FORGEROCK®

Connector Reference

/ ForgeRock Identity Management 6.0

Latest update: 6.0.0.7

Lana Frost

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

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.

Table of Contents

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 OpenICF connectors that are supported in a deployment of ForgeRock Identity Management (IDM). The guide focuses on getting the connectors installed and configured with IDM software.

This guide does not describe all OpenICF connectors. Additional connectors are available from ForgeRock's BackStage site.

This guide is written for anyone using supported OpenICF connectors with IDM software.

You do not need to have a complete understanding of IDM 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 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

This chapter provides a high-level overview of the supported connectors.

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

1.1. Connectors Supported With IDM 6

The following connectors are supported for use with IDM 6:

Adobe Marketing Cloud Connector	Salesforce Connector
CSV File Connector	SAP Connector
Database Table Connector	SCIM Connector
Google Apps Connector	Scripted REST Connector
Groovy Connector	Scripted SQL Connector
Kerberos Connector	ServiceNow Connector
LDAP Connector	SSH Connector
Marketo Connector	Workday Connector
Office 365 Connector	PowerShell Connector

Adobe Marketing Cloud Connector

The Adobe Marketing Cloud connector enables you to manage profiles in an Adobe Campaign data store.

For information about installing and configuring the Adobe Campaign Manager connector, see "*Adobe Marketing Cloud Connector*".

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.

For information about installing and configuring the CSV file connector, see "*CSV File Connector*".

Database Table Connector

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

For information about installing and configuring the Database Table connector, see "*Database Table Connector*".

Google Apps Connector

The Google Apps connector enables you to interact with Google's web applications.

For information about installing and configuring the Google Apps connector, see "*Google Apps Connector*".

Groovy Connector

The scripted Groovy Connector toolkit enables you to run a Groovy script for any OpenICF operation, such as search, update, create, and others, on any external resource.

For information about installing and configuring the Groovy connector, see "*Groovy Connector Toolkit*".

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). This connector enables you to manage Kerberos user principals from IDM.

For information about installing and configuring the Kerberos connector, see "*Kerberos Connector*".

LDAP Connector

The generic 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.

For information about installing and configuring the LDAP connector, see "*Generic LDAP Connector*".

Marketo Connector

The Marketo connector enables synchronization between IDM managed users and a Marketo Lead Database.

For information about installing and configuring the Marketo connector, see "*Marketo Connector*".

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.

For information about installing and configuring the Office 365 connector, see "*Office 365 Connector*".

PowerShell Connector

The scripted PowerShell Connector toolkit allows you to create a connector customized to communicate with Microsoft systems such as Azure AD and Active Directory.

For information about installing and configuring the PowerShell connector, see "*PowerShell Connector Toolkit*".

Salesforce Connector

The Salesforce connector enables provisioning, reconciliation, and synchronization between Salesforce and the IDM repository.

For information about installing and configuring the Salesforce connector, see "*Salesforce Connector*".

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.

For information about installing and configuring the SAP connector, see "*SAP Connector*".

SCIM Connector

The SCIM connector is based on the Simple Cloud Identity Management (SCIM) protocol and enables you to manage user and group accounts on any SCIM-compliant resource provider, such as Slack, Facebook or SalesForce.

For information about installing and configuring the SCIM connector, see "*SCIM Connector*".

Scripted REST Connector

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

For information about installing and configuring the Scripted REST connector, see "*Scripted REST Connector*".

Scripted SQL Connector

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

For information about installing and configuring the Scripted SQL connector, see "*Scripted SQL Connector*".

ServiceNow Connector

This connector enables you to manage objects in the ServiceNow platform, integrating with ServiceNow's REST API.

For information about installing and configuring the ServiceNow connector, see "*ServiceNow Connector*".

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 enables you to interact with any SSH server, using Groovy scripts for the OpenICF operations.

For information about installing and configuring the SSH connector, see "*SSH Connector*".

Workday Connector

The Workday connector enables you to synchronize user and organization accounts between IDM and Workday's cloud-based HR system.

For information about installing and configuring the Workday connector, see "*Workday Connector*".

Chapter 2 Generic LDAP Connector

The generic 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.

OpenICF 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 1.5.20.0 of the LDAP connector. Three sample LDAP connector configurations are provided in the `path/to/openidm/samples/example-configurations/provisioners/` directory:

- `provisioner.openicf-dsldap.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": "[1.4.0.0,2.0.0.0)"
   }
 }
```

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

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

```
"configurationProperties" : {
    "host" : "localhost",
    "port" : 1389,
    "ssl" : false,
    "startTLS" : false,
    "privateKeyAlias" : null,
    "alternateKeyStore" : null,
    "alternateKeyStoreType" : null,
    "alternateKeyStorePassword" " null,
    "principal" : "cn=Directory Manager",
    "credentials" : "password",
    "baseContexts" : [
        "dc=example,dc=com"
    ],
    "baseContextsToSynchronize" : [
        "dc=example,dc=com"
    ],
    "accountSearchFilter" : null,
    "accountSynchronizationFilter" : null,
    "groupSearchFilter" : null,
    "groupSynchronizationFilter" : null,
    "passwordAttributeToSynchronize" : null,
    "synchronizePasswords" : false,
    "removeLogEntryObjectClassFromFilter" : true,
    "modifiersNamesToFilterOut" : [ ],
    "passwordDecryptionKey" : null,
    "changeLogBlockSize" : 100,
    "attributesToSynchronize" : [ ],
    "changeNumberAttribute" : "changeNumber",
    "passwordDecryptionInitializationVector" : null,
    "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.

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.

`passwordAttributeToSynchronize`

IDM synchronizes password values on this attribute.

`passwordDecryptionInitializationVector`

This is a legacy attribute, and its value should remain set to `null`. To configure password synchronization between an LDAP server and IDM, use one of the password synchronization plugins, described in the Password Synchronization Plugin Guide.

`passwordDecryptionKey`

This is a legacy attribute, and its value should remain set to `null`. To configure password synchronization between an LDAP server and IDM, use one of the password synchronization plugins, described in the Password Synchronization Plugin Guide.

`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.

`synchronizePasswords`

This is a legacy attribute, and its value should remain set to `false`. To configure password synchronization between an LDAP server and IDM, use one of the password synchronization plugins, described in the Password Synchronization Plugin Guide.

`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.

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 enables you to define a list of DNs. During synchronization operations, the connector ignores changes made by these DNs.

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

To avoid this situation, you can specify a unique user in your LDAP directory, that will be used *only* for the LDAP connector. The unique user must be something other than `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. 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"
}
```

Note

By default, the LDAP directory handles referential integrity with regard to groups. In other words, it is the function of the directory to modify group membership when entries are renamed or deleted. If you want the connector to manage this functionality set `maintainLdapGroupMembership` to `true` in the provisioner file.

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 OpenICF 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 [a]	Required [b]
`filterWithOrInsteadOfAnd`	`boolean`	`false`		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.				
`objectClassesToSynchronize`	`String[]`	`['inetOrgPerson'`		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.				
`baseContextsToSynchronize`	`String[]`	`[]`		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.				
`attributesToSynchronize`	`String[]`	`[]`		Sync
The names of the attributes to synchronize. This ignores updates from the change log if they do not update any of the named attributes. For example, if only "department" is listed, then only changes that affect "department" will be processed. All other updates are ignored. If blank (the default), then all changes are processed.				
`changeNumberAttribute`	`String`	`changeNumber`		Sync
The name of the change number attribute in the change log entry.				
`modifiersNamesToFilterOut`	`String[]`	`[]`		Sync
The list of names (DNs) to filter from the changes. Changes with the attribute "modifiersName" that match entries in this list will be filtered out. The standard value is the administrator name used by this adapter, to prevent loops. Entries should be of the format "cn=Directory Manager".				
`credentials`	`GuardedString`	`null`	Yes	No
Password for the principal.				
`changeLogBlockSize`	`int`	`100`		Sync
The number of change log entries to fetch per query.				
`useTimestampsForSync`	`boolean`	`false`		Sync

Property	Type	Default	Encrypted [a]	Required [b]
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.				
`accountSynchronizationFilter`	`String`	`null`		Sync
An optional LDAP filter for the objects to synchronize. Because the change log is for all objects, this filter updates only objects that match the specified filter. If you specify a filter, an object will be synchronized only if it matches the filter and includes a synchronized object class.				
`removeLogEntryObjectClassFromFilter`	`boolean`	`true`		Sync
If this property is set (the default), the filter used to fetch change log entries does not contain the "changeLogEntry" object class, expecting that there are no entries of other object types in the change log.				
`alternateKeyStorePassword`	`GuardedString`	`null`	Yes	No
Password to use for the alternate keystore				
`groupSynchronizationFilter`	`String`	`null`		Sync
An optional LDAP filter for the objects to synchronize. Because the change log is for all objects, this filter updates only objects that match the specified filter. If you specify a filter, an object will be synchronized only if it matches the filter and includes a synchronized object class.				
`groupMemberAttribute`	`String`	`uniqueMember`		No
The name of the group attribute that will be updated with the distinguished name of the user when the user is added to the group.				
`accountSearchFilter`	`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.				
`privateKeyAlias`	`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.				
`ssl`	`boolean`	`false`		No
Select the check box to connect to the LDAP server using SSL.				
`maintainPosixGroupMembership`	`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.				
`checkAliveMinInterval`	`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.				

Property	Type	Default	Encrypted [a]	Required [b]
`groupSearchFilter`	`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.				
`referralsHandling`	`String`	`follow`		No
Defines how to handle LDAP referrals. Possible values can be follow, ignore or throw.				
`host`	`String`	`null`		No
The name or IP address of the host where the LDAP server is running.				
`maintainLdapGroupMembership`	`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.				
`resetSyncToken`	`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.				
`vlvSortAttribute`	`String`	`uid`		No
Specify the sort attribute to use for VLV indexes on the resource.				
`convertGTToISO8601`	`String[]`	`['whenCreated', 'whenChanged']`		No
Converts the Greenwich Time to ISO8601 format				
`baseContexts`	`String[]`	`[]`		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.				
`hostNameVerification`	`boolean`	`false`		No
If true, the connector will verify the hostname in the certificate (subject + alternative subject) against the defined hostNameVerifierPattern.				
`blockSize`	`int`	`100`		No
The maximum number of entries that can be in a block when retrieving entries in blocks.				
`groupObjectClasses`	`String[]`	`['top', 'groupOfUniqueN`		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.				

Property	Type	Default	Encrypted [a]	Required [b]
`accountUserNameAttributes`	`String[]`	`['uid', 'cn']`		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.				
`failover`	`String[]`	`[]`		No
List all servers that should be used for failover in case the preferred server fails. If the preferred server fails, JNDI will connect to the next available server in the list. List all servers in the form of "ldap://ldap.example.com:389/", which follows the standard LDAP v3 URLs described in RFC 2255. Only the host and port parts of the URL are relevant in this setting.				
`port`	`int`	`389`		No
TCP/IP port number used to communicate with the LDAP server.				
`convertADIntervalToISO8601`	`String[]`	`['pwdLastSet', 'accountExpires', 'lockoutTime', 'lastLogon']`		No
Converts the AD Interval to ISO8601				
`hostNameVerifierPattern`	`String`	`null`		No
A simple pattern used to match the hostname from the certificate. It can contains * character (server1.example.com, *.example.com)				
`passwordAttribute`	`String`	`userPassword`		No
The name of the LDAP attribute that holds the password. When changing a users password, the new password is set to this attribute.				
`useDNSSRVRecord`	`boolean`	`false`		No
If true, the connector will do a DNS query to find SRV records associated with the value set for host property ("_ldap._tcp.example.com" for example). Defaults to false.				
`getGroupMemberId`	`boolean`	`false`		No
Specifies whether to add an extra _memberId attribute to get the group members __UID__. CAUTION: Setting this property to true can incur a large performance cost on group handling.				
`lastCheckAlive`	`long`	`1618416343876`		No
The last time the connector was checked to see if it was alive				
`startTLS`	`boolean`	`false`		No
Specifies whether to use the startTLS operation to initiate a TLS/SSL session.				
`allowTreeDelete`	`boolean`	`false`		No
Connector can delete an entry (node) with leaf entry if this value is set to true (defaults to false). The LDAP control LDAP_SERVER_TREE_DELETE_OID (1.2.840.113556.1.4.805) is used.				

Property	Type	Default	Encrypted [a]	Required [b]
`respectResourcePasswordPolicyChange`	`boolean`	`false`		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.				
`uidAttribute`	`String`	`entryUUID`		No
The name of the LDAP attribute that is mapped to the OpenICF UID attribute.				
`principal`	`String`	`null`		No
The distinguished name with which to authenticate to the LDAP server.				
`accountObjectClasses`	`String[]`	`['top', 'person', 'organizationalI , 'inetOrgPerson'`		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.				
`alternateKeyStoreType`	`String`	`null`		No
Defines the type of the alternate key store. Valid values are JKS, JCEKS and PKCS12				
`passwordHashAlgorithm`	`String`	`null`		No
Indicates the algorithm that the Identity system should use to hash the password. Currently supported values are SSHA, SHA, SMD5, MD5 and WIN-AD (when AD is the target). A blank value indicates that the system will not hash passwords. This will cause clear text passwords to be stored in LDAP unless the LDAP server performs the hash (as Forgerocks OpenDJ does, for example).				
`alternateKeyStore`	`String`	`null`		No
Defines the filename of an alternate keystore. If specified, the connector will not use the default keystore specified by the javax.net.ssl.keyStore property.				
`authType`	`String`	`simple`		No
The authentication mechanism to use: Simple or SASL-GSSAPI. Defaults to "simple".				
`connectionTimeout`	`int`	`30000`		No
The timeout (in ms) before the connection attempt is aborted.				
`useBlocks`	`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.				

Property	Type	Default	Encrypted [a]	Required [b]
`readSchema`	`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.				
`usePagedResultControl`	`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.				
`useOldADGUIDFormat`	`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.				
`sendCAUDTxId`	`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.				
`gssapiLoginContext`	`String`	`null`		No
Defines the name used in the JAAS configuration file to define the JAAS login configuration. If null, it defaults to "org.identityconnectors.ldap.LdapConnector".				

[a] Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.
[b] A list of operations in this column indicates that the property is required for those operations.

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": "[1.5.1.4,1.6.0.0)"
  }
}
```

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 [a]	Required [b]
`headerPassword`	`String`	`password`		No
The CSV header that maps to the password for each row. Use this property when password-based authentication is required.				
`spaceReplacementString`	`String`	`_`		No
The character(s) used to replace spaces within column names.				
`csvFile`	`File`	`null`		Yes
The full path to the CSV file that is the data source for this connector.				
`newlineString`	`String`	`\n`		No
The character string in the CSV file that is used to terminate each line.				
`headerUid`	`String`	`uid`		No
The CSV header that maps to the uid (or name) for each row.				
`quoteCharacter`	`String`	`"`		No
The character in the CSV file that is used to encapsulate strings.				
`fieldDelimiter`	`String`	`,`		No
The character in the CSV file that is used to separate field values.				
`syncFileRetentionCount`	`int`	`3`		No

Property	Type	Default	Encrypted [a]	Required [b]
The number of historical copies of the CSV file to retain when performing synchronization operations.				

[a] Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.
[b] A list of operations in this column indicates that the property is required for those operations.

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 [a]	Required [b]
`connectionProperties`	`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.				
`propagateInterruptState`	`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.				
`useDisposableConnectionFacade`	`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.				
`defaultCatalog`	`String`	`null`		No
The default catalog of connections created by this pool.				
`validationInterval`	`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).				
`ignoreExceptionOnPreLoad`	`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.				
`jmxEnabled`	`boolean`	`true`		No
Register the pool with JMX or not. The default value is true.				
`commitOnReturn`	`boolean`	`false`		No
If autoCommit==false then the pool can complete the transaction by calling commit on the connection as it is returned to the pool If rollbackOnReturn==true then this attribute is ignored. Default value is false.				
`logAbandoned`	`boolean`	`false`		No
Flag to log stack traces for application code which abandoned a Connection. Logging of abandoned Connections adds overhead for every Connection borrow because a stack trace has to be generated. The default value is false.				

Property	Type	Default	Encrypted [a]	Required [b]
`maxIdle`	`int`	`100`		No
The maximum number of connections that should be kept in the pool at all times. Idle connections are checked periodically (if enabled) and connections that have been idle for longer than minEvictableIdleTimeMillis are released. The default value is derived from maxActive:100. (Also see testWhileIdle.)				
`testWhileIdle`	`boolean`	`false`		No
The indication of whether objects will be validated by the idle object evictor (if any). If an object fails to validate, it will be dropped from the pool. NOTE - for a true value to have any effect, the validationQuery parameter must be set to a non-null string. The default value is false and this property has to be set in order for the pool cleaner/test thread is to run (also see timeBetweenEvictionRunsMillis)				
`removeAbandoned`	`boolean`	`false`		No
Flag to remove abandoned connections if they exceed the removeAbandonedTimeout. If set to true a connection is considered abandoned and eligible for removal if it has been in use longer than the removeAbandonedTimeout Setting this to true can recover db connections from applications that fail to close a connection. See also logAbandoned The default value is false.				
`abandonWhenPercentageFull`	`int`	`0`		No
Connections that have been abandoned (timed out) wont get closed and reported up unless the number of connections in use are above the percentage defined by abandonWhenPercentageFull. The value should be between 0-100. The default value is 0, which implies that connections are eligible for closure as soon as removeAbandonedTimeout has been reached.				
`minIdle`	`int`	`10`		No
The minimum number of established connections that should be kept in the pool at all times. The connection pool can shrink below this number if validation queries fail. The default value is derived from initialSize:10. (Also see testWhileIdle.)				
`defaultReadOnly`	`Boolean`	`null`		No
The default read-only state of connections created by this pool. If not set then the setReadOnly method will not be called. (Some drivers dont support read only mode, ex: Informix)				
`maxWait`	`int`	`30000`		No
The maximum number of milliseconds that the pool will wait (when there are no available connections) for a connection to be returned before throwing an exception. Default value is 30000 (30 seconds)				
`logValidationErrors`	`boolean`	`false`		No
Set this to true to log errors during the validation phase to the log file. If set to true, errors will be logged as SEVERE. Default value is false for backwards compatibility.				
`driverClassName`	`String`	`null`		No
The fully qualified Java class name of the JDBC driver to be used. The driver has to be accessible from the same classloader as tomcat-jdbc.jar				

Property	Type	Default	Encrypted [a]	Required [b]
`name`	`String`	`Tomcat Connection Pool[3 -1207228264]`		No
Returns the name of the connection pool. By default a JVM unique random name is assigned.				
`useStatementFacade`	`boolean`	`true`		No
Returns true if this connection pool is configured to wrap statements in order to enable equals() and hashCode() methods to be called on the closed statements if any statement proxy is set.				
`initSQL`	`String`	`null`		No
A custom query to be run when a connection is first created. The default value is null.				
`validationQueryTimeout`	`int`	`-1`		No
The timeout in seconds before a connection validation queries fail. This works by calling java.test_sample.Statement.setQueryTimeout(seconds) on the statement that executes the validationQuery. The pool itself doesnt timeout the query, it is still up to the JDBC driver to enforce query timeouts. A value less than or equal to zero will disable this feature. The default value is -1.				
`validationQuery`	`String`	`null`		No
The SQL query that will be used to validate connections from this pool before returning them to the caller. If specified, this query does not have to return any data, it just cant throw a SQLException. The default value is null. Example values are SELECT 1(mysql), select 1 from dual(oracle), SELECT 1(MS Sql Server)				
`rollbackOnReturn`	`boolean`	`false`		No
If autoCommit==false then the pool can terminate the transaction by calling rollback on the connection as it is returned to the pool Default value is false.				
`alternateUsernameAllowed`	`boolean`	`false`		No
By default, the jdbc-pool will ignore the DataSource.getConnection(username,password) call, and simply return a previously pooled connection under the globally configured properties username and password, for performance reasons. The pool can however be configured to allow use of different credentials each time a connection is requested. To enable the functionality described in the DataSource.getConnection(username,password) call, simply set the property alternateUsernameAllowed to true. Should you request a connection with the credentials user1/password1 and the connection was previously connected using different user2/password2, 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.				
`validatorClassName`	`String`	`null`		No
The name of a class which implements the org.apache.tomcat.jdbc.pool.Validator interface and provides a no-arg constructor (may be implicit). If specified, the class will be used to create a Validator instance which is then used instead of any validation query to validate connections. The default value is null. An example value is com.mycompany.project.SimpleValidator.				
`suspectTimeout`	`int`	`0`		No

Property	Type	Default	Encrypted [a]	Required [b]
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.				
`useEquals`	`boolean`	`true`		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.				
`removeAbandonedTimeout`	`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.				
`defaultAutoCommit`	`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 setAutoCommit method will not be called.)				
`testOnConnect`	`boolean`	`false`		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.				
`jdbcInterceptors`	`String`	`null`		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.				
`initialSize`	`int`	`10`		No
The initial number of connections that are created when the pool is started. Default value is 10				
`defaultTransactionIsolation`	`int`	`-1`		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.				
`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

Property	Type	Default	Encrypted [a]	Required [b]
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 [a]	Required [b]
The maximum number of active connections that can be allocated from this pool at the same time. The default value is 100				
`username`	`String`	`null`		No
The connection username to be passed to our JDBC driver to establish a connection. Note that method DataSource.getConnection(username,password) by default will not use credentials passed into the method, but will use the ones configured here. See alternateUsernameAllowed property for more details.				
`table`	`String`	`TABLE_NAME`		Yes
Enter the name of the table in the database that contains the accounts.				

[a] Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.
[b] A list of operations in this column indicates that the property is required for those operations.

4.4.2. Basic Configuration Properties

Property	Type	Default	Encrypted [a]	Required [b]
`password`	`String`	`null`	Yes	Yes
The connection password to be passed to the JDBC driver to establish a connection. Note that method DataSource.getConnection(username,password) by default will not use credentials passed into the method, but will use the ones configured here. See alternateUsernameAllowed property for more details.				
`quoting`	`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.				
`keyColumn`	`String`	`KEY_COLUMN`		Yes
This mandatory column value will be used as the unique identifier for rows in the table.				
`passwordColumn`	`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.				
`disablePaging`	`boolean`	`false`		Yes
If true, optional paging in a query will be ignored by the connector. Defaults to false.				
`enableEmptyString`	`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.				
`rethrowAllSQLExceptions`	`boolean`	`true`		No

Property	Type	Default	Encrypted [a]	Required [b]
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.				
`nativeTimestamps`	`boolean`	`false`		No
Select to retrieve Timestamp data type of the columns in java.sql.Timestamp format from the database table.				
`allNative`	`boolean`	`false`		No
Select to retrieve all data types of columns in native format from the database table.				
`changeLogColumn`	`String`	`null`		Sync
The change log column stores the latest change time. Providing this value the Sync capabilities are activated.				
`suppressPassword`	`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.				
`inclusiveSync`	`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.				

[a] Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.
[b] A list of operations in this column indicates that the property is required for those operations.

Chapter 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, MS 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 ForgeRock's BackStage site.

IDM includes Active Directory and Azure sample scripts for the PowerShell connector that can help you get started with this toolkit. For more information, see "*Connecting to Active Directory With the PowerShell Connector*" in the *Samples Guide* and "*Connecting to Azure AD With the PowerShell Connector*" in the *Samples Guide*.

The sample scripts illustrate the following scenarios:

- Synchronization of users between Windows AD DS and IDM.
- Synchronization of users between Windows Azure AD and IDM.

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\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 ForgeRock's BackStage 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. Sample PowerShell scripts are provided in the `openidm\samples\` directory. The `scripted-powershell-with-ad` directory contains scripts for a connection to Active Directory, and the `scripted-powershell-with-azure-ad` contains scripts for a connection to Azure AD. 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 for the PowerShell connector from the `samples\provisioners` directory to your project's `conf` directory.

   IDM includes two sample PowerShell connector configurations:

   - Active Directory: `provisioner.openicf-adpowershell.json`
   - Azure AD: `provisioner.openicf-azureadpowershell.json`

   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 [a]	Required [b]
*operation*ScriptFileName	String	C:/openidm/AD/ADCreate.ps1, C:/openidm/samples/scripted-powershell-with-azure-ad/azureADScripts/AzureADDelete.ps1	No	Yes
The full path to the script that implements the corresponding OpenICF operation.				
VariablesPrefix	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.				
QueryFilterType	String	AdPsModule (for AD), Map (for Azure AD)	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: • Map - the query filter is a map • Ldap - the query filter is in LDAP search format, for example, "(cn=Joe)" • Native - the query filter is a native OpenICF query filter • AdPsModule - the query filter is compatible with the Active Directory PowerShell module, Get-ADUser Filter				
ReloadScriptOnExecution	Boolean	true	No	No
When true, the connector reloads the script from disk every time it is executed. This can be useful for debugging purposes. Set to false in production.				
UseInterpretersPool	Boolean	true	No	No
If true, the connector leverages the PowerShell RunSpace Pool.				

Property	Type	Example	Encrypted [a]	Required [b]
`MaxInterpretersPoolSize`	`Integer`	`5`	No	No
The maximum size of the interpreter pool.				
`MinInterpretersPoolSize`	`Integer`	`1`	No	No
The minimum size of the interpreter pool.				
`PoolCleanupInterval`	`Double`	`60`	No	No
Specifies the interval (in minutes) at which unused interpreter instances are discarded. To avoid cleaning up unused interpreter instances, set this property to `0`.				
`SubstituteUidAndNameInQueryFilter`	`Boolean`	`true`	No	No
Specifies whether the `__UID__` and `__NAME__` should be replaced by the value defined in the `NameAttributeName` and `UidAttributeName` in the query filter.				
`UidAttributeName`	`String`	`ObjectGUID` (AD), `ObjectId` (AzureAD)	No	No
The attribute on the resource that contains the object `__UID__`				
`NameAttributeName`	`String`	`DistinguishedNam` (AD), `UserPrincipalNam` (AzureAD)	No	No
The attribute on the resource that contains the object `__NAME__`				
`PsModulesToImport`	`Array`	`[ "ActiveDirecto , "C:/openidm /samples/ scripted- powershell- with-ad/tools /ADSISearch .psm1" ],` (AD), `["MSOnline"]` (AzureAD)	No	No
An array of additional PowerShell modules that the connector must import				
`Host`	`String`	(AD), (AzureAD)	No	Yes
The host name or IP address of the resource (Active Directory or Azure AD)				
`Port`	`Integer`	null	No	Yes
The port number on which the remote resource listens for connections				
`Login`	`String`	""	No	Yes
The user account in the remote resource that is used for the connection				
`Password`	`String`	null	Encrypted	Yes
The password of the user account that is used for the connection				

Property	Type	Example	Encrypted [a]	Required [b]
`CustomProperties`	`Array`	[ ]	No	No

An array of Strings to define custom configuration properties. Each property takes the format `"name=value"`. For example:

```
"configurationProperties" : {
...
    "CustomProperties" : ["baseContext = CN=Users,DC=example,DC=com" ],
...  },
```

The custom property can then be read from the PowerShell scripts as follows: `$base = $Connector .Configuration.PropertyBag.baseContext`

[a] Indicates whether the property value is considered confidential, and therefore encrypted in IDM.
[b] A list of operations in this column indicates that the property is required for those operations.

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" : "azureadpowershell",
  "enabled" : true,
  "config" : "config/provisioner.openicf/azureadpowershell",
  "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
}
```

The displayed output demonstrates a successful configuration of an Azure AD connector.

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 on an Azure AD system. 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/azureadpowershell/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 on a remote Azure AD instance:

```
$ 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/azureadpowershell/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/azureadpowershell/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. As an example, the following command deletes one user from an Azure AD deployment, based on their `_id`:

```
$ curl \
--header "X-OpenIDM-Username: openidm-admin"
 \
--header "X-OpenIDM-Password: openidm-admin"
 \
--request DELETE \
"http://localhost:8080/openidm/system/azureadpowershell/account/1d4c9276-6937-4d9e-9c60-67e8b4207f4e"
```

5.4.6. Running a Script on the Connector

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

`Configuration`

A handler to the connector's configuration object.

`Options`

A handler to the Operation Options.

`Operation`

The operation type that corresponds to the action (`RUNSCRIPTONCONNECTOR` in this case).

`Arguments`

A map of script arguments (this can be `null`).

The script can return any object that can be serialized by OpenICF, such as `Boolean`, `String`, `Array`, or `Dictionary`. If the object type cannot be serialized, such as `Hashtable`, the script fails with the error:

```
"error": "No serializer for class: System.Collections.Hashtable"
```

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

```
"systemActions" : [
    {
        "scriptId" : "MyScript",
        "actions" : [
            {
                "systemType" : ".*PowerShellConnector",
                "actionType" : "PowerShell",
                "actionFile" : "scripts/Myactionscript.ps1"
            }
        ]
    }
]
```

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

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request POST \
 "http://localhost:8080/openidm/system/powershell?
_action=script&scriptId=MyScript&param1=value1&param2=value2"
```

You can also call it through the IDM script engine, as follows:

```
openidm.action("/system/powershell","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:

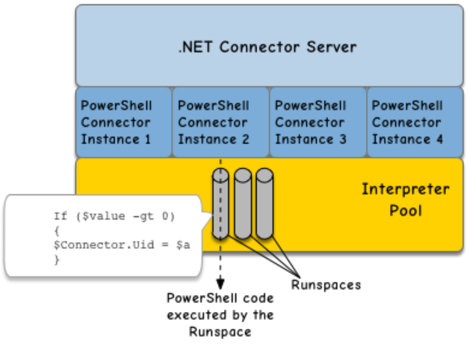
```
"systemActions" : [
    {
        "scriptId" : "SetExchange",
        "actions" : [
                                {
                                    "systemType" : ".*PowerShellConnector",
                                    "actionType" : "PowerShell",
                                    "actionSource" : "Set-Exchange $Connector.Arguments.dn"
                                }
                            ]
    }
]
```

5.5. Debugging Scripts Running in the PowerShell Connector

The PowerShell connector uses runspaces to execute the scripts for each action (create, update, search, and so on). A *runspace* is an instance of the Windows PowerShell interpreter within the PowerShell connector. A runspace essentially creates a new thread on an existing process. The connector can also use an interpreter pool and have several runspace instances running within the connector. This makes the connector more efficient under a heavy load. The interpreter pool can be shared between connector instances.

The following image shows how multiple connector instances use an interpreter pool with multiple runspaces:

PowerShell Connector and Runspaces


.NET Connector Server
PowerShell Connector Instance 1
PowerShell Connector Instance 2
PowerShell Connector Instance 3
PowerShell Connector Instance 4
Interpreter Pool
If ($value -gt 0) { $Connector.Uid = $a }
Runspaces
PowerShell code executed by the Runspace

PowerShell 5.0 includes several cmdlets related to runspace debugging. These cmdlets allow you to debug arbitrary runspaces, that is, runspaces other than the default PowerShell console or PowerShell ISE.

The examples shown in this section assume the following setup. Adjust the examples for your particular setup:

- You are using IDM 6 and version 1.4.5.0 of the PowerShell connector.

- IDM is running on a local UNIX host, distinct from the Windows host on which the PowerShell connector runs.
- You have already installed and configured version 1.5.20.0 of the .NET connector server on the remote Windows host. You have also installed and tested the PowerShell connector. See "Setting Up the PowerShell Connector" for more information.
- The remote Windows host includes the PowerShell module version 5.0 (available with the Windows Management Framework 5.0).
- You are using the PowerShell scripts and configuration provided with the sample described in "*Connecting to Azure AD With the PowerShell Connector*" in the *Samples Guide*.

Before you start, check that the interpreters pool is configured as follows in your PowerShell connector configuration (`provisioner.openicf-azureadpowershell.json`):

```
"configurationProperties" : {
...
    "UseInterpretersPool" : true,
    "MinInterpretersPoolSize" : 1,
    "MaxInterpretersPoolSize" : 1,
...
},
```

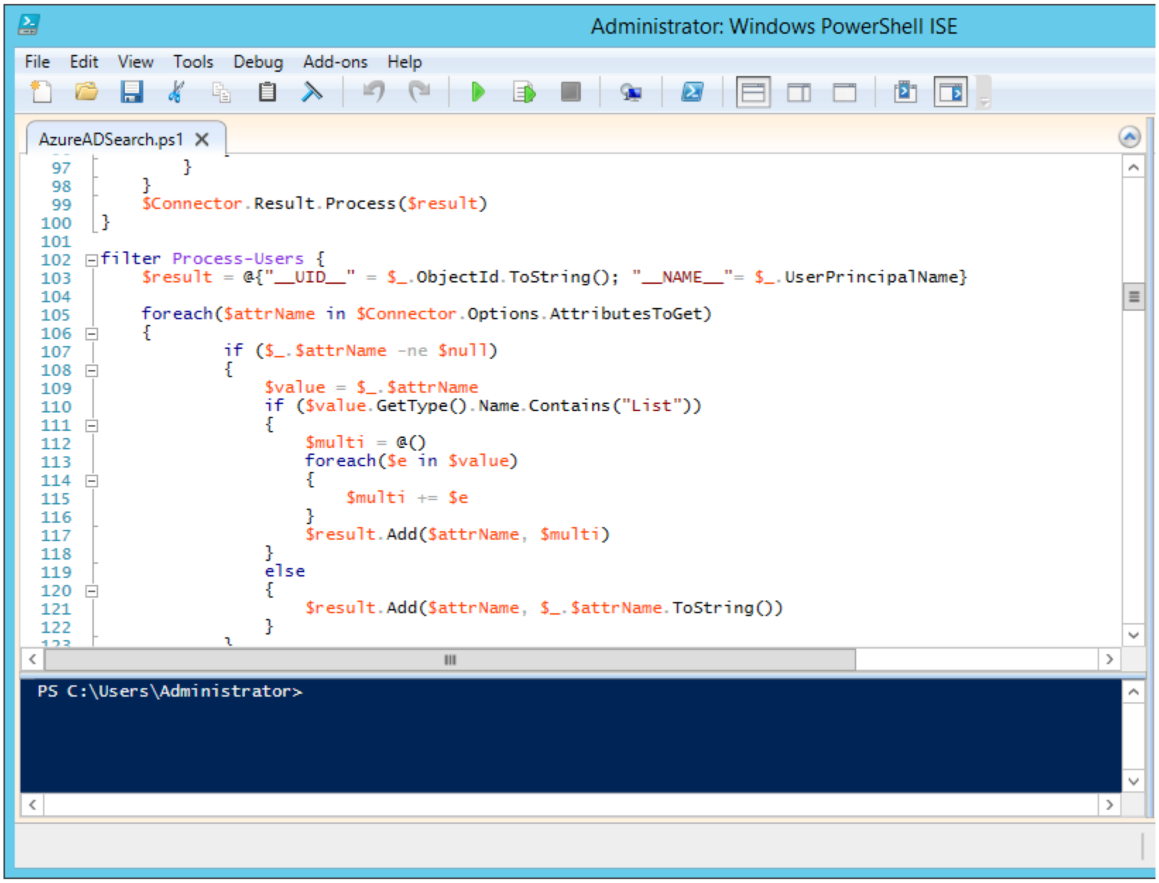
This configuration will make debugging easier.

Then follow these steps to set up debugging:

1. Connect the PowerShell ISE to the connector server.

   a. Start the PowerShell ISE in Administrator mode.

   b. From the PowerShell ISE, open the `AzureADSearch.ps1` sample script:


Administrator: Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
AzureADSearch.ps1

```
            }
        }
        $Connector.Result.Process($result)
}

filter Process-Users {
        $result = @{"__UID__" = $_.ObjectId.ToString(); "__NAME__"= $_.UserPrincipalName}

        foreach($attrName in $Connector.Options.AttributesToGet)
        {
                if ($_.$attrName -ne $null)
                {
                    $value = $_.$attrName
                    if ($value.GetType().Name.Contains("List"))
                    {
                        $multi = @()
                        foreach($e in $value)
                        {
                            $multi += $e
                        }
                        $result.Add($attrName, $multi)
                    }
                    else
                    {
                        $result.Add($attrName, $_.$attrName.ToString())
                    }
```

```
PS C:\Users\Administrator>
```

c. In the PowerShell ISE console, use the `Get-Process` cmdlet to obtain the process identifier of the Connector Server service:

```
PS C:\Program Files (x86)\ForgeRock\OpenICF> Get-Process -Name "connector*"

Handles  NPM(K)    PM(K)      WS(K)     CPU(s)     Id  SI ProcessName
-------  ------    -----      -----     ------     --  -- -----------
    576      56    79748      89484       1.16   1628   1 ConnectorServerService
```

d. Use the `EnterPSHostProcess` cmdlet to connect to the Connector Server service, specifying its process identifier:

```
PS C:\Program Files (x86)\ForgeRock\OpenICF> Enter-PSHostProcess -Id 1628

[Process:1628]: PS C:\Program Files (x86)\ForgeRock\OpenICF>
```

e. Use the `Get-Runspace` to look at the PowerShell interpreter pool that is embedded by the connector:

```
[Process:1628]: PS C:\Program Files (x86)\ForgeRock\OpenICF> Get-Runspace

 Id Name            ComputerName    Type          State         Availability
 -- ----            ------------    ----          -----         ------------
  1 Runspace1       localhost       Local         Opened        Available
  2 RemoteHost      localhost       Local         Opened        Busy
```

Note that there is one Runspace (`Runspace1`) open. Because you have set the maximum pool size to 1, that number should not increase.

2. Enter Debug mode and call a script:

   a. First, use the `Debug-RunSpace` cmdlet to attach to the available Runspace:

```
[Process:1628]: PS C:\Program Files (x86)\ForgeRock\OpenICF> Debug-Runspace  -Id 1
Debugging Runspace: Runspace1
To end the debugging session type the 'Detach' command at the debugger prompt, or type 'Ctrl+C'
 otherwise.
```

   b. Run any action over the IDM REST interface.

   The following example queries all user accounts:

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/system/azureadpowershell/account?_queryFilter=true"
Stopped at: $proceed = $TRUE
[DBG]: [Process:1628]: [Runspace1]: PS C:\Program Files (x86)\ForgeRock\OpenICF>>
```

   The output indicates that the debugger has been triggered and that it is stopped at the first line of code.

   c. Type `h` to list the debugging commands:

```
[DBG]: [Process:1628]: [Runspace1]: PS C:\Program Files (x86)\ForgeRock\OpenICF>> h

 s, stepInto         Single step (step into functions, scripts, etc.)
 v, stepOver         Step to next statement (step over functions, scripts, etc.)
 o, stepOut          Step out of the current function, script, etc.

 c, continue         Continue operation
 q, quit             Stop operation and exit the debugger
 d, detach           Continue operation and detach the debugger.

 k, Get-PSCallStack  Display call stack

 l, list             List source code for the current script.
                     Use "list" to start from the current line, "list <m>"
                     to start from line <m>, and "list <m> <n>" to list <n>
                     lines starting from line <m>

 <enter>             Repeat last command if it was stepInto, stepOver or list

 ?, h                displays this help message.


For instructions about how to customize your debugger prompt, type "help about_prompt".
```

d. To inspect the variables injected into the scripts by the connector, type `$Connector`:

```
[DBG]: [Process:1628]: [Runspace1]: PS C:\Program Files (x86)\ForgeRock\OpenICF>> $Connector

Key           Value
---           -----
Result        Org.ForgeRock.OpenICF.Connectors.MsPowerShell.MsPowerShellSearchResults
Action        SEARCH
Operation     SEARCH
Configuration Org.ForgeRock.OpenICF.Connectors.MsPowerShell.MsPowerShellConfiguration
ObjectClass   ObjectClass: __ACCOUNT__
Options       OperationOptions: Org.IdentityConnectors.Common.ReadOnlyDictionary`2[System.String
,System.Object]
```

e. Because the `$Connector` variable is a hash table, you can use dotted notation to inspect the various items.

The following example inspects the connector configuration:

```
[DBG]: [Process:1628]: [Runspace1]: PS C:\Program Files (x86)\ForgeRock\OpenICF>> $Connector
.Configuration

AuthenticateScriptFileName        : C:/openidm/samples/scripted-powershell-with-ad/tools/
ADAuthenticate.ps1
CreateScriptFileName              : C:/openidm/samples/scripted-powershell-with-ad/tools/ADCreate
.ps1
DeleteScriptFileName              : C:/openidm/samples/scripted-powershell-with-ad/tools/ADDelete
.ps1
ResolveUsernameScriptFileName     : C:/openidm/samples/scripted-powershell-with-ad/tools/
ADResolveUsername.ps1
SchemaScriptFileName              : C:/openidm/samples/scripted-powershell-with-ad/tools/ADSchema
.ps1
SearchScriptFileName              : C:/openidm/samples/scripted-powershell-with-ad/tools/ADSearch
.ps1
SyncScriptFileName                : C:/openidm/samples/scripted-powershell-with-ad/tools/ADSync
.ps1
TestScriptFileName                : C:/openidm/samples/scripted-powershell-with-ad/tools/ADTest
.ps1
UpdateScriptFileName              : C:/openidm/samples/scripted-powershell-with-ad/tools/ADUpdate
.ps1
VariablesPrefix                   : Connector
QueryFilterType                   : AdPsModule
ReloadScriptOnExecution           : True
UseInterpretersPool               : True
SubstituteUidAndNameInQueryFilter : True
UidAttributeName                  : ObjectGUID
NameAttributeName                 : DistinguishedName
PsModulesToImport                 : {ActiveDirectory}
Host                              : 192.168.1.103
Port                              : 389
Login                             : CN=bjensen,CN=users,DC=example,DC=com
Password                          : Org.IdentityConnectors.Common.Security.GuardedString
MinInterpretersPoolSize           : 1
MaxInterpretersPoolSize           : 1
...
```

The following example inspects the Search query:

```
[DBG]: [Process:1628]: [Runspace1]: PS C:\Program Files (x86)\ForgeRock\OpenICF>> $Connector.Query

Key              Value
---              -----
Not              False
Operation   STARTSWITH
Left       DisplayName
Right           Sprint
...
```

3. Use the `s` and `v` commands to step over and step into your scripts:

```
[DBG]: [Process:1628]: [Runspace1]: PS C:\Program Files (x86)\ForgeRock\OpenICF>> s

Stopped at: $searchBase = $Connector.Configuration.PropertyBag.baseContext
[DBG]: [Process:1628]: [Runspace1]: PS C:\Program Files (x86)\ForgeRock\OpenICF>> v

 Stopped at: $attrsToGet = "*"
[DBG]: [Process:1628]: [Runspace1]: PS C:\Program Files (x86)\ForgeRock\OpenICF>>
```

Note

You cannot use breakpoints with the debugger because it is running in memory.

4. Use the `l` command to check where you are in the script:

```
[DBG]: [Process:1628]: [Runspace1]: PS C:\Program Files (x86)\ForgeRock\OpenICF>> l

   78:
   79:  # Always put code in try/catch statement and make sure exceptions are re-thrown to connector
   80:  try
   81:  {
   82:   $searchBase = $Connector.Configuration.PropertyBag.baseContext
   83:*  $attrsToGet = "*"
   84:   $filter = "*"
   85:
   86:   if ( $Connector.Query ) {$filter = $Connector.Query}
   87:
   88:   switch ($Connector.ObjectClass.Type)
   89:   {
   90:    "__ACCOUNT__"
   91:    {
   92:     Get-ADUser -Filter $filter -SearchBase $searchBase -Properties $attrsToGet | Process-
Results
   93:    }
```

5. Close the session.

   When the script has completed, you will see the following message in the debugger console:

```
Command or script completed.
To end the debugging session type the 'Detach' command at the debugger prompt, or type 'Ctrl+C'
 otherwise.
```

   Type `Ctrl+C` to return to the process prompt or `exit` to exit the process and return to the console prompt.

   To debug the same script again, or to debug another script, you must call `Debug-Runspace` again before you send the query over the IDM REST interface.

Chapter 6
Groovy Connector Toolkit

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

The Groovy Connector Toolkit is not a complete connector in the traditional sense. Rather, it is a framework within which you must write your own Groovy scripts to address the requirements of your implementation.

6.1. Configuring Scripted Groovy Connectors

The Groovy Connector Toolkit is bundled in the JAR `openidm/connectors/groovy-connector-1.5.20.0.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
```

MS 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 [a]	Required [b]
`customSensitiveConfiguration`	`GuardedString`	`null`	Yes	No
Custom Sensitive Configuration script for Groovy ConfigSlurper				
`customConfiguration`	`String`	`null`		No
Custom Configuration script for Groovy ConfigSlurper				

Property	Type	Default	Encrypted [a]	Required [b]

[a] Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.
[b] A list of operations in this column indicates that the property is required for those operations.

6.3.1.2. Operation Script Files Properties

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

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

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

6.3.1.3. Groovy Engine configuration Properties

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

Property	Type	Default	Encrypted [a]	Required [b]
If set to true recompilation is enabled				

[a] Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.
[b] A list of operations in this column indicates that the property is required for those operations.

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 [a]	Required [b]
`customSensitiveConfiguration`	`GuardedString`	`null`	Yes	No
Custom Sensitive Configuration script for Groovy ConfigSlurper				
`customConfiguration`	`String`	`null`		No
Custom Configuration script for Groovy ConfigSlurper				

[a] Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.
[b] A list of operations in this column indicates that the property is required for those operations.

6.3.2.2. Operation Script Files Properties

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

Property	Type	Default	Encrypted [a]	Required [b]
The name of the file used to perform the RESOLVE_USERNAME operation.				
`searchScriptFileName`	`String`	`null`		Get Search
The name of the file used to perform the SEARCH operation.				
`updateScriptFileName`	`String`	`null`		Update
The name of the file used to perform the UPDATE operation.				
`schemaScriptFileName`	`String`	`null`		Schema
The name of the file used to perform the SCHEMA operation.				
`testScriptFileName`	`String`	`null`		Test
The name of the file used to perform the TEST operation.				
`syncScriptFileName`	`String`	`null`		Sync
The name of the file used to perform the SYNC operation.				

[a] Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.
[b] A list of operations in this column indicates that the property is required for those operations.

6.3.2.3. Groovy Engine configuration Properties

Property	Type	Default	Encrypted [a]	Required [b]
`targetDirectory`	`File`	`null`		No
Directory into which to write classes.				
`warningLevel`	`int`	`1`		No
Warning Level of the compiler				
`scriptExtensions`	`String[]`	`['groovy']`		No
Gets the extensions used to find groovy files				
`minimumRecompilationInterval`	`int`	`100`		No
Sets the minimum of time after a script can be recompiled.				
`scriptBaseClass`	`String`	`null`		No
Base class name for scripts (must derive from Script)				
`scriptRoots`	`String[]`	`null`		Yes
The root folder to load the scripts from. If the value is null or empty the classpath value is used.				
`tolerance`	`int`	`10`		No

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

[a] Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.
[b] A list of operations in this column indicates that the property is required for those operations.

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 enables you to interact with any REST API, using Groovy scripts for the ICF operations.

7.1. Configuring the Scripted REST Connector

The Scripted REST Connector is bundled in the JAR `openidm/connectors/scriptedrest-connector-1.5.20.0.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 [a]	Required [b]
`customSensitiveConfiguration`	`GuardedString`	`null`	Yes	No
Custom Sensitive Configuration script for Groovy ConfigSlurper				
`customConfiguration`	`String`	`null`		No
Custom Configuration script for Groovy ConfigSlurper				

[a] Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.
[b] A list of operations in this column indicates that the property is required for those operations.

7.4.1.2. Operation Script Files Properties

Property	Type	Default	Encrypted [a]	Required [b]
`createScriptFileName`	`String`	`null`		Create
The name of the file used to perform the CREATE operation.				
`customizerScriptFileName`	`String`	`null`		No
The script used to customize some function of the connector. Read the documentation for more details.				
`authenticateScriptFileName`	`String`	`null`		Authenticate
The name of the file used to perform the AUTHENTICATE operation.				

Property	Type	Default	Encrypted [a]	Required [b]
`scriptOnResourceScriptFileName`	`String`	`null`		Script On Resource
The name of the file used to perform the RUNSCRIPTONRESOURCE operation.				
`deleteScriptFileName`	`String`	`null`		Delete
The name of the file used to perform the DELETE operation.				
`resolveUsernameScriptFileName`	`String`	`null`		Resolve Username
The name of the file used to perform the RESOLVE_USERNAME operation.				
`searchScriptFileName`	`String`	`null`		Get Search
The name of the file used to perform the SEARCH operation.				
`updateScriptFileName`	`String`	`null`		Update
The name of the file used to perform the UPDATE operation.				
`schemaScriptFileName`	`String`	`null`		Schema
The name of the file used to perform the SCHEMA operation.				
`testScriptFileName`	`String`	`null`		Test
The name of the file used to perform the TEST operation.				
`syncScriptFileName`	`String`	`null`		Sync
The name of the file used to perform the SYNC operation.				

[a] Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.
[b] A list of operations in this column indicates that the property is required for those operations.

7.4.1.3. Groovy Engine configuration Properties

Property	Type	Default	Encrypted [a]	Required [b]
`targetDirectory`	`File`	`null`		No
Directory into which to write classes.				
`warningLevel`	`int`	`1`		No
Warning Level of the compiler				
`scriptExtensions`	`String[]`	`['groovy']`		No
Gets the extensions used to find groovy files				
`minimumRecompilationInterval`	`int`	`100`		No

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

[a] Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.
[b] A list of operations in this column indicates that the property is required for those operations.

7.4.1.4. Basic Configuration Properties Properties

Property	Type	Default	Encrypted [a]	Required [b]
`username`	`String`	`null`		No
The Remote user to authenticate with				
`password`	`GuardedString`	`null`	Yes	No
The Password to authenticate with				

Property	Type	Default	Encrypted [a]	Required [b]
`serviceAddress`	`URI`	`null`		Yes
The service URI (example: http://myservice.com/api)				
`proxyAddress`	`URI`	`null`		No
The optional Proxy server URI (example: http://myproxy:8080)				
`proxyUsername`	`String`	`null`		No
The username to authenticate with the proxy server				
`proxyPassword`	`GuardedString`	`null`	Yes	No
The password to authenticate with the proxy server				
`defaultAuthMethod`	`String`	`BASIC`		No
Authentication method used. Defaults to BASIC.				
`defaultContentType`	`String`	`application/json`		No
Default HTTP request content type. Defaults to JSON. Can be: TEXT, XML, HTML, URLENC, BINARY				
`defaultRequestHeaders`	`String[]`	`null`		No
Placeholder for default HTTP request headers.				
`OAuthTokenEndpoint`	`URI`	`null`		No
When using OAUTH, this property defines the endpoint where a new access token should be queried for (https://myserver.com/oauth2/token)				
`OAuthClientId`	`String`	`null`		No
The client identifier				
`OAuthClientSecret`	`GuardedString`	`null`	Yes	No
Secure client secret for OAUTH				
`OAuthRefreshToken`	`GuardedString`	`null`	Yes	No
The refresh token used to renew the access token for the refresh_token grant type				
`OAuthScope`	`String`	`null`		No
The optional scope				
`OAuthGrantType`	`String`	`CLIENT_CREDENTIAl`		No
The grant type to use. Can be CLIENT_CREDENTIALS (default) | REFRESH_TOKEN | AUTHORIZATION_CODE				

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

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

Chapter 8

Scripted SQL Connector

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

8.1. Configuring the Scripted SQL Connector

The Scripted SQL Connector is bundled in the JAR `openidm/connectors/scriptedsql-connector-1.5.20.0.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 OpenICF 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 Properties

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

Property	Type	Default	Encrypted [a]	Required [b]
`syncScriptFileName`	`String`	`null`		Sync
The name of the file used to perform the SYNC operation.				

[a] Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.
[b] A list of operations in this column indicates that the property is required for those operations.

8.3.1.2. Groovy Engine configuration Properties

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

Property	Type	Default	Encrypted [a]	Required [b]
If true, debugging code should be activated				
`classpath`	`String[]`	`[]`		No
Classpath for use during compilation.				
`verbose`	`boolean`	`false`		No
If true, the compiler should produce action information				

[a] Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.
[b] A list of operations in this column indicates that the property is required for those operations.

8.3.1.3. Configuration Properties

Property	Type	Default	Encrypted [a]	Required [b]
`password`	`String`	`null`	Yes	No
The connection password to be passed to our JDBC driver to establish a connection. Note that method DataSource.getConnection(username,password) by default will not use credentials passed into the method, but will use the ones configured here. See alternateUsernameAllowed property for more details.				
`customSensitiveConfiguration`	`GuardedString`	`null`	Yes	No
Custom Sensitive Configuration script for Groovy ConfigSlurper				
`customConfiguration`	`String`	`null`		No
Custom Configuration script for Groovy ConfigSlurper				
`connectionProperties`	`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.				
`propagateInterruptState`	`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.				
`useDisposableConnectionFacade`	`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.				
`defaultCatalog`	`String`	`null`		No
The default catalog of connections created by this pool.				
`validationInterval`	`long`	`3000`		No

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

Property	Type	Default	Encrypted [a]	Required [b]
The minimum number of established connections that should be kept in the pool at all times. The connection pool can shrink below this number if validation queries fail. Default value is derived from initialSize:10 (also see testWhileIdle)				
`defaultReadOnly`	`Boolean`	`null`		No
The default read-only state of connections created by this pool. If not set then the setReadOnly method will not be called. (Some drivers dont support read only mode, ex: Informix)				
`maxWait`	`int`	`30000`		No
The maximum number of milliseconds that the pool will wait (when there are no available connections) for a connection to be returned before throwing an exception. Default value is 30000 (30 seconds)				
`logValidationErrors`	`boolean`	`false`		No
Set this to true to log errors during the validation phase to the log file. If set to true, errors will be logged as SEVERE. Default value is false for backwards compatibility.				
`driverClassName`	`String`	`null`		No
The fully qualified Java class name of the JDBC driver to be used. The driver has to be accessible from the same classloader as tomcat-jdbc.jar				
`name`	`String`	`Tomcat Connection Pool[1 -1207228264]`		No
Returns the name of the connection pool. By default a JVM unique random name is assigned.				
`useStatementFacade`	`boolean`	`true`		No
If a statement proxy is set, wrap statements so that equals() and hashCode() methods can be called on closed statements.				
`initSQL`	`String`	`null`		No
A custom query to be run when a connection is first created. The default value is null.				
`validationQueryTimeout`	`int`	`-1`		No
The timeout in seconds before a connection validation queries fail. This works by calling java.test_sample.Statement.setQueryTimeout(seconds) on the statement that executes the validationQuery. The pool itself doesnt timeout the query, it is still up to the JDBC driver to enforce query timeouts. A value less than or equal to zero will disable this feature. The default value is -1.				
`validationQuery`	`String`	`null`		No
The SQL query that will be used to validate connections from this pool before returning them to the caller. If specified, this query does not have to return any data, it just cant throw a SQLException. The default value is null. Example values are SELECT 1(mysql), select 1 from dual(oracle), SELECT 1(MS Sql Server)				
`rollbackOnReturn`	`boolean`	`false`		No

Property	Type	Default	Encrypted [a]	Required [b]
If autoCommit==false then the pool can terminate the transaction by calling rollback on the connection as it is returned to the pool Default value is false.				
`alternateUsernameAllowed`	`boolean`	`false`		No
By default, the jdbc-pool will ignore the DataSource.getConnection(username,password) call, and simply return a previously pooled connection under the globally configured properties username and password, for performance reasons. The pool can however be configured to allow use of different credentials each time a connection is requested. To enable the functionality described in the DataSource.getConnection(username,password) call, simply set the property alternateUsernameAllowed to true. Should you request a connection with the credentials user1/password1 and the connection was previously connected using different user2/password2, 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.				
`dataSourceJNDI`	`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 dataSource attribute. Default value is null				
`validatorClassName`	`String`	`null`		No
The name of a class which implements the org.apache.tomcat.jdbc.pool.Validator interface and provides a no-arg constructor (may be implicit). If specified, the class will be used to create a Validator instance which is then used instead of any validation query to validate connections. The default value is null. An example value is com.mycompany.project.SimpleValidator.				
`suspectTimeout`	`int`	`0`		No
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.				
`useEquals`	`boolean`	`true`		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.				
`removeAbandonedTimeout`	`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.				
`defaultAutoCommit`	`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 setAutoCommit method will not be called.)				
`testOnConnect`	`boolean`	`false`		No

Property	Type	Default	Encrypted [a]	Required [b]
Validate the connection when connecting to the database for the first time. False by default. Set to true if you want to use the validationQuery as an init query.				
`jdbcInterceptors`	`String`	`null`		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.				
`initialSize`	`int`	`10`		No
The initial number of connections that are created when the pool is started. Default value is 10				
`defaultTransactionIsolation`	`int`	`-1`		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.				
`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

Property	Type	Default	Encrypted [a]	Required [b]
Time in milliseconds to keep this connection. When a connection is returned to the pool, the pool will check to see if the now - time-when-connected > maxAge has been reached, and if so, it closes the connection rather than returning it to the pool. The default value is 0, which implies that connections will be left open and no age check will be done upon returning the connection to the pool.				
`minEvictableIdleTimeMillis`	`int`	`60000`		No
The minimum amount of time an object may sit idle in the pool before it is eligible for eviction. The default value is 60000 (60 seconds).				
`timeBetweenEvictionRunsMillis`	`int`	`5000`		No
The number of milliseconds to sleep between runs of the idle connection validation/cleaner thread. This value should not be set under 1 second. It dictates how often we check for idle, abandoned connections, and how often we validate idle connections. The default value is 5000 (5 seconds).				
`testOnReturn`	`boolean`	`false`		No
The indication of whether objects will be validated before being returned to the pool. NOTE - for a true value to have any effect, the validationQuery parameter must be set to a non-null string. The default value is false.				
`useLock`	`boolean`	`false`		No
Use a lock when performing operations on the connection object. False by default. Set to true if you will use a separate background thread for idle and abandon checking (e.g. JMX clients). If the pool sweeper is enabled, a lock is used, regardless of this setting.				
`maxActive`	`int`	`100`		No
The maximum number of active connections that can be allocated from this pool at the same time. The default value is 100				
`username`	`String`	`null`		No
The connection username to be passed to our JDBC driver to establish a connection. Note that method DataSource.getConnection(username,password) by default will not use credentials passed into the method, but will use the ones configured here. See alternateUsernameAllowed property for more details.				

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

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

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 ForgeRock's BackStage site.
2. Copy the SAP connector JAR file (`sap-connector-1.5.0.0.jar`) to the `openidm/connectors` directory:

   ```
   $ cp ~/Downloads/sap-connector-1.5.0.0.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" : "1.5.0.0"
     },
     "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" : [ {
       "EMPLOYEENO" : "55099307",
       "SUBTYPE" : "0010",
       "VALIDEND" : "Fri Dec 31 00:00:00 CET 9999",
       "VALIDBEGIN" : "Fri March 18 00:00:00 CET 2016",
       "RECORDNR" : "000",
       "COMMTYPE" : "0010",
       "NAMEOFCOMMTYPE" : "E-mail",
       "ID" : "Maria.Gonzales@example.com"
     } ]
   ,
   ...
   ```

   By default, the connector sets the `VALIDBEGIN` date to the current date, and the `VALIDEND` date to the SAP "END" date (12/31/9999). You can specify different temporal constraints by including these properties in the JSON payload, with the format `YYYYMMDD`. For example:

```
{
  "EMAIL": {
     "ID": "maria.gonzales@example.com"
     "VALIDBEGIN": "20160401",
     "VALIDEND": "20161231"
  }
}
```

3. To change the value of an existing email account, provide a new value for the `ID`.

   The JSON payload of the change request must also include the `RECORDNR` attribute, as well as the `VALIDBEGIN` and `VALIDEND` dates, in SAP format (`YYYYMMDD`).

   The following example changes Maria Gonzales' email address to `maria.gonzales-admin@example.com`:

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --header "Content-Type: application/json" \
 --request PUT \
 --data '{
  "EMAIL": {
     "ID": "maria.gonzales-admin@example.com",
     "RECORDNR" : "000",
     "VALIDEND" : "99991231",
     "VALIDBEGIN" : "20000101"
  }
 }' \
 "http://localhost:8080/openidm/system/saphr/employee/55099307"
```

4. To change the temporal constraint (`VALIDEND` date) of the record, include the existing `VALIDEND` data in the JSON payload, and specify the new end date as a value of the `DELIMIT_DATE` attribute.

   The following example changes the end date of Maria Gonzale's new mail address to December 31st, 2016:

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --header "Content-Type: application/json" \
 --request PUT \
 --data '{
  "EMAIL": {
     "ID": "maria.gonzales-admin@example.com",
     "RECORDNR" : "000",
     "VALIDEND" : "99991231",
     "VALIDBEGIN" : "20000101",
     "DELIMIT_DATE": "20161231"
  }
 }' \
 "http://localhost:8080/openidm/system/saphr/employee/55099307"
```

5. To delete the email address of the record, send a PUT request with the current `RECORDNR`, `VALIDBEGIN`, and `VALIDEND` attributes, but without the `ID`.

The following request removes the email address from Maria Gonzales' record:

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --header "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 enables you to perform the following operations on SAP system user accounts:

- List all users
- List all activity groups (roles)
- Manage user profiles
- List all user companies
- Obtain a user's details
- Create a user
- Update a user
- Assign roles to a user
- Lock a user account
- Unlock a user account
- Delete a user account

Currently, the SAP connector cannot detect changes on the SAP system in real time. You must run a reconciliation operation to detect changes on the SAP system.

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": "1.5.0.0"
  },
  "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 OpenICF 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-issha, 1024}zrs3Zm/fX/l/KFGATp3kvOGlis3zLLiPmPVCDpJ9XF0=",
        "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
- `ADDTEL` - telephone numbers assigned to the user

In addition, the SAP connector supports the following OpenICF 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",
    "LANGUCPISO": "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",
        "TEL1_NUMBR": "33297603177",
        "E_MAIL": "katie.vaughan@example.com",
        "LANGU_CR_P": "E",
        "LANGUCPISO": "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",
      "BAPIPTEXT": "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",
      "BAPIPTEXT": "All SAP System authorizations",
      "BAPITYPE": "C",
      "BAPIAKTPS": "A"
    },
    {
      "BAPIPROF": "S_A.SYSTEM",
      "BAPIPTEXT": "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",
      "BAPIPTEXT": "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",
      "BAPIPTEXT": "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 [a]	Required [b]
`createScriptFileName`	`String`	`null`		Create
The name of the file used to perform the CREATE operation.				
`customizerScriptFileName`	`String`	`null`		No
The script used to customize some function of the connector. Read the documentation for more details.				
`resolveUsernameScriptFileName`	`String`	`null`		Resolve Username
The name of the file used to perform the RESOLVE_USERNAME operation.				
`updateScriptFileName`	`String`	`null`		Update
The name of the file used to perform the UPDATE operation.				
`schemaScriptFileName`	`String`	`null`		Schema
The name of the file used to perform the SCHEMA operation.				
`authenticateScriptFileName`	`String`	`null`		Authenticate
The name of the file used to perform the AUTHENTICATE operation.				
`scriptOnResourceScriptFileName`	`String`	`null`		Script On Resource
The name of the file used to perform the RUNSCRIPTONRESOURCE operation.				
`deleteScriptFileName`	`String`	`null`		Delete
The name of the file used to perform the DELETE operation.				
`searchScriptFileName`	`String`	`null`		Get Search
The name of the file used to perform the SEARCH operation.				
`testScriptFileName`	`String`	`null`		Test
The name of the file used to perform the TEST operation.				
`syncScriptFileName`	`String`	`null`		Sync
The name of the file used to perform the SYNC operation.				

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

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

9.7.2. Groovy Engine configuration Properties

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

Property	Type	Default	Encrypted [a]	Required [b]
If true, the compiler should produce action information				

[a] Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.
[b] A list of operations in this column indicates that the property is required for those operations.

9.7.3. Configuration Properties

Property	Type	Default	Encrypted [a]	Required [b]
`customSensitiveConfiguration`	`GuardedString`	`null`	Yes	No
Custom Sensitive Configuration script for Groovy ConfigSlurper				
`customConfiguration`	`String`	`null`		No
Custom Configuration script for Groovy ConfigSlurper				
`x509Cert`	`String`	`null`	Yes	No
Description is not available				

[a] Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.
[b] A list of operations in this column indicates that the property is required for those operations.

9.7.4. Basic Configuration Properties

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

Property	Type	Default	Encrypted [a]	Required [b]
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])				

[a] Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.
[b] A list of operations in this column indicates that the property is required for those operations.

9.7.5. SAP Jco Logs Configuration Properties

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

[a] Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.
[b] A list of operations in this column indicates that the property is required for those operations.

9.7.6. Advanced Configuration Properties

Property	Type	Default	Encrypted [a]	Required [b]
`msHost`	`String`	`null`		No
Specifies the host that the message server is running on				
`group`	`String`	`null`		No
Specifies the group name of the application servers, used when you log in to a logon group that uses load balancing				
`msServ`	`String`	`null`		No
Name of the service where the message server can be reached				

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

[a] Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.
[b] A list of operations in this column indicates that the property is required for those operations.

9.7.7. SAP Secure Network Connection Configuration Properties

Property	Type	Default	Encrypted [a]	Required [b]
`sncMode`	`String`	`0`		Yes
Flag used to activate SNC. Possible values are 0 (OFF) and 1 (ON).				
`sncQoP`	`String`	`3`		No
Specifies the security level to use for the connection. Possible values are 1 - Authentication only, 2 - Integrity protection, 3 - Privacy protection, 8 - Use the value from snc/data_protection/use on the application server, 9 - Use the value from snc/data_protection/max on the application server				
`sncLibrary`	`String`	`null`		No
Specifies the path to the external library that provides Secure Network Connection service. The default is the system-defined library as defined in the environment variable SNC_LIB.				
`sncPartnerName`	`String`	`null`		No
Specifies the AS ABAP SNC name, for example, "p:CN=ABC, O=MyCompany, C=US". You can find the application server SNC name in the profile parameter snc/identity/as on the AS ABAP.				
`sncMyName`	`String`	`null`		No
Specifies the connector SNC name, for example, "p:CN=OpenIDM, O=MyCompany, C=US". This parameter is optional, but you should set it to make sure that the correct SNC name is used for the connection.				
`sncSSO`	`String`	`0`		No
Specifies whether the connection should be configured for single sign-on (SSO). Possible values are 0 (OFF) and 1 (ON).				

[a] Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.
[b] A list of operations in this column indicates that the property is required for those operations.

9.7.8. JCo Connection Pool Configuration Properties

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

Property	Type	Default	Encrypted [a]	Required [b]
`expirationTime`	`String`	`60000`		No
Time in ms after that a free connection can be closed. Default is one minute.				
`maxGetTime`	`String`	`30000`		No
Maximum time in ms to wait for a connection, if the maximum allowed number of connections is allocated by the pool. Default is 30 seconds.				
`peakLimit`	`String`	`0`		No
Maximum number of active connections that can be created for a destination simultaneously. The default is 0 (unlimited).				
`expirationPeriod`	`String`	`60000`		No
Period in ms after that the destination checks the released connections for expiration. Default is one minute				

[a] Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.
[b] A list of operations in this column indicates that the property is required for those operations.

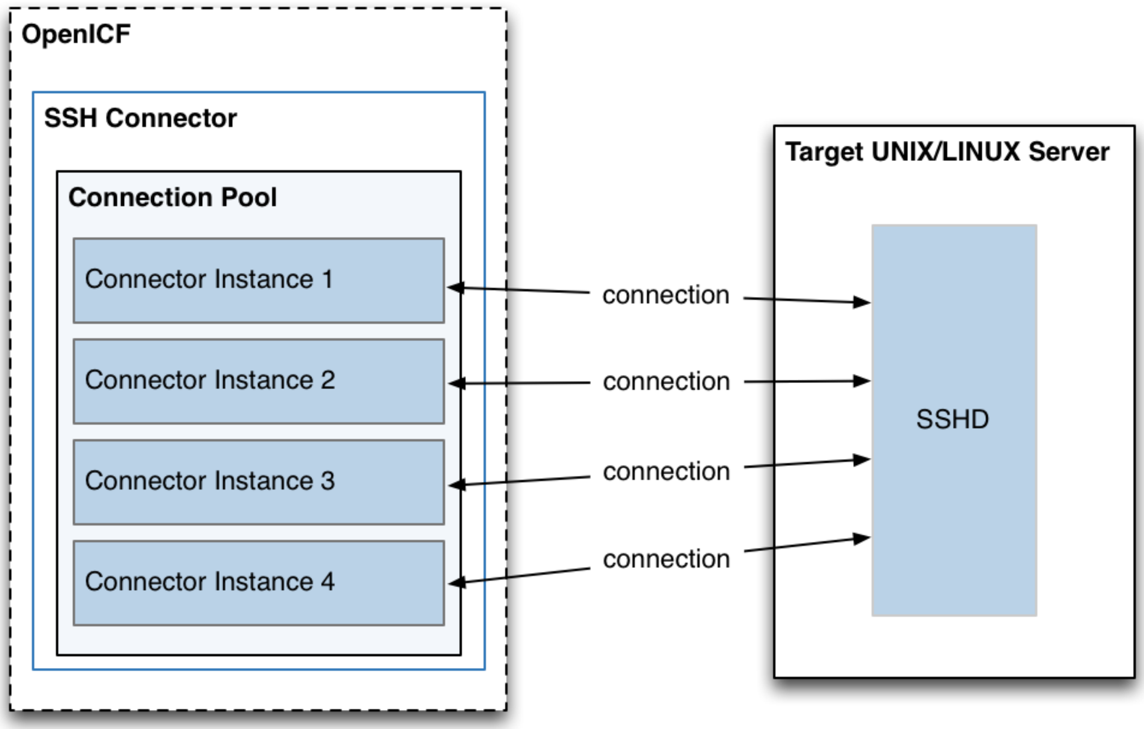
Chapter 10

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 enables you to interact with any SSH server, using Groovy scripts for the OpenICF 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-----",
              "MIIBugIBAAKBgQDcB0ztVMCFptpJhqlLNZSdN/5cDL3S7aOVy52Ae7vwwCqQPCQr",
              "6NyUk+wtkDr07NlYd3sg7a9hbsEnlYChsuX+/WUIvbOKdMfeqcQ+jKK26YdkTCGj",
              "g86dBj9JYhobSHDoQ9ov31pYN/cfW5BAZwkm9TdpEjHPvMIaOxx7GPGKWwIVALbD",
              "CEuf1yJk9UB7v0dmJS7bKkbxAoGARcbAuDP4rB6MsgAAkVwf+1sHXEiGPShYWrVV",
              "qBgCZ/S45ELqUuiaN/1N/nip/Cc/0SBPKqwl7o50CUg9GH9kTAjmXiwmbkwvtUv+",
              "Xjn5vCHS0w18yc3rGwyr2wj+D9KtDLFJ8+T5HmsbPoDQ3mIZ9xPmRQuRFfVMd9wr",
              "DY0Rs7cCgYAxjGjWDSKThowsvOUCiE0ySz6tWggHH3LTrS4Mfh2t0tnbUfrXq2cw",
              "3CN+T6brgnpYbyX5XI17p859C+cw90MD8N6vvBxaN8QMDRFk+hHNUeSy8gXeem9x",
              "O0vdIxCgKvA4dh5nSVb5VGKENEGNEHRlYxEPzbqlPa/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 OpenICF 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 OpenICF 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 [a]	Required [b]
`customSensitiveConfiguration`	`GuardedString`	`null`	Yes	No
Description is not available				
`createScriptFileName`	`String`	`null`		Create
Description is not available				
`targetDirectory`	`File`	`null`		No
Description is not available				
`customizerScriptFileName`	`String`	`null`		No
Description is not available				
`warningLevel`	`int`	`1`		No
Description is not available				
`authenticateScriptFileName`	`String`	`null`		Authenticate
Description is not available				
`scriptExtensions`	`String[]`	`['groovy']`		No
Description is not available				
`scriptOnResourceScriptFileName`	`String`	`null`		Script On Resource

Property	Type	Default	Encrypted [a]	Required [b]
Description is not available				
`minimumRecompilationInterval`	`int`	`100`		No
Description is not available				
`deleteScriptFileName`	`String`	`null`		Delete
Description is not available				
`scriptBaseClass`	`String`	`null`		No
Description is not available				
`scriptRoots`	`String[]`	`null`		Yes
Description is not available				
`customConfiguration`	`String`	`null`		No
Description is not available				
`resolveUsernameScriptFileName`	`String`	`null`		Resolve Username
Description is not available				
`searchScriptFileName`	`String`	`null`		Get Search
Description is not available				
`tolerance`	`int`	`10`		No
Description is not available				
`updateScriptFileName`	`String`	`null`		Update
Description is not available				
`debug`	`boolean`	`false`		No
Description is not available				
`classpath`	`String[]`	`[]`		No
Description is not available				
`disabledGlobalASTTransformations`	`String[]`	`null`		No
Description is not available				
`schemaScriptFileName`	`String`	`null`		Schema
Description is not available				

Property	Type	Default	Encrypted [a]	Required [b]
`verbose`	`boolean`	`false`		No
Description is not available				
`testScriptFileName`	`String`	`null`		Test
Description is not available				
`sourceEncoding`	`String`	`UTF-8`		No
Description is not available				
`syncScriptFileName`	`String`	`null`		Sync
Description is not available				
`recompileGroovySource`	`boolean`	`false`		No
Description is not available				

[a] Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.
[b] A list of operations in this column indicates that the property is required for those operations.

10.4.2. Basic Configuration Properties Properties

Property	Type	Default	Encrypted [a]	Required [b]
`host`	`String`	`null`		Yes
The hostname to connect to				
`port`	`int`	`22`		Yes
TCP port to use (defaults to 22)				
`user`	`String`	`null`		Yes
The user name used to login to remote server				
`password`	`GuardedString`	`null`	Yes	No
The password used to login to remote server				
`passphrase`	`GuardedString`	`null`	Yes	No
The passphrase used to read the private key when using Public Key authentication				
`privateKey`	`String[]`	`[]`	Yes	No
The base 64 encoded value (PEM) of the private key used for Public Key authentication				
`authenticationType`	`String`	`PASSWORD`		Yes
Defines which authentication type should be use: PASSWORD or PUBKEY (defaults to PASSWORD)				

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

[a] Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.
[b] A list of operations in this column indicates that the property is required for those operations.

Chapter 11 Google Apps Connector

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

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

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": "[1.4.0.0,2.0.0.0)"
},
```

The following excerpt shows the required configuration properties:

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

These configuration properties are fairly straightforward:

`domain`

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

`clientId`

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

`clientSecret`

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

refreshToken

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

For a sample Google Apps configuration that includes OAuth 2-based entries for `configurationProperties`, see "*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 Properties

Property	Type	Default	Encrypted [a]	Required [b]
`domain`	`String`	`null`		Yes
Internet domain name. See https://support.google.com/a/answer/177483?hl=en				
`clientId`	`String`	`null`		Yes
Client identifier issued to the client during the registration process.				
`clientSecret`	`GuardedString`	`null`	Yes	Yes
Client secret issued to the client during the registration process.				
`refreshToken`	`GuardedString`	`null`	Yes	Yes
The refresh token allows you to get a new access token that is good for another hour. Refresh tokens never expire, they can only be revoked by the user or programatically by your app.				
`proxyHost`	`String`	`null`		Yes

Property	Type	Default	Encrypted [a]	Required [b]
Defines an HTTP proxy host to use with the connection (example: "myproxy.home.com").				
`proxyPort`	`int`	`8080`		Yes
Defines an HTTP proxy port to use with the connection (defaults to 8080).				
`validateCertificate`	`boolean`	`true`		Yes
Validate the server certificate from the local truststore (defaults to true).				
`usersMaxResults`	`int`	`100`		No
Maximum number of Users to return. Acceptable values are 1 to 500, inclusive.				
`groupsMaxResults`	`int`	`200`		No
Maximum number of Groups to return. Acceptable values are 1 to 200, inclusive.				
`membersMaxResults`	`int`	`200`		No
Maximum number of Members to return. Acceptable values are greater than 1				
`listProductMaxResults`	`long`	`100`		No
Maximum number of Licenses to return. Acceptable values are 1 to 1000, inclusive.				
`listProductAndSkuMaxResults`	`long`	`100`		No
Maximum number of Licenses to return. Acceptable values are 1 to 1000, inclusive.				

[a] Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.
[b] A list of operations in this column indicates that the property is required for those operations.

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

OpenICF 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 OpenICF 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-1.5.20.0.jar`
- `/path/to/openidm/lib/expect4j-<version>.jar`
- `/path/to/openidm/lib/jsch-<version>.jar`

The Kerberos connector enables you to manage Kerberos user principals from IDM. The connector is provided in `/path/to/openidm/connectors/kerberos-connector-1.5.20.0.jar` and bundles a number of Groovy scripts to interact with a Kerberos admin server. Users of the Kerberos connector are not expected to edit the bundled Groovy scripts. The bundled scripts use the `kadmin` utility to communicate with the Kerberos server.

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

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

12.1. Kerberos Connector Schema

The Kerberos connector can only be used to manage the Kerberos `principal` object type (which maps to the OpenICF `__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 promtps, see this article.

`sudoCommand`

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

`echoOff`

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

`terminalType`

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

Default: `vt102`

`setLocale`

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

Default: `false`

locale

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

Default: `en_US.utf8`

`connectionTimeout`

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

Default: `5000`

`expectTimeout`

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

Default: `5000`

`authenticationType`

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

Default: `PASSWORD`

`throwOperationTimeoutException`

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

Default: `true`

`scriptRoots`

The path to the Groovy scripts that will perform the OpenICF 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-1.5.20.0.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 OpenICF 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 [a]	Required [b]
`customSensitiveConfiguration`	`GuardedString`	`null`	Yes	No
Description is not available				
`createScriptFileName`	`String`	`null`		Create
Description is not available				
`targetDirectory`	`File`	`null`		No
Description is not available				
`customizerScriptFileName`	`String`	`null`		No
Description is not available				
`warningLevel`	`int`	`1`		No
Description is not available				
`authenticateScriptFileName`	`String`	`null`		Authenticate
Description is not available				
`scriptExtensions`	`String[]`	`['groovy']`		No
Description is not available				
`scriptOnResourceScriptFileName`	`String`	`null`		Script On Resource
Description is not available				
`minimumRecompilationInterval`	`int`	`100`		No
Description is not available				
`deleteScriptFileName`	`String`	`null`		Delete
Description is not available				
`scriptBaseClass`	`String`	`null`		No
Description is not available				
`scriptRoots`	`String[]`	`null`		Yes
Description is not available				
`customConfiguration`	`String`	`null`		No
Description is not available				

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

Property	Type	Default	Encrypted [a]	Required [b]
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 [a]	Required [b]
Description is not available				
`throwOperationTimeoutException`	`boolean`	`true`		Yes
Description is not available				
`promptReadyTimeout`	`long`	`20`		No
Description is not available				

[a] Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.
[b] A list of operations in this column indicates that the property is required for those operations.

Chapter 13
Salesforce Connector

IDM provides a Salesforce connector, along with a sample connector configuration. The Salesforce connector enables provisioning, reconciliation, and synchronization between Salesforce and the IDM repository.

The Salesforce Connector is not an OpenICF connector, but a separate IDM module, based on the ForgeRock Common Resource API.

To use this connector, you need a Salesforce account, and a Connected App that has OAuth enabled, which will allow you to retrieve the required consumer key and consumer secret.

For additional instructions, and a sample Salesforce configuration, see "*Synchronizing Users Between Salesforce and IDM*" in the *Samples Guide*.

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 enables you to interact with leads in a Marketo database, using Groovy scripts for the OpenICF 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" : "1.5.20.0",
        "connectorName" : "org.forgerock.openicf.connectors.marketo.MarketoConnector"
    },
    ...
    "configurationProperties" : {
        "instance" : "<INSTANCE_FQDN>",
        "clientId" : "<CLIENT_ID>",
        "clientSecret" : "<CLIENT_SECRET>",
        "leadFields" : null,
        "partitionName" : null,
        "listName" : "<LEAD_LIST_NAME>",
    ...
```

instance

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

clientId

Locate the client ID in the details of your Marketo service `LaunchPoint`.

clientSecret

Locate the client secret in the details of your Marketo service `LaunchPoint`.

listName

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

You can also configure the Marketo connector through the Admin UI. Select Configure > Connectors > New Connector and select Marketo Connector - 1.5.20.0 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": "1.5.20.0"
    },
    "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 enables you to reconcile IDM users (including managed users and users who have registered through a social identity provider) with a Marketo leads database. To set up reconciliation to a Marketo database, copy the following sample mapping file to your project's `conf` directory:

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

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

If you have an existing mapping configuration (`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 Self-Service 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 [a]	Required [b]
`customSensitiveConfiguration`	`GuardedString`	`null`	Yes	No
Custom Sensitive Configuration script for Groovy ConfigSlurper				
`customConfiguration`	`String`	`null`		No
Custom Configuration script for Groovy ConfigSlurper				
`instance`	`String`	`null`		Yes
The Marketo-assigned FQDN for your instance				
`clientId`	`String`	`null`		Yes
Your OAuth2 client ID				
`clientSecret`	`GuardedString`	`null`	Yes	Yes
Your OAuth2 client secret				
`leadFields`	`String`	`null`		No
Comma-delimited list of lead fields to fetch; Leave empty for default set				
`partitionName`	`String`	`null`		No
Name of the partition in which to create and update leads; May be left empty				
`listName`	`String`	`null`		Yes
Name of the Marketo static list the connector will use to manage leads				
`accessToken`	`String`	`null`		Yes
The access token for the application				
`tokenExpiration`	`Long`	`null`		Yes
The expiration token for the application				

[a] Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.
[b] A list of operations in this column indicates that the property is required for those operations.

14.4.2. Operation Script Files Properties

Property	Type	Default	Encrypted [a]	Required [b]
`createScriptFileName`	`String`	`CreateMarketo.groovy`		Create
The name of the file used to perform the CREATE operation.				
`customizerScriptFileName`	`String`	`null`		No

Property	Type	Default	Encrypted [a]	Required [b]
The script used to customize some function of the connector. Read the documentation for more details.				
`authenticateScriptFileName`	`String`	`null`		Authenticate
The name of the file used to perform the AUTHENTICATE operation.				
`scriptOnResourceScriptFileName`	`String`	`null`		Script On Resource
The name of the file used to perform the RUNSCRIPTONRESOURCE operation.				
`deleteScriptFileName`	`String`	`DeleteMarketo.groovy`		Delete
The name of the file used to perform the DELETE operation.				
`resolveUsernameScriptFileName`	`String`	`null`		Resolve Username
The name of the file used to perform the RESOLVE_USERNAME operation.				
`searchScriptFileName`	`String`	`SearchMarketo.groovy`		Get Search
The name of the file used to perform the SEARCH operation.				
`updateScriptFileName`	`String`	`UpdateMarketo.groovy`		Update
The name of the file used to perform the UPDATE operation.				
`schemaScriptFileName`	`String`	`SchemaMarketo.groovy`		Schema
The name of the file used to perform the SCHEMA operation.				
`testScriptFileName`	`String`	`TestMarketo.groovy`		Test
The name of the file used to perform the TEST operation.				
`syncScriptFileName`	`String`	`null`		Sync
The name of the file used to perform the SYNC operation.				

[a] Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.
[b] A list of operations in this column indicates that the property is required for those operations.

14.4.3. Groovy Engine configuration Properties

Property	Type	Default	Encrypted [a]	Required [b]
`targetDirectory`	`File`	`null`		No

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

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

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

Chapter 15
Active Directory Connector

The Active Directory connector is a legacy connector, written in C# for the .NET platform. OpenICF 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 ForgeRock's BackStage 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:

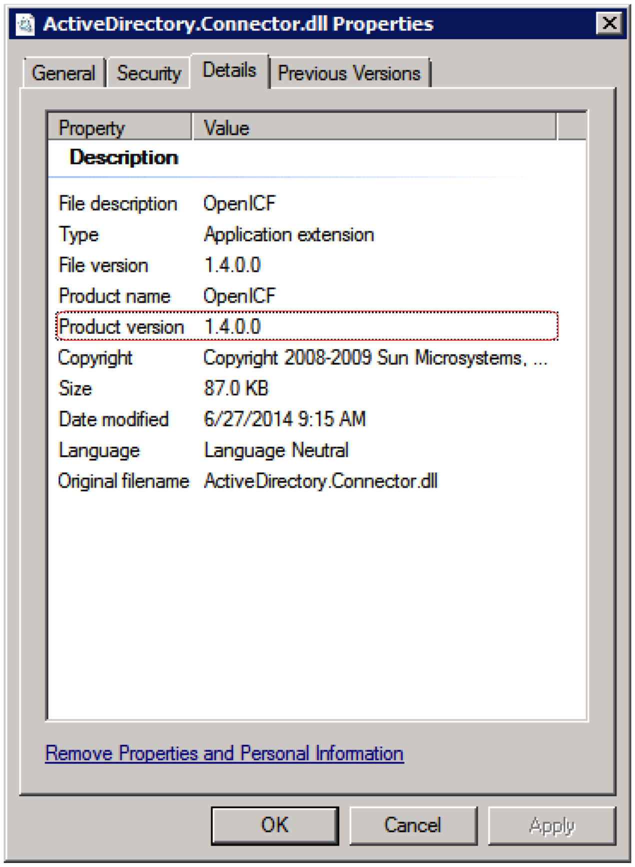
```
$ cd /path/to/openidm
$ cp samples/example-configurations/provisioners/provisioner.openicf-ad.json project-dir/conf/
```

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.


ActiveDirectory.Connector.dll Properties
General
Security
Details
Previous Versions
Remove Properties and Personal Information
OK
Cancel
Apply

Property	Value
Description	
File description	OpenICF
Type	Application extension
File version	1.4.0.0
Product name	OpenICF
Product version	1.4.0.0
Copyright	Copyright 2008-2009 Sun Microsystems, ...
Size	87.0 KB
Date modified	6/27/2014 9:15 AM
Language	Language Neutral
Original filename	ActiveDirectory.Connector.dll

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"     : "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 MS 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.

The Office 365 connector is available from ForgeRock's BackStage site. If you want to use this connector in production, contact ForgeRock Support.

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 [a]	Required [b]
`tenant`	`String`	`null`		Yes
Name of your Office365 tenant				
`clientId`	`String`	`null`		Yes

Property	Type	Default	Encrypted [a]	Required [b]
This value is provided by Office365				
`clientSecret`	`GuardedString`	`null`	Yes	Yes
This value is provided by Office365				
`accessToken`	`String`	`null`		Yes
This value is provided by Office365				
`tokenExpiration`	`Long`	`null`		No
This value is provided by Office365				
`refreshToken`	`String`	`null`		Yes
This value is provided by Office365				

[a] Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.
[b] A list of operations in this column indicates that the property is required for those operations.

16.3.2. Office365 AzureAD Configuration Properties Properties

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

[a] Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.
[b] A list of operations in this column indicates that the property is required for those operations.

Chapter 17 SCIM Connector

The SCIM connector is based on the Simple Cloud Identity Management (SCIM) protocol and enables you to manage user and group accounts on any SCIM-compliant resource provider, such as Slack or Facebook. The SCIM connector implements both 1.1 and 2.0 endpoints. The SCIM connector is bundled with IDM in the `connectors/` directory.

The SCIM connector uses the Apache HTTP client, which leverages the HTTP client connection pool, not the ICF connector pool.

Configure the SCIM Connector Using the Filesystem

1. Download provisioner.openicf-scim.json to your project's `conf/` directory.
2. Edit `conf/provisioner.openicf-scim.json`, as necessary. The following changes are required:
   - `"enabled" : true`
   - To specify the connection details to the SCIM resource provider, set the `configurationProperties`. The required properties vary, based on the `authenticationMethod`:

     `OAUTH`

     The minimum required properties are `grantType`, `SCIMEndpoint`, `tokenEndpoint`, `clientId`, and `clientSecret`.

     `BASIC`

     The minimum required properties are `user` and `password`.

     `TOKEN`

     The minimum required property is `authToken`.

   Sample Configuration Using OAUTH:

```
"configurationProperties" : {
    "SCIMEndpoint" : "https://example.com/scim",
    "SCIMVersion" : 1,
    "authenticationMethod" : "OAUTH",
    "user" : null,
    "password" : null,
    "tokenEndpoint" : "https://example.com/oauth2/token",
    "clientId" : "Kdvl...................j3fka",
    "clientSecret" : "xxxxxxxxxxxxxxxxxx",
    "acceptSelfSignedCertificates" : true,
    "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": "1.5.20.0"
    },
    "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 Properties

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

Property	Type	Default	Encrypted [a]	Required [b]
`grantType`	`String`	`null`		No
The OAuth2 grant type to use (client_credentials or refresh_token)				
`scope`	`String`	`null`		No
The OAuth2 scope to use.				
`acceptSelfSignedCertificates`	`boolean`	`false`		Yes
To be used for debug/test purposes. To be avoided in production. Defaults to false.				
`disableHostNameVerifier`	`boolean`	`false`		Yes
To be used for debug/test purposes. To be avoided in production. Defaults to false.				
`disableHttpCompression`	`boolean`	`false`		Yes
Content compression is enabled by default. Set this property to true to disable it. Defaults to false.				
`clientCertAlias`	`String`	`null`		Yes
If TLS Mutual Auth is needed, set this to the certificate alias from the keystore.				
`clientCertPassword`	`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.				
`maximumConnections`	`Integer`	`10`		Yes
Defines the max size of the http connection pool used. Defaults to 10.				
`httpProxyHost`	`String`	`null`		Yes
Defines the Hostname if an HTTP proxy is used between the connector and the SCIM service provider. Defaults to null.				
`httpProxyPort`	`Integer`	`null`		Yes
Defines the Port if an HTTP proxy is used between the connector and the SCIM service provider. Defaults to null.				
`httpProxyUsername`	`String`	`null`		Yes
Defines Proxy Username if an HTTP proxy is used between the connector and the SCIM service provider. Defaults to null.				
`httpProxyPassword`	`GuardedString`	`null`	Yes	Yes
Defines Proxy Password if an HTTP proxy is used between the connector and the SCIM service provider. Defaults to null.				
`connectionTimeout`	`int`	`30`		No
Defines a timeout for the underlying http connection in seconds. Defaults to 30.				

Property	Type	Default	Encrypted [a]	Required [b]

[a] Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.
[b] A list of operations in this column indicates that the property is required for those operations.

Chapter 18
Adobe Marketing Cloud Connector

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

To use this connector, you need an Adobe ID.

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 \
    -srcalias 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 - 1.5.20.0 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-na1.adobelogin.com",
    "tenant" : "https://example.adobesandbox.com/",
    "apiKey" : "",
    "techAccId" : "example@techacct.adobe.com",
    "orgId" : "example@AdobeOrg",
    "clientSecret" : "CLIENT_SECRET",
    "privateKey" : "PRIVATE_KEY"
},
...
```

`endpoint`

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

`imsHost`

> The Adobe Identity Management System (IMS) host. `ims-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": "1.5.20.0"
    },
    "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 Properties

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

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

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

18.4.2. Adobe Integration Properties Properties

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

[a] Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.
[b] A list of operations in this column indicates that the property is required for those operations.

Chapter 19
Workday Connector

Workday is a multi-tenant Software-as-a-Service (SaaS) application. The Workday connector enables you to synchronize user accounts between IDM and Workday's cloud-based HR system.

The connector supports reconciliation of users and organizations from Workday to an IDM repository, liveSync of users from Workday to IDM, and updating users in a Workday system.

To use the connector, you need a Workday instance with the required permissions and a set of credentials to access the instance, including the username, password, tenant name, and host name.

This connector is bundled with IDM in the `connectors/` directory (`workday-connector-1.5.20.0.jar`).

19.1. 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".

19.2. 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": "1.5.20.0",
      "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
}
```

19.3. 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"
}
```

19.4. 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
}
```

19.5. 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.

19.6. 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.

19.7. Workday Connector Configuration

The Workday Connector has the following configurable properties.

19.7.1. Configuration Properties

Property	Type	Default	Encrypted [a]	Required [b]
`includeManagementChainDataForWorker`	`Boolean`	`true`		No
Description is not available				
`includeOrganizationsForWorkers`	`Boolean`	`true`		No
Description is not available				
`includePersonalInformationForWorker`	`Boolean`	`true`		No
Description is not available				
`excludeCostCentersForWorkers`	`Boolean`	`false`		No
Description is not available				
`excludeCustomOrganizationsForWorker`	`Boolean`	`true`		No
Description is not available				
`includeRolesForWorkers`	`Boolean`	`false`		No
Description is not available				
`includeStaffingRestrictionsDataFor(`	`Boolean`	`false`		No
Description is not available				
`excludeMatrixOrganizationsForWorker`	`Boolean`	`true`		No
Description is not available				
`includeEmploymentInformationForWork`	`Boolean`	`true`		No
Description is not available				
`includeAccountProvisioningForWorker`	`Boolean`	`false`		No
Description is not available				
`excludeBusinessUnitHierarchiesForWo`	`Boolean`	`true`		No
Description is not available				
`includeRelatedPersonsForWorkers`	`Boolean`	`false`		No
Description is not available				

Property	Type	Default	Encrypted [a]	Required [b]
includePhotoForWorkers	Boolean	false		No
Description is not available				
excludeSupervisoryOrganizationsForW	Boolean	true		No
Description is not available				
excludeTeamsForWorkers	Boolean	false		No
Description is not available				
includeTransactionLogDataForWorkers	Boolean	true		No
Description is not available				
includeSupervisoryDataForOrganizati	Boolean	false		No
Description is not available				
excludeCompaniesForWorkers	Boolean	false		No
Description is not available				
includeAdditionalJobsForWorkers	Boolean	false		No
Description is not available				
excludeBusinessUnitsForWorkers	Boolean	false		No
Description is not available				
includeHierarchyDataForOrganization	Boolean	false		No
Description is not available				
includeEmployeeContractDataForWorke	Boolean	false		No
Description is not available				
includeUserAccountForWorkers	Boolean	true		No
Description is not available				
excludeRegionsForWorkers	Boolean	false		No
Description is not available				
includeRolesDataForOrganizations	Boolean	false		No
Description is not available				
includeMultipleManagersInManagement	Boolean	false		No
Description is not available				

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

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

19.7.2. Basic Configuration Properties Properties

Property	Type	Default	Encrypted [a]	Required [b]
`hostname`	`String`	`null`		Yes
The hostname for the Workday service. Example: https://[workday hostname]/ccx/service/[workday tenant]/. You need to configure the bracketed Workday hostname and tenant name to successfully connect to the proper instance.				
`tenant`	`String`	`null`		Yes
The tenant in URL for the Workday service. For example: https://[workday hostname]/ccx/service/[workday tenant]/. You need to configure the bracketed Workday hostname and tenant name to successfully connect to the proper instance.				
`username`	`String`	`null`		Yes
The user name for logging into the Workday service. It will be concatenated with the tenant name (user@tenant)				
`password`	`GuardedString`	`null`	Yes	Yes
The user password for logging into the Workday service				
`excludeInactiveWorkers`	`boolean`	`false`		No
Excludes from the response terminated employees or contingent workers whose contracts have ended (defaults to false)				
`excludeContingentWorkers`	`boolean`	`false`		No
Excludes contingent workers from inclusion in a query response.				
`excludeEmployees`	`boolean`	`false`		No
Excludes employees from inclusion in a query response.				
`connectionTimeout`	`int`	`30`		No
Specifies the amount of time, in seconds, that the client will attempt to establish a connection before it times out. The default is 30 seconds). Set to 0 for no timeout.				
`receiveTimeout`	`int`	`60`		No
Specifies the amount of time, in seconds, that the client will wait for a response before it times out. The default is 60. Set to 0 for no timeout.				
`pageSize`	`long`	`100`		No
Set the page size used for search operations (defaults to 100).				
`proxyHost`	`String`	`null`		No
If defined the connection to Workday will go through this HTTP proxy server				

Property	Type	Default	Encrypted [a]	Required [b]
`proxyPort`	`int`	`8080`		No
The HTTP proxy server port number (defaults to 8080).				
`xslTransformer`	`File`	`null`		No
The file path to the XSL File to get the custom attributes				
`asOfEffectiveDate`	`String`	`null`		No
Optional configuration of Response_Filter/As_Of_Effective_Date element. Valid values are: Date (http://www.w3.org/TR/xpath-functions/#date-time-values http://www.w3.org/TR/xmlschema-2/#dateTime-order) or Duration (http://www.w3.org/TR/xpath-functions/#dt-dayTimeDuration). If set to Duration, the effective date is calculated as current date + duration.				
`effectiveFrom`	`String`	`null`		No
Set the Get_Workers_Request/Request_Criteria/Transaction_Log_Criteria_Data/Transaction_Date_Range_Data/Effective_From for every outbound query request. Valid value could be Date (http://www.w3.org/TR/xpath-functions/#date-time-values http://www.w3.org/TR/xmlschema-2/#dateTime-order) or string Today representing the current time of the request.				
`effectiveThrough`	`String`	`null`		No
Set the Get_Workers_Request/Request_Criteria/Transaction_Log_Criteria_Data/Transaction_Date_Range_Data/Effective_Through for every outbound query request. Valid value could be Date (http://www.w3.org/TR/xpath-functions/#date-time-values http://www.w3.org/TR/xmlschema-2/#dateTime-order) or Duration (http://www.w3.org/TR/xpath-functions/#dt-dayTimeDuration)				

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

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

Chapter 20
ServiceNow Connector

This connector enables you to manage objects in the ServiceNow platform, integrating with ServiceNow's REST API. The connector is bundled with IDM in the `connectors/` directory (`servicenow-connector-1.5.20.0.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 - 1.5.20.0 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" : "4xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxee",
    "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": "1.5.20.0",
      "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": "2018-02-275 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 Properties

Property	Type	Default	Encrypted [a]	Required [b]
`instance`	`String`	`null`		Yes
URL of the ServiceNow instance, for example: dev00000.service-now.com				
`username`	`String`	`null`		Yes

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

[a] Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.
[b] A list of operations in this column indicates that the property is required for those operations.

Chapter 21 MongoDB Connector

The MongoDB connector is an implementation of the Scripted Groovy Connector Toolkit. This connector enables you to interact with a MongoDB document database, using Groovy scripts for the ICF operations.

The connector is bundled with IDM in the `connectors/` directory (`mongodb-connector-1.5.20.0.jar`).

Note

Version 1.5.20.0 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 on 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

A sample connector configuration (`provisioner.openicf-mongodb.json`) is provided in the `/path/to/openidm/samples/sync-with-mongodb/conf/` directory. You can 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": "1.5.0.0-M1",
      "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. Implemented Interfaces

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

21.3.1. 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. Configuration Properties

The following table lists the configuration properties for the MongoDB connector:

21.4.1. MongoDB Connector Configuration

The MongoDB Connector has the following configurable properties.

21.4.1.1. Configuration Properties

Property	**Type**	**Default**	**Encrypted** [a]	**Required** [b]
`customSensitiveConfiguration`	`GuardedString`	`null`	Yes	No
Custom Sensitive Configuration script for Groovy ConfigSlurper				
`customConfiguration`	`String`	`null`		No
Custom Configuration script for Groovy ConfigSlurper				

[a] Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.
[b] A list of operations in this column indicates that the property is required for those operations.

21.4.1.2. Operation Script Files Properties

Property	**Type**	**Default**	**Encrypted** [a]	**Required** [b]
`createScriptFileName`	`String`	`null`		Create
The name of the file used to perform the CREATE operation.				
`customizerScriptFileName`	`String`	`null`		No
The script used to customize some function of the connector. Read the documentation for more details.				
`authenticateScriptFileName`	`String`	`null`		Authenticate
The name of the file used to perform the AUTHENTICATE operation.				
`scriptOnResourceScriptFileName`	`String`	`null`		Script On Resource
The name of the file used to perform the RUNSCRIPTONRESOURCE operation.				
`deleteScriptFileName`	`String`	`null`		Delete
The name of the file used to perform the DELETE operation.				
`resolveUsernameScriptFileName`	`String`	`null`		Resolve Username
The name of the file used to perform the RESOLVE_USERNAME operation.				
`searchScriptFileName`	`String`	`null`		Get Search
The name of the file used to perform the SEARCH operation.				
`updateScriptFileName`	`String`	`null`		Update

Property	Type	Default	Encrypted [a]	Required [b]
The name of the file used to perform the UPDATE operation.				
`schemaScriptFileName`	`String`	`null`		Schema
The name of the file used to perform the SCHEMA operation.				
`testScriptFileName`	`String`	`null`		Test
The name of the file used to perform the TEST operation.				
`syncScriptFileName`	`String`	`null`		Sync
The name of the file used to perform the SYNC operation.				

[a] Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.
[b] A list of operations in this column indicates that the property is required for those operations.

21.4.1.3. Groovy Engine configuration Properties

Property	Type	Default	Encrypted [a]	Required [b]
`targetDirectory`	`File`	`null`		No
Directory into which to write classes.				
`warningLevel`	`int`	`1`		No
Warning Level of the compiler				
`scriptExtensions`	`String[]`	`['groovy']`		No
Gets the extensions used to find groovy files				
`minimumRecompilationInterval`	`int`	`100`		No
Sets the minimum of time after a script can be recompiled.				
`scriptBaseClass`	`String`	`null`		No
Base class name for scripts (must derive from Script)				
`scriptRoots`	`String[]`	`null`		Yes
The root folder to load the scripts from. If the value is null or empty the classpath value is used.				
`tolerance`	`int`	`10`		No
The error tolerance, which is the number of non-fatal errors (per unit) that should be tolerated before compilation is aborted.				
`debug`	`boolean`	`false`		No
If true, debugging code should be activated				
`classpath`	`String[]`	`[]`		No

Property	Type	Default	Encrypted [a]	Required [b]
Classpath for use during compilation.				
`disabledGlobalASTTransformations`	`String[]`	`null`		No
Sets a list of global AST transformations which should not be loaded even if they are defined in META-INF/ org.codehaus.groovy.transform.ASTTransformation files. By default, none is disabled.				
`verbose`	`boolean`	`false`		No
If true, the compiler should produce action information				
`sourceEncoding`	`String`	`UTF-8`		No
Encoding for source files				
`recompileGroovySource`	`boolean`	`false`		No
If set to true recompilation is enabled				

[a] Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.
[b] A list of operations in this column indicates that the property is required for those operations.

21.4.1.4. Basic Configuration Properties Properties

Property	Type	Default	Encrypted [a]	Required [b]
`connectionURI`	`String`	`null`		No
The MongoDB client connection URI, for example "mongodb://localhost:27017". Overides other connection parameters				
`host`	`String`	`localhost`		No
The MongoDB server host name (localhost by default).				
`port`	`int`	`27017`		No
The MongoDB server port number (27017 by default).				
`user`	`String`	`null`		No
The MongoDB username				
`password`	`GuardedString`	`null`	Yes	No
The password used to connect to MongoDB				
`userDatabase`	`String`	`null`		No
The name of the database in which the MongoDB user is defined				
`clusterAddresses`	`String[]`	`null`		No
A list of additional mongodbDB servers when connecting to a MongoDB cluster (["host1:27017","host2:27017",...]")				

Property	Type	Default	Encrypted [a]	Required [b]
dateAttributes	String[]	[]		No
Defines the list of attributes to convert to MongoDB BSON Date type on create/update.				
database	String	null		No
The database to use				
arrayAttributes	String[]	[]		No
Defines the list of attributes that should be considered as BSON Arrays.				
includeNullValue	boolean	false		No
If set to true, retains null values in the target MongoDB document (false by default).				
includeEmptyList	boolean	false		No
If set to true, retains null values in the target MongoDB document (false by default).				
dateFormat	String	yyyy-MM-dd'T'HH:mm:ss'Z'		No
Defines the date format to use for MongoDB Date attributes (defaults to ISO 8601 "yyyy-MM-ddTHH:mm:ssZ").				
timeZone	String	UTC		No
Defines the timezone to use for MongoDB Date attributes.				
ICFName	String	name		No
Defines the name to use in the target MongoDB document for the ICF __NAME__ attribute.				

[a] Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.
[b] A list of operations in this column indicates that the property is required for those operations.

21.4.1.5. Connection Configuration Properties Properties

Property	Type	Default	Encrypted [a]	Required [b]
sslEnabled	boolean	true		No
Use secure socket layer to connect to MongoDB (true by default)				
sslHostNameValidation	boolean	true		No
Defines if host name should be validated when SSL is enabled				
maxConnectionIdleTime	int	0		No
The maximum idle time for a pooled connection in ms (0 means no limit)				
maxConnectionLifeTime	int	0		No
The maximum life time for a pooled connection in ms (0 means no limit)				

Property	Type	Default	Encrypted [a]	Required [b]
`minConnectionsPerHost`	`int`	`0`		No
The minimum number of connections per host (must be >= 0)				
`maxConnectionsPerHost`	`int`	`5`		No
The maximum number of connections per host (must be > 0)				

[a] Indicates whether the property value is considered confidential, and therefore encrypted in OpenIDM.
[b] A list of operations in this column indicates that the property is required for those operations.

Appendix A. OpenICF Interfaces

This chapter describes all of the interfaces supported by the OpenICF 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 user name and password. If the connector does not implement the AuthenticateOp interface it can not be used in OpenIDM 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 OpenICF 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 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. OpenICF Operation Options

This chapter describes all of the predefined operation options by the OpenICF framework, 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 Container) that specifies how far beneath the specified 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 to use with Script on Resource and possibly others that specifies an account under which to execute the script/operation. The specified account will appear to have performed any action that the script/operation performs.

B.4. Run with Password

An option to use with Script on Resource and possibly others that specifies a password under which to execute the script/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 exactly the set of attributes that are 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) and/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 which is 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 which should be used for ordering the ConnectorObject 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, OpenICF 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, OpenICF 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 15 seconds.
- `"minEvictableIdleTimeMillis"` - the minimum period to wait before evicting an idle connector instance from the pool. The default period is `120000` milliseconds, or 12 seconds.

- `"minIdle"` - the minimum number of idle connector instances in the pool. The default value is `1` instance.