh5nSVb5VGKENEGNEHRLYxEPzbqlPa/C/ZvzIvdKXQIUQMoidPFC",
    "n9z+mE2dAADnPf2m9vk=",
    "-----END DSA PRIVATE KEY-----"
  ],
  ...
}
```

The default value for the `passphrase` property is `null`. If you do not set a passphrase for the private key, the passphrase value must be equal to an empty string.

You *must* set a value for the `password` property, because the connector uses `sudo` to perform actions on the SSH server.

The private key (PEM certificate) must be defined as a JSON String array.

The values of the `passphrase`, `password` and `privateKey` are encrypted when IDM loads the provisioner file.

Configuring the SSH Connector

IDM provides a sample connector configuration (`provisioner.openicf-ssh.json`) in the `/path/to/openidm/samples/ssh/conf/` directory. You can copy the sample connector configuration to your project's `conf/` directory, and adjust it to match your Kerberos environment.

Set the authentication properties, as described in "Configuring Authentication to the SSH Server". In addition, set at least the following properties:

host

Specify the hostname or IP address of the SSH server.

port

Set the port on which the SSH server listens.

Default: 22

user

The username of the account that connects to the SSH server.

This account must be able to `ssh` into the server, with the password provided in the next parameter.

password

The password of the account that is used to connect to the SSH server.

prompt

A string representing the remote SSH session prompt. This must be the exact prompt string, in the format `username@target:`, for example `admin@myserver:~$`. Include any trailing spaces.

The following list describes the configuration properties of the SSH connector shown in the sample connector configuration file. You can generally use the defaults provided in the sample connector configuration file, in most cases. For a complete list of all the configuration properties of the SSH connector, see "Configuration properties".

sudoCommand

A string that shows the full path to the **sudo** command, for example `/usr/bin/sudo`.

echoOff

If set to `true` (the default), the input command `echo` is disabled. If set to `false`, every character that is sent to the server is sent back to the client in the `expect()` call.

terminalType

Sets the terminal type to use for the session. The list of supported types is determined by your Linux/UNIX system. For more information, see the `terminfo` manual page (**man terminfo**).

Default: `vt102`

setLocale

If set to `true`, indicates that the default environment locale should be changed to the value of the `locale` property.

Default: `false`

locale

Sets the locale for the `LC_ALL`, `LANG` and `LANGUAGE` environment variables, if `setLocale` is set to `true`.

Default: `en_US.utf8`

`connectionTimeout`

Specifies the connection timeout to the remote server, in milliseconds.

Default: `5000`

`expectTimeout`

Specifies the timeout used by the `expect()` calls in scripts, in milliseconds.

Default: `5000`

`authenticationType`

Sets the authentication type, either `PASSWORD` or `PUBKEY`. For more information, see "Configuring Authentication to the SSH Server".

Default: `PASSWORD`

`throwOperationTimeoutException`

If `true`, the connector throws an exception when the `expectTimeout` is reached for an operation. Otherwise, the operation fails silently.

Default: `true`

`scriptRoots`

The path to the Groovy scripts that will perform the ICF operations, relative to your IDM installation directory. The sample connector configuration expects the scripts in `project-dir/tools`, so this parameter is set to `&{idm.instance.dir}/tools` in the sample configuration.

`classpath`

The directory in which the compiler should look for compiled classes. The default classpath, if not is specified, is `install-dir/lib`.

`*ScriptFileName`

The name of the Groovy script that is used for each ICF operation.

OpenICF Interfaces Implemented by the SSH Connector

The SSH Connector implements the following OpenICF interfaces.

Authenticate

Provides simple authentication with two parameters, presumed to be a user name and password.

Create

Creates an object and its `uid`.

Delete

Deletes an object, referenced by its `uid`.

Resolve Username

Resolves an object by its username and returns the `uid` of the object.

Schema

Describes the object types, operations, and options that the connector supports.

Script on Connector

Enables an application to run a script in the context of the connector. Any script that runs on the connector has the following characteristics:

- The script runs in the same execution environment as the connector and has access to all the classes to which the connector has access.
- The script has access to a `connector` variable that is equivalent to an initialized instance of the connector. At a minimum, the script can access the connector configuration.
- The script has access to any script-arguments passed in by the application.

Script on Resource

Runs a script on the target resource that is managed by this connector.

Search

Searches the target resource for all objects that match the specified object class and filter.

Sync

Polls the target resource for synchronization events, that is, native changes to objects on the target resource.

Test

Tests the connector configuration. Testing a configuration checks all elements of the environment that are referred to by the configuration are available. For example, the connector might make a physical connection to a host that is specified in the configuration to verify that it exists and that the credentials that are specified in the configuration are valid.

This operation might need to connect to a resource, and, as such, might take some time. Do not invoke this operation too often, such as before every provisioning operation. The test operation is not intended to check that the connector is alive (that is, that its physical connection to the resource has not timed out).

You can invoke the test operation before a connector configuration has been validated.

Update

Updates (modifies or replaces) objects on a target resource.

SSH Connector Configuration

The SSH Connector has the following configurable properties.

Configuration properties

Property	Type	Default	Encrypted ^a	Required ^b
<code>customSensitiveConfiguration</code>	<code>GuardedString</code>	<code>null</code>	Yes	No
Description is not available				
<code>createScriptFileName</code>	<code>String</code>	<code>null</code>		Create
Description is not available				
<code>targetDirectory</code>	<code>File</code>	<code>null</code>		No
Description is not available				
<code>customizerScriptFileName</code>	<code>String</code>	<code>null</code>		No
Description is not available				
<code>warningLevel</code>	<code>int</code>	<code>1</code>		No
Description is not available				
<code>authenticateScriptFileName</code>	<code>String</code>	<code>null</code>		Authenticate
Description is not available				
<code>scriptExtensions</code>	<code>String[]</code>	<code>['groovy']</code>		No
Description is not available				
<code>scriptOnResourceScriptFileName</code>	<code>String</code>	<code>null</code>		Script On Resource
Description is not available				
<code>minimumRecompilationInterval</code>	<code>int</code>	<code>100</code>		No
Description is not available				
<code>deleteScriptFileName</code>	<code>String</code>	<code>null</code>		Delete
Description is not available				
<code>scriptBaseClass</code>	<code>String</code>	<code>null</code>		No
Description is not available				

Property	Type	Default	Encrypted ^a	Required ^b
<code>scriptRoots</code>	<code>String[]</code>	<code>null</code>		Yes
Description is not available				
<code>customConfiguration</code>	<code>String</code>	<code>null</code>		No
Description is not available				
<code>resolveUsernameScriptFileName</code>	<code>String</code>	<code>null</code>		Resolve Username
Description is not available				
<code>searchScriptFileName</code>	<code>String</code>	<code>null</code>		Get Search
Description is not available				
<code>tolerance</code>	<code>int</code>	<code>10</code>		No
Description is not available				
<code>updateScriptFileName</code>	<code>String</code>	<code>null</code>		Update
Description is not available				
<code>debug</code>	<code>boolean</code>	<code>false</code>		No
Description is not available				
<code>classpath</code>	<code>String[]</code>	<code>[]</code>		No
Description is not available				
<code>disabledGlobalASTTransformations</code>	<code>String[]</code>	<code>null</code>		No
Description is not available				
<code>schemaScriptFileName</code>	<code>String</code>	<code>null</code>		Schema
Description is not available				
<code>verbose</code>	<code>boolean</code>	<code>false</code>		No
Description is not available				
<code>testScriptFileName</code>	<code>String</code>	<code>null</code>		Test
Description is not available				
<code>sourceEncoding</code>	<code>String</code>	<code>UTF-8</code>		No
Description is not available				
<code>syncScriptFileName</code>	<code>String</code>	<code>null</code>		Sync

Property	Type	Default	Encrypted ^a	Required ^b
Description is not available				
<code>recompileGroovySource</code>	<code>boolean</code>	<code>false</code>		No
Description is not available				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

Basic Configuration Properties

Property	Type	Default	Encrypted ^a	Required ^b
<code>host</code>	<code>String</code>	<code>null</code>		Yes
The hostname to connect to				
<code>port</code>	<code>int</code>	<code>22</code>		Yes
TCP port to use (defaults to 22)				
<code>user</code>	<code>String</code>	<code>null</code>		Yes
The user name used to login to remote server				
<code>password</code>	<code>GuardedString</code>	<code>null</code>	Yes	No
The password used to login to remote server				
<code>passphrase</code>	<code>GuardedString</code>	<code>null</code>	Yes	No
The passphrase used to read the private key when using Public Key authentication				
<code>privateKey</code>	<code>String[]</code>	<code>[]</code>	Yes	No
The base 64 encoded value (PEM) of the private key used for Public Key authentication				
<code>authenticationType</code>	<code>String</code>	<code>PASSWORD</code>		Yes
Defines which authentication type should be use: PASSWORD or PUBKEY (defaults to PASSWORD)				
<code>prompt</code>	<code>String</code>	<code>root@localhost:#</code>		Yes
A string representing the remote SSH session prompt (defaults to root@localhost:#)				
<code>sudoCommand</code>	<code>String</code>	<code>/usr/bin/sudo</code>		Yes
A string representing the sudo command (defaults to /usr/bin/sudo)				
<code>echoOff</code>	<code>boolean</code>	<code>true</code>		Yes
Disable the input command echo (default to true)				
<code>terminalType</code>	<code>String</code>	<code>vt102</code>		Yes

Property	Type	Default	Encrypted ^a	Required ^b
Defines the terminal type to use for the session (default to vt102)				
locale	String	en_US.utf8		Yes
Define the locale for LC_ALL, LANG and LANGUAGE environment variables to use if setLocale=true				
setLocale	boolean	false		Yes
Defines if the default environment locale should be changed with the value provided for locale (defaults to false)				
connectionTimeout	int	5000		Yes
Defines the connection timeout to the remote server in milliseconds (default to 5000)				
expectTimeout	long	5000		Yes
Defines the timeout used by the expect() calls in the scripts in milliseconds (default to 5000)				
throwOperationTimeoutException	boolean	true		Yes
Defines if an OperationTimeoutException should be thrown if any call to expect times out (defaults to true)				
promptReadyTimeout	long	20		No
Defines the "prompt ready" timeout for the promptReady() command in milliseconds (default to 20)				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

SAP SuccessFactors Connector

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

The SAP SuccessFactors connector lets you synchronize SAP SuccessFactors users with IDM managed users.

Before you start

Before you configure the connector, gather the following details:

Host

The SuccessFactors API hostname. For example, `apisalesdemo2.successfactors.eu`.

Client ID

The SuccessFactors API Key or client ID. To find this:

1. Open your SuccessFactors administrator account.
2. Open Manage OAuth2 Client Applications.
3. Select your registered OAuth2 Client Application.
4. Click View.
5. Copy the API key.

User ID

The API User ID of the SuccessFactors user who authenticates to the REST server.

Private Key

A private key. To configure this, generate a key pair from the X.509 certificate and copy the value of the private key.

Company ID

The API Company ID of the admin user. This is specified in the SuccessFactors login URL.

Person Segments

SuccessFactors person segments; for example, `EmpJob`, `EmpEmployment`, `PerPersonal`.

Install the SuccessFactors connector

Download the connector .jar file from the link: `{fr_download_site_url}[{fr_download_site_name}]`.

- If you are running the connector locally, place it in the `/path/to/openidm/connectors` directory; for example:

```
mv ~/Downloads/successfactors-connector-1.5.20.12.jar/path/to/openidm/connectors/
```

- If you are using a remote connector server (RCS), place it in the `/path/to/openicf/connectors` directory on the RCS.

Configure the SuccessFactors connector

Create a connector configuration using the Admin UI:

1. Select Configure > Connectors and click New Connector.
2. Enter a Connector Name.
3. Select SuccessFactors Connector - 1.5.20.12 as the Connector Type.

4. Provide the Base Connector Details.
5. Click Save.

When your connector is configured correctly, the connector displays as Active in the Admin UI.

Alternatively, test that the configuration is correct by running the following command:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system/Successfactors?action=test"
{
  "name" : "Successfactors",
  "enabled" : true,
  "config" : "config/provisioner.openicf/Successfactors",
  "connectorRef" : {
    "bundleVersion" : "${bundleVersion}",
    "bundleName" : "org.forgerock.openicf.connectors.successfactors-connector",
    "connectorName" : "org.forgerock.openicf.connectors.successfactors.SuccessFactorsConnector"
  },
  "displayName" : "SuccessFactors Connector",
  "objectTypes" : [ "__GROUP__", "__PERSON__", "__ACCOUNT__", "__ALL__" ],
  "ok" : true
}
```

If the command returns `"ok": true`, your connector was configured correctly, and can authenticate to the Cerner system.

Use the SuccessFactors connector

Actions on accounts

You can perform the following actions on a SAP SuccessFactors account:

+ *Create a SuccessFactors user*

The following example creates a user with every available attribute:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
--data '{
  "userId": "BJENSEN",
  "username": "bjensen",
  "ENABLE": true,
  "email": "bjensen@example.com",
  "firstName": "Barbara",
  "lastName": "Jensen",
  "country": "USA",
}
```

```
"married": false,
"timeZone": "US/Eastern",
"department": "Cloud",
"state": "New York",
"city": "New York City",
"jobLevel": "2",
"location": "40.6635°N 73.9387°W",
"__PASSWORD__": "Test@123",
"division": "Manufacturing",
"hireDate": "2021-07-26 00:00:00",
"dateOfBirth": "2012-08-22 00:00:00",
"__GROUP__": [
  {"groupId": "6895"},
  {"groupId": "6095"}
]
}' \
"http://localhost:8080/openidm/system/Successfactors/__ACCOUNT__?_action=create"
{
  "_id" : "BJENSEN",
  "userId" : "BJENSEN",
  "jobLevel" : "2",
  "__GROUP__" : [ {
    "groupId" : "1586",
    "groupName" : "$$EVERYONE$$"
  }, {
    "groupId" : "6895",
    "groupName" : "SAP_Managers"
  }, {
    "groupId" : "6095",
    "groupName" : "SAP_ONB2_ErrorFlowAdmins"
  } ],
  "department" : "Cloud",
  "dateOfBirth" : "2012-08-22 00:00:00",
  "lastModifiedDateTime" : "2022-11-02 09:13:49",
  "__ENABLE__" : true,
  "email" : "bjensen@example.com",
  "country" : "USA",
  "lastModified" : "2022-11-02 10:13:49",
  "location" : "40.6635°N 73.9387°W",
  "lastName" : "Jensen",
  "lastModifiedWithTZ" : "2022-11-02 09:13:49",
  "username" : "bjensen",
  "timeZone" : "US/Eastern",
  "city" : "New York City",
  "state" : "New York",
  "__NAME__" : "bjensen",
  "hireDate" : "2021-07-26 00:00:00",
  "married" : false,
  "division" : "Manufacturing",
  "firstName" : "Barbara"
}
```

Note

New users must have at least the `username`, `userId`, and `status` properties.

+ Query all users

The following example queries all SuccessFactors users:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/successfactors/__ACCOUNT__?_queryId=query-all-ids"
{
  "result": [
    { "_id": "1007373" },
    { "_id": "1007371" },
    { "_id": "1007376" },
    { "_id": "1007370" },
    { "_id": "1007377" }
  ],
  "resultCount": 5,
  "pagedResultsCookie": null,
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

+ Query a single user

The following example queries a single user by their ID:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/successfactors/__ACCOUNT__?_queryFilter=_id%20eq%20%22BJENSEN%22"
{
  "_id" : "BJENSEN",
  "userId" : "BJENSEN",
  "jobLevel" : "2",
  "__GROUP__" : [ {
    "groupId" : "1586",
    "groupName" : "$$EVERYONE$$"
  }, {
    "groupId" : "6895",
    "groupName" : "SAP_Managers"
  }, {
    "groupId" : "6095",
    "groupName" : "SAP_ONB2_ErrorFlowAdmins"
  } ]
}
```

```
} ],
"department" : "Cloud",
"dateOfBirth" : "2012-08-22 00:00:00",
"lastModifiedDateTime" : "2022-11-02 09:13:49",
"__ENABLE__" : true,
"email" : "bjensen@example.com",
"country" : "USA",
"lastModified" : "2022-11-02 10:13:49",
"location" : "40.6635°N 73.9387°W",
"lastName" : "Jensen",
"lastModifiedWithTZ" : "2022-11-02 09:13:49",
"username" : "bjensen",
"timeZone" : "US/Eastern",
"city" : "New York City",
"state" : "New York",
"__NAME__" : "bjensen",
"hireDate" : "2021-07-26 00:00:00",
"married" : false,
"division" : "Manufacturing",
"firstName" : "Barbara"
}
```

+ *Modify a user*

You can use the SuccessFactors connector to modify the following attributes of a user entry:

- `username`
- `email`
- `status`
- `country`
- `department`
- `timeZone`
- `jobLevel`
- `married`
- `city`
- `state`
- `division`
- `citizenship`
- `location`
- `firstName`
- `lastName`

- gender
- dateOfBirth
- jobCode

The following example updates the `division` property on a user:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "If-Match:*" \
--request PUT \
--data '{
  "division": "Engineering"
}' \
"http://localhost:8080/openidm/system/successfactors/__ACCOUNT__/BJENSEN"
{
  "_id" : "BJENSEN",
  "userId" : "BJENSEN",
  ...
  "division" : "Engineering",
  "firstName" : "Barbara"
}
```

+ Reset a user's password

The following example resets the password for a SuccessFactors user account:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request PATCH \
--data '[{
  "operation": "replace",
  "field": "__PASSWORD__",
  "value": "__CHANGE_ME__"
}]' \
"http://localhost:8080/openidm/system/successfactors/__ACCOUNT__/BJENSEN"
{
  "_id" : "BJENSEN",
  "userId" : "BJENSEN",
  ...
}
```

Note

The updated password is not included in the response object; however, the value is updated in the system.

+ Activate a user

The following example activates a user with the minimum required attributes:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
--data '{
  "username": "bjensen",
  "__ENABLE__": true,
  "firstName": "Barbara",
  "userId": "BJENSEN"
}' \
"http://localhost:8080/openidm/system/successfactors/__ACCOUNT__/BJENSEN"
{
  "_id": "BJENSEN",
  "userId": "BJENSEN",
  ...
  "__ENABLE__": true
}
```

+ Deactivate a user

The SuccessFactors connector does not support deleting accounts. To deactivate an unwanted account, set the account's `__ENABLE__` attribute value to `false`. A deactivated account remains in the SuccessFactors system and can still be queried by its ID, but cannot be accessed.

The following example deactivates a SuccessFactors account:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
--data '{
  "username": "bjensen",
  "__ENABLE__": false,
  "firstName": "Barbara",
  "userId": "BJENSEN"
}' \
"http://localhost:8080/openidm/system/successfactors/__ACCOUNT__/BJENSEN"
{
  _id: "BJENSEN"
}
```

+ Assign a user to a group

The following example assigns a user to a group:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "if-Match:*" \
--request PUT \
--data '{
  "_ENABLE_": true,
  "_GROUP_": [{"groupId":1001}]
}' \
"http://localhost:8080/openidm/system/successfactors/__ACCOUNT__/BJENSEN"
{
  "_id" : "BJENSEN",
  "userId" : "BJENSEN",
  "jobLevel" : "2",
  "_GROUP_" : [ {
    "groupId" : "1001",
    "groupName" : "Example Working Group"
  },
  ...
}
```

Actions on other objects

+ *Query all groups*

The following example queries all groups in the system:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "if-Match:*" \
--request GET \
"http://localhost:8080/openidm/system/successfactors/__GROUP__?_queryId=query-all-ids"
{
  "result": [
    {"_id": "6637"},
    {"_id": "2202"},
    {"_id": "1588"},
    {"_id": "6877"},
    {"_id": "2203"}
  ],
  "resultCount": 5,
  "pagedResultsCookie": null,
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

+ *Query a single group*

The following example queries a single group:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/successfactors/__GROUP__?/1001"
{
  "_id": "1001",
  "_NAME_": "1001",
  "groupName": "Example Working Group",
  "lastModifiedDate": "2015-01-04 23:29:38",
  "createdBy": "v4admin",
  "totalMemberCount": "33590",
  "activeMembershipCount": "2294",
  "groupID": "1001",
  "groupType": "permission"
}
```

+ Query all persons

The following example queries all persons in the system:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/successfactors/__PERSON__?_queryId=query-all-ids"
{
  "result": [
    {"_id": "69119"},
    {"_id": "69120"},
    {"_id": "69121"},
    {"_id": "80279"},
    {"_id": "80280"}
  ],
  "resultCount": 5,
  "pagedResultsCookie": null,
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

+ Query a single person

The following example queries a single person:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/successfactors/__PERSON__?_queryFilter=_id%20%22scarter%22"
{
```

```
"result": [{
  "_id": "scarter",
  "EmpJob_payGrade": "GR-08",
  "EmpEmployment_firstDateWorked": "2002-03-17 00:00:00",
  "PerPersonal_maritalStatus": "10819",
  "PerPersonal_nationality": "USA",
  "EmpEmployment_lastDateWorked": null,
  "EmpEmployment_userId": "scarter",
  "PerPersonal_personIdExternal": "scarter",
  "EmpEmployment_initialStockGrant": null,
  "PerPerson_countryOfBirth": "USA",
  "PerPersonal_endDate": "9999-12-31 00:00:00",
  "PerPersonal_firstName": "Sam",
  "EmpEmployment_eligibleForStock": null,
  "PerPersonal_lastName": "Carter",
  "EmpJob_payScaleArea": "USA/US2",
  "EmpJob_jobCode": "50070968",
  "PerPerson_regionOfBirth": null,
  "PerPersonal_startDate": "2002-03-17 00:00:00",
  "PerPerson_personIdExternal": "scarter",
  "PerPerson_lastModifiedDateTime": "2015-10-30 10:05:06",
  "EmpEmployment_lastModifiedDateTime": "2018-07-15 23:12:06",
  "PerPersonal_lastModifiedDateTime": "2018-10-25 23:51:29",
  "EmpJob_timezone": "US/Eastern",
  "PerPersonal_gender": "M",
  "PerPerson_dateOfBirth": "1983-02-15 00:00:00",
  "PerPersonal_nativePreferredLang": "10223",
  "EmpEmployment_serviceDate": null,
  "EmpEmployment_assignmentIdExternal": "scarter",
  "EmpJob_lastModifiedDateTime": "2020-06-23 10:50:43",
  "PerPerson_createdOn": "2015-01-05 23:34:22",
  "EmpJob_company": "1710",
  "EmpEmployment_originalStartDate": "2002-03-17 00:00:00",
  "EmpEmployment_endDate": null,
  "EmpJob_position": "3000325",
  "EmpJob_jobTitle": "Administrative Support",
  "PerPersonal_salutation": "10810",
  "EmpEmployment_seniorityDate": "2002-03-17 00:00:00",
  "PerPerson_createdDateTime": "2015-01-05 22:34:22",
  "EmpEmployment_professionalServiceDate": null,
  "EmpJob_startDate": "2017-01-01 00:00:00",
  "PerPersonal_middleName": null,
  "PerPerson_createdBy": "v4admin",
  "PerPersonal_preferredName": null,
  "PerPerson_lastModifiedBy": "scarter",
  "EmpJob_businessUnit": "CORP",
  "EmpJob_seqNumber": "1",
  "PerPerson_perPersonUuid": "87AF10389BCC4F29BC3F3A225B321E14",
  "EmpJob_location": "1710-2001",
  "EmpJob_managerId": "108743",
  "EmpJob_eventReason": "PAYOTH",
  "PerPerson_lastModifiedOn": "2015-10-30 11:05:06",
  "EmpJob_payScaleType": "USA/US2",
  "EmpJob_userId": "scarter",
  "EmpEmployment_initialOptionGrant": null,
  "EmpEmployment_personIdExternal": "scarter",
  "PerPerson_personId": "8",
  "___NAME___": "scarter"}],
"resultCount": 1,
```

```
{
  "pagedResultsCookie": null,
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

Account Status

Attribute	Description
userId	The user's User ID.
userName	The user's username.
status	The user's status.
firstName	The user's first name.
lastName	The user's last name.
mi	The user's middle name.
email	The user's email address.
dateOfBirth	The user's birthdate.
defaultFullName	The default full name for the user.
password	The user's password.
lastModifiedDateTime	The last modified date and time without time zone information.
country	The user's country of residence.
citizenship	The user's country of citizenship.
married	The user's marital status.
state	The state where the user lives.
city	The city where the user lives.
division	The division the user works in.
department	The department the user works in.
jobCode	The Job code of the user.
jobLevel	The Job level of the user.
timeZone	The user's time zone.
location	The user's location.
manager	The user's manager.
hireDate	The date the user was hired.
lastModifiedWithTZ	The last modified date and time with time zone information.
lastModified	The last modified date.

Group Attributes

The following group attributes are supported by the SuccessFactors Connector:

Attribute	Description
groupId	The unique ID of the group.
groupName	The name of the group.
groupType	The type of the group.
activeMembershipCount	The number of active members.
totalMemberCount	The number of total members.
deExcludePools	Users excluded from the group.
dgIncludePools	Users included in the group.
createdBy	The user who created the group.
lastModifiedDate	The last modified date.

Person Attributes

PerPerson Attributes

The following PerPerson attributes are supported by the SuccessFactors connector:

Attribute	Description
personIdExternal	An ID used to represent the person externally.
personId	An ID used to represent the person internally.
userId	The person's user ID.
dateOfBirth	The person's date of birth.
lastModifiedOn	The date the person was last modified.
lastModifiedDateTime	The time the person was last modified.
countryOfBirth	The country the person was born in.
createdBy	The ID of the user who created the person.
createdDateTime	The time the person was created.
lastModifiedBy	The ID of the last user to modify the person.
perPersonUuid	A UUID for the person.
regionOfBirth	The person's birth region.

PerPersonal Attributes

Attribute	Description
personIdExternal	An ID used to represent the employee externally.
endDate	The end date of the employment.
startDate	The start date of the employment.
firstName	The person's first name.
lastName	The person's last name.
gender	The person's gender.
nativePreferredLang	The person's preferred native language code.
salutation	The salutation to be used for the person.
maritalStatus	The person's marital status.
nationality	The person's nationality.
middleName	The person's middle name.
preferredName	The person's preferred name.
lastModifiedDateTime	The time when the PerPersonal was last updated.

EmpEmployment Attributes

Attribute	Description
personIdExternal	An ID used to represent the employee externally.
userId	The employee's user ID.
assignmentIdExternal	An assignment ID used to identify users across the suite.
firstDateWorked	The first date the employee worked.
endDate	The end date of the employment.
startDate	The start date of the employment.
eligibleForStock	Whether or not the user is eligible for stock.
initialOptionGrant	The initial grant value of the employment.
serviceDate	The service date of employment.
professionalServiceDate	The professional service date of employment.
initialStockGrant	The employment's initial stock grant.
seniorityDate	The date of seniority.
lastModifiedDateTime	The time when the EmpEmployment object was last updated.
lastDateWorked	The date of the last day the employee worked.

EmpJob Attributes

Attribute	Description
seqNumber	The sequence number associated with the job.
userId	The employee's user ID.
eventReason	The reason for action.
company	The company the job is for.
managerId	The ID of the manager of the job.
timezone	The time zone the job is in.
startDate	The date the job begins.
endDate	The date the job ends.
payGrade	The job's pay grade.
jobCode	The job's code.
position	The position of the job.
location	The job's location.
payScaleType	The payscale type for the job.
payScaleArea	The payscale area for the job.
businessUnit	The business unit the job belongs to.
lastModifiedDateTime	The date the job was last modified.

OpenICF Interfaces Implemented by the SuccessFactors Connector

The SuccessFactors Connector implements the following OpenICF interfaces.

Create

Creates an object and its **uid**.

Delete

Deletes an object, referenced by its **uid**.

Schema

Describes the object types, operations, and options that the connector supports.

Script on Connector

Enables an application to run a script in the context of the connector. Any script that runs on the connector has the following characteristics:

- The script runs in the same execution environment as the connector and has access to all the classes to which the connector has access.
- The script has access to a **connector** variable that is equivalent to an initialized instance of the connector. At a minimum, the script can access the connector configuration.
- The script has access to any script-arguments passed in by the application.

Search

Searches the target resource for all objects that match the specified object class and filter.

Sync

Polls the target resource for synchronization events, that is, native changes to objects on the target resource.

Test

Tests the connector configuration. Testing a configuration checks all elements of the environment that are referred to by the configuration are available. For example, the connector might make a physical connection to a host that is specified in the configuration to verify that it exists and that the credentials that are specified in the configuration are valid.

This operation might need to connect to a resource, and, as such, might take some time. Do not invoke this operation too often, such as before every provisioning operation. The test operation is not intended to check that the connector is alive (that is, that its physical connection to the resource has not timed out).

You can invoke the test operation before a connector configuration has been validated.

Update

Updates (modifies or replaces) objects on a target resource.

SuccessFactors Connector Configuration

The SuccessFactors Connector has the following configurable properties.

Configuration properties

Property	Type	Default	Encrypted ^a	Required ^b
host	String	null		Yes
Hostname of the target				
clientId	String	null		Yes

Property	Type	Default	Encrypted ^a	Required ^b
The client identifier				
userId	String	null		Yes
User id for authentication				
privateKey	GuardedString	null		Yes
The private key which is used for signing JWT				
companyId	String	null		Yes
Company id as present in target application				
personSegments	String	null		No
To retrieve data based on person segments				
pageSize	int	0		No
Page size for search operation				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

Basic configuration properties

Property	Type	Default	Encrypted ^a	Required ^b
maximumConnections	Integer	10		No
Provide the maximum connections				
connectionTimeout	int	600		No
Provide the maximum connection timeout in seconds				
httpProxyHost	String	null		No
Provide the HTTP proxy host				
httpProxyPort	Integer	null		No
Provide the HTTP proxy port				
httpProxyUsername	String	null		No
Provide the HTTP proxy username				
httpProxyPassword	GuardedString	null	Yes	No
Provide the HTTP proxy password				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

Workday Connector

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

Workday is a multi-tenant Software-as-a-Service (SaaS) application. The Workday connector enables you to synchronize user accounts between IDM and Workday's cloud-based HR system.

The connector supports reconciliation of users and organizations from Workday to an IDM repository, liveSync of users from Workday to IDM, and updating users in a Workday system.

To use the connector, you need a Workday instance with the required permissions and a set of credentials to access the instance, including the username, password, tenant name, and host name.

Download the Workday connector from the [ForgeRock BackStage download site](#) and place it in the `/path/to/openidm/connectors/` directory.

Download the Workday connector dependencies and copy them to the `/path/to/openidm/lib/` directory. If you are running the connector remotely, copy the dependencies to the `/path/to/openicf/lib/` directory on the remote system.

Configuring the Workday Connector

1. The easiest way to configure the connector is to use the Admin UI. Select Configure > Connectors > New Connector, then select Workday in the Connector Type field.

Alternatively, use the sample configuration file provided in `/path/to/openidm/samples/example-configurations/provisioners/provisioner.openicf-workday.json`. Copy that file to your project's `conf/` directory, and set `enabled` to `true`.

2. Edit the `configurationProperties` to specify the connection to the Workday instance, for example:

```
"configurationProperties" : {
  "hostname" : "example.workday.net",
  "tenant" : "example-tenant",
  "username" : "admin",
  "password" : "Passw0rd",
  ...
}
```

Set at least the following properties:

hostname

The fully qualified name of the Workday instance. The connector uses the `hostname` to construct the endpoint URL.

tenant

The tenant to which you are connecting. The connector uses the **tenant** name to construct the endpoint URL, and the complete username (in the form **username@tenant**).

username

The **username** used to log in to the Workday instance. Do not specify the complete username including the **tenant**. The connector constructs the complete username.

password

The **password** used to log in to the Workday instance.

connectionTimeout

The timeout (in milliseconds) that the connector should wait for a request to be sent to the Workday instance. The default timeout is 60000ms or one minute. Requests that take longer than a minute throw an exception.

receiveTimeout

The timeout (in milliseconds) that the connector waits to receive a response. The default timeout is 60000ms or one minute. Because the Workday can be slow, and the amount of information returned can be very large, you should set this parameter carefully to avoid unnecessary timeouts.

3. Check that the connector is retrieving the exact data that you need.

The **configurationProperties** also specify the data that the connector should retrieve with a number of boolean **include...** and **exclude...** properties. These properties can be divided as follows:

Worker types

By default, all worker types are retrieved, with the following settings:

- **excludeContingentWorkers** - exclude contingent workers from query results, **false** by default.
- **excludeEmployees** - exclude regular employees from query results, **false** by default.
- **excludeInactiveWorkers** - exclude inactive workers from query results, **false** by default.

Specific worker data

These parameters specify the properties that are returned for every worker included by the parameters in the previous section.

For performance reasons, set all of these to false initially, then include *only* the properties that you need.

```
includeWorkerDocuments
includeDevelopmentItems
includeRoles
includeQualifications
includeTransactionLogData
includeCareer
includeContingentWorkerTaxAuthorityFormInformation
includeUserAccount
includeFeedbackReceived
includeEmployeeContractData
includeSkills
includeAccountProvisioning
includeGoals
includeSuccessionProfile
includeBackgroundCheckData
includeEmployeeReview
includeManagementChainData
includeOrganizations
includePhoto
includeRelatedPersons
includeBenefitEligibility
includeTalentAssessment
includeBenefitEnrollments
includeCompensation
```

Specific organizational data

Included in the data of each worker is the organization to which the user belongs. If you have set `includeOrganizations` to `true`, you can specify the organizational data that should be *excluded* from the query response. By default, all organizational data is included.

To exclude data from a response, set its corresponding property to `true`. For performance reasons, set all of these to `true` initially, then include *only* the properties that you need:

```
excludeCompanies
excludeBusinessUnits
excludeCustomOrganizations
excludeMatrixOrganizations
excludeGiftHierarchies
excludeCostCenterHierarchies
excludeGrants
excludeProgramHierarchies
excludeFunds
excludeOrganizationSupportRoleData
excludeGifts
excludeBusinessUnitHierarchies
excludeCostCenters
```

```
excludePrograms
excludeSupervisoryOrganizations
excludeRegionHierarchies
excludeTeams
excludeLocationHierarchies
excludeRegions
excludePayGroups
excludeFundHierarchies
excludeGrantHierarchies
```

For information about all the configurable properties for this connector, see "Workday Connector Configuration".

Testing the Workday Connector

When your connector is configured correctly, you can test its status by running the following command:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system?_action=test"
[
{
  "name": "workday",
  "enabled": true,
  "config": "config/provisioner.openicf/workday",
  "connectorRef": {
    "bundleVersion": "1.5.20.12",
    "bundleName": "org.forgerock.openicf.connectors.workday-connector",
    "connectorName": "org.forgerock.openicf.connectors.workday.WorkdayConnector"
  },
  "displayName": "Workday Connector",
  "objectTypes": [
    "employee",
    "__ALL__"
  ],
  "ok": true
}
]
```

A status of `"ok": true` indicates that the connector can contact the Workday instance.

To retrieve the workers in the Workday system, run the following command:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
"http://localhost:8080/openidm/system/workday/employee?_queryId=query-all-ids"
{
  "result": [
    {
      "_id": "3aa5550b7fe348b98d7b5741afc65534",
      "employeeID": "21001"
    },
    {
      "_id": "0e44c92412d34b01ace61e80a47aaf6d",
      "employeeID": "21002"
    },
    {
      "_id": "3895af7993ff4c509cbea2e1817172e0",
      "employeeID": "21003"
    },
    ...
  ]
}
```

The first time the connector retrieves the employees from the Workday system, you might see the following warning in the console:

```
WARNING: Default key managers cannot be initialized: Invalid keystore format
java.io.IOException: Invalid keystore format
```

You can safely ignore this warning.

To retrieve a specific user, include the user's ID in the URL. For example:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
"http://localhost:8080/openidm/system/workday/employee/3aa5550b7fe348b98d7b5741afc65534"
{
  "_id": "3aa5550b7fe348b98d7b5741afc65534",
  "title": "Vice President, Human Resources",
  "country": "USA",
  "postalCode": "94111",
  "userID": "lmcneil",
  "hireDate": "2000-01-01-08:00",
  "address": [
    "3939 The Embarcadero"
  ],
  "state": "California",
  "postalAddress": "3939 The Embarcadero\nSan Francisco, CA 94111\nUnited States of America",
  "addressLastModified": "2011-06-20T13:54:02.023-07:00",
  "firstName": "Logan",
  "gender": "Female",
  "employeeID": "21001",
  "managerID": "21431",
}
```

```
{
  "email": "logan.mcneil@workday.net",
  "city": "San Francisco",
  "preferredName": "Logan McNeil",
  "birthDate": "1971-05-25-07:00",
  "active": true,
  "preferredFirstName": "Logan",
  "employee": true,
  "workerType": "Full time",
  "positionEffectiveDate": "2016-06-01-07:00",
  "preferredLastName": "McNeil",
  "dateActivated": "2000-01-01-08:00",
  "legalName": "Logan McNeil",
  "lastName": "McNeil",
  "mobile": [
    "+1 (415) 789-8904"
  ],
  "terminated": false
}
```

Reconciling Users from Workday to IDM

To reconcile users from Workday to the IDM repository, set up a mapping, either using the Admin UI or in a `sync.json` file in your project's `conf` directory. For information about mapping resources, see *"Mapping Data Between Resources"* in the *Synchronization Guide*.

When you have created a mapping, you can run reconciliation using the Admin UI or with a REST call similar to the following:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/recon?
_action=recon&mapping=systemWorkdayEmployee_managedUser&waitForCompletion=true"
{
  "_id": "db2bc7f4-e9a8-4315-9dd1-e2cdcd85ae6e-33099",
  "state": "SUCCESS"
}
```

Updating Users in the Workday System

The connector supports updates to system users only for the following properties:

- Account credentials (`username` and `password`)
- `email`
- `mobile` (telephone number)

The following command update's user `lmcneil`'s mobile number:

```
curl \
```

```
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-type: application/json" \
--request PATCH \
--data '[
{
  "operation": "replace",
  "field": "mobile",
  "value": "+1 (415) 859-4366"
}
]' \
"http://localhost:8080/openidm/system/workday/employee/3aa5550b7fe348b98d7b5741afc65534"
{
  "_id": "3aa5550b7fe348b98d7b5741afc65534",
  "title": "Vice President, Human Resources",
  "country": "USA",
  "postalCode": "94111",
  "userID": "lmcneil",
  "hireDate": "2000-01-01-08:00",
  "address": [
    "3939 The Embarcadero"
  ],
  "state": "California",
  "postalAddress": "3939 The Embarcadero\nSan Francisco, CA 94111\nUnited States of America",
  "addressLastModified": "2011-06-20T13:54:02.023-07:00",
  "firstName": "Logan",
  "gender": "Female",
  "employeeID": "21001",
  "managerID": "21431",
  "email": "logan.mcneil@workday.net",
  "city": "San Francisco",
  "preferredName": "Logan McNeil",
  "birthDate": "1971-05-25-07:00",
  "active": true,
  "preferredFirstName": "Logan",
  "employee": true,
  "workerType": "Full time",
  "positionEffectiveDate": "2016-06-01-07:00",
  "preferredLastName": "McNeil",
  "dateActivated": "2000-01-01-08:00",
  "legalName": "Logan McNeil",
  "lastName": "McNeil",
  "mobile": [
    "+1 (415) 859-4366"
  ],
  "terminated": false
}
```

Implementation Specifics

For PATCH requests, a connector can potentially add, remove, or replace an attribute value. The Workday connector does not implement the add or remove operations, so a PATCH request always replaces the entire attribute value with the new value.

OpenICF Interfaces Implemented by the Workday Connector

The Workday Connector implements the following OpenICF interfaces.

Schema

Describes the object types, operations, and options that the connector supports.

Script on Connector

Enables an application to run a script in the context of the connector. Any script that runs on the connector has the following characteristics:

- The script runs in the same execution environment as the connector and has access to all the classes to which the connector has access.
- The script has access to a `connector` variable that is equivalent to an initialized instance of the connector. At a minimum, the script can access the connector configuration.
- The script has access to any script-arguments passed in by the application.

Search

Searches the target resource for all objects that match the specified object class and filter.

Sync

Polls the target resource for synchronization events, that is, native changes to objects on the target resource.

Test

Tests the connector configuration. Testing a configuration checks all elements of the environment that are referred to by the configuration are available. For example, the connector might make a physical connection to a host that is specified in the configuration to verify that it exists and that the credentials that are specified in the configuration are valid.

This operation might need to connect to a resource, and, as such, might take some time. Do not invoke this operation too often, such as before every provisioning operation. The test operation is not intended to check that the connector is alive (that is, that its physical connection to the resource has not timed out).

You can invoke the test operation before a connector configuration has been validated.

Update

Updates (modifies or replaces) objects on a target resource.

Workday Connector Configuration

The Workday Connector has the following configurable properties.

Configuration properties

Property	Type	Default	Encrypted ^a	Required ^b
<code>includeManagementChainDataForWorkers</code>	Boolean	true		No
Description is not available				
<code>includeOrganizationsForWorkers</code>	Boolean	true		No
Description is not available				
<code>includePersonalInformationForWorkers</code>	Boolean	true		No
Description is not available				
<code>excludeCostCentersForWorkers</code>	Boolean	false		No
Description is not available				
<code>excludeCustomOrganizationsForWorkers</code>	Boolean	true		No
Description is not available				
<code>includeRolesForWorkers</code>	Boolean	false		No
Description is not available				
<code>includeStaffingRestrictionsDataForWorkers</code>	Boolean	false		No
Description is not available				
<code>excludeMatrixOrganizationsForWorkers</code>	Boolean	true		No
Description is not available				
<code>includeEmploymentInformationForWorkers</code>	Boolean	true		No
Description is not available				
<code>includeAccountProvisioningForWorkers</code>	Boolean	false		No
Description is not available				
<code>excludeBusinessUnitHierarchiesForWorkers</code>	Boolean	true		No
Description is not available				
<code>includeRelatedPersonsForWorkers</code>	Boolean	false		No
Description is not available				
<code>includePhotoForWorkers</code>	Boolean	false		No
Description is not available				
<code>excludeSupervisoryOrganizationsForWorkers</code>	Boolean	true		No
Description is not available				

Property	Type	Default	Encrypted ^a	Required ^b
<code>excludeTeamsForWorkers</code>	Boolean	false		No
Description is not available				
<code>includeTransactionLogDataForWorkers</code>	Boolean	true		No
Description is not available				
<code>includeSupervisoryDataForOrganizations</code>	Boolean	false		No
Description is not available				
<code>excludeCompaniesForWorkers</code>	Boolean	false		No
Description is not available				
<code>includeAdditionalJobsForWorkers</code>	Boolean	false		No
Description is not available				
<code>excludeBusinessUnitsForWorkers</code>	Boolean	false		No
Description is not available				
<code>includeHierarchyDataForOrganizations</code>	Boolean	false		No
Description is not available				
<code>includeEmployeeContractDataForWorkers</code>	Boolean	false		No
Description is not available				
<code>includeUserAccountForWorkers</code>	Boolean	true		No
Description is not available				
<code>excludeRegionsForWorkers</code>	Boolean	false		No
Description is not available				
<code>includeRolesDataForOrganizations</code>	Boolean	false		No
Description is not available				
<code>includeMultipleManagersInManagement</code>	Boolean	false		No
Description is not available				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

Basic Configuration Properties

Property	Type	Default	Encrypted ^a	Required ^b
hostname	String	null		Yes
The hostname for the Workday service. Example: https://[workday hostname]/ccx/service/[workday tenant]/. You need to configure the bracketed Workday hostname and tenant name to successfully connect to the proper instance.				
tenant	String	null		Yes
The tenant in URL for the Workday service. For example: https://[workday hostname]/ccx/service/[workday tenant]/. You need to configure the bracketed Workday hostname and tenant name to successfully connect to the proper instance.				
username	String	null		Yes
The user name for logging into the Workday service. It will be concatenated with the tenant name (user@tenant)				
password	GuardedString	null	Yes	Yes
The user password for logging into the Workday service				
excludeInactiveWorkers	boolean	false		No
Excludes from the response terminated employees or contingent workers whose contracts have ended (defaults to false)				
excludeContingentWorkers	boolean	false		No
Excludes contingent workers from inclusion in a query response.				
excludeEmployees	boolean	false		No
Excludes employees from inclusion in a query response.				
connectionTimeout	int	30		No
Specifies the amount of time, in seconds, that the client will attempt to establish a connection before it times out. The default is 30 seconds). Set to 0 for no timeout.				
receiveTimeout	int	60		No
Specifies the amount of time, in seconds, that the client will wait for a response before it times out. The default is 60. Set to 0 for no timeout.				
pageSize	long	100		No
Set the page size used for search operations (defaults to 100).				
proxyHost	String	null		No
If defined the connection to Workday will go through this HTTP proxy server				
proxyPort	int	8080		No

Property	Type	Default	Encrypted ^a	Required ^b
The HTTP proxy server port number (defaults to 8080).				
<code>xslTransformer</code>	File	null		No
The file path to the XSL File to get the custom attributes				
<code>asOfEffectiveDate</code>	String	null		No
Optional configuration of Response_Filter/As_Of_Effective_Date element. Valid values are: Date (http://www.w3.org/TR/xpath-functions/#date-time-values http://www.w3.org/TR/xmlschema-2/#dateTime-order) or Duration (http://www.w3.org/TR/xpath-functions/#dt-dayTimeDuration). If set to Duration, the effective date is calculated as current date + duration.				
<code>effectiveFrom</code>	String	null		No
Set the Get_Workers_Request/Request_Criteria/Transaction_Log_Criteria_Data/Transaction_Date_Range_Data/Effective_From for every outbound query request. Valid value could be Date (http://www.w3.org/TR/xpath-functions/#date-time-values http://www.w3.org/TR/xmlschema-2/#dateTime-order) or string Today representing the current time of the request.				
<code>effectiveThrough</code>	String	null		No
Set the Get_Workers_Request/Request_Criteria/Transaction_Log_Criteria_Data/Transaction_Date_Range_Data/Effective_Through for every outbound query request. Valid value could be Date (http://www.w3.org/TR/xpath-functions/#date-time-values http://www.w3.org/TR/xmlschema-2/#dateTime-order) or Duration (http://www.w3.org/TR/xpath-functions/#dt-dayTimeDuration)				
<code>externalFieldAndParameterCriteria</code>	String[]	null		No
A list of external fields to add to the search/query criteria.				

^a Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

^b A list of operations in this column indicates that the property is required for those operations.

Chapter 3

Configure Connectors

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

Connectors are configured through the ICF provisioner service. Each connector configuration is stored in a file in your project's `conf/` directory, and accessible over REST at the `openidm/conf` endpoint. Connector configuration files are named `project-dir/conf/provisioner.openicf-name` where `name` corresponds to the name of the connector.

If you are creating your own connector configuration files, *do not include additional dash characters (-) in the connector name*, as this might cause problems with the OSGi parser. For example, the name `provisioner.openicf-hrdb.json` is fine. The name `provisioner.openicf-hr-db.json` is not.

You can create a connector configuration in the following ways:

- Start with the sample provisioner files in the `/path/to/openidm/samples/example-configurations/provisioners` directory. For more information, see "Sample Provisioner Files".
- Set up connectors with the help of the Admin UI. Log in to the Admin UI at `https://localhost:8443/admin`, then continue with the process described in "Creating Connector Configurations With the Admin UI".
- Use the service that IDM exposes through the REST interface to create basic connector configuration files. For more information, see "Configure Connectors Over REST".
- Use the `cli.sh` or `cli.bat` scripts to generate a basic connector configuration. For more information, see "`configureconnector`" in the *Setup Guide*.

Sample Provisioner Files

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

A number of sample connector configurations are available in the `openidm/samples/example-configurations/provisioners` directory. To use these connector configurations, edit the configuration files as required, and copy them to your project's `conf` directory.

The following example shows a high-level connector configuration. The individual configuration objects are described in detail later in this section:

```
{
  "connectorRef"           : connector-ref-object,
  "producerBufferSize"    : integer,
  "connectorPoolingSupported" : boolean, true/false,
  "poolConfigOption"      : pool-config-option-object,
  "operationTimeout"      : operation-timeout-object,
  "configurationProperties" : configuration-properties-object,
  "syncFailureHandler"    : sync-failure-handler-object,
  "resultsHandlerConfig"  : results-handler-config-object,
  "excludeUnmodified"     : boolean, true/false,
  "objectTypes"           : object-types-object,
  "operationOptions"      : operation-options-object
}
```

Creating Connector Configurations With the Admin UI

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

To configure connectors in the Admin UI, select **Configure > Connector**. If your project has an existing connector configuration (for example, if you have started IDM with one of the sample configurations) click on that connector to edit it. If you're starting with a new project, click **New Connector** to configure a new connector.

The connectors displayed on the **Connectors** page reflect the provisioner files that are in your project's `conf/` directory. To add a new connector configuration, you can also copy a provisioner file from the `/path/to/openidm/samples/example-configurations/provisioners` directory, then edit it to fit your deployment.

When you add a new connector, the **Connector Type** dropdown list reflects the connector `.jar` files that are in the `/path/to/openidm/connectors` directory. You can have more than one connector configuration for a specific connector type. For example, you might use the LDAP connector to set up

two connector configurations—one to an Active Directory server and one to a ForgeRock Directory Services (DS) instance.

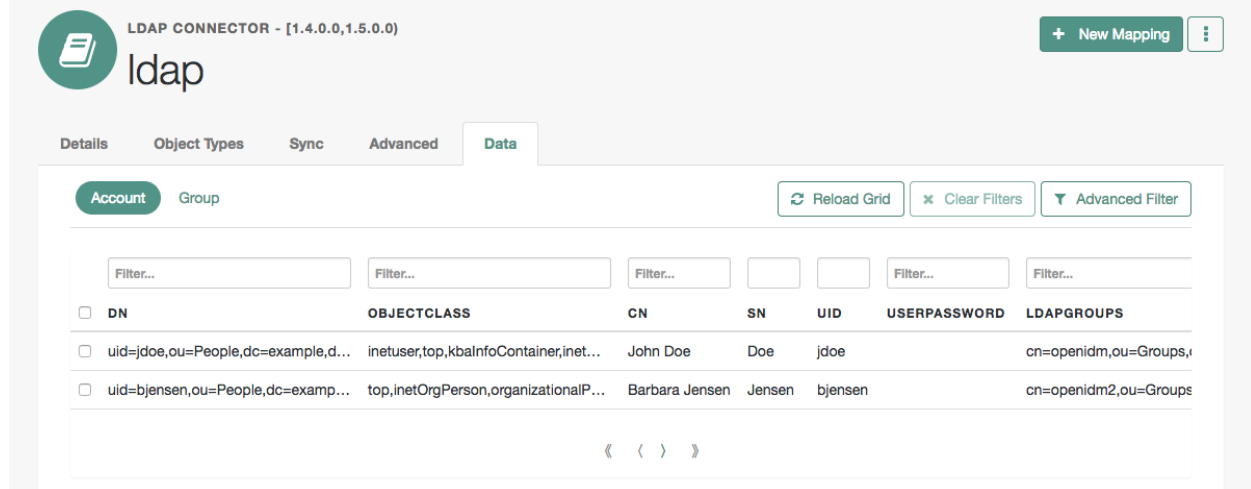
The Connector Types listed here do not include all supported connectors. The *scripted* connectors (such as scripted Groovy, scripted REST, scripted SQL, and PowerShell) are not available in the list of connector types. In general, the scripted connectors require extensive custom configuration changes, and a single HTML template to cover all possible permutations is not feasible. To add a scripted connector configuration, copy one of the example provisioner files in `/path/to/openidm/samples/example-configurations/provisioners` into your project's `conf` directory and edit the configuration directly in the provisioner file.

Additional connectors are available from the [ForgeRock BackStage download site](#), and can be downloaded to the `/path/to/openidm/connectors` directory. For connectors that are not bundled with IDM, the UI displays a generic template, based on the schema provided by the connector.

The tabs on the connector configuration screens correspond to the objects and properties described in the remaining sections of this chapter.

When a connector configuration is complete, and IDM is able to establish the connection to the remote resource, the Data tab displays the objects in that remote resource. For example, the following image shows the contents of a connected LDAP resource:

Data Tab For a Connected LDAP Resource



LDAP CONNECTOR - [1.4.0.0,1.5.0.0]

ldap

Details Object Types Sync Advanced **Data**

Account Group

Filter... Filter... Filter... Filter... Filter... Filter...

☐ DN
 ☐ uid=jdoe,ou=People,dc=example,d...
 ☐ uid=bjensen,ou=People,dc=examp...

OBJECTCLASS	CN	SN	UID	USERPASSWORD	LDAPGROUPS
inetuser,top,kbainfoContainer,inet...	John Doe	Doe	jdoe		cn=openidm,ou=Groups,i
top,inetOrgPerson,organizationalP...	Barbara Jensen	Jensen	bjensen		cn=openidm2,ou=Groups

« ‹ › »

You can search through these objects with either the Basic Filter shown in each column, or the Advanced Filter option, which lets you build many of the queries shown in "Define and Call Data Queries" in the *Object Modeling Guide*.

Configure Connectors Over REST

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

To create a new connector configuration over REST, follow these steps:

1. List the available connectors.
2. Generate the core configuration.
3. Add the target system properties, then connect to the target system to generate the final configuration.
4. Submit the final configuration to IDM.

This procedure walks you through creating a connector configuration over REST, for a CSV file connector.

1. List the available connectors.

In a default IDM installation, the available connectors are installed in the `openidm/connectors` directory. If you are using a remote connector server, additional connectors might be available in the `openicf/connectors` directory on the remote server.

Run the following command to list the available connectors:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system?_action=availableConnectors"
```

On a default IDM installation, this command returns the following output:

```
{
  "connectorRef": [
    {
      "displayName": "SSH Connector",
      "bundleVersion": "1.5.20.8",
      "systemType": "provisioner.openicf",
      "bundleName": "org.forgerock.openicf.connectors.ssh-connector",
      "connectorName": "org.forgerock.openicf.connectors.ssh.SSHConnector"
    },
    {
      "displayName": "ServiceNow Connector",
      "bundleVersion": "1.5.20.11",
      "systemType": "provisioner.openicf",
      "bundleName": "org.forgerock.openicf.connectors.servicenow-connector",
      "connectorName": "org.forgerock.openicf.connectors.servicenow.ServiceNowConnector"
    }
  ]
}
```

```
{
  "displayName": "Scripted SQL Connector",
  "bundleVersion": "1.5.20.8",
  "systemType": "provisioner.openicf",
  "bundleName": "org.forgerock.openicf.connectors.scriptedsql-connector",
  "connectorName": "org.forgerock.openicf.connectors.scriptedsql.ScriptedSQLConnector"
},
{
  "displayName": "Scripted REST Connector",
  "bundleVersion": "1.5.20.11",
  "systemType": "provisioner.openicf",
  "bundleName": "org.forgerock.openicf.connectors.scriptedrest-connector",
  "connectorName": "org.forgerock.openicf.connectors.scriptedrest.ScriptedRESTConnector"
},
{
  "displayName": "Scim Connector",
  "bundleVersion": "1.5.20.12",
  "systemType": "provisioner.openicf",
  "bundleName": "org.forgerock.openicf.connectors.scim-connector",
  "connectorName": "org.forgerock.openicf.connectors.scim.ScimConnector"
},
{
  "displayName": "Salesforce Connector",
  "bundleVersion": "1.5.20.11",
  "systemType": "provisioner.openicf",
  "bundleName": "org.forgerock.openicf.connectors.salesforce-connector",
  "connectorName": "org.forgerock.openicf.connectors.salesforce.SalesforceConnector"
},
{
  "displayName": "MongoDB Connector",
  "bundleVersion": "1.5.20.8",
  "systemType": "provisioner.openicf",
  "bundleName": "org.forgerock.openicf.connectors.mongodb-connector",
  "connectorName": "org.forgerock.openicf.connectors.mongodb.MongoDBConnector"
},
{
  "displayName": "Marketo Connector",
  "bundleVersion": "1.5.20.11",
  "systemType": "provisioner.openicf",
  "bundleName": "org.forgerock.openicf.connectors.marketo-connector",
  "connectorName": "org.forgerock.openicf.connectors.marketo.MarketoConnector"
},
{
  "displayName": "LDAP Connector",
  "bundleVersion": "1.5.20.12",
  "systemType": "provisioner.openicf",
  "bundleName": "org.forgerock.openicf.connectors.ldap-connector",
  "connectorName": "org.identityconnectors.ldap.LdapConnector"
},
{
  "displayName": "Kerberos Connector",
  "bundleVersion": "1.5.20.8",
  "systemType": "provisioner.openicf",
  "bundleName": "org.forgerock.openicf.connectors.kerberos-connector",
  "connectorName": "org.forgerock.openicf.connectors.kerberos.KerberosConnector"
},
{
  "displayName": "Scripted Poolable Groovy Connector",
  "bundleVersion": "1.5.5.0",
```

```

    "systemType": "provisioner.openicf",
    "bundleName": "org.forgerock.openicf.connectors.groovy-connector",
    "connectorName": "org.forgerock.openicf.connectors.groovy.ScriptedPoolableConnector"
  },
  {
    "displayName": "Scripted Groovy Connector",
    "bundleVersion": "1.5.20.8",
    "systemType": "provisioner.openicf",
    "bundleName": "org.forgerock.openicf.connectors.groovy-connector",
    "connectorName": "org.forgerock.openicf.connectors.groovy.ScriptedConnector"
  },
  {
    "displayName": "GoogleApps Connector",
    "bundleVersion": "1.5.20.12",
    "systemType": "provisioner.openicf",
    "bundleName": "org.forgerock.openicf.connectors.googleapps-connector",
    "connectorName": "org.forgerock.openicf.connectors.googleapps.GoogleAppsConnector"
  },
  {
    "displayName": "Database Table Connector",
    "bundleVersion": "1.5.20.8",
    "systemType": "provisioner.openicf",
    "bundleName": "org.forgerock.openicf.connectors.databasetable-connector",
    "connectorName": "org.identityconnectors.databasetable.DatabaseTableConnector"
  },
  {
    "displayName": "CSV File Connector",
    "bundleVersion": "1.5.20.11",
    "systemType": "provisioner.openicf",
    "bundleName": "org.forgerock.openicf.connectors.csvfile-connector",
    "connectorName": "org.forgerock.openicf.csvfile.CSVFileConnector"
  },
  {
    "displayName": "Adobe Marketing Cloud Connector",
    "bundleVersion": "1.5.20.11",
    "systemType": "provisioner.openicf",
    "bundleName": "org.forgerock.openicf.connectors.adobecm-connector",
    "connectorName": "org.forgerock.openicf.acm.ACMConnector"
  }
]
}

```

2. Generate a core configuration.

Locate the connector to configure from the previous step's output, and copy the JSON object to insert as the value of the `"connectorRef"` property in the `data` payload of the following command.

This example generates a core configuration for the CSV file connector:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request POST \
--data '{"connectorRef":
{
  "systemType": "provisioner.openicf",
  "bundleName": "org.forgerock.openicf.connectors.csvfile-connector",
  "connectorName": "org.forgerock.openicf.csvfile.CSVFileConnector",
  "displayName": "CSV File Connector",
  "bundleVersion": "1.5.20.11"
}
}' \
"http://localhost:8080/openidm/system?_action=createCoreConfig"
```

The command returns a connector configuration, similar to the following:

```
{
  "connectorRef": {
    "systemType": "provisioner.openicf",
    "bundleName": "org.forgerock.openicf.connectors.csvfile-connector",
    "connectorName": "org.forgerock.openicf.csvfile.CSVFileConnector",
    "displayName": "CSV File Connector",
    "bundleVersion": "1.5.20.11"
  },
  "poolConfigOption": {
    "maxObjects": 10,
    "maxIdle": 10,
    "maxWait": 150000,
    "minEvictableIdleTimeMillis": 120000,
    "minIdle": 1
  },
  "resultsHandlerConfig": {
    "enableNormalizingResultsHandler": false,
    "enableFilteredResultsHandler": false,
    "enableCaseInsensitiveFilter": false,
    "enableAttributesToGetSearchResultsHandler": true
  },
  "operationTimeout": {
    "CREATE": -1,
    "UPDATE": -1,
    "DELETE": -1,
    "TEST": -1,
    "SCRIPT_ON_CONNECTOR": -1,
    "SCRIPT_ON_RESOURCE": -1,
    "GET": -1,
    "RESOLVEUSERNAME": -1,
    "AUTHENTICATE": -1,
    "SEARCH": -1,
    "VALIDATE": -1,
    "SYNC": -1,
    "SCHEMA": -1
  },
  "configurationProperties": {
    "headerPassword": "password",
    "spaceReplacementString": "_",

```

```

    "csvFile": null,
    "newlineString": "\n",
    "headerUid": "uid",
    "quoteCharacter": "\"",
    "escapeCharacter": "\\",
    "fieldDelimiter": ",",
    "syncFileRetentionCount": 3
  }
}

```

3. Connect to the target system to generate the final configuration.

The configuration returned in the previous step is not functional. It does not include the required `configurationProperties` that are specific to the target system (such as the host name and port number of the target system, or the `csvFile` for a CSV file connector). It also doesn't include the complete list of `objectTypes` and `operationOptions`.

To connect to the target system, add values for the required `configurationProperties`, and submit the updated configuration in the data payload of the following command.

This example connects to the specified CSV file:

```

curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request POST \
--data '{
  "configurationProperties": {
    "headerPassword": "password",
    "spaceReplacementString": " ",
    "csvFile": "${idm.instance.dir}/data/csvConnectorData.csv",
    "newlineString": "\n",
    "headerUid": "uid",
    "quoteCharacter": "\"",
    "fieldDelimiter": ",",
    "syncFileRetentionCount": 3
  },
  "connectorRef": {
    "systemType": "provisioner.openicf",
    "bundleName": "org.forgerock.openicf.connectors.csvfile-connector",
    "connectorName": "org.forgerock.openicf.csvfile.CSVFileConnector",
    "displayName": "CSV File Connector",
    "bundleVersion": "1.5.20.11"
  },
  "poolConfigOption": {
    "maxObjects": 10,
    "maxIdle": 10,
    "maxWait": 150000,
    "minEvictableIdleTimeMillis": 120000,
    "minIdle": 1
  },
  "resultsHandlerConfig": {
    "enableNormalizingResultsHandler": true,
    "enableFilteredResultsHandler": true,
    "enableCaseInsensitiveFilter": false,

```

```
"enableAttributesToGetSearchResultsHandler": true
},
"operationTimeout": {
  "CREATE": -1,
  "UPDATE": -1,
  "DELETE": -1,
  "TEST": -1,
  "SCRIPT_ON_CONNECTOR": -1,
  "SCRIPT_ON_RESOURCE": -1,
  "GET": -1,
  "RESOLVEUSERNAME": -1,
  "AUTHENTICATE": -1,
  "SEARCH": -1,
  "VALIDATE": -1,
  "SYNC": -1,
  "SCHEMA": -1
}
}' \
"http://localhost:8080/openidm/system?_action=createFullConfig"
```

Note

The single quotes around the JSON object in the `--data` parameter prevent the command from being executed when a new line is encountered in the content. You can therefore include line feeds for readability.

With this command, IDM connects to the target resource, and attempts to read the schema, if it is available. It then iterates through the schema objects and attributes, and creates JSON representations of the supported objects and operations. The command output includes the JSON payload that you submitted, along with the `operationOptions` and `objectTypes`.

Important

Because IDM produces a full property set for all attributes and all object types in the schema, the resulting configuration can be very large. For an LDAP server, for example, IDM can generate a configuration containing several tens of thousands of lines. It might be useful to reduce the schema on the external resource to a minimum before you run the `createFullConfig` command.

4. When you have the final configuration, use a PUT request to add it to the IDM configuration, in the JSON payload of the following command:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request PUT \
--data '{complete-configuration}' \
"http://localhost:8080/openidm/config/provisioner.openicf-connector-name"
```

Alternatively, you can save the complete configuration in a file named `provisioner.openicf-connector-name.json`, and place the file in the `conf` directory of your project.

Setting the Connector Reference Properties

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

The following example shows a connector reference object:

```
"connectorRef" : {  
  "bundleName"      : "org.forgerock.openicf.connectors.csvfile-connector",  
  "bundleVersion"   : "[1.5.0.0,1.6.0.0)",  
  "connectorName"   : "org.forgerock.openicf.csvfile.CSVFileConnector",  
  "connectorHostRef" : "csv"  
}
```

`bundleName`

string, required

The `ConnectorBundle-Name` of the ICF connector.

`bundleVersion`

string, required

The `ConnectorBundle-Version` of the ICF connector. The value can be a single version (such as `1.4.0.0`) or a range of versions, which lets you support multiple connector versions in a single project.

You can specify a range of versions as follows:

- `[1.1.0.0,1.4.0.0]` indicates that all connector versions from 1.1 to 1.4, inclusive, are supported.
- `[1.1.0.0,1.4.0.0)` indicates that all connector versions from 1.1 to 1.4, including 1.1 but excluding 1.4, are supported.
- `(1.1.0.0,1.4.0.0]` indicates that all connector versions from 1.1 to 1.4, excluding 1.1 but including 1.4, are supported.
- `(1.1.0.0,1.4.0.0)` indicates that all connector versions from 1.1 to 1.4, exclusive, are supported.

When a range of versions is specified, IDM uses the latest connector that is available within that range. If your project requires a specific connector version, you must explicitly state the version in your connector configuration file, or constrain the range to address only the version that you need.

connectorName

string, required

The connector implementation class name.

connectorHostRef

string, optional

If the connector runs remotely, the value of this field must match the `name` field of the `RemoteConnectorServers` object in the connector server configuration file (`provisioner.openicf.connectorinfoprovider.json`). For example:

```
...
  "remoteConnectorServers" :
  [
    {
      "name" : "dotnet",
      ...
    }
  ]
...
```

If the connector runs locally, the value of this field can be one of the following:

- If the connector .jar is installed in `openidm/connectors/`, the value must be `"#LOCAL"`. This is currently the default, and recommended location.
- If the connector .jar is installed in `openidm/bundle/` (not recommended), the value must be `"osgi:service/org.forgerock.openicf.framework.api.osgi.ConnectorManager"`.

Setting the Pool Configuration

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

The `poolConfigOption` specifies the pool configuration for poolable connectors only (connectors that have `"connectorPoolingSupported" : true`). Non-poolable connectors ignore this parameter.

The following example shows a pool configuration option object for a poolable connector:

```
{
  "maxObjects"           : 10,
  "maxIdle"              : 10,
  "maxWait"              : 150000,
  "minEvictableIdleTimeMillis" : 120000,
  "minIdle"              : 1
}
```

maxObjects

The maximum number of idle and active instances of the connector.

maxIdle

The maximum number of idle instances of the connector.

maxWait

The maximum time, in milliseconds, that the pool waits for an object before timing out. A value of 0 means that there is no timeout.

minEvictableIdleTimeMillis

The maximum time, in milliseconds, that an object can be idle before it is removed. A value of 0 means that there is no idle timeout.

minIdle

The minimum number of idle instances of the connector.

Setting the Operation Timeouts

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

The operation timeout property enables you to configure timeout values per operation type. By default, no timeout is configured for any operation type. A sample configuration follows:

```
{
  "CREATE"           : -1,
  "TEST"             : -1,
  "AUTHENTICATE"     : -1,
  "SEARCH"           : -1,
  "VALIDATE"         : -1,
  "GET"              : -1,
  "UPDATE"           : -1,
  "DELETE"           : -1,
  "SCRIPT_ON_CONNECTOR" : -1,
  "SCRIPT_ON_RESOURCE" : -1,
  "SYNC"             : -1,
  "SCHEMA"           : -1
}
```

operation-name

Timeout in milliseconds

A value of `-1` disables the timeout.

Setting the Connection Configuration

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

The `configurationProperties` object specifies the configuration for the connection between the connector and the resource, and is therefore resource-specific.

The following example shows a configuration properties object for the default CSV sample resource connector:

```
"configurationProperties" : {
  "csvFile" : "&{idm.instance.dir}/data/csvConnectorData.csv"
}
```

property

Individual properties depend on the type of connector.

Setting the Synchronization Failure Configuration

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

The `syncFailureHandler` object specifies what should happen if a liveSync operation reports a failure for an operation. The following example shows a synchronization failure configuration:

```
{
  "maxRetries" : 5,
  "postRetryAction" : "logged-ignore"
}
```

maxRetries

positive integer or `-1`, required

The number of attempts that IDM should make to process a failed modification. A value of zero indicates that failed modifications should not be reattempted. In this case, the post retry action is executed immediately when a liveSync operation fails. A value of `-1` (or omitting the `maxRetries`

property, or the entire `syncFailureHandler` object) indicates that failed modifications should be retried an infinite number of times. In this case, no post retry action is executed.

postRetryAction

string, required

The action that should be taken if the synchronization operation fails after the specified number of attempts. The post retry action can be one of the following:

- `logged-ignore` - IDM ignores the failed modification, and logs its occurrence.
- `dead-letter-queue` - IDM saves the details of the failed modification in a table in the repository (accessible over REST at `repo/synchronisation/deadLetterQueue/provisioner-name`).
- `script` specifies a custom script that should be executed when the maximum number of retries has been reached.

For more information, see "Configure the LiveSync Retry Policy" in the *Synchronization Guide*.

Configuring How Results Are Handled

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

The `resultsHandlerConfig` object specifies how OpenICF returns results. These configuration properties do not apply to all connectors and depend on the interfaces that are implemented by each connector. For information about the interfaces that connectors support, see the *Connectors Guide*.

The following example shows a results handler configuration object:

```
"resultsHandlerConfig" : {
  "enableNormalizingResultsHandler" : true,
  "enableFilteredResultsHandler" : false,
  "enableCaseInsensitiveFilter" : false,
  "enableAttributesToGetSearchResultsHandler" : false
}
```

enableNormalizingResultsHandler

boolean, false by default

When this property is enabled, ICF normalizes returned attributes to ensure that they are filtered consistently. If the connector implements the attribute normalizer interface, enable the interface by setting this property to `true`. If the connector does not implement the attribute normalizer interface, the value of this property has no effect.

enableFilteredResultsHandler

boolean, false by default

Most connectors use the filtering and search capabilities of the remote connected system. In these cases, you can leave this property set to **false**. If the connector does not use the remote system's filtering and search capabilities, you *must* set this property to **true**.

All the non-scripted connectors, except for the CSV connector, use the filtering mechanism of the remote system. In the case of the CSV connector, the remote resource has no filtering mechanism, so you must set **enableFilteredResultsHandler** to **true**. For the scripted connectors, the setting will depend on how you have implemented the connector.

enableCaseInsensitiveFilter

boolean, false by default

This property applies only if **enableFilteredResultsHandler** is set to **true**. The filtered results handler is case-sensitive by default. For example, a search for `lastName = "Jensen"` will not match a stored user with `lastName : jensen`. When the filtered results handler is enabled, you can use this property to enable case-insensitive filtering. If you leave this property set to **false**, searches on that resource will be case-sensitive.

enableAttributesToGetSearchResultsHandler

boolean, false by default

By default, IDM determines which attributes should be retrieved in a search. If you set this property to **true**, the ICF framework removes *all* attributes from the READ/QUERY response, except for those that are specifically requested. For performance reasons, you should set this property to **false** for local connectors and to **true** for remote connectors.

Specifying What Attributes are Updated

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

The **excludeUnmodified** property determines which properties are updated during synchronization. When this property is set to **true**, synchronization operations update *only* the modified properties on a target resource, rather than the whole target object. In the default LDAP provisioner files, **excludeUnmodified** is set to **true**. This means that unmodified attributes are excluded by default during update operations.

Specifying the Supported Object Types

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

The `objectTypes` configuration specifies the object types (user, group, account, and so on) that are supported by the connector. The object names that you define here determine how the object is accessed in the URI. For example:

```
system/systemName/objectType
```

This configuration is based on the JSON Schema with the extensions described in the following section.

Attribute names that start or end with `__` are regarded as *special attributes* by OpenICF. The purpose of the special attributes in ICF is to enable someone who is developing a *new* connector to create a contract regarding how a property can be referenced, regardless of the application that is using the connector. In this way, the connector can map specific object information between an arbitrary application and the resource, without knowing how that information is referenced in the application.

These attributes have no specific meaning in the context of IDM, although some of the connectors that are bundled with IDM use these attributes. The generic LDAP connector, for example, can be used with ForgeRock Directory Services (DS), Active Directory, OpenLDAP, and other LDAP directories. Each of these directories might use a different attribute name to represent the same type of information. For example, Active Directory uses `unicodePassword` and DS uses `userPassword` to represent the same thing, a user's password. The LDAP connector uses the special OpenICF `__PASSWORD__` attribute to abstract that difference. In the same way, the LDAP connector maps the `__NAME__` attribute to an LDAP `dn`.

The ICF `__UID__` is a special case. The `__UID__` *must not* be included in the IDM configuration or in any update or create operation. This attribute denotes the unique identity attribute of an object and IDM always maps it to the `_id` of the object.

The following excerpt shows the configuration of an `account` object type:

```
{
  "account" : {
    "$schema" : "http://json-schema.org/draft-03/schema",
    "id" : "__ACCOUNT__",
    "type" : "object",
    "nativeType" : "__ACCOUNT__",
    "absentIfEmpty" : false,
    "absentIfNull" : true,
    "properties" : {
      "name" : {
        "type" : "string",
        "nativeName" : "__NAME__",
        "nativeType" : "JAVA_TYPE_PRIMITIVE_LONG",
        "flags" : [
```

```

        "NOT_CREATABLE",
        "NOT_UPDATEABLE",
        "NOT_READABLE",
        "NOT_RETURNED_BY_DEFAULT"
    ],
    "groups" : {
        "type" : "array",
        "items" : {
            "type" : "string",
            "nativeType" : "string"
        },
        "nativeName" : "__GROUPS__",
        "nativeType" : "string",
        "flags" : [
            "NOT_RETURNED_BY_DEFAULT"
        ]
    },
    "givenName" : {
        "type" : "string",
        "nativeName" : "givenName",
        "nativeType" : "string"
    },
    },
}
}
}

```

ICF supports an `__ALL__` object type that ensures that objects of every type are included in a synchronization operation. The primary purpose of this object type is to prevent synchronization errors when multiple changes affect more than one object type.

For example, imagine a deployment synchronizing two external systems. On system A, the administrator creates a user, `jdoe`, then adds the user to a group, `engineers`. When these changes are synchronized to system B, if the `__GROUPS__` object type is synchronized first, the synchronization will fail, because the group contains a user that does not yet exist on system B. Synchronizing the `__ALL__` object type ensures that user `jdoe` is created on the external system before he is added to the group `engineers`.

The `__ALL__` object type is assumed by default - you do not need to declare it in your provisioner configuration file. If it is not declared, the object type is named `__ALL__`. If you want to map a different name for this object type, declare it in your provisioner configuration. The following excerpt from a sample provisioner configuration uses the name `allobjects`:

```

"objectTypes": {
  "allobjects": {
    "$schema": "http://json-schema.org/draft-03/schema",
    "id": "__ALL__",
    "type": "object",
    "nativeType": "__ALL__"
  },
  ...
}

```

A liveSync operation invoked with no object type assumes an object type of `__ALL__`. For example, the following call invokes a liveSync operation on all defined object types in an LDAP system:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system/ldap?_action=liveSync"
```

Note

Using the `__ALL__` object type requires a mechanism to ensure the order in which synchronization changes are processed. Servers that use the `cn=changelog` mechanism to order sync changes, such as ForgeRock Directory Services (DS), Oracle DSEE, and the legacy Sun Directory Server, cannot use the `__ALL__` object type by default. Such servers must be forced to use timestamps to order their sync changes. For these LDAP server types, set `useTimestampsForSync` to `true` in the provisioner configuration.

LDAP servers that use timestamps rather than change logs (such as Active Directory GCs and OpenLDAP) can use the `__ALL__` object type without any additional configuration. Active Directory and Active Directory LDS, which use Update Sequence Numbers, can also use the `__ALL__` object type without additional configuration.

Adding Objects and Properties Using the UI

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

To add object types and properties to a connector configuration by using the Admin UI, select **Configure > Connectors**. Select the connector that you want to change, then select the **Object Types** tab.

In the case of the LDAP connector, the connector reads the schema from the remote resource to determine the object types and properties that can be added to its configuration. When you select one of these object types, you can think of it as a template. Edit the basic object type, as required, to suit your deployment.

To add a property to an object type, select the **Edit** icon next to the object type, then select **Add Property**.

Extending the Object Type Configuration

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

`nativeType`

string, optional

The native ICF object type.

The list of supported native object types is dependent on the resource, or on the connector. For example, an LDAP connector might have object types such as `__ACCOUNT__` and `__GROUP__`.

Specifying the Behavior For Empty Attributes

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

The `absentIfEmpty` and `absentIfNull` object class properties enable you to specify how attributes are handled during synchronization if their values are null (for single-valued attributes) or empty (for multi-valued attributes). You can set these properties per object type.

By default, these properties are set as follows:

```
"absentIfEmpty" : false
```

Multi-valued attributes whose values are empty are included in the resource response during synchronization.

```
"absentIfNull" : true
```

Single-valued attributes whose values are null are removed from the resource response during synchronization.

Extending the Property Type Configuration

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

nativeType

string, optional

The native ICF attribute type.

The following native types are supported:

```

JAVA_TYPE_BIGDECIMAL
JAVA_TYPE_BIGINTEGER
JAVA_TYPE_BYTE
JAVA_TYPE_BYTE_ARRAY
JAVA_TYPE_CHAR
JAVA_TYPE_CHARACTER
JAVA_TYPE_DATE
JAVA_TYPE_DOUBLE
JAVA_TYPE_FILE
JAVA_TYPE_FLOAT
JAVA_TYPE_GUARDEDBYTEARRAY
JAVA_TYPE_GUARDEDSTRING
JAVA_TYPE_INT
JAVA_TYPE_INTEGER
JAVA_TYPE_LONG
JAVA_TYPE_OBJECT
JAVA_TYPE_PRIMITIVE_BOOLEAN
JAVA_TYPE_PRIMITIVE_BYTE
JAVA_TYPE_PRIMITIVE_DOUBLE
JAVA_TYPE_PRIMITIVE_FLOAT
JAVA_TYPE_PRIMITIVE_LONG
JAVA_TYPE_STRING

```

Note

The `JAVA_TYPE_DATE` property is deprecated. Functionality may be removed in a future release. This property-level extension is an alias for `string`. Any dates assigned to this extension should be formatted per ISO 8601.

`nativeName`

string, optional

The native ICF attribute name.

`flags`

string, optional

The native ICF attribute flags. ICF supports the following attribute flags:

- `MULTIVALUED` - specifies that the property can be multivalued.

For multi-valued properties, if the property value type is anything other than a `string`, you *must* include an `items` property that declares the data type.

The following example shows the `entries` property of the `authentication` object in a provisioner file. The `entries` property is multi-valued, and its elements are of type `object`:

```
"authentication" : {
  ...
  "properties" : {
    ...
    "entries" : {
      "type" : "object",
      "required" : false,
      "nativeName" : "entries",
      "nativeType" : "object",
      "items" : {
        "type" : "object"
      },
      "flags" : [
        "MULTIVALUED"
      ]
    },
    ...
  },
  ...
}
```

- `NOT_CREATABLE`, `NOT_READABLE`, `NOT_UPDATEABLE`

In some cases, the connector might not support manipulating an attribute because the attribute can only be changed directly on the remote system. For example, if the `name` attribute of an account can only be created by Active Directory, and *never* changed by IDM, you would add `NOT_CREATABLE` and `NOT_UPDATEABLE` to the provisioner configuration for that attribute.

- `NOT_RETURNED_BY_DEFAULT`

Certain attributes such as LDAP groups or other calculated attributes might be expensive to read. To avoid returning these attributes in a default read of the object, unless they are explicitly requested, add the `NOT_RETURNED_BY_DEFAULT` flag to the provisioner configuration for that attribute.

You can also use this flag to prevent properties from being read by default during a synchronization operation. To synchronize changes to a target object, IDM performs an UPDATE rather than a PATCH. This causes *all* attributes that are mapped from the source to the target to be modified when the synchronization is processed (rather than only those attributes that have changed). Although the *value* of a property might not change, the property still registers an update. This behavior can be problematic for properties such as the `password`, which might have restrictions on updating with a similar value. To prevent such properties from being updated during synchronization, set the `NOT_RETURNED_BY_DEFAULT` flag, which effectively prevents the property from being read from the source during the synchronization. For example:

```

    "_PASSWORD_" : {
      "type" : "string",
      "nativeName" : "_PASSWORD_",
      "nativeType" : "JAVA_TYPE_GUARDEDSTRING",
      "flags" : [
        "NOT_RETURNED_BY_DEFAULT"
      ],
      "runAsUser" : true
    }
  }

```

- **REQUIRED** - specifies that the property is required in create operations. This flag sets the **required** property of an attribute as follows:

```

    "required" : true
  }

```

You can configure connectors to enable provisioning of any arbitrary property. For example, the following property definitions would enable you to provision image files, used as avatars, to **account** objects in a system resource. The first definition would work for a single photo encoded as a base64 string. The second definition would work for multiple photos encoded in the same way:

```

"attributeByteArray" : {
  "type" : "string",
  "nativeName" : "attributeByteArray",
  "nativeType" : "JAVA_TYPE_BYTE_ARRAY"
},

"attributeByteArrayMultivalue": {
  "type": "array",
  "items": {
    "type": "string",
    "nativeType": "JAVA_TYPE_BYTE_ARRAY"
  },
  "nativeName": "attributeByteArrayMultivalue"
},

```

Note

Do not use the dash character (-) in property names, like **last-name**. Dashes in names make JavaScript syntax more complex. If you cannot avoid the dash, write `source['last-name']` instead of `source.last-name` in your JavaScript scripts.

Configuring the Operation Options

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

The **operationOptions** object enables you to deny specific operations on a resource. For example, you can use this configuration object to deny **CREATE** and **DELETE** operations on a read-only resource to avoid IDM accidentally updating the resource during a synchronization operation.

The following example defines the options for the "SYNC" operation:

```
"operationOptions" : {
  "SYNC" : {
    "denied" : true,
    "onDeny" : "DO_NOTHING",
    "objectFeatures" : {
      "_ACCOUNT_" : {
        "denied" : true,
        "onDeny" : "THROW_EXCEPTION",
        "operationOptionInfo" : {
          "$schema" : "http://json-schema.org/draft-03/schema",
          "type" : "object",
          "properties" : {
            "_OperationOption-float" : {
              "type" : "number",
              "nativeType" : "JAVA_TYPE_PRIMITIVE_FLOAT"
            }
          }
        }
      },
      "_GROUP_" : {
        "denied" : false,
        "onDeny" : "DO_NOTHING"
      }
    }
  },
  ...
}
```

The ICF Framework supports the following operations:

- AUTHENTICATE
- CREATE
- DELETE
- GET
- RESOLVEUSERNAME
- SCHEMA
- SCRIPT_ON_CONNECTOR
- SCRIPT_ON_RESOURCE
- SEARCH
- SYNC
- TEST
- UPDATE

- `VALIDATE`

For detailed information on these operations, see the [ICF API documentation](#).

The `operationOptions` object has the following configurable properties:

`denied`

boolean, optional

This property prevents operation execution if the value is `true`.

`onDeny`

string, optional

If `denied` is `true`, then the service uses this value. Default value: `DO_NOTHING`.

- `DO_NOTHING`: On operation the service does nothing.
- `THROW_EXCEPTION`: On operation the service throws a `ForbiddenException` exception.

Chapter 4

Remote Connectors

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

In most cases, IDM bundles the connectors required to connect to remote resources, and assumes that the connector will run on the same host as IDM. Sometimes, a connector cannot run on the same host as IDM. This might be for security or network reasons, or because IDM runs in the cloud while the resource is on prem. Connectors that do not run on the same host as IDM are called *remote connectors*. To run remotely, a connector needs a *connector server*, that runs on the same host as the connector. The connector server lets IDM access the connector.

Running connectors remotely requires the following high-level steps:

1. Install a connector server (either .NET or Java) on your remote server.
2. If the connector you want to use is not bundled with the connector server, download it from the ForgeRock BackStage download site, and put the .jar or .dll file on your remote server, in the `/path/to/openicf/connectors/` directory.
3. Install any required connector dependencies on your remote server, in the `/path/to/openicf/lib/` directory.
4. Configure IDM to connect to the remote connector server.

For a list of supported connector server versions, and compatibility between versions, see "IDM / ICF Compatibility Matrix" in the *Release Notes*.

Install a Remote Connector Server

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

There are two types of remote connector servers: .NET and Java.

You need a .NET connector server if you are using the PowerShell connector to connect to an identity store. IDM communicates with the .NET connector server over the network, and the connector server runs the Powershell connector.

You need a Java connector server if your Java connector needs to run in a different JVM to IDM.

These procedures describe how to set up both connector server types on a remote host.

+ *Set Up a .NET Connector Server*

Set Up a .NET Connector Server

The .NET connector server is distributed in two file formats:

- `openicf-version-dotnet.msi` is a wizard that installs the connector server as a Windows service.
 - `openicf-version-dotnet.zip` is just a bundle of the files required to run the connector server.
1. Depending on how you want to install the connector server, download the corresponding file from the [ForgeRock BackStage download site](#).
 2. Follow one of these procedures to install the connector server:

+ *Install the Connector Server as a Service*

1. Double-click the `openicf-version-dotnet.msi` installation file and complete the wizard.

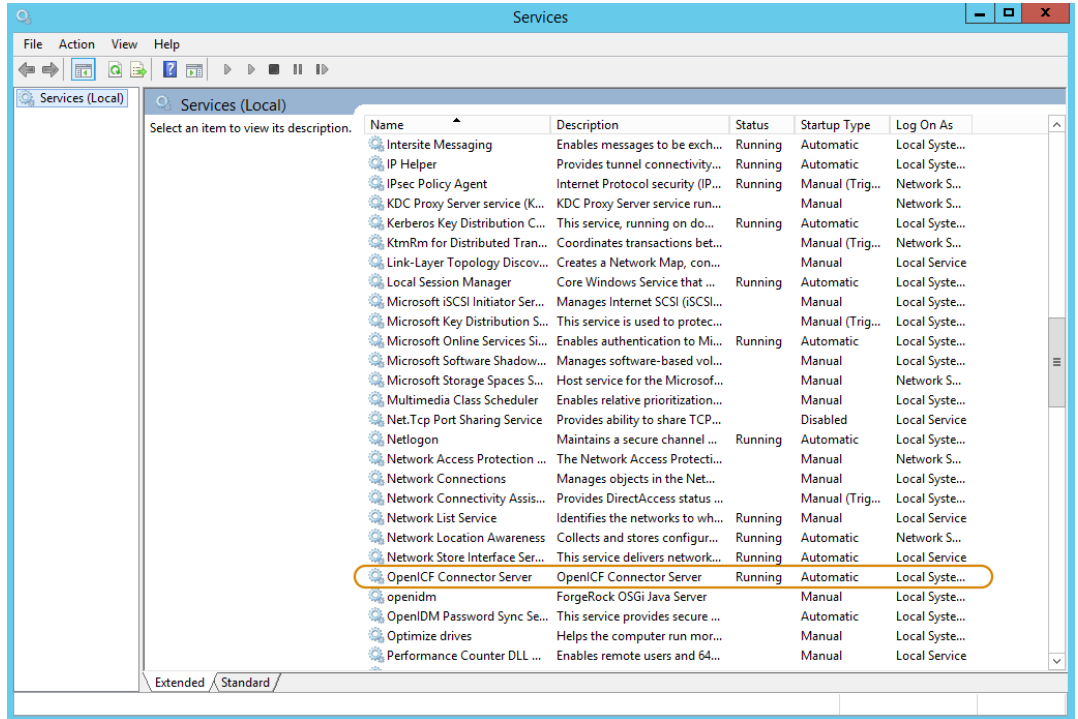
You must run the wizard as a user who has permission to start and stop a Windows service; otherwise, the service will not start.

Select Typical as the Setup Type.

When the wizard has completed, the connector server is installed as a Windows service.

2. Open the Microsoft Services Console and make sure that the connector server is listed there.

The name of the service is `OpenICF Connector Server`, by default.



3. Make sure that the connector server is not currently running. If it is running, use the Microsoft Services Console to stop it.

+ Unpack the Connector Server Zip

1. If you do *not* want to run the connector server as a Windows service, download and extract the `openicf-version-dotnet.zip` file.
2. If you have already extracted the .zip file and then decide to run the connector server as a service, install the service manually with the following command:

```
.\ConnectorServerService.exe /install /serviceName service-name
```

3. At the command prompt, change to the directory where the connector server was installed, for example:

```
cd "c:\Program Files (x86)\ForgeRock\OpenICF"
```

4. (Optional) By default, the connector server outputs log messages to a file named `connectorserver.log`, in the `\path\to\openicf` directory. To change the location of the log file, set the `initializeData` parameter in the configuration file. The following example sets the log directory to `C:\openicf\logs\connectorserver.log`:

```
<add name="file" type="System.Diagnostics.TextWriterTraceListener" initializeData="C:\openicf\logs\connectorserver.log" traceOutputOptions="DateTime">
  <filter type="System.Diagnostics.EventTypeFilter" initializeData="Information"/>
</add>
```

5. Run the **ConnectorServerService /setKey** command to set a secret key for the connector server. The key can be any string value. This example sets the secret key to `Passw0rd`:

```
ConnectorServerService /setKey Passw0rd
Key has been successfully updated.
```

This key is used by clients connecting to the connector server. The key that you set here must also be set in the IDM remote connector server configuration.

6. Edit the connector server configuration.

The connector server configuration is saved in a file named `ConnectorServerService.exe.Config` (in the directory where the connector server is installed).

Check and edit this file, as necessary, to reflect your installation. Specifically, verify that the `baseAddress` reflects the host and port on which the connector server is installed:

```
<system.serviceModel>
  <services>
    <service name="Org.ForgeRock.OpenICF.Framework.Service.WcfServiceLibrary.WcfWebsocket">
      <host>
        <baseAddresses>
          <add baseAddress="http://0.0.0.0:8759/openicf" />
        </baseAddresses>
      </host>
    </service>
  </services>
</system.serviceModel>
```

The `baseAddress` specifies the host and port on which the connector server listens, and is set to `http://0.0.0.0:8759/openicf` by default. If you set a host value other than the default `0.0.0.0`, connections from all IP addresses other than the one specified are denied.

If Windows firewall is enabled, you must create an inbound port rule to open the TCP port for the connector server (8759 by default). If you do not open the TCP port, IDM won't be able to contact the connector server. For more information, see the corresponding Microsoft documentation.

7. (Optional) Configure the connector server to use SSL:
 - a. Open a Powershell terminal as a user with administrator privileges, then change to the ICF installation directory:

```
cd 'C:\Program Files (x86)\ForgeRock\OpenICF'
```

- b. Use an existing CA certificate, or use the `New-SelfSignedCertificate` cmdlet to create a self-signed certificate:

```
New-SelfSignedCertificate -DnsName "dotnet", "dotnet.example.com" -CertStoreLocation "cert:
\LocalMachine\My"
PSParentPath: Microsoft.PowerShell.Security\Certificate::LocalMachine\My

Thumbprint                               Subject
-----
770F531F14AF435E963E14AD82B70A47A4BFFBF2  CN=dotnet
```

- c. Assign the certificate to the connector server:

```
.\ConnectorServerService.exe /setCertificate
Select certificate you want to use:
Index   Issued To           Thumbprint
-----
0) dotnet            770F531F14AF435E963E14AD82B70A47A4BFFBF2

0
Certificate Thumbprint has been successfully updated to
770F531F14AF435E963E14AD82B70A47A4BFFBF2.
```

- d. Bind the certificate to the connector server port (8759 by default). To bind the certificate:

- i. Use the `New-Guid` cmdlet to generate a new UUID:

```
New-Guid
Guid
----
0352cf0f-2e7a-4aee-801d-7f27f8344c77
```

- ii. Enter the `netsh http` console and add the certificate thumbprint generated in the previous step, and the UUID that you have just generated:

```
netsh
netsh>http
netsh http>add sslcert ipport=0.0.0.0:8759 certhash=770F5...FFBF2 appid={0352c...4c77}
SSL Certificate successfully added
```

- e. Change the connector server configuration (in the `ConnectorServerService.exe.Config` file) to use HTTPS and not HTTP.

Change `baseAddress="http..."` to `baseAddress="https..."`:

```
<host>
  <baseAddresses>
    ...
    <add baseAddress="https://0.0.0.0:8759/openicf"/>
  </baseAddresses>
</host>
```

Change `httpTransport` to `httpsTransport`:

```
<httpsTransport authenticationScheme="Basic" realm="OpenICF">
  <webSocketSettings transportUsage="Always" createNotificationOnConnection="true" .../>
</httpsTransport>
```

f. Export the certificate:

- i. Launch the certificate management MMC (`certlm.msc`).
- ii. Right-click the `dotnet` certificate, and select All Tasks > Export to launch the Certificate Export Wizard.
- iii. Select Next > No, do not export the private key > DER encoded binary X.509 (.CER) > Next.
- iv. Save the file in an accessible location (for example, `C:\Users\Administrator\Desktop\dotnet.cer`), and click Finish.

g. Import the certificate into the IDM truststore:

- i. Transfer the certificate from the Windows machine to the machine that's running IDM.
- ii. Change to the `openidm/security` directory and use the Java `keytool` command to import the certificate:

```
cd /path/to/openidm/security
keytool -import -alias dotnet -file ~/Downloads/dotnet.cer -keystore ./truststore
Enter keystore password: changeit
Owner: CN=dotnet
Issuer: CN=dotnet
Serial number: 1e3af7baed05ce834da5cd1bf1241835
Valid from: Tue Aug 08 15:58:32 SAST 2017 until: Wed Aug 08 16:18:32 SAST 2018
Certificate fingerprints:
  MD5: D1:B7:B7:46:C2:59:1A:3C:94:AA:65:99:B4:43:3B:E8
  SHA1: 77:0F:53:1F:14:AF:43:5E:96:3E:14:AD:82:B7:0A:47:A4:BF:FB:F2
  SHA256:
C0:52:E2:E5:E5:72:9D:69:F8:11:4C:B8:4C:E4:E3:1C:19:95:86:19:70:E5:31:FA:D8:81:4B:F2:AC:30:9C:73
Signature algorithm name: SHA256withRSA
Version: 3

...

Trust this certificate? [no]: yes
Certificate was added to keystore
```

h. When you configure the remote connector server, remember to set `"useSSL": true`.

8. (Optional) Check the trace settings under `system.diagnostics` in the connector server configuration file:

```
<system.diagnostics>
  <trace autoflush="true" indentsize="4">
    <listeners>
      <remove name="Default" />
      <add name="console" />
      <add name="file" />
    </listeners>
  </trace>
  <sources>
    <source name="ConnectorServer" switchName="switch1">
      <listeners>
        <remove name="Default" />
        <add name="file" />
      </listeners>
    </source>
  </sources>
  <switches>
    <add name="switch1" value="Information" />
  </switches>
  <sharedListeners>
    <add name="console" type="System.Diagnostics.ConsoleTraceListener" />
    <add name="file" type="System.Diagnostics.TextWriterTraceListener"
      initializeData="logs\ConnectorServerService.log"
      traceOutputOptions="DateTime">
      <filter type="System.Diagnostics.EventTypeFilter" initializeData="Information" />
    </add>
  </sharedListeners>
</system.diagnostics>
```

The connector server uses the standard .NET trace mechanism. For more information about tracing options, see [Microsoft's .NET documentation for `System.Diagnostics`](#).

The default trace settings are a good starting point. For less tracing, set the `EventTypeFilter`'s `initializeData` to `Warning` or `Error`. For very verbose logging, set the value to `Verbose` or `All`. The logging level has a direct effect on the Connector server performance, so take care when setting this level.

9. Start the .NET connector server in one of the following ways:

- Start the server as a Windows service, by using the Microsoft Services Console.

Locate the connector server service (`OpenICF connector server`), and click `Start the service` or `Restart the service`.

The service runs with the credentials of the "run as" user (`System`, by default).

- Start the server as a Windows service, by using the command line.

In the Windows Command Prompt, run the following command:

```
net start ConnectorServerService
```

To stop the service, run the following command:

```
net stop ConnectorServerService
```

- Start the server without using Windows services.

In the Windows Command Prompt, change to the connector server installation directory. The default location is `c:\> cd "c:\Program Files (x86)\ForgeRock\OpenICF"`.

Start the server with the following command:

```
ConnectorServerService.exe /run
```

Note

This command starts the connector server with the credentials of the current user. It does not start the server as a Windows service.

+ Set Up a Java Connector Server

Install a Java Connector Server on Unix/Linux

1. Download the ICF Java connector server from the ForgeRock BackStage download site.
2. Change to the appropriate directory and unpack the .zip file. The following command unzips the file in the current directory:

```
unzip openicf-zip-1.5.20.12.zip
```

3. Change to the `openicf` directory:

```
cd path/to/openicf
```

4. Review the `ConnectorServer.properties` file in the `/path/to/openicf/conf` directory, and adjust it to suit your deployment. For a complete list of properties in that file, see [Remote Connector Server Properties](#).
5. In server mode, the connector server uses a `connectorserver.key` property to authenticate the connection. The default value of the key is a hashed value of the string `changeit`. You cannot set this property directly in the configuration file. To change its value, use the command **ConnectorServer.sh /setKey**. This example sets the key value to `Passw0rd`:

```
/path/to/openicf/bin/ConnectorServer.sh /setKey Passw0rd
Key has been successfully updated.
```

6. Start the Java connector server:

```
/path/to/openicf/bin/ConnectorServer.sh /run
```

The connector server is now running, and listening on port 8759, by default.

Log files are available in the `/path/to/openicf/logs` directory.

```
ls logs/
Connector.log  ConnectorServer.log  ConnectorServerTrace.log
```

7. To stop the Java connector server, press CTRL + C, or `q` in the terminal where you started the server.

Install a Java Connector Server on Windows

1. Download the ICF Java connector server from the [ForgeRock BackStage download site](#).
2. Change to the appropriate directory and unpack the .zip file.
3. In a Command Prompt window, change to the `openicf` directory:

```
C:\>cd C:\path\to\openicf\bin
```

4. Review the `ConnectorServer.properties` file in the `\path\to\openicf\conf` directory, and adjust it to suit your deployment. For a complete list of properties in that file, see [Remote Connector Server Properties](#).
5. In server mode, the connector server uses a `connectorserver.key` property to authenticate the connection. The default value of the key is a hashed value of the string `changeit`. You cannot set this property directly in the configuration file. To change its value, use the **ConnectorServer.bat /setKey** command. This example sets the key value to `Passw0rd`:

```
c:\path\to\openicf>bin\ConnectorServer.bat /setKey Passw0rd
lib\framework\connector-framework.jar;lib\framework\connector-framework-internal
.jar;lib\framework\groovy-all.jar;lib\framework\icfl-over-slf4j.jar;lib\framework
\slf4j-api.jar;lib\framework\logback-core.jar;lib\framework\logback-classic.jar
```

6. You can either run the Java connector server as a Windows service, or start and stop it from the command line:

- To install the Java connector server as a Windows service, run the following command:

```
c:\path\to\openicf>bin\ConnectorServer.bat /install
```

If you install the connector server as a Windows service, you can use the Microsoft Services Console to start, stop, and restart the service. The Java Connector Service is named `OpenICFConnectorServerJava`.

To uninstall the Java connector server as a Windows service, run the following command:

```
c:\path\to\openicf>bin\ConnectorServer.bat /uninstall
```

- To start the Java connector server from the command line, enter the following command:

```
c:\path\to\openicf>bin\ConnectorServer.bat /run
```

7. The connector server is now running, and listening on port 8759, by default.

Log files are available in the `\path\to\openicf\logs` directory.

8. To stop the Java connector server, press `^ + C`.

+ Remote Connector Server Properties

Some of these configuration properties are only applicable if you configure the connector server in client mode. For more information, see "Configure IDM to Connect to a Remote Connector Server".

Note that all configuration properties are prefixed with `connectorserver.` in the configuration file. The prefixes are not shown here so that the table is easier to read.

Property	RCS Mode (Server or Client)	Description	Example
<code>url</code>	Client	URL of the server on which IDM runs.	<code>wss://openidm.example.com:8443/openicf^a</code>
<code>proxyHost</code>	Client	Proxy server host.	
<code>proxyPort</code>	Client	Proxy server port number.	
<code>proxyPrincipal</code>	Client	Proxy server principal.	
<code>proxyPassword</code>	Client	Proxy server password.	
<code>housekeepingInterval</code>	Client	WebSocket connections housekeeping interval, in seconds.	600
<code>groupCheckInterval</code>	Client	WebSocket groups check interval, in seconds.	900
<code>websocketConnections</code>	Client	Number of WebSocket connections to open.	2
<code>connectionTtl</code>	Client	Time to live of a WebSocket connection, in seconds.	600
<code>tokenEndpoint</code>	Client	Token endpoint from which to retrieve the access token, if you are using OAuth2 to authenticate against AM.	<code>https://am.example.com/am/oauth2/realms/root/access_token</code>
<code>scope</code>	Client	OAuth2 token scope, if you are using OAuth2 to authenticate against AM.	<code>fr:idm:*</code>

Property	RCS Mode (Server or Client)	Description	Example
<code>clientId</code>	Client	OAuth2 Client ID for which to request an access token. ^b	<code>connectorServer</code>
<code>clientSecret</code>	Client	OAuth2 Client Secret.	<code>openidm</code>
<code>connectorServerName</code>	Both	Name of the remote connector client. This name is used to identify the remote connector server in the list of connector reference objects. The name must be lower case alphanumeric characters (<code>^[a-z0-9]*\$</code>), and must match the <code>name</code> property in the <code>provisioner.openicf.connectorinfoprovider.json</code> file on your IDM server.	<code>rsc1</code>
<code>pingPongInterval</code>	Both	WebSocket Ping/Pong interval, in seconds. The purpose of the ping is to keep connections alive (for firewalls or load balancers that honor connections in use). If your firewall or load balancer does not honor connections in use (that is, connections are timed out, regardless of their usage), the ping has no effect and you should disable it. Set this property to <code>0</code> to disable the ping.	<code>300</code>
<code>useSSL</code>	Both	Whether the connection between IDM and the connector server should be over SSL.	<code>false/true</code> ^c
<code>trustStoreFile</code>	Both	The IDM truststore file. You do not need to set this property if the IDM certificate is a CA-signed certificate.	<code>security/truststore.pkcs12</code>

Property	RCS Mode (Server or Client)	Description	Example
<code>trustStoreType</code>	Both	The IDM truststore type. You do not need to set this property if the IDM certificate is a CA-signed certificate.	<code>PKCS12</code>
<code>trustStorePass</code>	Both	The IDM truststore password. You do not need to set this property if the IDM certificate is a CA-signed certificate.	<code>changeit</code>
<code>keyStoreFile</code>	Both	The IDM keystore file. You do not need to set this property if the IDM certificate is a CA-signed certificate.	<code>security/keyStore.pkcs12</code>
<code>keyStoreType</code>	Both	The IDM keystore type. You do not need to set this property if the IDM certificate is a CA-signed certificate.	<code>PKCS12</code>
<code>keyStorePass</code>	Both	The IDM keystore password. You do not need to set this property if the IDM certificate is a CA-signed certificate.	<code>changeit</code>
<code>keyPass</code>	Both	The IDM certificate password. You do not need to set this property if the IDM certificate is a CA-signed certificate.	<code>changeit</code>
<code>libDir</code>	Both	Directory on the connector server host in which connector library file dependencies are located (relative to <code>/path/to/openicf/</code>).	<code>lib</code>
<code>bundleDir</code>	Both	Directory on the connector server host in which connector .jar files are located (relative to <code>/path/to/openicf/</code>).	<code>connectors</code>
<code>loggerClass</code>	Both	The connector server logger class.	<code>org.forgerock.openicf.common.logging.slf4j.Slf4JLog</code>

Property	RCS Mode (Server or Client)	Description	Example
<code>port</code>	Server	Port on which the connector server listens for the connection from IDM.	8759
<code>principal</code>	Server	Principal to authenticate to the connector server. This property is not used if the connector server obtains its access token through ForgeRock® Access Management (AM) (which is the case when IDM is running in ForgeRock Identity Cloud).	anonymous
<code>password</code>	Server	Password to authenticate to the connector server. This property is not used if the connector server obtains its access token through AM (which is the case when IDM is running in ForgeRock Identity Cloud).	<i>changeit</i>

^a Note the `wss` (WebSocket) transport protocol and the `openicf` endpoint.

^b

Important

If the connector server is authenticating against AM, you must update your IDM authentication configuration (in `conf/authentication.json`). Add a user mapping for this client ID in the `rsFilter` authentication module configuration. For more information, see "rsFilter" in the *Security Guide*.

^c In Client mode (when the connection uses `wss`), the connection *must* be over SSL, so this property must be set to `true`.

Configure IDM to Connect to a Remote Connector Server

Important

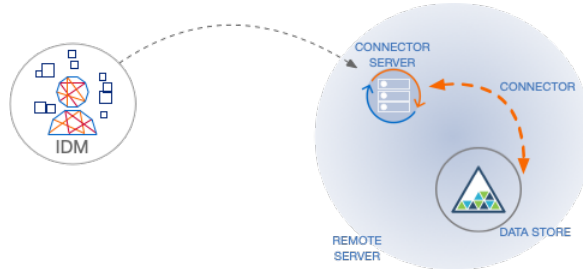
Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

You configure a connector server to run in one of two modes:

Server mode

In server mode, IDM initiates the connection to the remote connector server. Run the connector server in server mode if IDM can initiate the connection and has access through any firewalls.

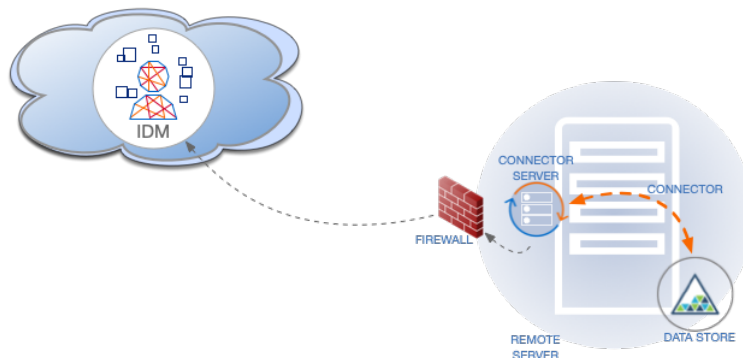
The following diagram shows a connector server in server mode:



Client mode

In client mode, IDM waits for the remote connector server to initiate a connection. Run the connector server in client mode if your data store is "on prem" and protected by a firewall or DMZ. In this case, IDM cannot initiate a connection to the connector server.

The following diagram shows a connector server in client mode:



For failover purposes, you can configure a *group* of remote connector servers, in either server or client mode. Failover is particularly important when you configure a connector server in client mode because IDM has no way of knowing whether the connector server is available.

This example shows how to retrieve the connector server types over REST:

+ *List the Remote Connector Server Types*

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system?_action=availableConnectorServers"
{
  "connectorServers": [
    {
      "displayName": "Remote Connector Server",
      "systemType": "provisioner.openicf",
      "type": "remoteConnectorServer"
    },
    {
      "displayName": "Remote Connector Servers Group",
      "systemType": "provisioner.openicf",
      "type": "remoteConnectorServersGroup"
    },
    {
      "displayName": "Remote Connector Server in Client mode",
      "systemType": "provisioner.openicf",
      "type": "remoteConnectorClient"
    },
    {
      "displayName": "Remote Connector Servers Group in Client mode",
      "systemType": "provisioner.openicf",
      "type": "remoteConnectorClientsGroup"
    }
  ]
}
```

Configure a Remote Connector Server in Server Mode

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

The following commands configure a remote connector server in server mode:

+ Create a Core Connector Server Configuration (Server Mode)

To generate the core configuration, use the `createConnectorServerCoreConfig` action on the `system` endpoint. Include at least the remote connector server `type` (`remoteConnectorServer`) and the `systemType` in the JSON payload. The `systemType` is always `provisioner.openicf`, regardless of the connector server type:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request POST \
--data '{
  "type": "remoteConnectorServer",
  "systemType": "provisioner.openicf"
}' \
"http://localhost:8080/openidm/system?_action=createConnectorServerCoreConfig"
{
  "displayName": "",
  "proxyPassword": null,
  "proxyHost": null,
  "enabled": true,
  "useSSL": false,
  "proxyPort": 8080,
  "port": "",
  "name": "",
  "host": "",
  "proxyUser": null,
  "housekeepingInterval": 600,
  "connectionGroupCheckInterval": 900,
  "pingPongInterval": 300,
  "key": "password",
  "websocketConnections": 2
}
```

IDM returns the required configuration properties for a connector server in server mode. The configuration that is returned is not functional. It does not contain the specific property values, such as the host name and port of the remote connector server.

+ Create a New Connector Server Configuration in Server Mode

Use the output returned in the previous example to create your complete connector server configuration. Specify at least the **host** and **port** of the remote connector server, and use a PUT request on the **config** endpoint. Note that this step creates a connector server configuration on IDM. The values of these properties must match the connector server configuration specified in the `ConnectorServer.properties` file on the remote connector server:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request PUT \
--data '{
  "_id": "provisioner.openicf.connectorinfoprovider",
  "connectorsLocation": "connectors",
  "enabled": true,
  "remoteConnectorServers": [
    {
      "type": "remoteConnectorServer",
      "displayName": "Remote Connector Server 1",

```

```

        "proxyPassword": null,
        "proxyHost": null,
        "enabled": true,
        "useSSL": false,
        "proxyPort": 8080,
        "port": 8759,
        "name": "rcs1",
        "host": "rcs.example.com",
        "proxyUser": null,
        "housekeepingInterval": 600,
        "connectionGroupCheckInterval": 900,
        "pingPongInterval": 300,
        "key": "Passw0rd",
        "webSocketConnections": 2
    }
}
}, \
"http://localhost:8080/openidm/config/provisioner.openicf.connectorinfoprovider"
{
  "_id": "provisioner.openicf.connectorinfoprovider",
  "connectorsLocation": "connectors",
  "enabled": true,
  "remoteConnectorServers": [
    {
      "type": "remoteConnectorServer",
      "displayName": "Remote Connector Server 1",
      "proxyPassword": null,
      "proxyHost": null,
      "enabled": true,
      "useSSL": false,
      "proxyPort": 8080,
      "port": 8759,
      "name": "rcs1",
      "host": "rcs.example.com",
      "proxyUser": null,
      "housekeepingInterval": 600,
      "connectionGroupCheckInterval": 900,
      "pingPongInterval": 300,
      "key": {
        "$crypto": {
          "type": "x-simple-encryption",
          "value": {
            "cipher": "AES/CBC/PKCS5Padding",
            "stableId": "openidm-sym-default",
            "salt": "3Mq1UJuZXqANx2AzUtbFbg==",
            "data": "4WHBEI3nSVWJ2DfIs2dPZg==",
            "keySize": 16,
            "purpose": "idm.config.encryption",
            "iv": "BvFAQ4sjwJCNy2e7WZPkGw==",
            "mac": "ximBz/BlqC8SEsBTuYQX5Q=="
          }
        }
      },
      "webSocketConnections": 2
    }
  ]
}

```

```
}
```

Configure a Remote Connector Server in Client Mode

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

In client mode, the connector server initiates the TCP connection to IDM. Therefore, IDM doesn't need any host, port or other connection details to the connector server.

+ Create a Core Connector Server Configuration (Client Mode)

To generate the core configuration, use the `createConnectorServerCoreConfig` action on the `system` endpoint. Include at least the remote connector server `type` (`remoteConnectorClient`) and the `systemType` in the JSON payload. The `systemType` is always `provisioner.openicf`, regardless of the connector server type:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request POST \
--data '{
  "type" : "remoteConnectorClient",
  "systemType" : "provisioner.openicf"
}' \
"http://localhost:8080/openidm/system?_action=createConnectorServerCoreConfig"
{
  "displayName": "",
  "name": "",
  "enabled": true,
  "useSSL": false
}
```

IDM returns the basic configuration properties for a connector server in client mode. The configuration that is returned is not functional. It does not contain the required configuration property values, such as the name the remote connector server.

+ Create a New Connector Server Configuration in Client Mode

Use the output returned in the previous example to create your complete connector server configuration. Specify at least the `name` of the remote connector server, and use a PUT request on the `config` endpoint. Note that this step creates a connector server configuration on IDM. The values of these properties must match the connector server configuration specified in the `ConnectorServer.properties` file on the remote connector server:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request PUT \
--data '{
  "_id": "provisioner.openicf.connectorinfoprovider",
  "connectorsLocation": "connectors",
  "enabled": true,
  "remoteConnectorClients": [
    {
      "displayName": "On premise 1",
      "name": "onprem",
      "enabled": true
    }
  ]
}' \
"http://localhost:8080/openidm/config/provisioner.openicf.connectorinfoprovider"
{
  "_id": "provisioner.openicf.connectorinfoprovider",
  "connectorsLocation": "connectors",
  "enabled": true,
  "remoteConnectorClients": [
    {
      "displayName": "On premise 1",
      "name": "onprem",
      "enabled": true,
      "useSSL": false
    }
  ]
}
```

Configure Failover Between Remote Connector Servers

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

To prevent the connector server from being a single point of failure, you can specify a list of remote connector servers that the connector can target. To set up a failover configuration, you create either a `remoteConnectorServersGroup` or a `remoteConnectorClientsGroup` and list the remote connector servers. The connector attempts to contact the first connector server in the list. If that connector server is down, it proceeds to the next connector server.

+ *Configure Failover For Connector Servers in Server Mode*

This example configures a `remoteConnectorServersGroup` that lists two remote connector servers, on hosts `remote-host-1` and `remote-host-2`. The connector servers are listed, by their `name` property. You can configure multiple groups and multiple servers per group.

First, generate the core configuration to obtain the required properties:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request POST \
--data '{
  "type": "remoteConnectorServersGroup",
  "systemType": "provisioner.openicf"
}' \
"http://localhost:8080/openidm/system?_action=createConnectorServerCoreConfig"
{
  "displayName": "",
  "name": "",
  "serversList": [],
  "algorithm": "failover"
}
```

Use the output returned in the previous example to create your connector server group configuration. Use a PUT request on the **config** endpoint:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request PUT \
--data '{
  "_id": "provisioner.openicf.connectorinfoprovider",
  "connectorsLocation": "connectors",
  "enabled": true,
  "remoteConnectorServers": [
    {
      "type": "remoteConnectorServersGroup",
      "displayName": ".NET Failover Group",
      "name": "dotnet-ha",
      "algorithm": "failover",
      "serversList": [
        {"name": "remote-host-1"},
        {"name": "remote-host-2"}
      ]
    }
  ]
}' \
"http://localhost:8080/openidm/config/provisioner.openicf.connectorinfoprovider"
{
  "_id": "provisioner.openicf.connectorinfoprovider",
  "connectorsLocation": "connectors",
  "enabled": true,
  "remoteConnectorServers": [
    {
      "type": "remoteConnectorServersGroup",
      "displayName": ".NET Failover Group",
      "name": "dotnet-ha",
      "algorithm": "failover",
      "serversList": [
        {

```

```

        "name": "remote-host-1"
      },
      {
        "name": "remote-host-2"
      }
    ]
  }
}

```

The **algorithm** can be either **failover** or **roundrobin**. If the algorithm is **failover**, requests are always sent to the first connector server in the list, unless it is unavailable; in which case, requests are sent to the next connector server in the list. If the algorithm is **roundrobin**, requests are distributed equally between the connector servers in the list, in the order in which they are received.

Your connector configuration (**provisioner.openicf-connector-name.json**) references the remote connector server group, rather than a single remote connector server. For example, the following excerpt of a PowerShell connector configuration file references the **dotnet-ha** connector server group created in the previous example:

```

{
  "connectorRef" : {
    "bundleName" : "MsPowerShell.Connector",
    "connectorName" : "Org.ForgeRock.OpenICF.Connectors.MsPowerShell.MsPowerShellConnector",
    "connectorHostRef" : "dotnet-ha",
    "bundleVersion" : "[1.4.2.0,1.5.0.0)"
  },
  ...
}

```

+ *Configure Failover For Connector Servers in Client Mode*

This example configures a **remoteConnectorClientsGroup** that lists two remote connector servers, on hosts **remote-host-1** and **remote-host-2**. The connector servers are listed, by their **name** property. You can configure multiple groups and multiple servers per group.

First, generate the core configuration to obtain the required properties:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request POST \
--data '{
  "type": "remoteConnectorClientsGroup",
  "systemType": "provisioner.openicf"
}' \
"http://localhost:8080/openidm/system?_action=createConnectorServerCoreConfig"
{
  "displayName": "",
  "name": "",
  "serversList": [],
  "algorithm": "failover"
}
```

Use the output returned in the previous example to create your connector server group configuration. Use a PUT request on the **config** endpoint:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request PUT \
--data '{
  "_id": "provisioner.openicf.connectorinfoprovider",
  "connectorsLocation": "connectors",
  "enabled": true,
  "remoteConnectorClients": [
    {
      "type": "remoteConnectorClientsGroup",
      "displayName": ".NET Failover Group",
      "name": "dotnet-ha",
      "algorithm": "failover",
      "serversList": [
        { "name": "remote-host-1" },
        { "name": "remote-host-2" }
      ]
    }
  ]
}' \
"http://localhost:8080/openidm/config/provisioner.openicf.connectorinfoprovider"
{
  "_id": "provisioner.openicf.connectorinfoprovider",
  "connectorsLocation": "connectors",
  "enabled": true,
  "remoteConnectorClients": [
    {
      "type": "remoteConnectorClientsGroup",
      "displayName": ".NET Failover Group",
      "name": "dotnet-ha",
      "algorithm": "failover",
      "serversList": [
        {
          "name": "remote-host-1"
        }
      ]
    }
  ]
}
```

```

    },
    {
      "name": "remote-host-2"
    }
  ]
}

```

The **algorithm** can be either **failover** or **roundrobin**. If the algorithm is **failover**, requests are always sent to the first connector server in the list, unless it is unavailable; in which case, requests are sent to the next connector server in the list. If the algorithm is **roundrobin**, requests are distributed equally between the connector servers in the list, in the order in which they are received.

Your connector configuration (**provisioner.openicf-connector-name.json**) references the remote connector server group, rather than a single remote connector server. For example, the following excerpt of a PowerShell connector configuration file references the **dotnet-ha** connector server group created in the previous example:

```

{
  "connectorRef" : {
    "bundleName" : "MsPowerShell.Connector",
    "connectorName" : "Org.ForgeRock.OpenICF.Connectors.MsPowerShell.MsPowerShellConnector",
    "connectorHostRef" : "dotnet-ha",
    "bundleVersion" : "[1.4.2.0,1.5.0.0)"
  },
  ...
}

```

Secure the Connection to the Connector Server With SSL

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

The SSL configuration for a connector server depends on whether you are running the connector server in server mode or in client mode:

- In **server mode**, the connector server needs a public/private key pair and a certificate (either self-signed or signed by a CA). The certificate is sent to the client (IDM) during the SSL handshake. For IDM to trust the certificate, the certificate must be imported into the IDM truststore.
- In **client mode**, the connector server initiates the connection to IDM. IDM sends its certificate during the SSL handshake. The CA that signed the IDM certificate (or the IDM self-signed certificate) must be imported into the truststore of the connector server. If you use TLS Mutual Authentication, IDM requests the certificate from the connector server during the SSL handshake. The connector server needs a public/private key pair and a certificate to achieve TLS Mutual Authentication with IDM.

1. Generate the connector server private/public key pair and create a new PKCS12 keystore:

```
keytool \  
-genkeypair \  
-keyalg EC \  
-alias icf-rcs \  
-dname "CN=icf.example.com,O=Example Corp,C=FR" \  
-keystore rcsKeystore \  
-storetype PKCS12 \  
-storepass changeit \  

```

2. Verify the contents of the new keystore:

```
keytool \  
-list \  
-v \  
-keystore rcsKeystore  
Enter keystore password: changeit  
Keystore type: PKCS12  
Keystore provider: SUN  
  
Your keystore contains 1 entry  
  
Alias name: icf-rcs  
Creation date: Jul 13, 2020  
Entry type: PrivateKeyEntry  
Certificate chain length: 1  
Certificate[1]:  
Owner: CN=icf.example.com, O=Example Corp, C=FR  
Issuer: CN=icf.example.com, O=Example Corp, C=FR  
Serial number: 611e093d  
Valid from: Mon Jul 13 23:58:49 SAST 2020 until: Sun Oct 11 23:58:49 SAST 2020  
Certificate fingerprints:  
  SHA1: Fingerprint  
  SHA256: Fingerprint  
Signature algorithm name: SHA256withECDSA  
Subject Public Key Algorithm: 256-bit EC key  
...
```

3. Export the connector server certificate:

```
keytool \  
-export \  
-alias icf-rcs \  
-file rcs.cert \  
-keystore rcsKeystore.pkcs12  
Enter keystore password: changeit  
Certificate stored in file <rcs.cert>
```

4. If you are not using a self-signed certificate, have the certificate signed by a Certificate Authority (CA):
 - a. Create a Certificate Signing Request (CSR):

```
keytool \
-keystore rcsKeystore.pkcs12 \
-certreq \
-alias icf-rcs \
-file rcs.csr
```

```
more rcs.csr
-----BEGIN NEW CERTIFICATE REQUEST-----

MIIKTCCA9QCAQAwVzELMAkGA1UEBhMCRLIx CzA JBgNVBAgTAKZSMQswCQYDVQQH
xZ47rzcY60rElh8+/TYG50NRqcQYMzm4CefCrhxTm6dHW4XQEa24tHmHdUmEaVys
A1UdDgQWBBSivxV9AzgbrIo3gG6vCBlnAXf3wjANBgIghkgBZQMEAwIFAANAADA9
...
AhxL791/ikf1hqx0D3uttV7qumg+TNednsgtk6u0Ah0AlInk+1LBeyUkQA7iUH/
3KLYWog/Npu5USdCeA==

-----END NEW CERTIFICATE REQUEST-----
```

b. Submit the CSR to your CA for signature.

5. Import the signed certificate into the connector server keystore:

```
keytool \
-importcert \
-trustcacerts \
-file rcs.cert \
-keystore rcsKeystore.pkcs12 \
-storetype pkcs12 \
-alias icf-rcs
Enter keystore password: changeit
Certificate reply was installed in keystore
```

Note

If your CA certificate is not trusted, you might need to import the CA certificate into the keystore too.

6. Import the connector server certificate into the IDM truststore:

```
keytool \  
-import \  
-alias icf-rcs \  
-keystore /path/to/openidm/truststore \  
-file rcs.cert  
Enter keystore password: changeit  
Owner: CN=icf.example.com, O=Example Corp, C=FR  
Issuer: CN=icf.example.com, O=Example Corp, C=FR  
Serial number: 611e093d  
Valid from: Fri Apr 05 16:04:04 CEST 2019 until: Mon Aug 17 16:04:04 CEST 2020  
Certificate fingerprints:  
    MD5: Fingerprint  
    SHA1: Fingerprint  
    SHA256: Fingerprint  
Signature algorithm name: SHA256withRSA  
Subject Public Key Algorithm: 2048-bit DSA key  
Version: 1  
Trust this certificate? [no]: yes  
Certificate was added to keystore
```

1. Generate the connector server private/public key pair and create a new PKCS12 keystore:

```
keytool \  
-genkeypair \  
-keyalg EC \  
-alias icf-rcs \  
-dname "CN=icf.example.com,O=Example Corp,C=FR" \  
-keystore rcsKeystore \  
-storetype PKCS12 \  
-storepass changeit \  

```

2. Verify the contents of the new keystore:

```
keytool \  
-list \  
-v \  
-keystore rcsKeystore  
Enter keystore password: changeit  
Keystore type: PKCS12  
Keystore provider: SUN  
  
Your keystore contains 1 entry  
  
Alias name: icf-rcs  
Creation date: Jul 13, 2020  
Entry type: PrivateKeyEntry  
Certificate chain length: 1  
Certificate[1]:  
Owner: CN=icf.example.com, O=Example Corp, C=FR  
Issuer: CN=icf.example.com, O=Example Corp, C=FR  
Serial number: 611e093d  
Valid from: Mon Jul 13 23:58:49 SAST 2020 until: Sun Oct 11 23:58:49 SAST 2020  
Certificate fingerprints:  
    SHA1: Fingerprint  
    SHA256: Fingerprint  
Signature algorithm name: SHA256withECDSA  
Subject Public Key Algorithm: 256-bit EC key  
...
```

3. Export the connector server certificate:

```
keytool \  
-export \  
-alias icf-rcs \  
-file rcs.cert \  
-keystore rcsKeystore.pkcs12  
Enter keystore password: changeit  
Certificate stored in file <rcs.cert>
```

4. If you are not using a self-signed certificate, have the certificate signed by a Certificate Authority (CA):

a. Create a Certificate Signing Request (CSR):

```
keytool \  
-keystore rcsKeystore.pkcs12 \  
-certreq \  
-alias icf-rcs \  
-file rcs.csr
```

```
more rcs.csr  
-----BEGIN NEW CERTIFICATE REQUEST-----  
  
MIIKTCQA9QCAQAwVzELMAkGA1UEBhMCRLIx CzAJBgNVBAGTAkZSMQswCQYDVQQH  
xZ47rzcY60rElh8+/TYG50NRqcQYMzm4CefCrhxTm6dHW4XQEa24tHmHdUmEaVys  
A1UdDgQWBBSivxV9AzgbrIo3gG6vCBlnAXf3wjANBgIghkgBZQMEAwIFAANAADA9  
...  
AhxL791/ikf1hqx0D3uttV7qumg+TNednsgtk6u0Ah0AlINK+1LBeyUkQA7iUHy/  
3KLYWog/Npu5USdCeA==  
  
-----END NEW CERTIFICATE REQUEST-----
```

b. Submit the CSR to your CA for signature.

5. Import the signed certificate into the connector server keystore:

```
keytool \  
-importcert \  
-trustcacerts \  
-file rcs.cert \  
-keystore rcsKeystore.pkcs12 \  
-storetype pkcs12 \  
-alias icf-rcs  
Enter keystore password: changeit  
Certificate reply was installed in keystore
```

Note

If your CA certificate is not trusted, you might need to import the CA certificate into the keystore too.

6. Import the connector server certificate into the IDM truststore:

```
keytool \  
-import \  
-alias icf-rcs \  
-keystore /path/to/openidm/truststore \  
-file rcs.cert  
Enter keystore password: changeit  
Owner: CN=icf.example.com, O=Example Corp, C=FR  
Issuer: CN=icf.example.com, O=Example Corp, C=FR  
Serial number: 611e093d  
Valid from: Fri Apr 05 16:04:04 CEST 2019 until: Mon Aug 17 16:04:04 CEST 2020  
Certificate fingerprints:  
    MD5: Fingerprint  
    SHA1: Fingerprint  
    SHA256: Fingerprint  
Signature algorithm name: SHA256withRSA  
Subject Public Key Algorithm: 2048-bit DSA key  
Version: 1  
Trust this certificate? [no]: yes  
Certificate was added to keystore
```

7. Export the IDM self-signed certificate:

```
keytool \  
-export \  
-alias openidm-localhost \  
-keystore keystore.jceks \  
-storetype jceks \  
-file idm.cert \  
Enter keystore password: changeit  
Certificate stored in file <idm.cert>
```

8. Import the IDM self-signed certificate into the connector server truststore:

```
keytool \  
-import \  
-alias openidm-localhost \  
-keystore /path/to/rcs/security/truststore.pkcs12 \  
-storetype pkcs12 \  
-file idm.cert  
Enter keystore password: changeit  
  
Owner: CN=openidm-localhost, O=OpenIDM Self-Signed Certificate, OU=None, L=None, ST=None, C=None  
Issuer: CN=openidm-localhost, O=OpenIDM Self-Signed Certificate, OU=None, L=None, ST=None, C=None  
Serial number: 16981c79d8d  
Valid from: Wed Feb 13 15:35:36 CET 2019 until: Thu Mar 15 15:35:36 CET 2029  
Certificate fingerprints:  
    MD5: fingerprint  
    SHA1: fingerprint  
    SHA256: fingerprint  
Signature algorithm name: SHA512withRSA  
Subject Public Key Algorithm: 2048-bit RSA key  
Version: 3  
Trust this certificate? [no]: yes  
  
Certificate was added to keystore
```

Install Connector Dependencies

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

Many connectors depend on third-party libraries. In most cases, these libraries are bundled with IDM (if you are running the connector locally), or with the connector server (if you are running the connector remotely). In some cases, you'll need to download certain libraries. For local connectors, place these libraries in the `/path/to/openidm/lib/` directory. For remote connectors, place them in the `/path/to/openicf/lib/` directory.

The following table lists the connector dependencies and indicates which ones must be downloaded:

Dependencies for bundled connectors	
Connector	Dependencies
Adobe Marketing Cloud Connector	<ul style="list-style-type: none"> <code>bundle/httpclient-osgi-4.5.2.jar</code>
CSV File Connector	<ul style="list-style-type: none"> <code>bundle/super-csv-2.4.0.jar</code>
Database Table Connector	No external dependencies. However, you must include the JDBC driver for the database that you are targeting in the <code>/path/to/openidm/lib/</code> directory.
DocuSign Connector	<ul style="list-style-type: none"> <code>lib/java-jwt-3.4.0.jar</code>
GoogleApps Connector	<ul style="list-style-type: none"> <code>bundle/httpclient-osgi-4.5.2.jar</code> <code>bundle/httpcore-osgi-4.4.5.jar</code> <code>bundle/jackson-core-2.9.4.jar</code> <code>lib/google-api-client-1.19.0.jar</code> <code>lib/google-api-services-admin-directory-directory_v1-rev41-1.19.0.jar</code> <code>lib/google-api-services-licensing-v1-rev34-1.19.0.jar</code> <code>lib/google-http-client-1.19.0.jar</code> <code>lib/google-http-client-jackson2-1.19.0.jar</code> <code>lib/google-oauth-client-1.19.0.jar</code> <code>lib/google-oauth-client-java6-1.19.0.jar</code>
Scripted Groovy Connector	No external dependencies
Scripted Poolable Groovy Connector	No external dependencies
Kerberos Connector	<ul style="list-style-type: none"> <code>lib/groovy-connector-1.5.20.8</code>

Dependencies for bundled connectors	
Connector	Dependencies
	<ul style="list-style-type: none"> lib/ssh-connector-1.5.20.8
LDAP Connector	No external dependencies
Marketo Connector	<ul style="list-style-type: none"> lib/groovy-connector-1.5.20.8
MongoDB Connector	<ul style="list-style-type: none"> lib/groovy-connector-1.5.20.8
SCIM Connector	<ul style="list-style-type: none"> bundle/httpclient-osgi-4.5.2.jar bundle/httpcore-osgi-4.4.5.jar bundle/jackson-annotations-2.9.4.jar bundle/jackson-core-2.9.4.jar bundle/jackson-databind-2.9.4.jar
Scripted REST Connector	<ul style="list-style-type: none"> bundle/httpclient-osgi-4.5.2.jar bundle/httpcore-osgi-4.4.5.jar lib/commons-collections-3.2.2.jar lib/groovy-connector-1.5.20.8 lib/http-builder-0.7.1.jar lib/json-lib-2.3-jdk15.jar lib/xml-resolver-1.2.jar
Scripted SQL Connector	<ul style="list-style-type: none"> bundle/tomcat-juli-8.5.23.jar lib/groovy-connector-1.5.20.8 lib/tomcat-jdbc-8.5.23.jar
ServiceNow Connector	<ul style="list-style-type: none"> bundle/httpclient-osgi-4.5.2.jar lib/json-20170516.jar
SSH Connector	<ul style="list-style-type: none"> lib/expect4j-1.9.jar lib/groovy-connector-1.5.20.8 lib/jsch-0.1.54.jar
Workday Connector	<p>These dependencies are public, and can be downloaded from any maven public repo, such as https://mvnrepository.com/:</p> <ul style="list-style-type: none"> lib/cxf-core-3.2.2.jar lib/cxf-rt-bindings-soap-3.2.2.jar lib/cxf-rt-databinding-jaxb-3.2.2.jar

Dependencies for bundled connectors	
Connector	Dependencies
	<ul style="list-style-type: none"> lib/cxf-rt-frontend-jaxws-3.2.2.jar lib/cxf-rt-frontend-simple-3.2.2.jar lib/cxf-rt-security-3.2.2.jar lib/cxf-rt-transports-http-3.2.2.jar lib/cxf-rt-ws-security-3.2.2.jar lib/cxf-rt-wsdl-3.2.2.jar lib/wsdl4j-1.6.3.jar lib/wss4j-bindings-2.2.1.jar lib/wss4j-policy-2.2.1.jar lib/wss4j-ws-security-common-2.2.1.jar lib/wss4j-ws-security-dom-2.2.1.jar lib/wss4j-ws-security-policy-stax-2.2.1.jar lib/wss4j-ws-security-stax-2.2.1.jar lib/xmlschema-core-2.2.3.jar lib/xmlsec-2.1.1.jar

Example: Use the CSV Connector to Reconcile Users in a Remote CSV Data Store

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

This example shows reconciliation of users stored in a CSV file on a remote machine. The remote Java connector server lets IDM synchronize its repository with the remote CSV file.

The example assumes that a remote Java connector server is installed and running on a host named `remote-host`.

The example uses the small CSV data set provided with the *Getting Started* sample (`hr.csv`). The CSV connector runs as a *remote connector*, on the host where the Java connector server is running. Before you start, copy the CSV data file from the *Getting Started* sample (`/path/to/openidm/samples/getting-`

`started/data/hr.csv`) to an accessible location on the machine that hosts the remote Java connector server. For example:

```
cd /path/to/openidm/samples/getting-started/data/
scp hr.csv testuser@remote-host:/home/testuser/csv-sample/data/
Password:*****
hr.csv      100% 651      0.6KB/s   00:00
```

Configure IDM for the Remote CSV Connector Example

Before you start, copy the following files to your `/path/to/openidm/conf` directory:

- A customized mapping file for this example.
- `/openidm/samples/example-configurations/provisioners/provisioner.openicf.connectorinfoprovider.json`

A sample connector server configuration.

- `/openidm/samples/example-configurations/provisioners/provisioner.openicf-csvfile.json`

A sample connector configuration file.

1. Edit the remote connector server configuration file (`provisioner.openicf.connectorinfoprovider.json`) to match your network setup.

The following example indicates that the Java connector server is running on the host `remote-host`, listening on the default port, and configured with a secret key of `Passw0rd`:

```
{
  "remoteConnectorServers" : [
    {
      "name" : "csv",
      "host" : "remote-host",
      "port" : 8759,
      "useSSL" : false,
      "key" : "Passw0rd"
    }
  ]
}
```

The `name` that you set in this file will be referenced in the `connectorHostRef` property of the connector configuration, in the next step.

The `key` that you specify here must match the password that you set when you installed the Java connector server.

2. Edit the CSV connector configuration file (`provisioner.openicf-csvfile.json`) as follows:

```
{
  "connectorRef" : {
    "connectorHostRef" : "csv",
    "bundleName" : "org.forgerock.openicf.connectors.csvfile-connector",
    "bundleVersion" : "[1.5.0.0,1.6.0.0)",
    "connectorName" : "org.forgerock.openicf.csvfile.CSVFileConnector"
  },
  ...
  "configurationProperties" : {
    "csvFile" : "/home/testuser/csv-sample/data/hr.csv"
  }
}
```

- The `connectorHostRef` property sets the remote connector server to use, and refers to the `name` property you specified in the `provisioner.openicf.connectorinfoprovider.json` file.
- The `bundleVersion` : "[1.5.0.0,1.6.0.0)", must either be exactly the same as the version of the CSV connector that you are using or, if you specify a range, the CSV connector version must be included in this range.
- The `csvFile` property must specify the absolute path to the CSV data file that you copied to the remote host on which the Java Connector Server is running.

3. Start IDM:

```
/path/to/openidm/startup.sh
```

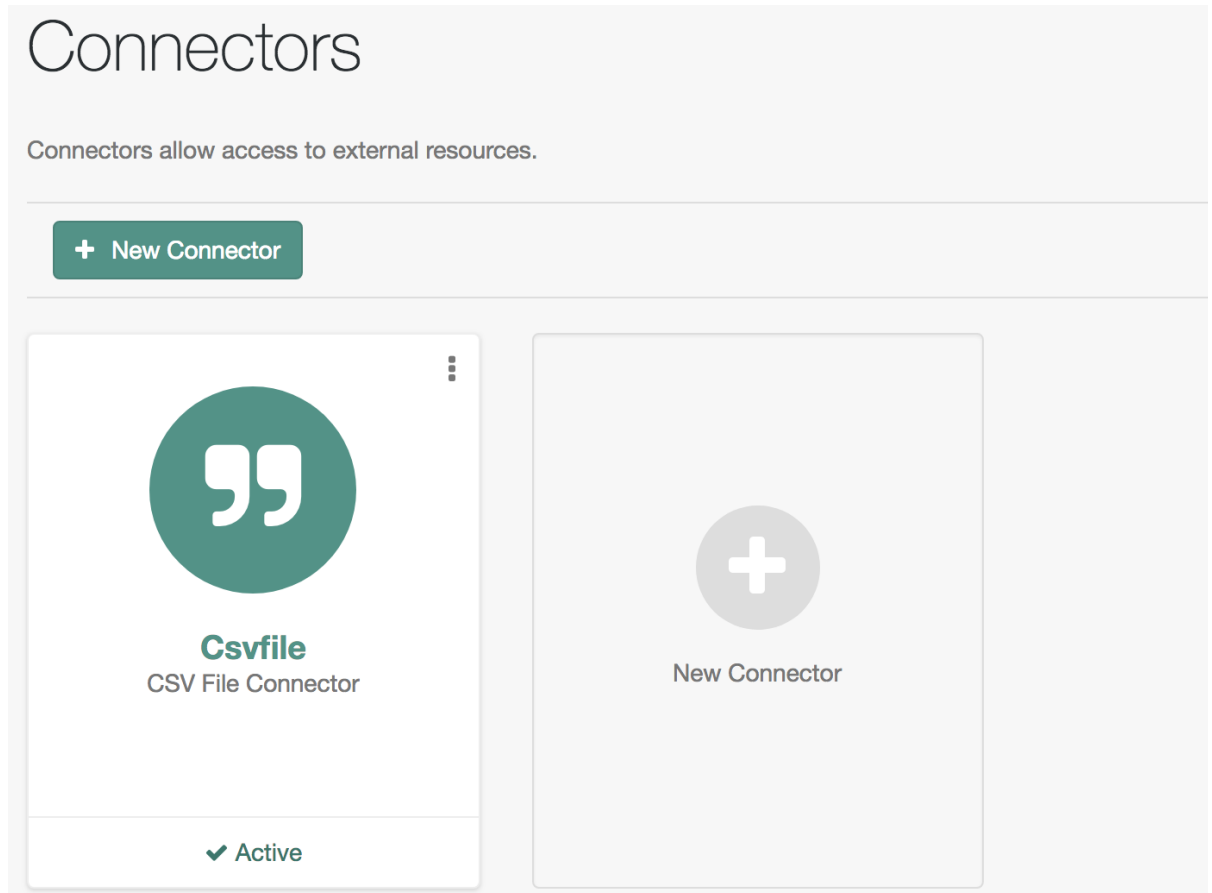
4. Verify that IDM can reach the remote connector server and that the CSV connector has been configured correctly:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system?_action=test"
[
{
  "name": "csv",
  "enabled": true,
  "config": "config/provisioner.openicf/csv",
  "objectTypes": [
    "__ALL__",
    "account"
  ],
  "connectorRef": {
    "bundleName": "org.forgerock.openicf.connectors.csvfile-connector",
    "connectorName": "org.forgerock.openicf.csvfile.CSVFileConnector",
    "bundleVersion": "[1.5.0.0,1.6.0.0)"
  },
  "displayName": "CSV File Connector",
  "ok": true
}
]
```

The connector must return `"ok": true`.

Alternatively, use the Admin UI to verify that IDM can reach the remote connector server and that the CSV connector is active. Log in to the Admin UI (<https://localhost:8443/openidm/admin>) and select Configure > Connectors. The CSV connector should be listed on the Connectors page, and its status should be Active.

Connectors Tab Showing an Active CSV Connector



5. To test that the connector has been configured correctly, run a reconciliation operation as follows:
 1. Select Configure > Mappings and click the `systemCsvAccounts_managedUser` mapping.
 2. Click Reconcile.

If the reconciliation is successful, the three users from the remote CSV file should have been added to the managed user repository.

To check this, select **Manage > User**.

Chapter 5

Check External System Status Using REST

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

After a connection has been configured, external systems are accessible over the REST interface at the URL <http://localhost:8080/openidm/system/connector-name>. Aside from accessing the data objects within the external systems, you can test the availability of the systems themselves.

To list the external systems that are connected to an IDM instance, use the **test** action on the URL <http://localhost:8080/openidm/system/>. The following example shows an IDM system with two connected LDAP systems:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system?_action=test"
[
  {
    "name": "ldap",
    "enabled": true,
    "config": "config/provisioner.openicf/ldap",
    "connectorRef": {
      "bundleVersion": "[1.4.0.0,1.6.0.0)",
      "bundleName": "org.forgerock.openicf.connectors.ldap-connector",
      "connectorName": "org.identityconnectors.ldap.LdapConnector"
    },
    "displayName": "LDAP Connector",
    "objectTypes": [
      "ALL",
      "account",
      "group"
    ],
    "ok": true
  },
  {
    "name": "ldap2",
    "enabled": true,
    "config": "config/provisioner.openicf/ldap2",
    "connectorRef": {
      "bundleVersion": "[1.4.0.0,1.6.0.0)",
      "bundleName": "org.forgerock.openicf.connectors.ldap-connector",
      "connectorName": "org.identityconnectors.ldap.LdapConnector"
    }
  }
]
```

```

    },
    "displayName": "LDAP Connector",
    "objectTypes": [
      "ALL",
      "account",
      "group"
    ],
    "ok": true
  }
]

```

The status of the system is provided by the `ok` parameter. If the connection is available, the value of this parameter is `true`.

To obtain the status for a single system, include the name of the connector in the URL, for example:

```

curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system/ldap?_action=test"
{
  "name": "ldap",
  "enabled": true,
  "config": "config/provisioner.openicf/ldap",
  "connectorRef": {
    "bundleVersion": "[1.4.0.0,1.6.0.0)",
    "bundleName": "org.forgerock.openicf.connectors.ldap-connector",
    "connectorName": "org.identityconnectors.ldap.LdapConnector"
  },
  "displayName": "LDAP Connector",
  "objectTypes": [
    "ALL",
    "account",
    "group"
  ],
  "ok": true
}

```

If there is a problem with the connection, the `ok` parameter returns `false`, with an indication of the error. In the following example, the LDAP server named `ldap`, running on `localhost:1389`, is down:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system/ldap?_action=test"
{
  "name": "ldap",
  "enabled": true,
  "config": "config/provisioner.openicf/ldap",
  "connectorRef": {
    "bundleVersion": "[1.4.0.0,1.6.0.0]",
    "bundleName": "org.forgerock.openicf.connectors.ldap-connector",
    "connectorName": "org.identityconnectors.ldap.LdapConnector"
  },
  "displayName": "LDAP Connector",
  "objectTypes": [
    "ALL",
    "account",
    "group"
  ],
  "error": "javax.naming.CommunicationException: localhost:1389 [Root exception
is java.net.ConnectException: Connection refused (Connection refused)]",
  "ok": false
}
```

To test the validity of a connector configuration, use the `testConfig` action and include the configuration in the command. For example:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request POST \
--data '{
  "configurationProperties": {
    "headerPassword": "password",
    "csvFile": "&{idm.instance.dir}/data/csvConnectorData.csv",
    "newlineString": "\n",
    "headerUid": "uid",
    "quoteCharacter": "\"",
    "fieldDelimiter": ",",
    "syncFileRetentionCount": 3
  },
  "connectorRef": {
    "systemType": "provisioner.openicf",
    "bundleName": "org.forgerock.openicf.connectors.csvfile-connector",
    "connectorName": "org.forgerock.openicf.csvfile.CSVFileConnector",
    "displayName": "CSV File Connector",
    "bundleVersion": "[1.5.0.0,1.6.0.0]"
  },
  "poolConfigOption": {
    "maxObjects": 10,
    "maxIdle": 10,
    "maxWait": 150000,
    "minEvictableIdleTimeMillis": 120000,
    "minIdle": 1
  }
}
```

```

},
"resultsHandlerConfig": {
  "enableNormalizingResultsHandler": true,
  "enableFilteredResultsHandler": true,
  "enableCaseInsensitiveFilter": false,
  "enableAttributesToGetSearchResultsHandler": true
},
"operationTimeout": {
  "CREATE": -1,
  "UPDATE": -1,
  "DELETE": -1,
  "TEST": -1,
  "SCRIPT_ON_CONNECTOR": -1,
  "SCRIPT_ON_RESOURCE": -1,
  "GET": -1,
  "RESOLVEUSERNAME": -1,
  "AUTHENTICATE": -1,
  "SEARCH": -1,
  "VALIDATE": -1,
  "SYNC": -1,
  "SCHEMA": -1
}
}, \
"http://localhost:8080/openidm/system?_action=testConfig"

```

If the configuration is valid, the command returns `"ok": true`, for example:

```

{
  "ok": true
}

```

If the configuration is not valid, the command returns an error, indicating the problem with the configuration. For example, the following result is returned when the LDAP connector configuration is missing a required property (in this case, the `baseContexts` to synchronize):

```

{
  "error": "org.identityconnectors.framework.common.exceptions.ConfigurationException: The list of base contexts cannot be empty",
  "name": "ldap",
  "ok": false
}

```

The `testConfig` action requires a running IDM instance, as it uses the REST API, but does not require an active connector instance for the connector whose configuration you want to test.

Chapter 6

Remove a Connector

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

If you have reason to remove a connector, be careful. If you remove a connector used in a mapping, while it's part of a scheduled task, you may see unintended consequences.

If you're removing a connector, consider the following checklist. Depending on your configuration, this list may not be comprehensive:

- Consider the remote resource. Make sure you no longer need data from that resource, and that the resource no longer requires data from IDM.
- Open the `sync.json` file for your project. Delete the code block associated with the mapping.
- Review the `schedule-recon.json` file. If it contains the schedule for a single operation, delete the file, or update it as a schedule for a different mapping.

When these steps are complete, you can delete the connector configuration file, typically named `provisioner-*.json`.

You can also delete the connector via the Admin UI. Log in as `openidm-admin` and select Configure > Connectors. Find the target connector, select the vertical ellipsis > *widget*. In the pop-up menu that appears, press Delete. The Admin UI will automatically make the specified changes to the noted configuration files.

Appendix A. ICF Interfaces

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

The ICF framework supports the following interfaces:

Note

Certain connectors may support only a subset of these interfaces.

AttributeNormalizer

Normalize attributes to ensure consistent filtering.

Authenticate

Provides simple authentication with two parameters, presumed to be a username and password. IDM requires the connector to implement the `AuthenticateOp` interface in order to provide pass-through authentication.

Batch

Execute a series of operations in a single request. If a resource does not support batch operations, the connector will not implement the batch operation interface. The ICF framework will still support batched requests but the operations will be executed iteratively through the connector.

Connector Event

Subscribe for notification of any specified event on the target resource. This operation can be used in the context of IoT device reports, to receive notification of events such as low battery signals, inactive devices, and so on.

Create

Create an object and return its UID.

Delete

Delete an object by its UID.

Get

Get an object by its UID.

PoolableConnector

Use pools of target resources.

Resolve Username

Resolve an object to its UID based on its username.

Schema

Describe supported object types, operations, and options.

Script on Connector

Allow script execution on the connector.

Script On Resource

Allow script execution on the resource.

Search

Allow searches for resource objects.

Connectors that implement *only* this interface can only be used for reconciliation operations.

Sync

Poll for synchronization events, which are native changes to target objects.

Sync Event

Subscribe for notification of synchronization events, which are native changes to target objects.

Test

Test the connection configuration, including connecting to the resource.

Update

Allows an authorized caller to update (modify or replace) objects on the target resource.

Update Attribute Values

Allows an authorized caller to update (modify or replace) attribute values on the target resource. This operation is more advanced than the **UpdateOp** operation, and provides better performance and atomicity semantics.

Appendix B. ICF Operation Options

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

The ICF framework supports the following predefined operation options:

Note

Certain connectors may support only a subset of these options.

Scope

An option to use with Search (in conjunction with the Container option) that specifies how far beneath the container to search. Must be one of the following values:

- `SCOPE_OBJECT`
- `SCOPE_ONE_LEVEL`
- `SCOPE_SUBTREE`

Container

An option to use with Search that specifies the container under which to perform the search. Must be of type `QualifiedUid`. Should be implemented for those object classes whose `ObjectClassInfo.isContainer()` returns true.

Run as User

An option that specifies an account under which to execute the script or operation. The specified account will appear to have performed any action that the script or operation performs.

Run with Password

An option to use with Script on Resource that specifies a password under which to execute the script or operation.

Attributes to Get

Determines which attributes to retrieve during Search and Sync. This option overrides the default behavior, which is for the connector to return the precise set of attributes identified as returned by default in the schema for that connector.

This option allows a client application to request additional attributes that would not otherwise not be returned (generally because such attributes are more expensive for a connector to fetch and to format) or to request only a subset of the attributes that would normally be returned.

Paged Results Cookie

An option to use with Search that specifies an opaque cookie, used by the connector to track its position in the set of query results.

Paged Results Offset

An option to use with Search that specifies the index within the result set of the first result which should be returned.

Page Size

An option to use with Search that specifies the requested page results page size.

Sort Keys

An option to use with Search that specifies the sort keys that should be used for ordering the connector object returned by search request.

Fail on Error

This option is used with the Batch operation to specify whether the batch process should be aborted when the first error is encountered. The default behavior is to continue processing regardless of errors.

Require Serial

This option instructs the connector to execute batched requests in a serial manner, if possible. The default behavior of the Batch operation is to execute requests in parallel, for speed and efficiency. In either case the task ID must be reflected in the response for each task so that tasks can be correctly reordered.

Appendix C. Connection Pooling Configuration

Important

Connectors continue to be released outside the IDM release. For the latest documentation, refer to the ICF documentation.

Certain connectors support the ability to be pooled. For a pooled connector, ICF maintains a pool of connector instances and reuses these instances for multiple provisioning and reconciliation operations. When an operation must be executed, an existing connector instance is taken from the connector pool. If no connector instance exists, a new instance is initialized. When the operation has been executed, the connector instance is released back into the connector pool, ready to be used for a subsequent operation.

For an unpooled connector, a new connector instance is initialized for every operation. When the operation has been executed, ICF disposes of the connector instance.

Because the initialization of a connector is an expensive operation, reducing the number of connector initializations can substantially improve performance.

To configure connection pooling, set the following values in the connector configuration file `poolConfigOptions` property:

`maxObjects`

The maximum number of connector instances in the pool (both idle and active). The default value is `10` instances.

`maxIdle`

The maximum number of idle connector instances in the pool. The default value is `10` idle instances.

`maxWait`

The maximum period to wait for a free connector instance to become available before failing. The default period is `150000` milliseconds, or 150 seconds.

`minEvictableIdleTimeMillis`

The minimum period to wait before evicting an idle connector instance from the pool. The default period is `120000` milliseconds, or 120 seconds.

`minIdle`

The minimum number of idle connector instances in the pool. The default value is `1` instance.

IDM Glossary

correlation query	A correlation query specifies an expression that matches existing entries in a source repository to one or more entries in a target repository. A correlation query might be built with a script, but it is not the same as a correlation script. For more information, see <i>"Correlating Source Objects With Existing Target Objects"</i> in the <i>Synchronization Guide</i> .
correlation script	A correlation script matches existing entries in a source repository, and returns the IDs of one or more matching entries on a target repository. While it skips the intermediate step associated with a correlation query , a correlation script can be relatively complex, based on the operations of the script.
entitlement	An entitlement is a collection of attributes that can be added to a user entry via roles. As such, it is a specialized type of assignment . A user or device with an entitlement gets access rights to specified resources. An entitlement is a property of a managed object.
JCE	Java Cryptographic Extension, which is part of the Java Cryptography Architecture, provides a framework for encryption, key generation, and digital signatures.
JSON	JavaScript Object Notation, a lightweight data interchange format based on a subset of JavaScript syntax. For more information, see the JSON site .
JSON Pointer	A JSON Pointer defines a string syntax for identifying a specific value within a JSON document. For information about JSON Pointer syntax, see the JSON Pointer RFC .

JWT	JSON Web Token. As noted in the JSON Web Token draft IETF Memo , "JSON Web Token (JWT) is a compact URL-safe means of representing claims to be transferred between two parties." For IDM, the JWT is associated with the JWT_SESSION authentication module.
managed object	An object that represents the identity-related data managed by IDM. Managed objects are configurable, JSON-based data structures that IDM stores in its pluggable repository. The default configuration of a managed object is that of a user, but you can define any kind of managed object, for example, groups or roles.
mapping	A policy that is defined between a source object and a target object during reconciliation or synchronization. A mapping can also define a trigger for validation, customization, filtering, and transformation of source and target objects.
OSGi	A module system and service platform for the Java programming language that implements a complete and dynamic component model. For more information, see What is OSGi? Currently, only the Apache Felix container is supported.
reconciliation	During reconciliation, comparisons are made between managed objects and objects on source or target systems. Reconciliation can result in one or more specified actions, including, but not limited to, synchronization.
resource	An external system, database, directory server, or other source of identity data to be managed and audited by the identity management system.
REST	Representational State Transfer. A software architecture style for exposing resources, using the technologies and protocols of the World Wide Web. REST describes how distributed data objects, or resources, can be defined and addressed.
role	IDM distinguishes between two distinct role types - provisioning roles and authorization roles. For more information, see "Managed Roles" in the <i>Object Modeling Guide</i> .
source object	In the context of reconciliation, a source object is a data object on the source system, that IDM scans before attempting to find a corresponding object on the target system. Depending on the defined mapping, IDM then adjusts the object on the target system (target object).
synchronization	The synchronization process creates, updates, or deletes objects on a target system, based on the defined mappings from the source system. Synchronization can be scheduled or on demand.

system object	A pluggable representation of an object on an external system. For example, a user entry that is stored in an external LDAP directory is represented as a system object in IDM for the period during which IDM requires access to that entry. System objects follow the same RESTful resource-based design principles as managed objects.
target object	In the context of reconciliation, a target object is a data object on the target system, that IDM scans after locating its corresponding object on the source system. Depending on the defined mapping, IDM then adjusts the target object to match the corresponding source object.