



# Connector Reference

/ ForgeRock Identity Management 6.5

Latest update: 6.5.2.0

Lana Frost

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

---

Copyright © 2011-2018 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, <http://fontawesome.io>.

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

Preface .....	viii
1. About This Guide .....	viii
2. Accessing Documentation Online .....	viii
3. Using the ForgeRock.org Site .....	ix
1. Connector Overview .....	1
2. Generic LDAP Connector .....	3
2.1. Setting Up the Generic LDAP Connector .....	3
2.2. Configuring the LDAP Connector to Use SSL and StartTLS .....	9
2.3. Controlling What the LDAP Connector Synchronizes .....	11
2.4. Using the Generic LDAP Connector With Active Directory .....	12
2.5. Constructing the LDAP Search Filter .....	20
2.6. OpenICF Interfaces Implemented by the LDAP Connector .....	21
2.7. LDAP Connector Configuration .....	22
3. CSV File Connector .....	28
3.1. Configuring the CSV File Connector .....	28
3.2. OpenICF Interfaces Implemented by the CSV File Connector .....	28
3.3. CSV File Connector Configuration .....	30
4. Database Table Connector .....	32
4.1. Configuring the Database Table Connector .....	32
4.2. Implementation Specifics .....	33
4.3. OpenICF Interfaces Implemented by the Database Table Connector .....	33
4.4. Database Table Connector Configuration .....	35
5. PowerShell Connector Toolkit .....	42
5.1. Before You Start .....	42
5.2. Setting Up the PowerShell Connector .....	42
5.3. Configuring the PowerShell Connector .....	43
5.4. Testing the PowerShell Connector .....	45
6. Groovy Connector Toolkit .....	50
6.1. Configuring Scripted Groovy Connectors .....	50
6.2. Implemented Interfaces .....	52
6.3. Configuration Properties .....	55
6.4. Debugging Scripts Called From the Groovy Connector .....	60
7. Scripted REST Connector .....	61
7.1. Configuring the Scripted REST Connector .....	61
7.2. Using the Scripted REST Connector With a Proxy Server .....	61
7.3. Implemented Interfaces .....	61
7.4. Configuration Properties .....	63
8. Scripted SQL Connector .....	68
8.1. Configuring the Scripted SQL Connector .....	68
8.2. Implemented Interfaces .....	68
8.3. Configuration Properties .....	69
9. SAP Connector .....	78
9.1. Before You Start .....	78
9.2. Using the SAP Connector With an SAP HR System .....	79

9.3. Using the SAP Connector to Manage SAP Basis System (R/3) Users .....	85
9.4. Configuring the SAP Connector For SNC .....	102
9.5. Implementation Specifics .....	102
9.6. OpenICF Interfaces Implemented by the SAP Connector .....	103
9.7. SAP Connector Configuration .....	105
10. SSH Connector .....	111
10.1. Configuring Authentication to the SSH Server .....	112
10.2. Configuring the SSH Connector .....	113
10.3. OpenICF Interfaces Implemented by the SSH Connector .....	116
10.4. SSH Connector Configuration .....	117
11. Google Apps Connector .....	121
11.1. Configuring the Google Apps Connector .....	121
11.2. OpenICF Interfaces Implemented by the GoogleApps Connector .....	122
11.3. GoogleApps Connector Configuration .....	123
11.4. Using the Google Apps Connector With a Proxy Server .....	124
11.5. Supported Resource Types .....	125
11.6. Functional Limitations .....	125
11.7. Supported Search Filters .....	126
12. Kerberos Connector .....	127
12.1. Kerberos Connector Schema .....	127
12.2. Configuring the Kerberos Connector .....	128
12.3. OpenICF Interfaces Implemented by the Kerberos Connector .....	131
12.4. Kerberos Connector Configuration .....	132
13. Salesforce Connector .....	137
13.1. Before You Configure the Salesforce Connector .....	137
13.2. Configuring the Salesforce Connector .....	137
13.3. Implementation Specifics .....	141
13.4. OpenICF Interfaces Implemented by the Salesforce Connector .....	141
13.5. Salesforce Connector Configuration .....	142
14. Marketo Connector .....	145
14.1. Reconciling Users With a Marketo Leads Database .....	147
14.2. Implementation Specifics .....	148
14.3. OpenICF Interfaces Implemented by the Marketo Connector .....	148
14.4. Marketo Connector Configuration .....	149
15. Active Directory Connector .....	153
15.1. Configuring the Active Directory Connector .....	153
15.2. Using PowerShell Scripts With the Active Directory Connector .....	158
16. Office 365 Connector .....	161
16.1. Implementation Specifics .....	161
16.2. OpenICF Interfaces Implemented by the O365 Connector .....	161
16.3. O365 Connector Configuration .....	162
17. SCIM Connector .....	164
17.1. Implementation Specifics .....	166
17.2. Using the SCIM Connector With a Proxy Server .....	166
17.3. OpenICF Interfaces Implemented by the Scim Connector .....	167
17.4. Scim Connector Configuration .....	168
18. Adobe Marketing Cloud Connector .....	171

18.1. Before You Start .....	171
18.2. Configuring the Adobe Marketing Cloud Connector .....	172
18.3. OpenICF Interfaces Implemented by the Adobe Marketing Cloud Connector .....	174
18.4. Adobe Marketing Cloud Connector Configuration .....	175
19. Workday Connector .....	177
19.1. Before You Start .....	177
19.2. Configuring the Workday Connector .....	177
19.3. Testing the Workday Connector .....	180
19.4. Reconciling Users from Workday to IDM .....	182
19.5. Updating Users in the Workday System .....	182
19.6. Implementation Specifics .....	183
19.7. OpenICF Interfaces Implemented by the Workday Connector .....	184
19.8. Workday Connector Configuration .....	185
20. ServiceNow Connector .....	189
20.1. Before You Start .....	189
20.2. Configuring the Connector .....	189
20.3. Managing Users With the ServiceNow Connector .....	191
20.4. Implementation Specifics .....	196
20.5. OpenICF Interfaces Implemented by the ServiceNow Connector .....	196
20.6. ServiceNow Connector Configuration .....	197
21. MongoDB Connector .....	199
21.1. Before You Start .....	199
21.2. Configuring the MongoDB Connector .....	199
21.3. OpenICF Interfaces Implemented by the MongoDB Connector .....	200
21.4. MongoDB Connector Configuration .....	202
22. HubSpot Connector .....	207
22.1. Installing and Configuring the HubSpot Connector .....	207
22.2. Implementation Specifics .....	208
22.3. Using the HubSpot Connector With a Proxy Server .....	208
22.4. OpenICF Interfaces Implemented by the Hubspot Connector .....	209
22.5. Hubspot Connector Configuration .....	210
23. AWS Connector .....	212
23.1. Before you start .....	212
23.2. Install the AWS connector .....	212
23.3. Configure the AWS connector .....	213
23.4. Use the AWS connector .....	213
23.5. OpenICF Interfaces Implemented by the AWS Connector .....	220
23.6. AWS Connector Configuration .....	221
24. Cerner Connector .....	223
24.1. Before you start .....	223
24.2. Install the Cerner connector .....	223
24.3. Configure the Cerner connector .....	223
24.4. Use the Cerner connector .....	224
24.5. OpenICF Interfaces Implemented by the Cerner Connector .....	231
24.6. Cerner Connector Configuration .....	232
25. Epic Connector .....	234

25.1. Before you start .....	234
25.2. Install the Epic connector .....	235
25.3. Configure the Epic connector .....	235
25.4. Use the Epic connector .....	236
25.5. OpenICF Interfaces Implemented by the Epic Connector .....	255
25.6. Epic Connector Configuration .....	256
26. MS Graph API Java Connector .....	259
26.1. Before You Start .....	259
26.2. Configure the MS Graph API Connector .....	261
26.3. Test the MS Graph API Connector .....	261
26.4. Manage User Licenses .....	266
26.5. Synchronize Accounts Between IDM and Azure .....	269
26.6. OpenICF Interfaces Implemented by the MSGraphAPI Connector .....	269
26.7. MSGraphAPI Connector Configuration .....	270
27. PeopleSoft Connector .....	272
27.1. Before you start .....	272
27.2. Install the PeopleSoft connector .....	272
27.3. Configure the PeopleSoft connector .....	273
27.4. Use the PeopleSoft connector .....	274
27.5. OpenICF Interfaces Implemented by the PeopleSoft Connector .....	285
27.6. PeopleSoft Connector Configuration .....	286
28. SAP S/4HANA Connector .....	288
28.1. Before you start .....	288
28.2. Install the SAP S/4HANA connector .....	288
28.3. Configure the SAP S/4HANA connector .....	288
28.4. Use the SAP S/4HANA connector .....	289
28.5. OpenICF Interfaces Implemented by the SAP HANA Connector .....	299
28.6. SAP HANA Connector Configuration .....	300
A. ICF Interfaces .....	301
A.1. AttributeNormalizer .....	301
A.2. Authenticate .....	301
A.3. Batch .....	301
A.4. Connector Event .....	302
A.5. Create .....	302
A.6. Delete .....	302
A.7. Get .....	302
A.8. PoolableConnector .....	302
A.9. Resolve Username .....	302
A.10. Schema .....	302
A.11. Script on Connector .....	302
A.12. Script On Resource .....	303
A.13. Search .....	303
A.14. Sync .....	303
A.15. Sync Event .....	303
A.16. Test .....	303
A.17. Update .....	303
A.18. Update Attribute Values .....	303

---

B. ICF Operation Options .....	304
B.1. Scope .....	304
B.2. Container .....	304
B.3. Run as User .....	304
B.4. Run with Password .....	305
B.5. Attributes to Get .....	305
B.6. Paged Results Cookie .....	305
B.7. Paged Results Offset .....	305
B.8. Page Size .....	305
B.9. Sort Keys .....	305
B.10. Fail on Error .....	305
B.11. Require Serial .....	306
C. Connection Pooling Configuration .....	307

# Preface

ForgeRock Identity Platform™ serves as the basis for our simple and comprehensive Identity and Access Management solution. We help our customers deepen their relationships with their customers, and improve the productivity and connectivity of their employees and partners. For more information about ForgeRock and about the platform, see <https://www.forgerock.com>.

## 1. About This Guide

This guide describes the ICF connectors that are supported in a deployment of ForgeRock Identity Management. The guide focuses on getting the connectors installed and configured with ForgeRock Identity Management software.

This guide does not describe all ICF connectors. Additional connectors are available from the ForgeRock BackStage download site.

This guide is written for anyone using supported ICF connectors with ForgeRock Identity Management software.

You do not need to have a complete understanding of ForgeRock Identity Management to learn something from this guide, although a background in identity management and maintaining web application software can help. You do need some background in managing services on your operating systems and in your application servers. You can nevertheless get started with this guide, and learn more as you go along.

## 2. Accessing Documentation Online

ForgeRock publishes comprehensive documentation online:

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

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

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



### 3. Using the ForgeRock.org Site

The [ForgeRock.org](https://forgerock.org) site has links to source code for ForgeRock open source software, as well as links to the ForgeRock forums and technical blogs.

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

## Chapter 1

# Connector Overview

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](#).

For instructions on building connector configurations interactively, see "Configuring Connectors" in the *Integrator's Guide*.

All the connectors described in this guide are supported. This list indicates the connectors that are *bundled* with IDM 6.5.2.0:

<p>The Adobe Marketing Cloud connector lets you manage profiles in an Adobe Campaign data store.</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>The Database Table connector enables provisioning to a single table in a JDBC database.</p>
<p>The Google Apps connector lets you interact with Google's web applications.</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>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). This connector lets you manage Kerberos user principals from IDM.</p>
<p>The LDAP connector is based on 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.</p>	<p>The Marketo connector lets you synchronize between IDM managed users and a Marketo Leads Database.</p>	<p>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.</p>
<p>The Salesforce connector enables provisioning, reconciliation, and synchronization between Salesforce and the IDM repository.</p>	<p>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.</p>	<p>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.</p>

<p>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.</p>	<p>The ServiceNow connector lets you manage objects in the ServiceNow platform, integrating with ServiceNow's REST API.</p>	<p>The SSH connector is an implementation of the Scripted Groovy Connector, and 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.</p>
<p>The Workday connector lets you synchronize user accounts between IDM and Workday's cloud-based HR system.</p>		

This list indicates the connectors that are not bundled with IDM 6.5.2.0 but available from the ForgeRock Download Center:

<p>The AWS connector lets you interact with the AWS IAM service.</p>	<p>The Cerner connector lets you interact with Cerner healthcare IT systems.</p>	<p>The Epic connector lets you interact with Epic health systems.</p>
<p>The HubSpot connector lets you synchronize HubSpot contacts and companies with managed objects in an IDM repository.</p>	<p>The PeopleSoft connector lets you interact with Oracle PeopleSoft systems.</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>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 S/4HANA connector lets you synchronize user accounts between IDM and the SAP S/4HANA service.</p>	

## Chapter 2

# Generic LDAP Connector

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.

ICF provides a legacy Active Directory (AD) .NET connector. Note, however, that the AD Connector is deprecated and support for its use with IDM will be discontinued in a future release. For simple Active Directory (and Active Directory LDS) deployments, the generic LDAP Connector works better than the Active Directory connector, in most circumstances. Using the generic LDAP connector avoids the need to install a remote connector server in the overall deployment. In addition, the generic LDAP connector has significant performance advantages over the Active Directory connector. For more complex Active Directory deployments, use the PowerShell Connector Toolkit, as described in "*PowerShell Connector Toolkit*".

## 2.1. Setting Up the Generic LDAP Connector

IDM bundles version 1.5.20.8 of the LDAP connector. Three 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.
- `provisioner.openicf-adldsldap.json` provides a sample LDAP connector configuration for an Active Directory Lightweight Directory Services (AD LDS) 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": "[1.5.19.0,1.6.0.0]"
  }
}
```

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" : "cn=Directory Manager",
  "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 *Configuring Referential Integrity* in the *Developer's 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. For a DS resource, you should use the `entryUUID`. Although 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`.

### 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.

### vlvSortAttribute

Attribute used as the sort key for virtual list view.

### sendCAUDTxId

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

## 2.2. 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:

```
$ cd /path/to/openidm/security
$ keytool \
  -importcert \
  -alias server-cert \
  -keystore 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'
```

## 2.3. 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 operations includes liveSync (synchronization of changes from the LDAP server to IDM) and implicit sync (synchronization from IDM to the LDAP server).

### 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.

### 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.

### 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.

### 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. Automatic synchronization includes liveSync and implicit synchronization operations. For more information, see "Types of Synchronization" in the *Integrator's Guide*

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.

### modifiersNamesToFilterOut

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

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 `cn=directory manager`, for example `cn=openidmuser`. 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=openidmuser`) 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 synchronized 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.

## 2.4. Using the Generic LDAP Connector With Active Directory

The LDAP connector provides functionality specifically for managing Active Directory users and groups. This connector is supported with Active Directory Domain Controllers, Active Directory Global Catalogues, and Active Directory Lightweight Directory Services (LDS).

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.

## 2.4.1. 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" \
--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.

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.

## 2.4.2. 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" \
--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" \
--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"
}
```

### 2.4.3. Adding Users to Active Directory Groups

With the sample provisioner file, you cannot change the groups of which a user is a member from the user side. Effectively, you can add members to a group but you cannot add groups to a member. (This is also the case if you configure the connector through the Admin UI.)

To change this behavior, add the `ldapGroups` property to the `account` object in your provisioner file. For example:

```
"ldapGroups" : {
  "type" : "array",
  "items" : {
    "type" : "string",
    "nativeType" : "string"
  },
  "nativeName" : "ldapGroups",
  "nativeType" : "string"
},
```

When the connector configuration includes `ldapGroups`, you can update a user's group membership by patching their user entry. The following command adds user Brian Smith, created previously, to the `Employees` group:

```
$ curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request PATCH \
--data '[
  {
    "operation": "add",
    "field": "/ldapGroups/-",
    "value": "CN=Employees,DC=example,DC=com"
  }
]' \
"http://localhost:8080/openidm/system/ad/account/e1418d64-096c-4cb0-b903-ebb66562d99d"
```

## 2.4.4. 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`.

## 2.4.5. 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.

## 2.5. 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))
```

## 2.6. 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.

# 2.7. LDAP Connector Configuration

The LDAP Connector has the following configurable properties.

## 2.7.1. Configuration properties

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<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

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<p>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.</p>				
<code>changeNumberAttribute</code>	<code>String</code>	<code>changeNumber</code>		Sync
<p>The name of the change number attribute in the change log entry.</p>				
<code>modifiersNamesToFilterOut</code>	<code>String[]</code>	<code>[]</code>		Sync
<p>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".</p>				
<code>credentials</code>	<code>GuardedString</code>	<code>null</code>	Yes	No
<p>Password for the principal.</p>				
<code>changeLogBlockSize</code>	<code>int</code>	<code>100</code>		Sync
<p>The number of change log entries to fetch per query.</p>				
<code>useTimestampsForSync</code>	<code>boolean</code>	<code>false</code>		Sync
<p>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.</p>				
<code>accountSynchronizationFilter</code>	<code>String</code>	<code>null</code>		Sync
<p>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.</p>				
<code>removeLogEntryObjectClassFromFilter</code>	<code>boolean</code>	<code>true</code>		Sync
<p>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.</p>				
<code>alternateKeyStorePassword</code>	<code>GuardedString</code>	<code>null</code>	Yes	No
<p>Password to use for the alternate keystore</p>				
<code>groupSynchronizationFilter</code>	<code>String</code>	<code>null</code>		Sync
<p>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.</p>				
<code>groupMemberAttribute</code>	<code>String</code>	<code>uniqueMember</code>		No
<p>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.</p>				



Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<code>accountSearchFilter</code>	String	null		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>	String	null		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>	boolean	false		No
Select the check box to connect to the LDAP server using SSL.				
<code>maintainPosixGroupMembership</code>	boolean	false		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>	long	60		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>	String	null		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.				
<code>referralsHandling</code>	String	follow		No
Defines how to handle LDAP referrals. Possible values can be follow, ignore or throw.				
<code>host</code>	String	null		No
The name or IP address of the host where the LDAP server is running.				
<code>maintainLdapGroupMembership</code>	boolean	false		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	never		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	uid		No
Specify the sort attribute to use for VLV indexes on the resource.				

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<code>convertGTTtoISO8601</code>	<code>String[]</code>	<code>['whenCreated', 'whenChanged']</code>		No
Converts the Greenwich Time to ISO8601 format				
<code>baseContexts</code>	<code>String[]</code>	<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>	<code>boolean</code>	<code>false</code>		No
If true, the connector will verify the hostname in the certificate (subject + alternative subject) against the defined <code>hostNameVerifierPattern</code> .				
<code>blockSize</code>	<code>int</code>	<code>100</code>		No
The maximum number of entries that can be in a block when retrieving entries in blocks.				
<code>groupObjectClasses</code>	<code>String[]</code>	<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>	<code>String[]</code>	<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.				
<code>failover</code>	<code>String[]</code>	<code>[]</code>		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.				
<code>port</code>	<code>int</code>	<code>389</code>		No
TCP/IP port number used to communicate with the LDAP server.				
<code>convertADIntervalToISO8601</code>	<code>String[]</code>	<code>['pwdLastSet', 'accountExpires', 'lockoutTime', 'lastLogon']</code>		No
Converts the AD Interval to ISO8601				
<code>hostNameVerifierPattern</code>	<code>String</code>	<code>null</code>		No
A simple pattern used to match the hostname from the certificate. It can contains * character (server1.example.com, *.example.com)				
<code>passwordAttribute</code>	<code>String</code>	<code>userPassword</code>		No

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
The name of the LDAP attribute that holds the password. When changing a users password, the new password is set to this attribute.				
<code>useDNSSRVRecord</code>	<code>boolean</code>	<code>false</code>		No
If true, the connector will do a DNS query to find SRV records associated with the value set for host property (" <code>_ldap._tcp.example.com</code> " for example). Defaults to false.				
<code>getGroupMemberId</code>	<code>boolean</code>	<code>false</code>		No
Specifies whether to add an extra <code>_memberId</code> attribute to get the group members <code>__UID__</code> . CAUTION: Setting this property to true can incur a large performance cost on group handling.				
<code>lastCheckAlive</code>	<code>long</code>	<code>1659981153983</code>		No
The last time the connector was checked to see if it was alive				
<code>startTLS</code>	<code>boolean</code>	<code>false</code>		No
Specifies whether to use the startTLS operation to initiate a TLS/SSL session.				
<code>allowTreeDelete</code>	<code>boolean</code>	<code>false</code>		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.				
<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>alternateKeyStoreType</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

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
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 Forgerocks OpenDJ does, for example).				
<code>alternateKeyStore</code>	String	null		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>	String	simple		No
The authentication mechanism to use: Simple or SASL-GSSAPI. Defaults to "simple".				
<code>connectionTimeout</code>	int	30000		No
The timeout (in ms) before the connection attempt is aborted.				
<code>useBlocks</code>	boolean	false		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>	boolean	true		No
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".				

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.

## Chapter 3

# CSV File Connector

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.

## 3.1. 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": "[1.5.19.0,1.6.0.0)"
  }
}
```

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.

## 3.2. 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.

# 3.3. CSV File Connector Configuration

The CSV File Connector has the following configurable properties.

## 3.3.1. Configuration properties

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<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.				
<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

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
The character in the CSV file that is used to separate field values.				
<code>syncFileRetentionCount</code>	<code>int</code>	<code>3</code>		No
The number of historical copies of the CSV file to retain when performing synchronization operations.				

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.



## Chapter 4

# Database Table Connector

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

## 4.1. 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" :
{
  "quoting" : "",
  "host" : "localhost",
  "port" : "3306",
  "user" : "root",
  "password" : "",
  "database" : "contractordb",
  "table" : "people",
  "keyColumn" : "UNIQUE_ID",
  "passwordColumn" : "",
  "jdbcDriver" : "com.mysql.jdbc.Driver",
  "jdbcUrlTemplate" : "jdbc:mysql://%h:%p/%d",
  "enableEmptyString" : false,
  "rethrowAllSQLExceptions" : true,
  "nativeTimestamps" : true,
  "allNative" : false,
  "validConnectionQuery" : null,
  "changeLogColumn" : "CHANGE_TIMESTAMP",
  "datasource" : "",
  "jndiProperties" : null
},
```

The mandatory configurable properties are as follows:

### database

The JDBC database that contains the table to which you are provisioning.

### 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.

## 4.2. 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 *Integrator's Guide*.

## 4.3. 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.

## 4.4. Database Table Connector Configuration

The Database Table Connector has the following configurable properties.

### 4.4.1. Configuration properties

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<code>connectionProperties</code>	String	null		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>	boolean	false		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>	boolean	true		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>	String	null		No
The default catalog of connections created by this pool.				
<code>validationInterval</code>	long	3000		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>	boolean	false		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>	boolean	true		No
Register the pool with JMX or not. The default value is true.				
<code>commitOnReturn</code>	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.				
<code>logAbandoned</code>	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.				

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<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. Idle connections are checked periodically (if enabled) and connections that have been idle for longer than <code>minEvictableIdleTimeMillis</code> are released. The default value is derived from <code>maxActive:100</code> . (Also see <code>testWhileIdle</code> .)				
<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 <code>validationQuery</code> 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 <code>timeBetweenEvictionRunsMillis</code> )				
<code>removeAbandoned</code>	<code>boolean</code>	<code>false</code>		No
Flag to remove abandoned connections if they exceed the <code>removeAbandonedTimeout</code> . If set to true a connection is considered abandoned and eligible for removal if it has been in use longer than the <code>removeAbandonedTimeout</code> Setting this to true can recover db connections from applications that fail to close a connection. See also <code>logAbandoned</code> The default value is false.				
<code>abandonWhenPercentageFull</code>	<code>int</code>	<code>0</code>		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 <code>abandonWhenPercentageFull</code> . The value should be between 0-100. The default value is 0, which implies that connections are eligible for closure as soon as <code>removeAbandonedTimeout</code> has been reached.				
<code>minIdle</code>	<code>int</code>	<code>10</code>		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 <code>initialSize:10</code> . (Also see <code>testWhileIdle</code> .)				
<code>defaultReadOnly</code>	<code>Boolean</code>	<code>null</code>		No
The default read-only state of connections created by this pool. If not set then the <code>setReadOnly</code> method will not be called. (Some drivers dont support read only mode, ex: Informix)				
<code>maxWait</code>	<code>int</code>	<code>30000</code>		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)				
<code>logValidationErrors</code>	<code>boolean</code>	<code>false</code>		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.				
<code>driverClassName</code>	<code>String</code>	<code>null</code>		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 <code>tomcat-jdbc.jar</code>				

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<code>name</code>	<code>String</code>	<code>Tomcat Connection Pool[1 -153647080]</code>		No
Returns the name of the connection pool. By default a JVM unique random name is assigned.				
<code>useStatementFacade</code>	<code>boolean</code>	<code>true</code>		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.				
<code>initSQL</code>	<code>String</code>	<code>null</code>		No
A custom query to be run when a connection is first created. The default value is null.				
<code>validationQueryTimeout</code>	<code>int</code>	<code>-1</code>		No
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.				
<code>validationQuery</code>	<code>String</code>	<code>null</code>		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 can't throw a <code>SQLException</code> . The default value is null. Example values are <code>SELECT 1(mysql)</code> , <code>select 1 from dual(oracle)</code> , <code>SELECT 1(MS Sql Server)</code>				
<code>rollbackOnReturn</code>	<code>boolean</code>	<code>false</code>		No
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 false.				
<code>alternateUsernameAllowed</code>	<code>boolean</code>	<code>false</code>		No
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 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.				
<code>validatorClassName</code>	<code>String</code>	<code>null</code>		No
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> .				
<code>suspectTimeout</code>	<code>int</code>	<code>0</code>		No

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
Timeout value in seconds. Similar to to the removeAbandonedTimeout value but instead of treating the connection as abandoned, and potentially closing the connection, this simply logs the warning if logAbandoned 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.				
<code>useEquals</code>	<code>boolean</code>	<code>true</code>		No
Set to true if you wish the ProxyConnection class to use String.equals and set to false when you wish to use == when comparing method names. This property does not apply to added interceptors as those are configured individually. The default value is true.				
<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 setAutoCommit method will not be called.)				
<code>testOnConnect</code>	<code>boolean</code>	<code>false</code>		No
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 validationQuery as an init query.				
<code>jdbcInterceptors</code>	<code>String</code>	<code>null</code>		No
A semicolon separated list of classnames extending org.apache.tomcat.jdbc.pool.JdbcInterceptor 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 java.test_sample.Connection 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

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
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				
fairQueue	boolean	true		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.				
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
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.				
maxActive	int	100		No



Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
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>	String	null		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>	String	TABLE_NAME		Yes
Enter the name of the table in the database that contains the accounts.				

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.

## 4.4.2. Basic Configuration

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<code>password</code>	String	null	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>	String	NONE		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>	String	KEY_COLUMN		Yes
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

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
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				

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.

## Chapter 5

# PowerShell Connector Toolkit

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 "*Connecting to Active Directory With the PowerShell Connector*" in the *Samples Guide*.

## 5.1. 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 "Installing and Configuring a .NET Connector Server" in the *Integrator's Guide*.
- PowerShell version 4.0 or above.
- The PowerShell Connector Toolkit.

## 5.2. 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.

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

2. Configure IDM to connect to the .NET connector server.

To do so, copy the remote connector provisioner file from the `openidm\samples\provisioners` directory to your project's `conf\` directory, and edit the file to match your configuration.

```
PS C:\ > cd \path\to\openidm
PS C:\path\to\openidm > cp samples\example-configurations\provisioners\provisioner.openicf
.connectorinfoprovider.json conf
```

For instructions on editing this file, see "Configuring IDM to Connect to the .NET Connector Server" in the *Integrator's Guide*.

3. Download the PowerShell Connector Toolkit archive (`mspowershell-connector-1.4.5.0.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",
...
```

5. Copy the sample connector configuration file (`provisioner.openicf-adpowershell.json`) from the `samples\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.

## 5.3. Configuring the PowerShell Connector

Your PowerShell connector configuration file should include the following properties:

Property	Type	Example	Encrypted <sup>a</sup>	Required <sup>b</sup>
<code>operationScriptFileName</code>	String	<code>C:/openidm/AD/ADCreate.ps1,</code>	No	Yes

Property	Type	Example	Encrypted <sup>a</sup>	Required <sup>b</sup>
The full path to the script that implements the corresponding OpenICF operation.				
<code>VariablesPrefix</code>	String	Connector	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	AdPsModule (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"> <li>• <code>Map</code> - the query filter is a map</li> <li>• <code>Ldap</code> - the query filter is in LDAP search format, for example, "(cn=Joe)"</li> <li>• <code>Native</code> - the query filter is a native OpenICF query filter</li> <li>• <code>AdPsModule</code> - the query filter is compatible with the Active Directory PowerShell module, <code>Get-ADUser Filter</code></li> </ul>				
<code>ReloadScriptOnExecution</code>	Boolean	true	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	true	No	No
If <code>true</code> , the connector leverages the PowerShell RunSpace Pool.				
<code>MaxInterpretersPoolSize</code>	Integer	5	No	No
The maximum size of the interpreter pool.				
<code>MinInterpretersPoolSize</code>	Integer	1	No	No
The minimum size of the interpreter pool.				
<code>PoolCleanupInterval</code>	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.				
<code>SubstituteUidAndNameInQueryFilter</code>	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.				
<code>UidAttributeName</code>	String	ObjectGUID	No	No
The attribute on the resource that contains the object <code>__UID__</code>				
<code>NameAttributeName</code>	String	DistinguishedName	No	No
The attribute on the resource that contains the object <code>__NAME__</code>				
<code>PsModulesToImport</code>	Array	[ "ActiveDirectory", "C:/openidm/samples/scripted-	No	No

Property	Type	Example	Encrypted <sup>a</sup>	Required <sup>b</sup>
		<code>powershell-with-ad/tools/ADSIsearch.psm1"],</code>		
An array of additional PowerShell modules that the connector must import				
<b>Host</b>	String	<code>ad.example.com</code>	No	Yes
The host name or IP address of the Active Directory server				
<b>Port</b>	Integer	<code>null</code>	No	Yes
The port number on which the remote resource listens for connections				
<b>Login</b>	String	<code>""</code>	No	Yes
The user account in the remote resource that is used for the connection				
<b>Password</b>	String	<code>null</code>	Encrypted	Yes
The password of the user account that is used for the connection				
<b>CustomProperties</b>	Array	<code>[ ]</code>	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>				
The custom property can then be read from the PowerShell scripts as follows: <code>\$base = \$Connector.Configuration.PropertyBag.baseContext</code>				

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in IDM.

<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.

## 5.4. 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.

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

### 5.4.1. 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" \
--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/](#) subdirectory of that server.

## 5.4.2. 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 *Integrator's Guide*.

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

## 5.4.3. 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"
\
--request POST
\
--header "content-type: application/json"
\
--data '{
  "PasswordNeverExpires": false,
  "AlternateEmailAddresses": ["Robert.Smith@example.com"],
  "LastName": "Smith",
  "PreferredLanguage": "en-US",
  "FirstName": "Robert",
  "UserPrincipalName": "Robert.Smith@example.onmicrosoft.com",
  "DisplayName": "Robert Smith"
}' \
"http://localhost:8080/openidm/system/adpowershell/account?_action=create"
```

#### 5.4.4. 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"
\
--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"
```



### 5.4.5. 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" \
--request DELETE \
"http://localhost:8080/openidm/system/adpowershell/account/1d4c9276-6937-4d9e-9c60-67e8b4207f4e"
```

### 5.4.6. Running a Script on the Connector

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 (`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" \
--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"
      }
    ]
  }
]
```

## Chapter 6

# Groovy Connector Toolkit

ICF provides a generic Groovy Connector Toolkit that lets you 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.

## 6.1. 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.

### 6.1.1. Validating Pooled Connections

The scripted SQL connector uses the Tomcat JDBC Connection Pool to managed its connections. Occasionally, a JDBC resource that is accessed by the scripted SQL connector might become unavailable for a period. When the resource comes back online, IDM is generally 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 this 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,
  }
}
```

## 6.1.2. 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>'}",
```

## 6.2. Implemented Interfaces

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

### 6.2.1. 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.

## 6.2.2. 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.

# 6.3. Configuration Properties

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

## 6.3.1. Scripted Groovy Connector Configuration

The Scripted Groovy Connector has the following configurable properties.

### 6.3.1.1. Configuration properties

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<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				



Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
----------	------	---------	------------------------	-----------------------

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.

### 6.3.1.2. Operation Script Files

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<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.				

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.

### 6.3.1.3. Groovy Engine configuration

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<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
Encoding for source files				
<code>recompileGroovySource</code>	boolean	false		No

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
If set to true recompilation is enabled				

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.

## 6.3.2. Scripted Poolable Groovy Connector Configuration

The Scripted Poolable Groovy Connector has the following configurable properties.

### 6.3.2.1. Configuration properties

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<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				

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.

### 6.3.2.2. Operation Script Files

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<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.				
<code>resolveUsernameScriptFileName</code>	<code>String</code>	<code>null</code>		Resolve Username

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
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.				

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.

### 6.3.2.3. Groovy Engine configuration

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<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

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
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				

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.

## 6.4. Debugging Scripts Called From the Groovy Connector

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.](#)

## Chapter 7

# Scripted REST Connector

The Scripted REST connector is an implementation of the Scripted Groovy Connector Toolkit. This connector lets you interact with any REST API, using Groovy scripts for the ICF operations.

### Note

The Scripted REST connector is not a *poolable* connector.

## 7.1. Configuring the Scripted REST Connector

The Scripted REST Connector is bundled in the JAR `openidm/connectors/scriptedrest-connector-1.5.20.8.jar`.

A sample connector configuration and scripts are provided in the `/path/to/openidm/samples/scripted-rest-with-dj/` directory and described in "*Connecting 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*.

## 7.2. 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",  
  ...  
},
```

## 7.3. Implemented Interfaces

The following table lists the ICF interfaces that are implemented for the scripted REST connector:

### 7.3.1. 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.

# 7.4. Configuration Properties

The following table lists the configuration properties for the scripted REST connector:

## 7.4.1. Scripted REST Connector Configuration

The Scripted REST Connector has the following configurable properties.

### 7.4.1.1. Configuration properties

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<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				

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.

### 7.4.1.2. Operation Script Files

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<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



Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
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.				

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.

### 7.4.1.3. Groovy Engine configuration

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<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				

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<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				

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.

#### 7.4.1.4. Basic Configuration Properties

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<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

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
The Password to authenticate with				
<code>serviceAddress</code>	URI	null		Yes
The service URI (example: http://myservice.com/api)				
<code>proxyAddress</code>	URI	null		No
The optional Proxy server URI (example: http://myproxy:8080)				
<code>proxyUsername</code>	String	null		No
The username to authenticate with the proxy server				
<code>proxyPassword</code>	GuardedString	null	Yes	No
The password to authenticate with the proxy server				
<code>defaultAuthMethod</code>	String	BASIC		No
Authentication method used. Defaults to BASIC.				
<code>defaultContentType</code>	String	application/ json		No
Default HTTP request content type. Defaults to JSON. Can be: TEXT, XML, HTML, URLENC, BINARY				
<code>defaultRequestHeaders</code>	String[]	null		No
Placeholder for default HTTP request headers.				
<code>OAuthTokenEndpoint</code>	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)				
<code>OAuthClientId</code>	String	null		No
The client identifier				
<code>OAuthClientSecret</code>	GuardedString	null	Yes	No
Secure client secret for OAUTH				
<code>OAuthRefreshToken</code>	GuardedString	null	Yes	No
The refresh token used to renew the access token for the refresh_token grant type				
<code>OAuthScope</code>	String	null		No
The optional scope				
<code>OAuthGrantType</code>	String	CLIENT_CREDENTIALS		No
The grant type to use. Can be CLIENT_CREDENTIALS (default)   REFRESH_TOKEN   AUTHORIZATION_CODE				

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
----------	------	---------	------------------------	-----------------------

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.

## Chapter 8

# Scripted SQL Connector

The Scripted SQL connector is an implementation of the Scripted Groovy Connector Toolkit. This connector lets you interact with any SQL database, using Groovy scripts for the ICF operations.

## 8.1. 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 "*Connecting 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*.

## 8.2. Implemented Interfaces

The following table lists the ICF interfaces that are implemented for the scripted SQL connector:

### 8.2.1. 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.

# 8.3. Configuration Properties

The following table lists the configuration properties for the scripted SQL connector:

## 8.3.1. Scripted SQL Connector Configuration

The Scripted SQL Connector has the following configurable properties.

### 8.3.1.1. Operation Script Files

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<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
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.				

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.

### 8.3.1.2. Groovy Engine configuration

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<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.				
<code>sourceEncoding</code>	String	UTF-8		No
Encoding for source files				
<code>recompileGroovySource</code>	boolean	false		No
If set to true recompilation is enabled				
<code>minimumRecompilationInterval</code>	int	100		No
Sets the minimum of time after a script can be recompiled.				
<code>debug</code>	boolean	false		No
If true, debugging code should be activated				
<code>classpath</code>	String[]	[]		No
Classpath for use during compilation.				
<code>verbose</code>	boolean	false		No
If true, the compiler should produce action information				

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.



<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.

### 8.3.1.3. Configuration properties

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<code>password</code>	String	null	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>	GuardedString	null	Yes	No
Custom Sensitive Configuration script for Groovy ConfigSlurper				
<code>customConfiguration</code>	String	null		No
Custom Configuration script for Groovy ConfigSlurper				
<code>connectionProperties</code>	String	null		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>	boolean	false		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>	boolean	true		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>	String	null		No
The default catalog of connections created by this pool.				
<code>validationInterval</code>	long	3000		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>	boolean	false		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>	boolean	true		No
Register the pool with JMX or not. The default value is true.				

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<code>commitOnReturn</code>	<code>boolean</code>	<code>false</code>		No
If <code>autoCommit==false</code> then the pool can complete the transaction by calling <code>commit</code> on the connection as it is returned to the pool. If <code>rollbackOnReturn==true</code> then this attribute is ignored. Default value is <code>false</code> .				
<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 <code>false</code> .				
<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 <code>maxActive:100</code> . Idle connections are checked periodically (if enabled) and connections that been idle for longer than <code>minEvictableIdleTimeMillis</code> will be released. (also see <code>testWhileIdle</code> )				
<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 <code>validationQuery</code> parameter must be set to a non-null string. The default value is <code>false</code> and this property has to be set in order for the pool cleaner/test thread is to run (also see <code>timeBetweenEvictionRunsMillis</code> )				
<code>removeAbandoned</code>	<code>boolean</code>	<code>false</code>		No
Flag to remove abandoned connections if they exceed the <code>removeAbandonedTimeout</code> . If set to true a connection is considered abandoned and eligible for removal if it has been in use longer than the <code>removeAbandonedTimeout</code> . Setting this to true can recover db connections from applications that fail to close a connection. See also <code>logAbandoned</code> . The default value is <code>false</code> .				
<code>abandonWhenPercentageFull</code>	<code>int</code>	<code>0</code>		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 <code>abandonWhenPercentageFull</code> . The value should be between 0-100. The default value is 0, which implies that connections are eligible for closure as soon as <code>removeAbandonedTimeout</code> has been reached.				
<code>minIdle</code>	<code>int</code>	<code>10</code>		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 <code>initialSize:10</code> (also see <code>testWhileIdle</code> )				
<code>defaultReadOnly</code>	<code>Boolean</code>	<code>null</code>		No
The default read-only state of connections created by this pool. If not set then the <code>setReadOnly</code> method will not be called. (Some drivers dont support read only mode, ex: Informix)				
<code>maxWait</code>	<code>int</code>	<code>30000</code>		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)				

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<code>logValidationErrors</code>	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.				
<code>driverClassName</code>	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				
<code>name</code>	String	Tomcat Connection Pool[4 -153647080]		No
Returns the name of the connection pool. By default a JVM unique random name is assigned.				
<code>useStatementFacade</code>	boolean	true		No
If a statement proxy is set, wrap statements so that equals() and hashCode() methods can be called on closed statements.				
<code>initSQL</code>	String	null		No
A custom query to be run when a connection is first created. The default value is null.				
<code>validationQueryTimeout</code>	int	-1		No
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 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.				
<code>validationQuery</code>	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 <code>SQLException</code> . The default value is null. Example values are <code>SELECT 1(mysql)</code> , <code>select 1 from dual(oracle)</code> , <code>SELECT 1(MS Sql Server)</code>				
<code>rollbackOnReturn</code>	boolean	false		No
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 false.				
<code>alternateUsernameAllowed</code>	boolean	false		No
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 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.				

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<code>dataSourceJNDI</code>	String	null		No
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 null				
<code>validatorClassName</code>	String	null		No
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> .				
<code>suspectTimeout</code>	int	0		No
Timeout value in seconds. Similar to 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.				
<code>useEquals</code>	boolean	true		No
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.				
<code>removeAbandonedTimeout</code>	int	60		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>	Boolean	null		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>	boolean	false		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>	String	null		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>	int	10		No
The initial number of connections that are created when the pool is started. Default value is 10				
<code>defaultTransactionIsolation</code>	int	-1		No

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
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.				
numTestsPerEvictionRun	int	0		No
Property not used in tomcat-jdbc-pool.				
url	String	null		No
The URL used to connect to the database.				
testOnBorrow	boolean	false		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				
fairQueue	boolean	true		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.				
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).				

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<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
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.				
<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.				

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.

## Chapter 9

# SAP Connector

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

## 9.1. Before You Start

1. Download the SAP connector from the ForgeRock BackStage download site.
2. Copy the SAP connector JAR file (`sap-connector-1.5.1.0.jar`) to the `openidm/connectors` directory:

```
$ cp ~/Downloads/sap-connector-1.5.1.0.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
```

## 9.2. 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.

### 9.2.1. 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`.

2. 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).

## 9.2.2. 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.

```
$ cd 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" \
--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" : "1.5.1.0"
  },
  "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" \
--request GET \
"http://localhost:8080/openidm/system/saphr/employee?_queryId=query-all-ids"
{
  "result" : [ {
    "_id" : "00000010",
    "_NAME_" : "00000010"
  }, {
    "_id" : "00000069",
    "_NAME_" : "00000069"
  }, {
    "_id" : "00000070",
    "_NAME_" : "00000070"
  }
]
,
...

```

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" \
--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"
  }
}
,
...
}

```

### 9.2.3. 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" \
--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 "Content-Type: application/json" \
--request PUT \
--data '{
  "EMAIL": { "ID": "maria.gonzales@example.com" }
}' \
"http://localhost:8080/openidm/system/saphr/employee/55099307"
{
  "_id" : "55099307",
  "EMAIL" : [ {
    "EMPLOYEEENO" : "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"
  }
}
```

- 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 "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"
```

- 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 Gonzales' new mail address to December 31st, 2016:

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

- 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 "Content-Type: application/json" \
--request PUT \
--data '{
  "EMAIL": {
    "RECORDNR" : "000",
    "VALIDEND" : "99991231",
    "VALIDBEGIN" : "20000101"
  }
}' \
"http://localhost:8080/openidm/system/saphr/employee/55099307"
```

## 9.3. Using the SAP Connector to Manage SAP Basis System (R/3) Users

The SAP Connector lets you 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.

### 9.3.1. 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).

## 9.3.2. 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.

```
$ cd 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" \
--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": "1.5.1.0"
  },
  "displayName": "Sap Connector",
  "ok": true
}
```

### 9.3.3. Using the SAP Connector to Manage SAP R/3 Users

This section provides sample commands for managing users in an SAP system.

#### 9.3.3.1. 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" \
--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
}

```

### 9.3.3.2. 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" \
--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

### 9.3.3.3. 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 "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 into 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 "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 "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"
  }
}
,
...
```

### 9.3.3.3.1. 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
- **ADDTTEL** - 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 "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)
}

```

### 9.3.3.4. Updating SAP User Accounts

The following sections provide sample commands for updating an existing user account.

### 9.3.3.4.1. 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 "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 "Content-Type: application/json" \
--header "If-Match: *" \
--request PUT \
--data '{
  "__LOCK_OUT__": false
}' \
"http://localhost:8080/openidm/system/mysap/user/KVAUGHAN"
```

### 9.3.3.4.2. 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 "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",
    "TELI_NUMBR": "33297603177",
    "E_MAIL": "katie.vaughan@example.com",
    "LANGU_CR_P": "E",
    "LANGUCPIISO": "EN"
  }
}' \
"http://localhost:8080/openidm/system/mysap/user/KVAUGHAN"
```

### 9.3.3.4.3. 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 "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" \
--request PUT \
--data '{
  "__PASSWORD__": "MyPassw0rd",
  "__PASSWORD_EXPIRED__": false
}'
"http://localhost:8080/openidm/system/mysap/user/KVAUGHAN"
```

### 9.3.3.5. 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" \
--request DELETE \
"http://localhost:8080/openidm/system/mysap/user/KVAUGHAN"
```

The command returns the complete user object that was deleted.

### 9.3.3.6. 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.

#### 9.3.3.6.1. 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 "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.

### 9.3.3.6.2. 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 "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"
}
    
```

### 9.3.3.6.3. 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 "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.

### 9.3.3.7. 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.

### 9.3.3.7.1. 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 "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.

### 9.3.3.7.2. 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 "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"
}
```

### 9.3.3.7.3. 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 "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.

## 9.4. 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 Properties".

## 9.5. 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.

## 9.5.1. 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 the corresponding SAP Note 1287410 at <https://service.sap.com/sap/support/notes/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.

## 9.6. 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.

## 9.7. SAP Connector Configuration

The SAP Connector has the following configurable properties.

### 9.7.1. Operation Script Files Properties

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<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
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.				

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.

## 9.7.2. Groovy Engine configuration Properties

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<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.				
<code>sourceEncoding</code>	String	UTF-8		No
Encoding for source files				
<code>recompileGroovySource</code>	boolean	false		No
If set to true recompilation is enabled				
<code>minimumRecompilationInterval</code>	int	100		No
Sets the minimum of time after a script can be recompiled.				
<code>debug</code>	boolean	false		No
If true, debugging code should be activated				
<code>classpath</code>	String[]	[]		No
Classpath for use during compilation.				
<code>verbose</code>	boolean	false		No

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
If true, the compiler should produce action information				

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.

### 9.7.3. Configuration Properties

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<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>x509Cert</code>	<code>String</code>	<code>null</code>	Yes	No
The X509 certificate supplied for authentication.				

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.

### 9.7.4. Basic Configuration Properties

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<code>asHost</code>	<code>String</code>	<code>null</code>		Yes
The FQDN of your SAP Application Server, for example sap.example.com				
<code>gwHost</code>	<code>String</code>	<code>null</code>		Yes
SAP gateway host name				
<code>gwServ</code>	<code>String</code>	<code>null</code>		Yes
SAP gateway service				
<code>user</code>	<code>String</code>	<code>null</code>		Yes
SAP Logon user				
<code>password</code>	<code>GuardedString</code>	<code>null</code>	Yes	Yes
SAP Logon password				
<code>client</code>	<code>String</code>	<code>000</code>		Yes
SAP client				
<code>systemNumber</code>	<code>String</code>	<code>00</code>		Yes

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
SAP system number				
language	String	EN		Yes
SAP Logon language				
destination	String	OPENIDM		Yes
SAP JCo destination name				
directConnection	boolean	true		Yes
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])				

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.

### 9.7.5. SAP Jco Logs Configuration Properties

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
trace	String	0		No
Enable/disable RFC trace (0 or 1)				
cpicTrace	String	0		No
Enable/disable CPIC trace [0..3]				

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.

### 9.7.6. Advanced Configuration Properties

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
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				

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
r3Name	String	null		No
Specifies the name of the SAP system, used when you log in to a logon group that uses load balancing				

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.

### 9.7.7. SAP Secure Network Connection Configuration Properties

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
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).				

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.

### 9.7.8. JCo Connection Pool Configuration Properties

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
poolCapacity	String	1		No
Maximum number of idle connections kept open by the destination. 0 = no connection pooling. Default is 1.				

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<code>expirationTime</code>	String	60000		No
Time in ms after that a free connection can be closed. Default is one minute.				
<code>maxGetTime</code>	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.				
<code>peakLimit</code>	String	0		No
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				

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.

## Chapter 10

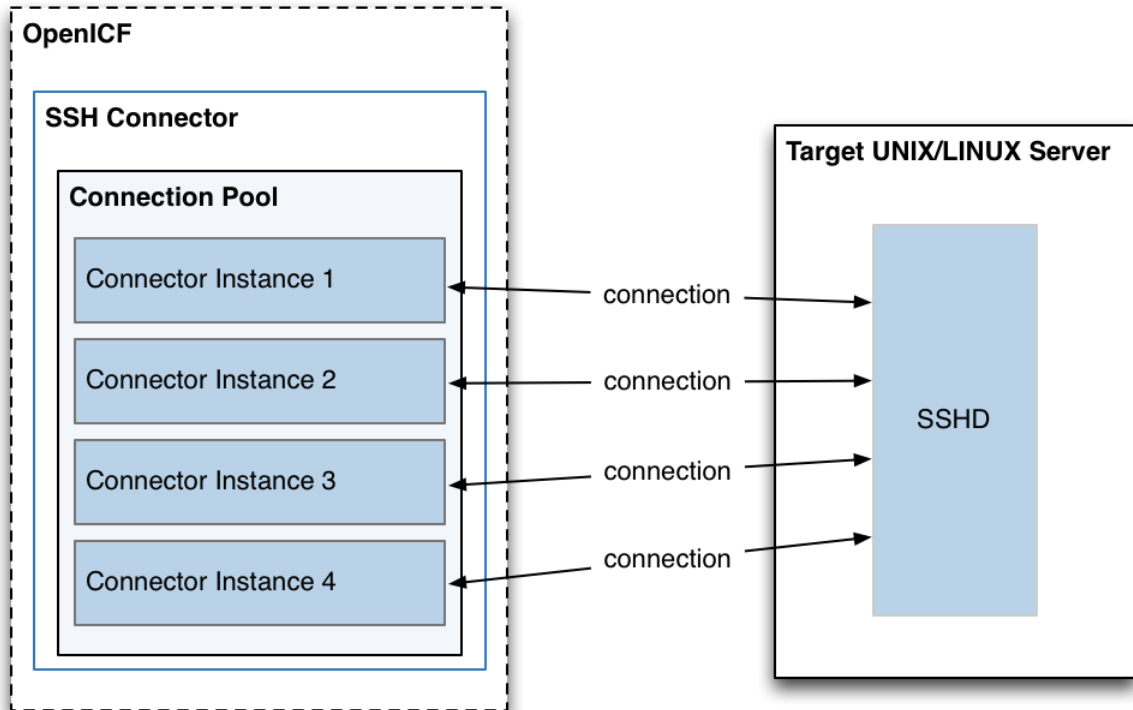
# SSH Connector

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 lets you 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:





## 10.1. 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-----",
    "MIIBugIBAABgQDcB0ztVMCFptJhqLLNZSdN/5cDL3S7a0Vy52Ae7vwwCqQPCQr",
    "6NyUk+wtkDr07NLYd3sg7a9hbsEnLYChsuX+/WUIvb0KdMfeqcQ+jKK26YdkTCGj",
    "g86dBj9JYhobSHDoQ9ov31pYN/cfW5BAZwkm9TdpEjHPvMIA0xx7GPGKwWIVALbD",
    "CEuflyJk9UB7v0dmJS7bKkbxAoGARcbAuDP4rB6MsgAAkVwf+1sHXEiGpShYWrVV",
    "qBgCZ/S45ELqUuiaN/1N/nip/Cc/0SBPKqwl7o50CUg9GH9kTAjmXiwmBkwvtUv+",
    "Xjn5vCHS0w18yc3rGwyr2wj+D9KtDLFJ8+T5HmsbPoDQ3mIZ9xPmRQURfVMD9wr",
    "DY0Rs7cCgYAxjGjWDSKThovsvOUCiE0ySz6tWggHH3LTrS4Mfh2t0tnbUfrXq2cw",
    "3CN+T6brgnpYbyX5XI17p859C+cw90MD8N6vvBxaN8QMDRFk+hHNUeSy8gXeem9x",
    "00vdIxCGkvA4dh5nSVb5VGKENEGNEHRLYxEPzblPa/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.

## 10.2. 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`.

#### `reloadScriptOnExecution`

By default, scripts are loaded and compiled when a connector instance is created and initialized. Setting `reloadScriptOnExecution` to `true` makes the connector load and compile the script every time it is called. Do not set this property to `true` in a production environment, because it will have a significant impact on performance.

Default: `false`

#### `*ScriptFileName`

The name of the Groovy script that is used for each ICF operation.

## 10.3. 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.

# 10.4. SSH Connector Configuration

The SSH Connector has the following configurable properties.

## 10.4.1. Configuration properties

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<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

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
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				
<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				

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<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
Description is not available				
<code>recompileGroovySource</code>	<code>boolean</code>	<code>false</code>		No
Description is not available				

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.

## 10.4.2. Basic Configuration Properties

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<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)				



Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<code>prompt</code>	String	<code>root@localhost:#</code>		Yes
A string representing the remote SSH session prompt (defaults to <code>root@localhost:#</code> )				
<code>sudoCommand</code>	String	<code>/usr/bin/sudo</code>		Yes
A string representing the sudo command (defaults to <code>/usr/bin/sudo</code> )				
<code>echoOff</code>	boolean	<code>true</code>		Yes
Disable the input command echo (default to true)				
<code>terminalType</code>	String	<code>vt102</code>		Yes
Defines the terminal type to use for the session (default to <code>vt102</code> )				
<code>locale</code>	String	<code>en_US.utf8</code>		Yes
Define the locale for <code>LC_ALL</code> , <code>LANG</code> and <code>LANGUAGE</code> environment variables to use if <code>setLocale=true</code>				
<code>setLocale</code>	boolean	<code>false</code>		Yes
Defines if the default environment locale should be changed with the value provided for <code>locale</code> (defaults to false)				
<code>connectionTimeout</code>	int	<code>5000</code>		Yes
Defines the connection timeout to the remote server in milliseconds (default to 5000)				
<code>expectTimeout</code>	long	<code>5000</code>		Yes
Defines the timeout used by the <code>expect()</code> calls in the scripts in milliseconds (default to 5000)				
<code>throwOperationTimeoutException</code>	boolean	<code>true</code>		Yes
Defines if an <code>OperationTimeoutException</code> should be thrown if any call to <code>expect</code> times out (defaults to true)				
<code>promptReadyTimeout</code>	long	<code>20</code>		No
Defines the "prompt ready" timeout for the <code>promptReady()</code> command in milliseconds (default to 20)				

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.

## Chapter 11

# Google Apps Connector

IDM bundles a Google Apps connector, along with a sample connector configuration. The Google Apps connector lets you 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](#).

## 11.1. 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": "[1.5.19.0,1.6.0.0]"
},
```

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 "*Synchronizing Accounts With the Google Apps Connector*" in the *Samples Guide*.

## 11.2. 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.

# 11.3. GoogleApps Connector Configuration

The GoogleApps Connector has the following configurable properties.

## 11.3.1. Basic Configuration Properties

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<code>domain</code>	String	null		Yes
Internet domain name. See <a href="https://support.google.com/a/answer/177483?hl=en">https://support.google.com/a/answer/177483?hl=en</a>				
<code>clientId</code>	String	null		Yes
Client identifier issued to the client during the registration process.				
<code>clientSecret</code>	GuardedString	null	Yes	Yes
Client secret issued to the client during the registration process.				
<code>refreshToken</code>	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.				
<code>proxyHost</code>	String	null		Yes

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
Defines an HTTP proxy host to use with the connection (example: "myproxy.home.com").				
<code>proxyPort</code>	<code>int</code>	<code>8080</code>		Yes
Defines an HTTP proxy port to use with the connection (defaults to 8080).				
<code>validateCertificate</code>	<code>boolean</code>	<code>true</code>		Yes
Validate the server certificate from the local truststore (defaults to true).				
<code>usersMaxResults</code>	<code>int</code>	<code>100</code>		No
Maximum number of Users to return. Acceptable values are 1 to 500, inclusive.				
<code>groupsMaxResults</code>	<code>int</code>	<code>200</code>		No
Maximum number of Groups to return. Acceptable values are 1 to 200, inclusive.				
<code>membersMaxResults</code>	<code>int</code>	<code>200</code>		No
Maximum number of Members to return. Acceptable values are greater than 1				
<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.				

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.

## 11.4. 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.

## 11.5. 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}

## 11.6. Functional Limitations

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

- 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.
- The Google Apps connector does not support custom schemas. The connector is therefore not able to read or update attributes associated with custom schemas in your Google Apps domain.
- 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.

## 11.7. 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}

## Chapter 12

# Kerberos Connector

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 lets you 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 lets you 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

## 12.1. 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`

## 12.2. 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.

### 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`

### throwOperationException

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`.

### reloadScriptOnExecution

By default, scripts are loaded and compiled when a connector instance is created and initialized. Setting `reloadScriptOnExecution` to true makes the connector load and compile the script every time it is called. Do not set this property to `true` in a production environment, because it will have a significant impact on performance.

Default: `false`

### \*ScriptFileName

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

## 12.3. 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.

# 12.4. Kerberos Connector Configuration

The Kerberos Connector has the following configurable properties.

## 12.4.1. Configuration properties

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<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				
<code>customConfiguration</code>	<code>String</code>	<code>null</code>		No
Description is not available				

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<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
Description is not available				
<code>host</code>	String	null		Yes

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
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
Description is not available				
expectTimeout	long	5000		Yes



Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
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				

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.

## Chapter 13

# Salesforce Connector

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 "*Synchronizing Users Between Salesforce and IDM*" in the *Samples Guide*.

## 13.1. 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.

## 13.2. 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:

```
$ cd /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 - 1.5.20.8 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.

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" \
"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.cs1.my.salesforce.com",
  "scope": "id api refresh_token",
  "issued_at": "1417182949781",
  "id": "https://login.salesforce.com/id/00DS0000003K4fUMAS/00530000009hWLCAM"
}
```

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" \
--request POST \
"http://localhost:8080/openidm/system/salesforce?_action=test"
{
  "name": "salesforce",
  "enabled": true,
  "config": "config/provisioner.openicf/salesforce",
  "connectorRef": {
    "bundleVersion": "[1.5.19.0,1.6.0.0)",
    "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.

## 13.3. 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.

## 13.4. 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.

# 13.5. Salesforce Connector Configuration

The Salesforce Connector has the following configurable properties.

### 13.5.1. Basic Configuration Properties

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<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 <sup>a</sup>	Required <sup>b</sup>
The proxy username to use with a proxy that requires authentication				
proxyPassword	GuardedString	null		No
The proxy user password to use with a proxy that requires authentication				

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.

## Chapter 14

# Marketo Connector

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 "*Configuring Social Identity Providers*" in the *Integrator's Guide*.

The Marketo connector is an implementation of the Scripted Groovy Connector Toolkit, and lets you 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" : "[1.5.19.0,1.6.0.0)",
    "connectorName" : "org.forgerock.openicf.connectors.marketo.MarkettoConnector"
  },
  ...
  "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 - 1.5.20.8 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" \
--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": "[1.5.19.0,1.6.0.0)"
    },
    "displayName": "Marketo Connector",
    "ok": true
  }
]
```

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

## 14.1. Reconciling Users With a Marketo Leads Database

The Marketo connector lets you 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 (`sync.json`), add the content of this sample `sync.json` to your existing file.

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 "Configuring Synchronization Filters With User Preferences" in the *Integrator's Guide*.

## 14.2. 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.

## 14.3. 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.

## 14.4. Marketo Connector Configuration

The Marketo Connector has the following configurable properties.

### 14.4.1. Configuration properties

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

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<code>customConfiguration</code>	String	null		No
Custom Configuration script for Groovy ConfigSlurper				
<code>instance</code>	String	null		Yes
The Marketo-assigned FQDN for your instance				
<code>clientId</code>	String	null		Yes
Your OAuth2 client ID				
<code>clientSecret</code>	GuardedString	null	Yes	Yes
Your OAuth2 client secret				
<code>leadFields</code>	String	null		No
Comma-delimited list of lead fields to fetch; Leave empty for default set				
<code>partitionName</code>	String	null		No
Name of the partition in which to create and update leads; May be left empty				
<code>listName</code>	String	null		Yes
Name of the Marketo static list the connector will use to manage leads				
<code>accessToken</code>	String	null		Yes
The access token for the application				
<code>tokenExpiration</code>	Long	null		Yes
The expiration token for the application				

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.

## 14.4.2. Operation Script Files

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<code>createScriptFileName</code>	String	CreateMarketo .groovy		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.				

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<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.				

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.

### 14.4.3. Groovy Engine configuration

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<code>targetDirectory</code>	File	null		No
Directory into which to write classes.				
<code>warningLevel</code>	int	1		No
Warning Level of the compiler				



Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<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				

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.

## Chapter 15

# Active Directory Connector

The Active Directory connector is a legacy connector, written in C# for the .NET platform. ICF connects to Active Directory over ADSI, the native connection protocol for Active Directory. The connector therefore requires a .NET connector server that has access to the ADSI .dll files.

### Important

The AD Connector is deprecated and support for its use with IDM will be discontinued in a future release. For simple Active Directory (and Active Directory LDS) deployments, the generic LDAP Connector works better than the Active Directory connector, in most circumstances. Using the generic LDAP connector avoids the need to install a remote connector server in the overall deployment. In addition, the generic LDAP connector has significant performance advantages over the Active Directory connector. For more complex Active Directory deployments, use the PowerShell Connector Toolkit, as described in "*PowerShell Connector Toolkit*".

## 15.1. Configuring the Active Directory Connector

Before you configure the Active Directory Connector, make sure that the .NET Connector Server is installed, configured and started, and that IDM has been configured to use the Connector Server. For more information, see "Installing and Configuring a .NET Connector Server" in the *Integrator's Guide*.

### *Setting Up the Active Directory Connector*

1. Download the Active Directory (AD) Connector from the ForgeRock BackStage download site.
2. Extract the contents of the AD Connector zip file into the directory in which you installed the Connector Server (by default `c:\Program Files (x86)\Identity Connectors\Connector Server>`).

Note that the files, specifically the connector itself (`ActiveDirectory.Connector.dll`) must be directly under the `path\to\Identity Connectors\Connector Server` directory, and *not* in a subdirectory.

**Note**

If the account that is used to install the Active Directory connector is different from the account under which the Connector Server runs, you must give the Connector Server runtime account the rights to access the Active Directory connector log files.

3. A sample Active Directory Connector configuration file is provided in `openidm/samples/example-configurations/provisioners/provisioner.openicf-ad.json`. On the IDM host, copy the sample Active Directory connector configuration file to your project's `conf/` directory:

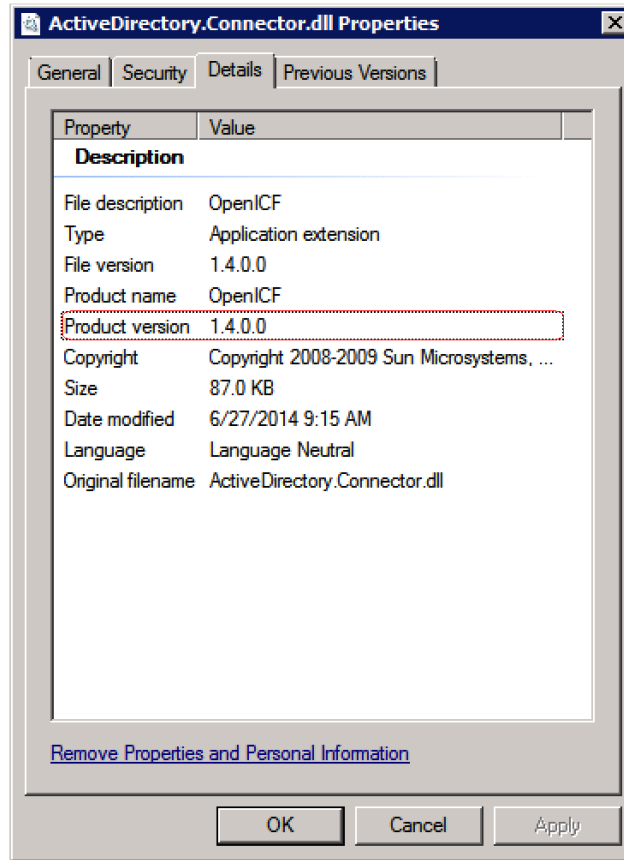
```
$ cd /path/to/openidm
$ cp samples/example-configurations/provisioners/provisioner.openicf-ad.json project-dir/conf/
```

4. Edit the Active Directory connector configuration to match your Active Directory deployment.

Specifically, check and edit the `configurationProperties` that define the connection details to the Active Directory server.

Also, check that the `bundleVersion` of the connector matches the version of the `ActiveDirectory.Connector.dll` in the Connector Server directory. The bundle version can be a range that includes the version of the connector bundle. To check the `.dll` version:

- Right click on the `ActiveDirectory.Connector.dll` file and select Properties.
- Select the Details tab and note the Product Version.



The following configuration extract shows sample values for the `connectorRef` and `configurationProperties`:

```
...
"connectorRef" :
  {
    "connectorHostRef" : "dotnet",
    "connectorName" : "Org.IdentityConnectors.ActiveDirectory.ActiveDirectoryConnector",
    "bundleName" : "ActiveDirectory.Connector",
    "bundleVersion" : "[1.4.0.0,1.5.0.0)"
  },
  ...
"configurationProperties" :
  {
    "DirectoryAdminName" : "EXAMPLE\\Administrator",
    "DirectoryAdminPassword" : "Passw0rd",
    "ObjectClass" : "User",
    "Container" : "dc=example,dc=com",
    "CreateHomeDirectory" : true,
    "LDAPHostName" : "192.0.2.0",
    "SearchChildDomains" : false,
    "DomainName" : "example",
    "SyncGlobalCatalogServer" : null,
    "SyncDomainController" : null,
    "SearchContext" : ""
  },
},
```

The main configurable properties are as follows:

#### **connectorHostRef**

Must point to an existing connector info provider configuration in `project-dir/conf/provisioner.openicf.connectorinfoprovider.json`. The `connectorHostRef` property is required because the Active Directory connector must be installed on a .NET connector server, which is always *remote*, relative to IDM.

#### **DirectoryAdminName and DirectoryAdminPassword**

Specify the credentials of an administrator account in Active Directory, that the connector will use to bind to the server.

The `DirectoryAdminName` can be specified as a bind DN, or in the format `DomainName\samaccountname`.

#### **SearchChildDomains**

Specifies if a Global Catalog (GC) should be used. This parameter is used in search and query operations. A Global Catalog is a read-only, partial copy of the entire forest, and is never used for create, update or delete operations.

Boolean, false by default.

#### **LDAPHostName**

Specifies a particular Domain Controller (DC) or Global Catalog (GC), using its hostname. This parameter is used for query, create, update, and delete operations.

If `SearchChildDomains` is set to `true`, this specific GC will be used for search and query operations. If the `LDAPHostName` is null (as it is by default), the connector will allow the ADSI libraries to pick up a valid DC or GC each time it needs to perform a query, create, update, or delete operation.

### SyncGlobalCatalogServer

Specifies a Global Catalog server name for sync operations. This property is used in combination with the `SearchChildDomains` property.

If a value for `SyncGlobalCatalogServer` is set (that is, the value is not `null`) and `SearchChildDomains` is set to `true`, this GC server is used for sync operations. If no value for `SyncGlobalCatalogServer` is set and `SearchChildDomains` is set to `true`, the connector allows the ADSI libraries to pick up a valid GC.

### SyncDomainController

Specifies a particular DC server for sync operations. If no DC is specified, the connector picks up the first available DC and retains this DC in future sync operations.

The updated configuration is applied immediately.

5. Check that the connector has been configured correctly by running the following command:

```
$ curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--request POST \
"http://localhost:8080/openidm/system?_action=test"
```

The command must return `"ok" : true` for the Active Directory connector.

6. The connector is now configured. To verify the configuration, perform a RESTful GET request on the remote system URL, for example:

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

This request should return the user accounts in the Active Directory server.

7. (Optional) To configure reconciliation or liveSync between IDM and Active Directory, create a synchronization configuration file (`sync.json`) in your project's `conf/` directory.

The synchronization configuration file defines the attribute mappings and policies that are used during reconciliation.

The following is a simple example of a `sync.json` file for Active Directory:

```
{
  "mappings" : [
    {
      "name" : "systemADAccounts_managedUser",
      "source" : "system/ActiveDirectory/account",
      "target" : "managed/user",
      "properties" : [
        { "source" : "cn", "target" : "displayName" },
        { "source" : "description", "target" : "description" },
        { "source" : "givenName", "target" : "givenName" },
        { "source" : "mail", "target" : "email" },
        { "source" : "sn", "target" : "familyName" },
        { "source" : "sAMAccountName", "target" : "userName" }
      ],
      "policies" : [
        { "situation" : "CONFIRMED", "action" : "UPDATE" },
        { "situation" : "FOUND", "action" : "UPDATE" },
        { "situation" : "ABSENT", "action" : "CREATE" },
        { "situation" : "AMBIGUOUS", "action" : "EXCEPTION" },
        { "situation" : "MISSING", "action" : "UNLINK" },
        { "situation" : "SOURCE_MISSING", "action" : "DELETE" },
        { "situation" : "UNQUALIFIED", "action" : "DELETE" },
        { "situation" : "UNASSIGNED", "action" : "DELETE" }
      ]
    }
  ]
}
```

8. To test the synchronization, run a reconciliation operation as follows:

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

If reconciliation is successful, the command returns a reconciliation run ID, similar to the following:

```
{"_id":"0629d920-e29f-4650-889f-4423632481ad","state":"ACTIVE"}
```

9. Query the internal repository, using either a **curl** command, or the IDM Admin UI, to make sure that the users in your Active Directory server were provisioned into the repository.

## 15.2. Using PowerShell Scripts With the Active Directory Connector

The Active Directory connector supports PowerShell scripting. The following example shows a simple PowerShell script that is referenced in the connector configuration and can be called over the REST interface.

## Note

External script execution is disabled on system endpoints by default. For testing purposes, you can enable script execution over REST, on system endpoints by adding the `script` action to the system object, in the `access.js` file. For example:

```
$ more /path/to/openidm/script/access.js
...
{
  "pattern" : "system/ActiveDirectory",
  "roles" : "internal/role/openidm-admin",
  "methods" : "action",
  "actions" : "script"
},
```

Be aware that scripts passed to clients imply a security risk in production environments. If you need to expose a script for direct external invocation, it might be better to write a custom authorization function to constrain the script ID that is permitted. Alternatively, do not expose the script action for external invocation, and instead, expose a custom endpoint that can make only the desired script calls. For more information about using custom endpoints, see "Creating Custom Endpoints to Launch Scripts" in the *Integrator's Guide*.

The following PowerShell script creates a new Microsoft SQL user with a username that is specified when the script is called. The script sets the user's password to `Passw0rd` and, optionally, gives the user a role. Save this script as `project-dir/script/createUser.ps1`:

```
if ($loginName -ne $NULL) {
  [System.Reflection.Assembly]::LoadWithPartialName('Microsoft.SqlServer.SMO') | Out-Null
  $sqlSrv = New-Object ('Microsoft.SqlServer.Management.Smo.Server') ('WIN-C2MSQ8G1TCA')

  $login = New-Object -TypeName ('Microsoft.SqlServer.Management.Smo.Login') ($sqlSrv, $loginName)
  $login.LoginType = 'SqlLogin'
  $login.PasswordExpirationEnabled = $false
  $login.Create('Passw0rd')
  # The next two lines are optional, and to give the new login a server role, optional
  $login.AddToRole('sysadmin')
  $login.Alter()
} else {
  $Error_Message = [string]"Required variables 'loginName' is missing!"
  Write-Error $Error_Message
  throw $Error_Message
}
```

Now edit the Active Directory connector configuration to reference the script. Add the following section to the connector configuration file (`project-dir/conf/provisioner.openicf-ad.json`):



```
"systemActions" : [
  {
    "scriptId" : "ConnectorScriptName",
    "actions" : [
      {
        "systemType" : ".*ActiveDirectoryConnector",
        "actionType" : "Shell",
        "actionSource" : "@echo off \r\n echo %loginName%\r\n"
      },
      {
        "systemType" : ".*ActiveDirectoryConnector",
        "actionType" : "PowerShell",
        "actionFile" : "script/createUser.ps1"
      }
    ]
  }
]
```

To call the PowerShell script over the REST interface, use the following request, specifying the `userName` as input:

```
$ curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--request POST \
"http://localhost:8080/openidm/system/ActiveDirectory/?
_action=script&scriptId=ConnectorScriptName&scriptExecuteMode=resource&loginName=myUser"
```

## Chapter 16

# Office 365 Connector

The Office 365 connector uses the O365 Graph API to manage Azure AD users and groups. This connector uses the OData 3.0 specification and can be used, with minor modifications, to connect to any OData 3 provider. Note that OData 2, 3 and 4 are not interchangeable and this connector can only function against OData 3 providers.

### Important

The Office 365 connector is deprecated and support for its use with IDM will be discontinued in a future release.

This chapter lists the implemented interfaces and configurable properties for the Office 365 connector.

## 16.1. Implementation Specifics

For PATCH requests, a connector can potentially add, remove, or replace an attribute value. The Office 365 connector does not implement the add or remove operations, so a PATCH request always replaces the entire attribute value with the new value.

## 16.2. OpenICF Interfaces Implemented by the O365 Connector

The O365 Connector implements the following OpenICF interfaces.

### 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.

## 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.

# 16.3. O365 Connector Configuration

The O365 Connector has the following configurable properties.

## 16.3.1. Office365 OAuth2 Configuration Properties Properties

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<code>tenant</code>	<code>String</code>	<code>null</code>		Yes

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
Name of your Office365 tenant				
<code>clientId</code>	String	null		Yes
This value is provided by Office365				
<code>clientSecret</code>	GuardedString	null	Yes	Yes
This value is provided by Office365				
<code>accessToken</code>	String	null		Yes
This value is provided by Office365				
<code>tokenExpiration</code>	Long	null		No
This value is provided by Office365				
<code>refreshToken</code>	String	null		Yes
This value is provided by Office365				

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.

### 16.3.2. Office365 AzureAD Configuration Properties Properties

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<code>accountEntitySet</code>	String	User		Yes
The name AzureAD uses to declare account objects in its data payloads				
<code>accountURIComponent</code>	String	users		Yes
The name used in a URI path to specify an account target object				
<code>groupEntitySet</code>	String	Group		Yes
The name AzureAD uses to declare group objects in its data payloads				
<code>groupURIComponent</code>	String	groups		Yes
The name used in a URI path to specify an account target object				

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.

## Chapter 17

# SCIM Connector

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 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.

### *Configure the SCIM Connector Using the Filesystem*

1. Copy the sample connector configuration file (`/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" : "xxxxxxxxxxxxxxxxxxxx",  
  "acceptSelfSignedCertificates" : true,  
  "disableHostNameVerifier" : true,  
  "maximumConnections" : 10,  
  "httpProxyHost" : null,  
  "httpProxyPort" : null  
},
```

#### Note

On startup, IDM encrypts the value of the `clientSecret`.

### Configure the SCIM Connector Using the Admin UI

1. From the navigation bar, click Configure > Connectors.
2. On the Connectors page, click New Connector.

The New Connector page displays.

3. In the General Details area, from the Connector Type drop-down list, select Scim Connector - *Version#*.
4. Enter other details, as necessary, and click Save.

After the connector is properly configured, you can test its status:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--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": "[1.5.19.0,1.6.0.0)"
    },
    "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.

## 17.1. 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.

## 17.2. 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,
  ...
},
```

## 17.3. 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.

# 17.4. Scim Connector Configuration

The Scim Connector has the following configurable properties.

## 17.4.1. Basic Configuration Properties

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<b>SCIMEndpoint</b>	String	null		Yes
The HTTP URL defining the root for the SCIM endpoint (https://myserver.com/service/scim)				
<b>SCIMVersion</b>	Integer	1		Yes
Defines the SCIM protocol version. Values can be either 1 or 2. Default is 1				
<b>authenticationMethod</b>	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				
<b>user</b>	String	null		Yes
In case of BASIC authentication type, this property defines the remote user.				
<b>password</b>	GuardedString	null	Yes	No
In case of BASIC authentication type, this property defines the remote password.				
<b>tokenEndpoint</b>	String	null		No
When using OAuth, this property defines the endpoint where a new access token should be requested (https://myserver.com/oauth2/token)				
<b>clientId</b>	String	null		Yes
Secure client identifier for OAuth2				
<b>clientSecret</b>	GuardedString	null	Yes	No
Secure client secret for OAuth2				
<b>authToken</b>	GuardedString	null	Yes	No
Some service providers (Slack for instance) use static authentication tokens.				
<b>refreshToken</b>	GuardedString	null		Yes
Used by the refresh_token grant type				

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<code>grantType</code>	String	null		No
The OAuth2 grant type to use (client_credentials or refresh_token)				
<code>scope</code>	String	null		No
The OAuth2 scope to use.				
<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.				

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
authorizationTokenPrefix	String	Bearer		No
The prefix to be used in the Authorization HTTP header for Token authentication. Defaults to "Bearer".				

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.

## Chapter 18

# Adobe Marketing Cloud Connector

The Adobe Marketing Cloud connector lets you 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.

## 18.1. 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:

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

- b. Export the certificate:

```
$ openssl pkcs12 \
  -in keystore.p12 \
  -nokeys \
  -out cert.pem
```

- c. Export unencrypted private key:

```
$ openssl pkcs12 \  
-in keystore.p12 \  
-nodes \  
-nocerts \  
-out 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.

## 18.2. 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 - 1.5.20.8 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.adobe.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-na1.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" \
--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": "[1.5.19.0,1.6.0.0]"
    },
    "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.

## 18.3. 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.

# 18.4. Adobe Marketing Cloud Connector Configuration

The Adobe Marketing Cloud Connector has the following configurable properties.

## 18.4.1. Basic configuration properties

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

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.



<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.

## 18.4.2. Adobe Integration Properties

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

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.

## Chapter 19

# Workday Connector

Workday is a multi-tenant Software-as-a-Service (SaaS) application. The Workday connector lets you 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.

This connector is bundled with IDM in the `connectors/` directory (`workday-connector-1.5.20.8.jar`).

## 19.1. Before You Start

The Workday connector depends on a number of libraries. Some of these libraries are bundled with IDM and some need to be downloaded separately. The downloaded libraries must be placed in the `/path/to/openidm/lib` directory if you are running the connector locally. All dependencies must be copied to the `/path/to/openicf/lib/` directory on the connector server if you are running the connector remotely.

For a list of dependencies, and an indication of whether they are bundled with IDM, see "Installing Connector Dependencies" in the *Integrator's Guide*.

## 19.2. 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 into the Workday instance. Do not specify the complete username including the **tenant**. The connector constructs the complete username.

**password**

The **password** used to log into 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".

## 19.3. 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" \
  --request POST \
  "http://localhost:8080/openidm/system?_action=test"
[
  {
    "name": "workday",
    "enabled": true,
    "config": "config/provisioner.openicf/workday",
    "connectorRef": {
      "bundleVersion": "[1.5.19.0,1.6.0.0)",
      "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" \
--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" \
--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
}
```

## 19.4. 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 Source Objects to Target Objects" in the *Integrator's 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" \
--request POST \
"http://localhost:8080/openidm/recon?
action=recon&mapping=systemWorkdayEmployee_managedUser&waitForCompletion=true"
{
  "_id": "db2bc7f4-e9a8-4315-9dd1-e2cdcd85ae6e-33099",
  "state": "SUCCESS"
}
```

## 19.5. 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 "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
}

```

## 19.6. 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.



## 19.7. 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.

## 19.8. Workday Connector Configuration

The Workday Connector has the following configurable properties.

### 19.8.1. Configuration properties

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<code>includeManagementChainDataForWorker</code>	Boolean	true		No
Description is not available				
<code>includeOrganizationsForWorkers</code>	Boolean	true		No
Description is not available				
<code>includePersonalInformationForWorker</code>	Boolean	true		No
Description is not available				
<code>excludeCostCentersForWorkers</code>	Boolean	false		No
Description is not available				
<code>excludeCustomOrganizationsForWorker</code>	Boolean	true		No
Description is not available				
<code>includeRolesForWorkers</code>	Boolean	false		No
Description is not available				
<code>includeStaffingRestrictionsDataForC</code>	Boolean	false		No
Description is not available				
<code>excludeMatrixOrganizationsForWorker</code>	Boolean	true		No
Description is not available				
<code>includeEmploymentInformationForWork</code>	Boolean	true		No
Description is not available				
<code>includeAccountProvisioningForWorker</code>	Boolean	false		No
Description is not available				
<code>excludeBusinessUnitHierarchiesForWc</code>	Boolean	true		No
Description is not available				
<code>includeRelatedPersonsForWorkers</code>	Boolean	false		No
Description is not available				

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<code>includePhotoForWorkers</code>	Boolean	false		No
Description is not available				
<code>excludeSupervisoryOrganizationsForWorkers</code>	Boolean	true		No
Description is not available				
<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				

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.

## 19.8.2. Basic Configuration Properties

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<code>hostname</code>	String	null		Yes
The hostname for the Workday service. Example: <code>https://[workday hostname]/ccx/service/[workday tenant]/</code> . You need to configure the bracketed Workday hostname and tenant name to successfully connect to the proper instance.				
<code>tenant</code>	String	null		Yes
The tenant in URL for the Workday service. For example: <code>https://[workday hostname]/ccx/service/[workday tenant]/</code> . You need to configure the bracketed Workday hostname and tenant name to successfully connect to the proper instance.				
<code>username</code>	String	null		Yes
The user name for logging into the Workday service. It will be concatenated with the tenant name (user@tenant)				
<code>password</code>	GuardedString	null	Yes	Yes
The user password for logging into the Workday service				
<code>excludeInactiveWorkers</code>	boolean	false		No
Excludes from the response terminated employees or contingent workers whose contracts have ended (defaults to false)				
<code>excludeContingentWorkers</code>	boolean	false		No
Excludes contingent workers from inclusion in a query response.				
<code>excludeEmployees</code>	boolean	false		No
Excludes employees from inclusion in a query response.				
<code>connectionTimeout</code>	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.				
<code>receiveTimeout</code>	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.				
<code>pageSize</code>	long	100		No
Set the page size used for search operations (defaults to 100).				
<code>proxyHost</code>	String	null		No
If defined the connection to Workday will go through this HTTP proxy server				

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<b>proxyPort</b>	int	8080		No
The HTTP proxy server port number (defaults to 8080).				
<b>xslTransformer</b>	File	null		No
The file path to the XSL File to get the custom attributes				
<b>asOfEffectiveDate</b>	String	null		No
Optional configuration of Response_Filter/As_Of_Effective_Date element. Valid values are: Date ( <a href="http://www.w3.org/TR/xpath-functions/#date-time-values">http://www.w3.org/TR/xpath-functions/#date-time-values</a> <a href="http://www.w3.org/TR/xmlschema-2/#dateTime-order">http://www.w3.org/TR/xmlschema-2/#dateTime-order</a> ) or Duration ( <a href="http://www.w3.org/TR/xpath-functions/#dt-dayTimeDuration">http://www.w3.org/TR/xpath-functions/#dt-dayTimeDuration</a> ). If set to Duration, the effective date is calculated as current date + duration.				
<b>effectiveFrom</b>	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 ( <a href="http://www.w3.org/TR/xpath-functions/#date-time-values">http://www.w3.org/TR/xpath-functions/#date-time-values</a> <a href="http://www.w3.org/TR/xmlschema-2/#dateTime-order">http://www.w3.org/TR/xmlschema-2/#dateTime-order</a> ) or string Today representing the current time of the request.				
<b>effectiveThrough</b>	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 ( <a href="http://www.w3.org/TR/xpath-functions/#date-time-values">http://www.w3.org/TR/xpath-functions/#date-time-values</a> <a href="http://www.w3.org/TR/xmlschema-2/#dateTime-order">http://www.w3.org/TR/xmlschema-2/#dateTime-order</a> ) or Duration ( <a href="http://www.w3.org/TR/xpath-functions/#dt-dayTimeDuration">http://www.w3.org/TR/xpath-functions/#dt-dayTimeDuration</a> )				

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.

## Chapter 20

# ServiceNow Connector

This connector lets you manage objects in the ServiceNow platform, integrating with ServiceNow's REST API. The connector is bundled with IDM in the `connectors/` directory (`servicenow-connector-1.5.20.8.jar`).

## 20.1. 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`

## 20.2. 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 - 1.5.20.8 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" \
--request POST \
"http://localhost:8080/openidm/system?_action=test"
[
{
```

```
"name": "serviceNow",
"enabled": true,
"config": "config/provisioner.openicf/serviceNow",
"connectorRef": {
  "bundleVersion": "[1.5.19.0,1.6.0.0)",
  "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.

## 20.3. 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" \
--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" \
--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": "02826bf03710200044e0bfc8bcbe5d3f",
"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 "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 "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 "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, you must specify a mapping between managed users and ServiceNow users. For more information, see "Configuring Synchronization Between Two Resources" in the *Integrator's Guide*.

The following example assumes that a mapping has been created and runs a reconciliation operation from ServiceNow to the managed user repository:

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

## 20.4. 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.

## 20.5. 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.

## 20.6. ServiceNow Connector Configuration

The ServiceNow Connector has the following configurable properties.

### 20.6.1. Basic configuration properties

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<code>instance</code>	<code>String</code>	<code>null</code>		Yes
URL of the ServiceNow instance, for example: dev00000.service-now.com				
<code>username</code>	<code>String</code>	<code>null</code>		Yes

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
An API user in ServiceNow that can consume the REST API				
<code>password</code>	<code>GuardedString</code>	<code>null</code>	Yes	Yes
Password for the user				
<code>clientID</code>	<code>String</code>	<code>null</code>		Yes
Client ID of the OAuth application in ServiceNow				
<code>clientSecret</code>	<code>GuardedString</code>	<code>null</code>	Yes	Yes
Client Secret for the preceding Client ID				
<code>pageSize</code>	<code>int</code>	<code>100</code>		No
Default page size				

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.

## Chapter 21

# MongoDB Connector

The MongoDB connector is an implementation of the Scripted Groovy Connector Toolkit. This connector lets you 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.

## 21.1. 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.

## 21.2. 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" \
--request POST \
"http://localhost:8080/openidm/system?_action=test"
[
  {
    "name": "mongodb",
    "enabled": true,
    "config": "config/provisioner.openicf/mongodb",
    "connectorRef": {
      "bundleVersion": "[1.5.19.0,1.6.0.0)",
      "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.

## 21.3. 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.

# 21.4. MongoDB Connector Configuration

The MongoDB Connector has the following configurable properties.

## 21.4.1. Configuration properties

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<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				

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.

## 21.4.2. Operation Script Files

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<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 <sup>a</sup>	Required <sup>b</sup>
<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.				

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.

### 21.4.3. Groovy Engine configuration

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<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 <sup>a</sup>	Required <sup>b</sup>
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				

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.

#### 21.4.4. Basic Configuration Properties

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<code>connectionURI</code>	<code>String</code>	<code>null</code>		No
The MongoDB client connection URI, for example "mongodb://localhost:27017". Overrides other connection parameters				
<code>host</code>	<code>String</code>	<code>localhost</code>		No
The MongoDB server host name (localhost by default).				
<code>port</code>	<code>int</code>	<code>27017</code>		No
The MongoDB server port number (27017 by default).				
<code>user</code>	<code>String</code>	<code>null</code>		No
The MongoDB username				

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<code>password</code>	<code>GuardedString</code>	<code>null</code>	Yes	No
The password used to connect to MongoDB				
<code>userDatabase</code>	<code>String</code>	<code>null</code>		No
The name of the database in which the MongoDB user is defined				
<code>clusterAddresses</code>	<code>String[]</code>	<code>null</code>		No
A list of additional mongodDB servers when connecting to a MongoDB cluster (["host1:27017","host2:27017",...])				
<code>dateAttributes</code>	<code>String[]</code>	<code>[]</code>		No
Defines the list of attributes to convert to MongoDB BSON Date type on create/update.				
<code>database</code>	<code>String</code>	<code>null</code>		No
The database to use				
<code>arrayAttributes</code>	<code>String[]</code>	<code>[]</code>		No
Defines the list of attributes that should be considered as BSON Arrays.				
<code>includeNullValue</code>	<code>boolean</code>	<code>false</code>		No
If set to true, retains null values in the target MongoDB document (false by default).				
<code>includeEmptyList</code>	<code>boolean</code>	<code>false</code>		No
If set to true, retains null values in the target MongoDB document (false by default).				
<code>dateFormat</code>	<code>String</code>	<code>yyyy-MM-dd'T'HH:mm:ss'Z'</code>		No
Defines the date format to use for MongoDB Date attributes (defaults to ISO 8601 "yyyy-MM-ddTHH:mm:ssZ").				
<code>timeZone</code>	<code>String</code>	<code>UTC</code>		No
Defines the timezone to use for MongoDB Date attributes.				
<code>ICFName</code>	<code>String</code>	<code>name</code>		No
Defines the name to use in the target MongoDB document for the ICF <code>__NAME__</code> attribute.				

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.

### 21.4.5. Connection Configuration Properties

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<code>sslEnabled</code>	<code>boolean</code>	<code>true</code>		No

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
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 $\geq 0$ )				
<code>maxConnectionsPerHost</code>	<code>int</code>	<code>5</code>		No
The maximum number of connections per host (must be $> 0$ )				

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.

## Chapter 22

# HubSpot Connector

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 *"Synchronizing 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`.

## 22.1. Installing and Configuring the HubSpot Connector

To install the HubSpot connector, download the connector jar from the ForgeRock BackStage download site and place it in the `/path/to/openidm/connectors` directory:

```
$ mv ~/Downloads/hubspot-connector-1.5.2.0.jar /path/to/openidm/connectors/
```

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

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" \
--request POST \
"http://localhost:8080/openidm/system?_action=test"
[
  {
    "name": "hubspot",
    "enabled": true,
    "config": "config/provisioner.openicf/hubspot",
    "connectorRef": {
      "bundleVersion": "[1.5.19.0,1.6.0.0)",
      "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.

## 22.2. 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.

## 22.3. 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,
  ...
},
```

## 22.4. 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.

## 22.5. Hubspot Connector Configuration

The Hubspot Connector has the following configurable properties.

### 22.5.1. Basic Configuration Properties Properties

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

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.

### 22.5.2. Advanced Connection Properties Properties

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

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
----------	------	---------	------------------------	-----------------------

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.

## Chapter 23

# AWS Connector

AWS Identity and Access Management (IAM) is a web service for securely controlling access to AWS services. The AWS connector lets you manage and synchronize accounts between AWS and IDM managed user objects. An AWS administrator account is required for this connector to work.

## 23.1. Before you start

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

### Access Key ID

The access key ID is a globally unique IAM user identifier to access the AWS service API.

### Secret Key ID

The secret key is a password to access the AWS service API.

### Role ARN

Amazon Resource Name (ARN) for the role which has IAM Full Access permissions.

### Credentials Expiration

Time (in seconds) to configure the duration in which the temporary credentials would expire.  
Optional.

### Region

The region where the AWS instance is hosted.

## 23.2. Install the AWS 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/aws-connector-1.5.20.8.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.

## 23.3. Configure the AWS connector

Create a connector configuration using the Admin UI:

1. Select Configure > Connectors and click New Connector.
2. Enter a Connector Name.
3. Select AWS Connector - 1.5.20.8 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/aws?action=test"
{
  "name": "aws",
  "enabled": true,
  "config": "config/provisioner.openicf/aws",
  "connectorRef": {
    "bundleVersion": "[1.5.19.0,1.6.0.0)",
    "bundleName": "org.forgerock.openicf.connectors.aws-connector",
    "connectorName": "org.forgerock.openicf.connectors.aws.AwsConnector"
  },
  "displayName": "AWS Connector",
  "objectTypes": [
    "__ACCOUNT__",
    "__ALL__"
  ],
  "ok": true
}
```

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

## 23.4. Use the AWS connector

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

Attribute	Description
<code>__USER__</code>	The username of the user. Only alphanumeric characters, and <code>+ =, .@_ -</code> symbols are supported. Required.
UserID	Auto-generated user id.
Path	The path to the created user (used to define a hierarchy-based structure). Default value is <code>/</code> .
<code>__PASSWORD__</code>	Password for the user account.
Arn	Amazon Resource Name (ARN), used to uniquely identify the AWS resource. For more information on ARNs, see Amazon Resource Names (ARNs) in the AWS documentation.
CreatedDate	Date of profile creation, in ISO 8601 date-time format.
PasswordLastUsed	Date the password was last used.
PermissionBoundary	The ARN of the policy that is used to set the permissions boundary for the user.
Tags	A list of customizable key-value pairs. For more information about tags on AWS, see <a href="#">Tagging AWS resources</a> in the AWS documentation. For example: <pre> "Tags": [{   "Key": "Department",   "Value": "Accounting" }] </pre>

You can use the AWS connector to perform the following actions on an AWS account:

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"
}' \
"http://localhost:8080/openidm/system/aws/__ACCOUNT__?_action=create"
{
  "_id": "bjensen",
  "Path": "/",
  "UserId": "AIDAW3FY74V57KNBRIDU6",
  "__NAME__": "bjensen",
  "Arn": "arn:aws:iam::470686885243:user/bjensen",
  "CreateDate": "Thu Jun 02 16:46:39 PDT 2022"
}
```

### Note

When you create a new user, you must specify at least `__NAME__`. See the list of available attributes for more information.

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:

- `__USER__`
- `__PASSWORD__`
- `Path`
- `PermissionBoundary`
- `Tags`

For example, to add a new tag 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",
  "Tags": [{
    "Key": "Project",
    "Value": "Meteor"
  }]
}' \
"http://localhost:8080/openidm/system/aws/__ACCOUNT__/bjensen"
{
  "_id": "bjensen",
  "Path": "/",
  "UserId": "AIDAW3FY74V57KNBRIDU6",
  "__NAME__": "bjensen",
  "Arn": "arn:aws:iam::470686885243:user/bjensen",
  "CreateDate": "Thu Jun 02 16:46:39 PDT 2022",
  "Tags": [
    {
      "Project": "Meteor"
    }
  ]
}
```

The following example queries all AWS 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/aws/__ACCOUNT__?_queryId=query-all-ids"
{
  "result": [
    {
      "_id": "bjensen"
    },
    {
      "_id": "frank@example.com"
    },
    {
      "_id": "testFR4User"
    },
    {
      "_id": "testFR5User"
    },
    {
      "_id": "testFR6User"
    }
  ],
  "resultCount": 5,
  "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/aws/__ACCOUNT__/bjensen"
{
  "_id": "bjensen",
  "Path": "/",
  "UserId": "AIDAW3FY74V57KNBRIDU6",
  "__NAME__": "bjensen",
  "Arn": "arn:aws:iam::470686885243:user/bjensen",
  "CreateDate": "Thu Jun 02 16:46:39 PDT 2022",
  "Tags": [
    {
      "Project": "Meteor"
    }
  ]
}
```

To reset the password for AWS 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 PATCH
\
--data ' [{
  "operation": "add",
  "field": "__PASSWORD__",
  "value": "Passw0rd@123!"
}]' \
"http://localhost:8080/openidm/system/aws/__ACCOUNT__/bjensen"
{
  "_id": "bjensen",
  "Path": "/",
  "UserId": "AIDAW3FY74V57KNBRIDU6",
  "__NAME__": "bjensen",
  "Arn": "arn:aws:iam::470686885243:user/bjensen",
  "CreateDate": "Thu Jun 02 16:46:39 PDT 2022",
  "Tags": [
    {
      "Project": "Meteor"
    }
  ]
}
```

#### Note

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

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

The following example deletes an AWS 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/aws/__ACCOUNT__/bjensen"
{
  "id": "bjensen",
  "Path": "/",
  "UserId": "AIDAW3FY74V57KNBRIDU6",
  "__NAME__": "bjensen",
  "Arn": "arn:aws:iam::470686885243:user/bjensen",
  "CreateDate": "Thu Jun 02 16:46:39 PDT 2022",
  "Tags": [
    {
      "Project": "Meteor"
    }
  ]
}
```

## 23.5. OpenICF Interfaces Implemented by the AWS Connector

The AWS 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.

# 23.6. AWS Connector Configuration

The AWS Connector has the following configurable properties.

## 23.6.1. Basic group of connector

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<code>accessKeyId</code>	String	null		Yes
Provide the Access Key ID to access the AWS IAM Service API				
<code>secretKey</code>	GuardedString	null	Yes	Yes
Provide the Secret Key ID to access the AWS IAM Service API				
<code>roleArn</code>	String	null		Yes
Provide the Amazon Resource Name specifying the Role				
<code>region</code>	String	null		No
Provide the Regions				
<code>pageSize</code>	int	100		No
Provide the Page Size				

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<code>credentialsExpiration</code>	<code>int</code>	<code>3600</code>		No
Provide the temporary credentials expiration time in seconds				
<code>proxyHost</code>	<code>String</code>	<code>null</code>		No
Provide the ProxyHost				
<code>proxyPort</code>	<code>Integer</code>	<code>null</code>		No
Provide the ProxyPort				
<code>proxyUsername</code>	<code>String</code>	<code>null</code>		No
Provide the Proxy Username				
<code>proxyPassword</code>	<code>GuardedString</code>	<code>null</code>		No
Provide the Proxy Password				
<code>connectionTimeout</code>	<code>Integer</code>	<code>10000</code>		No
Provide the Maximum Connection Timeout in milliseconds				
<code>maxConnections</code>	<code>Integer</code>	<code>10</code>		No
Provide the number of Maximum Connections				

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.

## Chapter 24

# Cerner Connector

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.

## 24.1. Before you start

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

- The bearer token associated with your system account.
- Your Cerner tenant ID.

## 24.2. 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-1.5.20.8.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.

## 24.3. 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 - 1.5.20.8 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": "[1.5.19.0,1.6.0.0)",
    "bundleName": "org.forgerock.openicf.connectors.cerner-connector",
    "connectorName": "org.forgerock.openicf.connectors.cerner.CernerConnector"
  },
  "displayName": "Cerner Connector",
  "objectTypes": [
    "_ACCOUNT_",
    "_ALL_"
  ],
  "ok": true
}
```

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

## 24.4. Use the Cerner connector

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

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`. See the list of available attributes in [Update a Cerner user](#) for more information.

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

Attribute	Notes										
<code>___NAME___</code>	The user's name, in a <code>FAMILY</code> , <code>GIVEN</code> format. Required.										
<code>birthDate</code>	Must be in <code>YYYY-MM-DD</code> format.										
<code>gender</code>	Accepted values are <code>MALE</code> , <code>FEMALE</code> , <code>OTHER</code> , <code>UNKNOWN</code> .										
<code>given</code>	The user's first name. Required.										
<code>family</code>	The user's last name. Required.										
<code>name</code>	<table border="1"> <tr> <td><code>given</code></td> <td></td> </tr> <tr> <td><code>middle</code></td> <td></td> </tr> <tr> <td><code>family</code></td> <td></td> </tr> <tr> <td><code>suffix</code></td> <td></td> </tr> <tr> <td><code>prefix</code></td> <td></td> </tr> </table>	<code>given</code>		<code>middle</code>		<code>family</code>		<code>suffix</code>		<code>prefix</code>	
<code>given</code>											
<code>middle</code>											
<code>family</code>											
<code>suffix</code>											
<code>prefix</code>											

Attribute	Notes	
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, 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, see the <i>Internet Assigned Numbers Authority</i> (IANA) language subtag registry.	

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"
}
```

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"
}

```

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___"
}
```

## 24.5. 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.

# 24.6. Cerner Connector Configuration

The Cerner Connector has the following configurable properties.

## 24.6.1. Configuration properties

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

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

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.

## Chapter 25

# Epic Connector

Epic is a healthcare-related service which handles patient medical records. The Epic connector lets you manage and synchronize accounts between Epic and the IDM managed user objects. An Epic administrator account on the Epic system you wish to connect to is required for this connector to work.

## 25.1. Before you start

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

- Client ID
- Username
- Password
- Private key (Generate an RSA keypair and convert to PKCS8)

### Note

To generate your private key:

1. Generate and download an RSA key pair.
2. Run the following command to convert the RSA private key to PKCS8 format:

```
openssl pkcs8 -topk8 -nocrypt -in privatekey.pem -out epic_pkcs8_private_key.pem
```
3. After generating the private key in PKCS8 format, remove `-----BEGIN PRIVATE KEY-----` and `-----END PRIVATE KEY-----` from the generated PKCS8 private key file.
4. Remove any escape characters such as `\n` or `\r`.

- REST Endpoint (optional)
- SOAP Endpoint (optional)
- Max Records (optional)
- User template file path (Optional)

- User sub template file path (Optional)
- In Basket classification file path (Optional)
- Group file path (Optional)

#### Note

The **user template**, **user sub template**, **in basket**, and **group** file paths are local paths that are accessible to the IDM or `#{rcs.abbr}` instance.

## 25.2. Install the Epic 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/epic-connector-1.5.20.8.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.

## 25.3. Configure the Epic connector

Create a connector configuration using the Admin UI:

1. Select Configure > Connectors and click New Connector.
2. Enter a Connector Name.
3. Select Epic Connector - 1.5.20.8 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/Epic?_action=test"
{
  "name": "Epic",
  "enabled": true,
  "config": "config/provisioner.openicf/Epic",
  "connectorRef": {
    "bundleVersion": "[1.5.19.0,1.6.0.0)",
    "bundleName": "org.forgerock.openicf.connectors.epic-connector",
    "connectorName": "org.forgerock.openicf.connectors.epic.EpicConnector"
  },
  "displayName": "Epic Connector",
  "objectTypes": [
    "__ACCOUNT__",
    "__ALL__"
  ],
  "ok": true
}

```

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

### 25.3.1. Configure connection pooling

The Epic connector supports connection pooling, which can substantially improve the performance of the connector. The basic connection pooling configuration is described in "*Connection Pooling Configuration*".

## 25.4. Use the Epic connector

The following resources are supported by the Epic connector:

Connector Resource	Epic Resource Type
<code>__ACCOUNT__</code>	Users
Department	Departments
<code>__GROUP__</code>	Groups
Provider	Linked Provider
UserTemplate	User Template
UserSubTemplate	User Sub Template

Connector Resource	Epic Resource Type
InBasketClassifications	In Basket Classifications

You can use the Epic connector to perform the following actions on an Epic account:

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 '{
  "UserID": "8675309",
  "__NAME__": "Walter, Taylor"
}' \
"http://localhost:8080/openidm/system/Epic/___ACCOUNT___?_action=create"
{
  "_id": "8675309",
  "UserComplexName": {
    "FirstName": "Taylor",
    "GivenNameInitials": "",
    "MiddleName": "",
    "LastName": "Walter",
    "LastNamePrefix": "",
    "SpouseLastName": "",
    "SpousePrefix": "",
    "Suffix": "",
    "AcademicTitle": "",
    "PrimaryTitle": "",
    "SpouseLastNameFirst": false,
    "FatherName": "",
    "GrandfatherName": ""
  },
  "BlockStatus": {
    "IsBlocked": false,
    "Reason": "",
    "Comment": ""
  },
  "__GROUP__": [],
  "ContactComment": "Initial contact created via web service",
  "UserID": "8675309",
  "__NAME__": "WALTER, TAYLOR",
  "UsersManagers": [],
  "InBasketClassifications": [],
  "__ENABLE__": true
}
```

### Note

- When you create a new user, you must specify *at least* `UserID` and `__NAME__`.
- The maximum length of `__NAME__` is 100 characters. The format for `userName` is LAST, FIRST MI format.

- The maximum number of characters for `UserID` is 15.

The following example queries all Epic 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/Epic/__ACCOUNT__?_queryId=query-all-ids"
{
  "result": [
    {
      "_id": "dsully"
    },
    {
      "_id": "999999999"
    },
    {
      "_id": "admin@ACECompany.com"
    },
    {
      "_id": "extuser320"
    },
    {
      "_id": "Achong"
    },
    {
      "_id": "dsewell"
    },
    {
      "_id": "8675309"
    },
    {
      "_id": "atestuser"
    },
    {
      "_id": "Amaraphornc"
    },
    {
      "_id": "jocampo"
    }
  ],
  "resultCount": 10,
  "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/Epic/___ACCOUNT___/8675309"
{
  "_id": "8675309",
  "UserComplexName": {
    "FirstName": "Taylor",
    "GivenNameInitials": "",
    "MiddleName": "",
    "LastName": "Walter",
    "LastNamePrefix": "",
    "SpouseLastName": "",
    "SpousePrefix": "",
    "Suffix": "",
    "AcademicTitle": "",
    "PrimaryTitle": "",
    "SpouseLastNameFirst": false,
    "FatherName": "",
    "GrandfatherName": ""
  },
  "BlockStatus": {
    "IsBlocked": false,
    "Reason": "",
    "Comment": ""
  },
  "___GROUP___": [],
  "ContactComment": "Initial contact created via web service",
  "UserID": "8675309",
  "___NAME___": "WALTER, TAYLOR",
  "UsersManagers": [],
  "InBasketClassifications": [],
  "___ENABLE___": true
}
```

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

### User accounts

- `___ENABLE___`
- `___GROUP___`
- `___NAME___` (Required)
- `___PASSWORD___`
- `UserID`



- UserIDType
- UserAlias
- UserPhotoPath
- Sex
- Notes
- Provider
- LinkedProviderID
- Department
- ContactComment
- ContactDate
- SystemLoginID
- LDAPOverrideID
- DefaultLoginDepartmentID
- ReportGrouper1
- ReportGrouper2
- ReportGrouper3
- CategoryReportGrouper
- InBasketClassifications
- UsersManagers
- PrimaryManager
- DefaultTemplateID
- UserTemplate
- UserSubtemplateIDs
- UserComplexName - UserComplexName has the following sub-attributes:
  - FirstName
  - GivenNameInitials

- `MiddleName`
- `LastName`
- `LastNamePrefix`
- `SpouseLastName`
- `SpousePrefix`
- `SpouseLastNameFirst`
- `Suffix`

#### Note

When updating a user, `__NAME__` overrides the `FirstName`, `LastName` and `MiddleName` of `UserComplexName` attributes.

- `IsActive`
- `BlockStatus` - `BlockStatus` has the following sub-attributes:
  - `IsBlocked`
  - `BlockStatus.Comment`

### *In Basket Classifications*

- `__UID__`
- `__NAME__`

### *Groups*

- `__UID__`
- `__NAME__`
- `Type`

### *User Templates*

- `__UID__`
- `__NAME__`

## User Sub Templates

- `__UID__`
- `__NAME__`

## Provider

- `__UID__`
- `__NAME__`
- ExternalID
- Title
- NPI ID
- Provider Type
- Specialty
- Practice Name
- Street Address
- Phone

For example, to add a `Suffix` 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__ ": "Walter, Taylor",
  "UserComplexName": {
    "Suffix": "Junior"
  }
}' \
"http://localhost:8080/openidm/system/Epic/ __ACCOUNT__ /8675309"
{
  "_id": "8675309",
  "UserComplexName": {
    "FirstName": "Taylor",
```

```

    "GivenNameInitials": "",
    "MiddleName": "",
    "LastName": "Walter",
    "LastNamePrefix": "",
    "SpouseLastName": "",
    "SpousePrefix": "",
    "Suffix": "Jr.",
    "AcademicTitle": "",
    "PrimaryTitle": "",
    "SpouseLastNameFirst": false,
    "FatherName": "",
    "GrandfatherName": ""
  },
  "BlockStatus": {
    "IsBlocked": false,
    "Reason": "",
    "Comment": ""
  },
  "_GROUP_": [],
  "UserID": "8675309",
  "_NAME_": "WALTER, TAYLOR JR.",
  "UsersManagers": [],
  "InBasketClassifications": [],
  "_ENABLE_": true
}

```

To reset the password for Epic 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@123!",
  "_NAME_": "Walter, Taylor"
}' \
"http://localhost:8080/openidm/system/Epic/_ACCOUNT_/8675309"
{
  "id": "8675309",
  "UserComplexName": {
    "FirstName": "Taylor",
    "GivenNameInitials": "",
    "MiddleName": "",
    "LastName": "Walter",
    "LastNamePrefix": "",
    "SpouseLastName": "",
    "SpousePrefix": "",
    "Suffix": "",
    "AcademicTitle": "",
    "PrimaryTitle": "",

```

```

    "SpouseLastNameFirst": false,
    "FatherName": "",
    "GrandfatherName": ""
  },
  "BlockStatus": {
    "IsBlocked": false,
    "Reason": "",
    "Comment": ""
  },
  "__GROUP__": [],
  "UserID": "8675309",
  "__NAME__": "WALTER, TAYLOR",
  "UsersManagers": [],
  "InBasketClassifications": [],
  "__ENABLE__": true
}

```

The following example activates a user with the minimum required attributes, and updates their 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 '{
  "__NAME__": "Walter, Taylorupdate",
  "__ENABLE__": "true"
}' \
"http://localhost:8080/openidm/system/Epic/___ACCOUNT___/8675309"
{
  "_id": "8675309",
  "UserComplexName": {
    "FirstName": "Taylorupdate",
    "GivenNameInitials": "",
    "MiddleName": "",
    "LastName": "Walter",
    "LastNamePrefix": "",
    "SpouseLastName": "",
    "SpousePrefix": "",
    "Suffix": "",
    "AcademicTitle": "",
    "PrimaryTitle": "",
    "SpouseLastNameFirst": false,
    "FatherName": "",
    "GrandfatherName": ""
  },
  "BlockStatus": {
    "IsBlocked": false,
    "Reason": "",
    "Comment": ""
  }
}

```

```

},
  "__GROUP__": [],
  "UserID": "8675309",
  "__NAME__": "WALTER, TAYLORUPDATE",
  "UsersManagers": [],
  "InBasketClassifications": [],
  "__ENABLE__": true
}

```

The following example deactivates 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" \
  --header "Content-Type: application/json" \
  --header "if-Match:*" \
  --request PUT \
  --data '{ \
    "__NAME__": "TAYLOR, WALTER", \
    "__ENABLE__": false \
  }' \
  "http://localhost:8080/openidm/system/Epic/__ACCOUNT__/8675309"
{
  "_id": "8675309",
  "UserComplexName": {
    "FirstName": "Taylor",
    "GivenNameInitials": "",
    "MiddleName": "",
    "LastName": "Walter",
    "LastNamePrefix": "",
    "SpouseLastName": "",
    "SpousePrefix": "",
    "Suffix": "",
    "AcademicTitle": "",
    "PrimaryTitle": "",
    "SpouseLastNameFirst": false,
    "FatherName": "",
    "GrandfatherName": ""
  },
  "BlockStatus": {
    "IsBlocked": false,
    "Reason": "",
    "Comment": ""
  },
  "__GROUP__": [],
  "UserID": "8675309",
  "__NAME__": "WALTER, TAYLOR",
  "UsersManagers": [],
  "InBasketClassifications": [],
  "__ENABLE__": false
}

```

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

#### Note

A deleted account technically remains in the Epic repository, but cannot be queried by its ID.

The following example deletes an Epic 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/Epic/___ACCOUNT___/8675309"
{
  "_id": "8675309",
  "UserComplexName": {
    "FirstName": "Taylor",
    "GivenNameInitials": "",
    "MiddleName": "",
    "LastName": "Walter",
    "LastNamePrefix": "",
    "SpouseLastName": "",
    "SpousePrefix": "",
    "Suffix": "",
    "AcademicTitle": "",
    "PrimaryTitle": "",
    "SpouseLastNameFirst": false,
    "FatherName": "",
    "GrandfatherName": ""
  },
  "BlockStatus": {
    "IsBlocked": false,
    "Reason": "",
    "Comment": ""
  },
  "___GROUP___": [],
  "UserID": "8675309",
  "___NAME___": "WALTER, TAYLOR",
  "UsersManagers": [],
  "InBasketClassifications": [],
  "___ENABLE___": false
}
```

You can then confirm the account has been deleted by querying the **UserID**:

```
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/Epic/__ACCOUNT__/8675309"
{
  "code": 404,
  "reason": "Not Found",
  "message": "Object 8675309 not found on system/Epic/__ACCOUNT__"
}
```

Additionally, all supported resources can be queried:

To query all departments:

```
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/Epic/Department?_queryId=query-all-ids"
{
  "result": [
    {
      "_id": "40"
    },
    {
      "_id": "56"
    },
    {
      "_id": "71"
    },
    {
      "_id": "77"
    },
    {
      "_id": "58"
    },
    [ ... ]
    {
      "_id": "46"
    },
    {
      "_id": "10120160"
    },
    {
      "_id": "1002020"
    }
  ]
}
```



```

  },
  {
    "_id": "31"
  },
  {
    "_id": "83"
  },
  {
    "_id": "115"
  }
],
"resultCount": 548,
"pagedResultsCookie": null,
"totalPagedResultsPolicy": "NONE",
"totalPagedResults": -1,
"remainingPagedResults": -1
}

```

To query all providers:

```

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/Epic/Provider?_queryId=query-all-ids"
{
  "result": [
    {
      "_id": "116"
    },
    {
      "_id": "E3087"
    },
    {
      "_id": "E4000"
    },
    {
      "_id": "E4913"
    },
    {
      "_id": "E5335"
    },
    {
      "_id": "E4716"
    },
    {
      "_id": "E5370"
    },
    [ ... ]
    {
      "_id": "E4001"
    },
    {

```

```

    "_id": "E4002"
  },
  {
    "_id": "E5137"
  },
  {
    "_id": "E5199"
  },
  {
    "_id": "E4003"
  },
  {
    "_id": "E4694"
  },
  {
    "_id": "E4004"
  },
  {
    "_id": "E4005"
  },
  {
    "_id": "E5019"
  },
  {
    "_id": "E4843"
  }
],
"resultCount": 2560,
"pagedResultsCookie": null,
"totalPagedResultsPolicy": "NONE",
"totalPagedResults": -1,
"remainingPagedResults": -1
}

```

To query a specific provider:

```

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/Epic/Provider/E4716"
{
  "_id": "E4716",
  "Specialty": "Family Medicine",
  "UID": "E4716",
  "Provider Type": "Physician",
  "__NAME__": "WELLHIVE, PROVIDER"
}

```

To query all user templates:

```
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/Epic/UserTemplate?_queryId=query-all-ids"
{
  "result": [
    {
      "_id": "T00004"
    },
    {
      "_id": "T00024"
    },
    {
      "_id": "T00033"
    },
    {
      "_id": "T00038"
    },
    {
      "_id": "T00076"
    },
    {
      "_id": "T00077"
    },
    {
      "_id": "T00078"
    },
    {
      "_id": "T00088"
    },
    {
      "_id": "T00089"
    },
    {
      "_id": "T00090"
    },
    {
      "_id": "T1000601"
    },
    {
      "_id": "T1002020"
    },
    {
      "_id": "T1020101"
    },
    {
      "_id": "T1020102"
    },
    [ ... ]
    {
      "_id": "T8888001"
    },
  ],
  {

```

```

    "_id": "T8889901"
  },
  {
    "_id": "T9998001"
  }
],
"resultCount": 431,
"pagedResultsCookie": null,
"totalPagedResultsPolicy": "NONE",
"totalPagedResults": -1,
"remainingPagedResults": -1
}

```

To query a specific user template:

```

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/Epic/UserTemplate/T8888001"
{
  "_id": "T8888001",
  "_UID_": "T8888001",
  "_NAME_": "RESEARCH ADMINISTRATOR TEMPLATE"
}

```

To query all user sub templates:

```

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/Epic/UserSubTemplate?_queryId=query-all-ids"
{
  "result": [
    {
      "_id": "ST00007"
    },
    {
      "_id": "ST00030"
    },
    {
      "_id": "ST10200"
    },
    {
      "_id": "ST10201"
    }
  ]
}

```

```

},
{
  "_id": "ST10202"
},
{
  "_id": "ST10203"
},
{
  "_id": "ST10204"
},
[ ... ]
{
  "_id": "ST10401"
},
{
  "_id": "ST10402"
},
{
  "_id": "ST10700"
},
{
  "_id": "ST107001"
},
{
  "_id": "T5080002"
},
{
  "_id": "T99901"
},
{
  "_id": "TCVREPSUB"
}
],
"resultCount": 91,
"pagedResultsCookie": null,
"totalPagedResultsPolicy": "NONE",
"totalPagedResults": -1,
"remainingPagedResults": -1
}

```

To query a specific user sub template:

```

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/Epic/UserSubTemplate/T8888001"
{
  "_id": "T00007",
  "NAME": "EXCEL MEDICAL",
  "UID": "T00007"
}

```

To query all In Basket classifications:

```

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/Epic/InBasketClassifications?_queryId=query-all-ids"
{
  "result": [
    {
      "_id": "1"
    },
    {
      "_id": "2"
    },
    {
      "_id": "15"
    },
    {
      "_id": "29"
    },
    {
      "_id": "30"
    },
    {
      "_id": "31"
    },
    {
      "_id": "84"
    },
    {
      "_id": "85"
    },
    {
      "_id": "100"
    },
    {
      "_id": "140"
    },
    {
      "_id": "141"
    },
    {
      "_id": "212"
    }
  ],
  "resultCount": 12,
  "pagedResultsCookie": null,
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}

```

To query a specific In Basket classification:

```
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/Epic/InBasketClassifications/140"
{
  "_id": "140",
  "__NAME__": "Model AP Pt Clinical Msg Pool",
  "__UID__": "140"
}
```

To query all groups:

```
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/Epic/__GROUP__?_queryId=query-all-ids"
{
  "result": [
    {
      "_id": "1"
    },
    {
      "_id": "2"
    },
    {
      "_id": "3"
    },
    {
      "_id": "4"
    },
    {
      "_id": "5"
    },
    {
      "_id": "6"
    },
    {
      "_id": "7"
    },
    {
      "_id": "1000"
    },
    {

```

```
{
  "_id": "1001"
},
{
  "_id": "1002"
},
{
  "_id": "1003"
},
{
  "_id": "1004"
},
{
  "_id": "1005"
},
{
  "_id": "1006"
}
],
"resultCount": 14,
"pagedResultsCookie": null,
"totalPagedResultsPolicy": "NONE",
"totalPagedResults": -1,
"remainingPagedResults": -1
}
```

To query a specific 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" \
--request GET \
"http://localhost:8080/openidm/system/Epic/__GROUP__/1000"
{
  "_id": "1000",
  "__NAME__": "Community",
  "__UID__": "1000",
  "Type": "Community",
  "Undefined3": "Customer~Epic Customer"
}
```

## 25.5. OpenICF Interfaces Implemented by the Epic Connector

The Epic 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.

# 25.6. Epic Connector Configuration

The Epic Connector has the following configurable properties.

## 25.6.1. Basic configuration properties

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<code>clientId</code>	String	null		Yes
Provide the Client ID to authorize the Epic APIs				
<code>privateKey</code>	GuardedString	null	Yes	Yes
Provide the Private key in pkcs8 format				
<code>userName</code>	String	null		Yes
Provide the Username required for Connection				
<code>password</code>	GuardedString	null	Yes	Yes
Provide the Password required for Connection				
<code>userTemplatesFilePath</code>	String	null		No
Provide the location of User Template file				
<code>subTemplatesFilePath</code>	String	null		No
Provide the location of User Subtemplate file				
<code>inBasketFilePath</code>	String	null		No
Provide the location of In Basket Classifications File				
<code>groupsFilePath</code>	String	null		No
Provide the location of Group File				
<code>maxRecords</code>	int	50		No
Provide the Maximum records for search operation				
<code>maxConnections</code>	Integer	10		No
Provide the Maximum connections				
<code>connectionTimeout</code>	int	600		No
Provide the Maximum Connection Timeout in seconds				
<code>accessToken</code>	GuardedString	null	Yes	No
Provide the Access token to establish connectivity with the target				
<code>tokenValidity</code>	Long	null		No
Provide the Validity period of the token				
<code>httpProxyHost</code>	String	null		No
Provide the HTTP Proxy Host				

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<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>restEndpoint</code>	String	<code>https:// apporchard .epic.com/ interconnect- aocurprd-oauth/</code>		No
The HTTP URL for the REST End point (https://myserver.com/service/)				
<code>soapEndpoint</code>	String	<code>https:// apporchard .epic.com/ interconnect -aocurprd- username/</code>		No
The HTTP URL for the SOAP End point (https://myserver.com/service/)				

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.

## Chapter 26

# MS Graph API Java Connector

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, 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 bundled with IDM, and available from the ForgeRock Download Center. The connector bundles all its dependencies.

## 26.1. 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:
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:

**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.
  - 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.

## 26.2. Configure the MS Graph API Connector

1. IDM bundles version 1.5.20.8 of the MS Graph API connector in the `openidm/connectors` directory. Alternatively, download the connector `.jar` file from the ForgeRock Download Center
2. Create a configuration for the connector.

Configure the MS Graph API connector through the Admin UI (select Configure > Connectors), or over REST.

Alternatively, copy the sample connector configuration file from `/path/to/openidm/samples/example-configurations/provisioners/provisioner.openicf-azuread.json` 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"  
}
```

## 26.3. 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:

```
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": "azuread",
  "enabled": true,
  "config": "config/provisioner.openicf.azuread",
  "connectorRef": {
    "bundleVersion": "[1.5.19.0,1.6.0.0]",
    "bundleName": "org.forgerock.openicf.connectors.msgraphapi-connector",
    "connectorName": "org.forgerock.openicf.connectors.msgraphapi.MSGraphAPIConnector"
  },
  "displayName": "MSGraphAPI Connector",
  "objectTypes": [
    "subscribedSku",
    "team",
    "user",
    "_ALL_",
    "group"
  ],
  "ok": true
}
```

A status of `"ok": true` indicates that the connector is configured correctly.

This command retrieves a list of users in your Azure tenant. You can also use any system-enabled filter, such as those described in "Constructing Queries" in the *Integrator's 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/azuread/user?_queryId=query-all-ids"
{
  "result": [
    {
      "_id": "c48be8cc-5846-4059-95e8-a7acb6a6ec31"
    },
    {
      "_id": "c7fe57e2-3159-45e1-b67a-435232fd88d9"
    },
    {
      "_id": "9e714b5c-345a-430c-93f5-d8c6f9a2f225"
    },
    ...
  ],
  ...
}
```

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/azuread/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"
}
```

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/azuread/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"
}

```

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": "MyNewPassw0rd"
} ]' \
"http://localhost:8080/openidm/system/azuread/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"
}
```

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/system/azuread/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"
}

```

## 26.4. Manage User Licenses

The MS Graph API connector lets you list the available licenses in your Azure data source, and manage those licenses for specific users:

This command lists the values of the read-only `subscribedSku` object. For more information about this object class, see the corresponding [Microsoft documentation](#):

```

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/azuread/subscribedSku?_queryFilter=true"
{
  "result": [
    {
      "_id": "5ee8xxxx-xxxx-xxxx-xxxx-76dc2c2c30bc_f245ecc8-xxxx-xxxx-xxxx-xxxx114de5f3",
      "prepaidUnits": {
        "warning": 0,
        "enabled": 1,
        "suspended": 0
      },
      "skuId": "f245ecc8-xxxx-xxxx-xxxx-xxxx114de5f3",
      "skuPartNumber": "0365_BUSINESS_PREMIUM",
      "capabilityStatus": "Enabled",
      "appliesTo": "User",
      "consumedUnits": 1,
      "__NAME__": "0365_BUSINESS_PREMIUM",
      "servicePlans": [
        {
          "servicePlanName": "RMS_S_BASIC",
          "provisioningStatus": "PendingProvisioning",
          "appliesTo": "Company",
          "servicePlanId": "31cxxxxxxxxxxxxxxxxxxxxxxxxxxxx122"
        },
        {
          "servicePlanName": "POWER_VIRTUAL_AGENTS_0365_P2",
          "provisioningStatus": "PendingProvisioning",
          "appliesTo": "User",
          "servicePlanId": "041xxxxx-xxxx-xxxx-xxxx-xxxxxxxxxae"
        }
      ],
      {
        "servicePlanName": "CDS_0365_P2",

```

```

    "provisioningStatus": "PendingProvisioning",
    "appliesTo": "User",
    "servicePlanId": "95bxxxxx-xxxx-xxxx-xxxx-xxxxxxxxx95a"
  },
  ...
]
}
],
...
}

```

Each user object can include a read-only `licenses` property that contains an array of objects (maps).

This command lists a specific user's licenses:

```

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/azuread/user/c48be8cc-5846-4059-95e8-a7acbf6aec31?_fields=licenses"
{
  "_id": "c48be8cc-5846-4059-95e8-a7acbf6aec31",
  "licenses": [
    {
      "skuPartNumber": "0365_BUSINESS_PREMIUM",
      "servicePlans": [
        {
          "servicePlanName": "RMS_S_BASIC",
          "provisioningStatus": "PendingProvisioning",
          "appliesTo": "Company",
          "servicePlanId": "31cxxxxx-xxxx-xxxx-xxxx-xxxxxxxxx122"
        },
        {
          "servicePlanName": "POWER_VIRTUAL_AGENTS_0365_P2",
          "provisioningStatus": "PendingProvisioning",
          "appliesTo": "Company",
          "servicePlanId": "041xxxxx-xxxx-xxxx-xxxx-xxxxxxxxxae"
        },
        {
          "servicePlanName": "CDS_0365_P2",
          "provisioningStatus": "PendingProvisioning",
          "appliesTo": "Company",
          "servicePlanId": "95bxxxxx-xxxx-xxxx-xxxx-xxxxxxxxx95a"
        },
        ...
      ],
      "id": "c8noxssxEqoxxxLCwxxxxRfKvxxxth8nxxx5fM",
      "skuId": "f24xxxxx-xxxx-xxxx-xxxx-xxxxxxxxx5f3"
    }
  ]
}

```

You cannot manipulate a user's `licenses` property directly, because it is read-only. To add or remove licenses for a user, set the `addLicenses` or `removeLicenses` properties when you create or update the user.

**Note**

The connector does not currently support PATCH `add` or PATCH `remove` operations. PATCH `replace` is supported because it is the equivalent of a PUT operation.

This command updates an existing user entry to add a license with the `skuId f24xxxxx-xxxx-xxxx-xxxx-xxxxxxxxx5f3`:

```
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-None-Match: *"
\
--request PUT
\
--data '{
  "addLicenses": [
    {
      "skuId": "f24xxxxx-xxxx-xxxx-xxxx-xxxxxxxxx5f3"
    }
  ]
}' \
"http://localhost:8080/openidm/system/azuread/user/c48be8cc-5846-4059-95e8-a7acbf6aec31"
```

This command updates the user entry to remove the license with `skuId f24xxxxx-xxxx-xxxx-xxxx-xxxxxxxxx5f3`:

```
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 '{
  "removeLicenses": "f24xxxxx-xxxx-xxxx-xxxx-xxxxxxxxx5f3"
}' \
"http://localhost:8080/openidm/system/azuread/user/c48be8cc-5846-4059-95e8-a7acbf6aec31"
```

## 26.5. 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 the sample configuration file at `/path/to/openidm/samples/sync-with-azuread/conf/sync.json` as a starting point.

## 26.6. 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.

# 26.7. MSGraphAPI Connector Configuration

The MSGraphAPI Connector has the following configurable properties.

## 26.7.1. Basic Configuration Properties

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<code>tenant</code>	String	null		Yes
The Azure AD tenant name or id				
<code>clientId</code>	String	null		Yes
The clientID used by the connector during the OAuth flow				
<code>clientSecret</code>	GuardedString	null	Yes	No
The client secret used by the connector during the OAuth flow				
<code>httpProxyHost</code>	String	null		No
The Http proxy host				
<code>httpProxyPort</code>	Integer	null		No
The Http proxy port				
<code>httpProxyUsername</code>	String	null		No
The Http proxy user name				
<code>httpProxyPassword</code>	GuardedString	null	Yes	No

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
The Http proxy user password				
<code>performHardDelete</code>	<code>boolean</code>	<code>false</code>		No
If set to true, the Azure object will be deleted permanently on delete operation.				
<code>readRateLimit</code>	<code>String</code>	<code>null</code>		No
Define throttling for read operations either per seconds ("30/sec") or per minute ("100/min").				
<code>writeRateLimit</code>	<code>String</code>	<code>null</code>		No
Define throttling for write operations (create/update/delete) either per second ("30/sec") or per minute ("100/min").				

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.



## Chapter 27

# PeopleSoft Connector

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.

## 27.1. 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.

## 27.2. 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-1.5.20.8.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 `xref:generate-psjoe`.
  - `psft.jar` is created by the following commands:

```
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
```

```
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
```

5. Copy `psjoe.jar` and generated jar into `/path/to/openicf/lib`.

## 27.3. 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 - 1.5.20.8 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": "[1.5.19.0,1.6.0.0)",
    "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.

## 27.4. Use the PeopleSoft connector

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

Attribute	Description				
<code>__NAME__</code>	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 <code>IDType</code> and <code>AttributeValue</code> as sub-attributes. For example:</p> <pre>"IDTypes": [{   "IDType": "EMP",   "AttributeValue": "0001" }]</pre> <p style="text-align: center;"><i>Supported ID types</i></p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>ID Type</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>BID</td> <td>Bidder</td> </tr> </tbody> </table>	ID Type	Name	BID	Bidder
ID Type	Name				
BID	Bidder				

Attribute	Description															
	<b>ID Type</b>	<b>Name</b>														
	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 style="text-align: center;"><i>Supported Languages</i></p> <table border="1" data-bbox="722 1242 1322 1513"> <thead> <tr> <th data-bbox="722 1242 1022 1281">Language</th> <th data-bbox="1022 1242 1322 1281">Code</th> </tr> </thead> <tbody> <tr> <td data-bbox="722 1281 1022 1319">Arabic</td> <td data-bbox="1022 1281 1322 1319">ARA</td> </tr> <tr> <td data-bbox="722 1319 1022 1357">Afrikaans</td> <td data-bbox="1022 1319 1322 1357">AFR</td> </tr> <tr> <td data-bbox="722 1357 1022 1395">Bulgarian</td> <td data-bbox="1022 1357 1322 1395">BUL</td> </tr> <tr> <td data-bbox="722 1395 1022 1433">Simplified Chinese</td> <td data-bbox="1022 1395 1322 1433">ZHS</td> </tr> <tr> <td data-bbox="722 1433 1022 1472">Traditional Chinese</td> <td data-bbox="1022 1433 1322 1472">ZHT</td> </tr> <tr> <td data-bbox="722 1472 1022 1513">Croatian</td> <td data-bbox="1022 1472 1322 1513">CRO</td> </tr> </tbody> </table>		Language	Code	Arabic	ARA	Afrikaans	AFR	Bulgarian	BUL	Simplified Chinese	ZHS	Traditional Chinese	ZHT	Croatian	CRO
Language	Code															
Arabic	ARA															
Afrikaans	AFR															
Bulgarian	BUL															
Simplified Chinese	ZHS															
Traditional Chinese	ZHT															
Croatian	CRO															

Attribute	Description	
	Language	Code
	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><b>Note</b></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 <b>Y</b> (Yes) or <b>N</b> (No).								
EmailUser	Email preference of the user. Must be either <b>Y</b> (Yes) or <b>N</b> (No).								
AlternateUserID	Fallback user to route to if the user is unavailable. This must be filled out if you specify <b>EffectiveDateFrom</b> or <b>EffectiveDateTo</b> .								
EffectiveDateFrom	Effective start date that a user will be unavailable. Must be in <b>MM/DD/YYYY</b> format.								
EffectiveDateTo	Effective end date, marking when a user will become available again. Must be in <b>MM/DD/YYYY</b> format.								
EmailAddresses	<p>List of email addresses associated with the user. This is an object, with <b>EmailType</b>, <b>EmailAddress</b>, and <b>PrimaryEmail</b> as sub-attributes. For example:</p> <pre data-bbox="735 1163 1320 1293"> "EmailAddresses": [{   "EmailType": "BUS",   "EmailAddress": "test@example.com",   "PrimaryEmail": "Y" }] </pre> <p style="text-align: center;"><i>Supported email types</i></p> <table border="1" data-bbox="725 1380 1320 1532"> <thead> <tr> <th>Email Code</th> <th>Email Type</th> </tr> </thead> <tbody> <tr> <td><b>BB</b></td> <td>Blackberry</td> </tr> <tr> <td><b>HOME</b></td> <td>Home</td> </tr> <tr> <td><b>WORK</b></td> <td>Work</td> </tr> </tbody> </table>	Email Code	Email Type	<b>BB</b>	Blackberry	<b>HOME</b>	Home	<b>WORK</b>	Work
Email Code	Email Type								
<b>BB</b>	Blackberry								
<b>HOME</b>	Home								
<b>WORK</b>	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 <b>RoleName</b> and <b>Dynamic</b> as sub-attributes. For example: <pre>"Roles": [{   "RoleName": "PeopleSoft User" }]</pre>	
<u>PASSWORD</u>	The password for the user.	
ConfirmPassword	Used to confirm the password of the user. This needs to match the user's password.	

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

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.

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](#)

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
}

```

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
}

```

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.

```

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
}

```

```

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  
}
```

## 27.5. 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.

## 27.6. PeopleSoft Connector Configuration

The PeopleSoft Connector has the following configurable properties.

### 27.6.1. Configuration properties

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<code>host</code>	<code>String</code>	<code>null</code>		Yes
Host name or IP address to connect to PeopleSoft server				
<code>port</code>	<code>int</code>	<code>0</code>		Yes
Port to connect to PeopleSoft server				
<code>userId</code>	<code>String</code>	<code>null</code>		Yes
The userid used to login to PeopleSoft server				

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
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				

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.



## Chapter 28

# SAP S/4HANA Connector

The SAP S/4HANA connector lets you manage and synchronize accounts between SAP S/4HANA and IDM managed user objects. An SAP S/4HANA administrator account is required for this connector to work.

## 28.1. Before you start

Before you configure the connector, log in to your SAP S/4HANA administrator account and note the following:

### Username

Inbound Communication user of SAP S/4HANA.

### Password

Inbound Communication user password of SAP S/4HANA.

### Tenant ID

Which tenant the SAP S/4HANA instance is hosted on.

## 28.2. Install the SAP S/4HANA 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/saphana-connector-1.5.20.8.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.

## 28.3. Configure the SAP S/4HANA connector

Create a connector configuration using the Admin UI:

1. Select Configure > Connectors and click New Connector.
2. Enter a Connector Name.
3. Select SAP HANA Connector - 1.5.20.8 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/saphana?_action=test"
{
  "name": "saphana",
  "enabled": true,
  "config": "config/provisioner.openicf/saphana",
  "connectorRef": {
    "bundleVersion": "[1.5.19.0,1.6.0.0)",
    "bundleName": "org.forgerock.openicf.connectors.saphana-connector",
    "connectorName": "org.forgerock.openicf.connectors.saphana.SapHanaConnector"
  },
  "displayName": "SAP HANA Connector",
  "objectTypes": [
    "_ACCOUNT_",
    "_ALL_"
  ],
  "ok": true
}
```

If the command returns `"ok": true`, your connector has been configured correctly, and can authenticate to the SAP S/4HANA server.

## 28.4. Use the SAP S/4HANA connector

The following SAP S/4HANA account attributes are supported by the SAP S/4HANA connector:

Attribute	Description
PersonUUID	Unique identifier for a user.
PersonExternalID	The external ID of the user. This can only include uppercase letters, numbers, <code>.</code> , <code>-</code> , and <code>_</code> . Required.

Attribute	Description
UserId	Auto-generated user id.
PersonID	Auto-generated user id.
__USER__	Login username for the user. This can only include uppercase letters, numbers, ., -, and _. Required.
FirstName	First name of the user. Required.
LastName	Last name of the user. Required.
MiddleName	Middle name of the user.
LockedIndicator	Status of the user. Either <b>active</b> or <b>inactive</b> .
GenderCode	Gender of the user. Permitted values are <b>1</b> (Male), or <b>2</b> (Female). Required; use <b>0</b> to leave it unspecified. If specified, this must match the values listed in <b>FormOfAddress</b> .
FormOfAddress	Salutation of the user. Permitted values are <b>00001</b> or <b>1</b> (Mr), or <b>00002</b> or <b>2</b> (Mrs). Values must match <b>GenderCode</b> .
StartDate	Start date for the created user, in YYYY-MM-DD format. Required.
EndDate	End date for the created user, in YYYY-MM-DD format.
PersonFullName	Full name of the user.
AcademicTitle	Academic title of the user. Permitted values are <b>0001</b> (DR.), <b>0002</b> (PROF.), <b>0003</b> (PROF. DR.), <b>0004</b> (B.A.), <b>0005</b> (MBA), or <b>0006</b> (PH.D.).
CorrespondenceLanguage	The correspondence language for the user. For example, `DE`, or `EN`.
AdditionalLastName	Additional last name of the user.
BirthName	Birth name of the user.
NickName	Nickname of the user.
Initials	Initials of the user.
AcademicSecondTitle	Academic secondary title of the user. Permitted values are <b>0001</b> (DR.), <b>0002</b> (PROF.), <b>0003</b> (PROF. DR.), <b>0004</b> (B.A.), <b>0005</b> (MBA), or <b>0006</b> (PH.D.).
NameSupplement	Supplemental titles of the user. Permitted values are <b>0001</b> or <b>1</b> (Earl), <b>0002</b> or <b>2</b> (Freifrau), <b>0003</b> or <b>3</b> (Freiherr), <b>0004</b> or <b>4</b> (Fürst), <b>0005</b> or <b>5</b> (Fürstin), <b>0006</b> or <b>6</b> (Graf), <b>0007</b> or <b>7</b> (Gräfin), and <b>0008</b> or <b>8</b> (Sir).
PhoneInformation	Object with the following sub-attributes: <ul style="list-style-type: none"> <li>• <b>PhoneNumberType</b>: Type of phone number. Permitted values are <b>ECPC</b> (Cell phone), or <b>ECPB</b> (Landline).</li> </ul>

Attribute	Description
	<ul style="list-style-type: none"> <li>• <b>CountryDialingCode</b>: Country dialing code, such as <b>1</b> (USA) or <b>33</b> (France). Numbers only, <b>+</b> is not allowed.</li> <li>• <b>PhoneNumberSubscriberID</b>: Phone number of the user.</li> </ul>
EmailAddress	Email address of the user.
CompanyCode	Predefined code of the company of the user. Required.
WorkAgreementStatus	Status of the work agreement for the user. Permitted values are <b>0</b> (Inactive), or <b>1</b> (Active).
LogonLanguageCode	Language code for the user.
DateFormatCode	<p>What format dates should be displayed in. For example, <b>6</b> formats dates according to ISO 8601 (YYYY-MM-DD). Available codes:</p> <ul style="list-style-type: none"> <li>• <b>1</b> - DD.MM.YYYY (Gregorian Date)</li> <li>• <b>2</b> - MM/DD/YYYY (Gregorian Date)</li> <li>• <b>3</b> - MM-DD-YYYY (Gregorian Date)</li> <li>• <b>4</b> - YYYY.MM.DD (Gregorian Date)</li> <li>• <b>5</b> - YYYY/MM/DD (Gregorian Date)</li> <li>• <b>6</b> - YYYY-MM-DD (Gregorian Date, ISO 8601)</li> <li>• <b>7</b> - GYY.MM.DD (Japanese Date)</li> <li>• <b>8</b> - GYY/MM/DD (Japanese Date)</li> <li>• <b>9</b> - GYY-MM-DD (Japanese Date)</li> <li>• <b>A</b> - YYYY/MM/DD (Islamic Date 1)</li> <li>• <b>B</b> - YYYY/MM/DD (Islamic Date 2)</li> <li>• <b>C</b> - YYYY/MM/DD (Iranian Date)</li> </ul>
TimeFormatCode	<p>What format times should be displayed in. For example, <b>0</b> formats times in a 24 hour format (14:35:59 PM). Available codes:</p> <ul style="list-style-type: none"> <li>• <b>0</b> - 24 Hour Format (Example: 12:05:10)</li> <li>• <b>1</b> - 12 Hour Format (Example: 12:05:10 PM)</li> <li>• <b>2</b> - 12 Hour Format (Example: 12:05:10 pm)</li> <li>• <b>3</b> - Hours from 0 to 11 (Example: 00:05:10 PM)</li> <li>• <b>4</b> - Hours from 0 to 11 (Example: 00:05:10 pm)</li> </ul>

Attribute	Description
TimeZoneCode	Time zone code of the user.
DecimalFormatCode	What decimal notation numbers should be displayed in. Available codes: <ul style="list-style-type: none"> <li>• 1.234.567,89</li> <li>• X - 1,234,567.89</li> <li>• Y - 1 234 567,89</li> </ul>
Role	Role assignment of the user.
PersonWorkAgreementUUID	Unique ID of the work agreement associated with the user.
PersonWorkAgreementExternalID	External ID of the work agreement associated with the user.
PersonWorkAgreementType	Role of the work agreement associated with the user. Permitted values are 1 (User), or 3 (Service performer).

#### Note

The following attributes are mapped in the connector automatically:

- PersonWorkAgreementUUID is mapped to PersonExternalID
- PersonWorkAgreementExternalID is mapped to PersonExternalID
- PersonWorkAgreementType is assigned to its default value

You can use the SAP S/4HANA connector to perform the following actions on an SAP S/4HANA account:

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",
  "FirstName": "Barbara",
  "LastName": "Jensen",
  "StartDate": "2022-06-02",
  "PersonExternalID": "BJENSEN",
  "CompanyCode": 1010,
```

```
"GenderCode": 2
}\
"http://localhost:8080/openidm/system/saphana/___ACCOUNT___?_action=create"
{
  "_id": "9980004320",
  "TimeFormatCode": "0",
  "PersonFullName": "Barbara Jensen",
  "___UID___": "9980004320",
  "FirstName": "Barbara",
  "UserID": "CB9980004320",
  "___NAME___": "BJENSEN",
  "DecimalFormatCode": "X",
  "StartDate": "2022-06-02",
  "LogonLanguageCode": "EN",
  "LastName": "Jensen",
  "PersonExternalID": "BJENSEN",
  "PersonUUID": "FA163EA9-3617-1EEC-B8DA-AD865EF3B625",
  "___ENABLE___": true,
  "TimeZoneCode": "CET",
  "EndDate": "9999-12-31",
  "DateFormatCode": "1"
}
```

### Note

When you create a new user, you must specify *at least*:

- `___NAME___`
- `PersonExternalID`
- `FirstName`
- `LastName`
- `GenderCode`
- `StartDate`
- `CompanyCode`

See the list of available attributes for more information.

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:

- `___USER___`
- `PersonExternalID`
- `FirstName`
- `LastName`
- `GenderCode`

- `EmailAddress`
- `PhoneInformation`
- `PersonFullName`
- `AcademicTitle`
- `CorrespondenceLanguage`
- `MiddleName`
- `AdditionalLastName`
- `BirthName`
- `NickName`
- `Initials`
- `AcademicSecondTitle`
- `NameSupplement`
- `WorkAgreementStatus`
- `CompanyCode`
- `StartDate`
- `EndDate`
- `LockedIndicator`
- `DateFormatCode`
- `DecimalFormatCode`
- `TimeFormatCode`
- `TimeZoneCode`
- `LogonLanguageCode`
- `Role`

#### Note

When updating the validity period for a user, both `StartDate` and `EndDate` are required.

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_": "BJENSEN",
  "FirstName": "Barbara",
  "LastName": "Jensen",
  "StartDate": "2022-06-02",
  "PersonExternalID": "BJENSEN",
  "CompanyCode": 1010,
  "GenderCode": 2,
  "EmailAddress": "bjensen@example.com"
}' \
"http://localhost:8080/openidm/system/saphana/___ACCOUNT___/9980004320"
{
  "_id": "9980004320",
  "TimeFormatCode": "0",
  "PersonFullName": "Barbara Jensen",
  "_UID_": "9980004320",
  "EmailAddress": "bjensen@example.com",
  "FirstName": "Barbara",
  "UserID": "CB9980004320",
  "_NAME_": "BJENSEN",
  "DecimalFormatCode": "X",
  "StartDate": "2022-06-02",
  "LogonLanguageCode": "EN",
  "LastName": "Jensen",
  "PersonExternalID": "BJENSEN",
  "PersonUUID": "FA163EA9-3617-1EEC-B8DA-AD865EF3B625",
  "_ENABLE_": true,
  "TimeZoneCode": "CET",
  "EndDate": "9999-12-31",
  "DateFormatCode": "1"
}
```

The following example queries all SAP S/4HANA 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/saphana/___ACCOUNT___?_queryId=query-all-ids"
{
  "result": [
```



```
{
  "_id": "9980000000"
},
{
  "_id": "9980000002"
},
{
  "_id": "9980000004"
},
{
  "_id": "9980000006"
},
[ ... ]
{
  "_id": "9980004314"
},
{
  "_id": "9980004316"
},
{
  "_id": "9980004318"
},
{
  "_id": "9980004320"
}
],
"resultCount": 2139,
"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/saphana/___ACCOUNT___/9980004320"
{
  "_id": "9980004320",
  "TimeFormatCode": "0",
  "PersonFullName": "Barbara Jensen",
  "___UID___": "9980004320",
  "EmailAddress": "bjensen@example.com",
  "FirstName": "Barbara",
  "UserID": "CB9980004320",
  "___NAME___": "BJENSEN",
  "DecimalFormatCode": "X",
  "StartDate": "2022-06-02",
  "LogonLanguageCode": "EN",
  "LastName": "Jensen",
  "PersonExternalID": "BJENSEN",
  "PersonUUID": "FA163EA9-3617-1EEC-B8DA-AD865EF3B625",
  "___ENABLE___": true,
  "TimeZoneCode": "CET",
  "EndDate": "9999-12-31",
  "DateFormatCode": "1"
}

```

```

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",
  "___ENABLE___": true
}' \
"http://localhost:8080/openidm/system/saphana/___ACCOUNT___/9980004320"
{
  "_id": "9980004320",
  "TimeFormatCode": "0",
  "PersonFullName": "Barbara Jensen",
  "___UID___": "9980004320",
  "EmailAddress": "bjensen@example.com",
  "FirstName": "Barbara",

```

```

"UserID": "CB9980004320",
"__NAME__": "BJENSEN",
"DecimalFormatCode": "X",
"StartDate": "2022-06-02",
"LogonLanguageCode": "EN",
"LastName": "Jensen",
"PersonExternalID": "BJENSEN",
"PersonUUID": "FA163EA9-3617-1EEC-B8DA-AD865EF3B625",
"__ENABLE__": true,
"TimeZoneCode": "CET",
"EndDate": "9999-12-31",
"DateFormatCode": "1"
}

```

```

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",
  "__ENABLE__": false
}' \
"http://localhost:8080/openidm/system/saphana/__ACCOUNT__/9980004320"
{
  "id": "9980004320",
  "TimeFormatCode": "0",
  "PersonFullName": "Barbara Jensen",
  "UID": "9980004320",
  "EmailAddress": "bjensen@example.com",
  "FirstName": "Barbara",
  "UserID": "CB9980004320",
  "__NAME__": "BJENSEN",
  "DecimalFormatCode": "X",
  "StartDate": "2022-06-02",
  "LogonLanguageCode": "EN",
  "LastName": "Jensen",
  "PersonExternalID": "BJENSEN",
  "PersonUUID": "FA163EA9-3617-1EEC-B8DA-AD865EF3B625",
  "__ENABLE__": false,
  "TimeZoneCode": "CET",
  "EndDate": "9999-12-31",
  "DateFormatCode": "1"
}

```

## 28.5. OpenICF Interfaces Implemented by the SAP HANA Connector

The SAP HANA Connector implements the following OpenICF interfaces.

### Create

Creates an object and 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.

## 28.6. SAP HANA Connector Configuration

The SAP HANA Connector has the following configurable properties.

### 28.6.1. Basic configuration properties

Property	Type	Default	Encrypted <sup>a</sup>	Required <sup>b</sup>
<code>userName</code>	String	null		Yes
Provide the Username to authorize the SAP HANA APIs				
<code>password</code>	GuardedString	null	Yes	Yes
Provide the Password to authorize the SAP HANA APIs				
<code>tenantId</code>	String	null		Yes
Provide the Tenant ID to identify your custom SAP HANA APIs				
<code>pageSize</code>	Integer	1000		No
Page Size for search operation				
<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				

<sup>a</sup> Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.

<sup>b</sup> A list of operations in this column indicates that the property is required for those operations.

# Appendix A. ICF Interfaces

This chapter describes all of the interfaces supported by the ICF framework, along with notes about their implementation. Specific connectors may support only a subset of these interfaces.

## A.1. AttributeNormalizer

Normalize attributes to ensure consistent filtering.

## A.2. 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.

## A.3. 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.

## A.4. 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.

## A.5. Create

Create an object and return its UID.

## A.6. Delete

Delete an object by its UID.

## A.7. Get

Get an object by its UID.

## A.8. PoolableConnector

Use pools of target resources.

## A.9. Resolve Username

Resolve an object to its UID based on its username.

## A.10. Schema

Describe supported object types, operations, and options.

## A.11. Script on Connector

Allow script execution on the connector.

## A.12. Script On Resource

Allow script execution on the resource.

## A.13. Search

Allow searches for resource objects.

Connectors that implement *only* this interface can only be used for reconciliation operations.

## A.14. Sync

Poll for synchronization events, which are native changes to target objects.

## A.15. Sync Event

Subscribe for notification of synchronization events, which are native changes to target objects.

## A.16. Test

Test the connection configuration, including connecting to the resource.

## A.17. Update

Allows an authorized caller to update (modify or replace) objects on the target resource.

## A.18. 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

This chapter describes the predefined operation options, along with notes about their use. Specific connectors may support only a subset of these options.

### B.1. 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`

### B.2. 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.

### B.3. 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.

## B.4. Run with Password

An option to use with Script on Resource that specifies a password under which to execute the script or operation.

## B.5. 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.

## B.6. 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.

## B.7. 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.

## B.8. Page Size

An option to use with Search that specifies the requested page results page size.

## B.9. 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.

## B.10. 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.

## B.11. 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

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.