Connectors Guide

/ OpenIDM 5

Latest update: 5.0.1.1

Lana Frost

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

Abstract

Guide to the connectors that are supported with OpenIDM software. The guide provides 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 OpenIDM. The guide focuses on getting the connectors installed and configured with OpenIDM 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 OpenIDM software.

You do not need to have a complete understanding of OpenIDM 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. Formatting Conventions

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

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

Command-line, terminal sessions are formatted as follows:

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

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

Program listings are formatted as follows:

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

3. Accessing Documentation Online

ForgeRock publishes comprehensive documentation online:

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

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

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

4. Using the ForgeRock.org Site

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

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

Chapter 1

Connector Overview

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

For instructions on building connector configurations interactively, see "Creating Default Connector Configurations" in the *Integrator's Guide*.

1.1. Connectors Supported With OpenIDM 5

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 OpenDJ, 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*".

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*".

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*".

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*".

Scripted SAP Connector

The scripted 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 "*Scripted SAP 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*".

Salesforce Connector

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

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

XML File Connector

The XML File connector is really useful only in a demonstration context and should not be used in the general provisioning of XML data stores.

For information about configuring the XML File connector, see "*XML File Connector*".

Active Directory Connector

The Active Directory connector is a legacy connector, written in C# for the .NET platform.

For information about installing and configuring the Active Directory connector, see "*Active Directory 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 OpenDJ, 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 will be deprecated in a future OpenICF release, and, ultimately, support for its use with OpenIDM will be discontinued. 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

OpenIDM 5 bundles version 1.4.3.0 of the LDAP connector. Three sample LDAP connector configurations are provided in the `path/to/openidm/samples/provisioners/` directory:

- `provisioner.openicf-opendjldap.json` provides a sample LDAP connector configuration for an OpenDJ directory 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 OpenDJ. 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,
    "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" : "dn",
    "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.

If you use the LDAP connector over SSL, set the `ssl` property to `true`, and the `port` to `636` in the connector configuration file. You must also specify the path to a truststore in your project's `conf/system.properties` file. A truststore is provided by default at `openidm/security/truststore`. Add the following line to the `system.properties` file, substituting the path to your own truststore if you do not want to use the default:

```
# Set the truststore
javax.net.ssl.trustStore=/path/to/openidm/security/truststore
```

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 an OpenDJ server).

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, OpenIDM 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, OpenIDM 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 `OR` 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` OR the object class `inetOrgPerson` is considered as an account entry. The value of this property must not include the `top` object class. You can override the value of this property by specifying the user object classes during the create operation.

`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. OpenIDM ignores change log updates that do not include any of the specified attributes. If empty, OpenIDM 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 OpenIDM 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`, OpenIDM modifies group membership when entries are renamed or deleted.

In the sample LDAP connector configuration file provided with OpenIDM, this property is set to `false`. This means that LDAP group membership is not modified when entries are renamed or deleted in OpenIDM. 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 OpenDJ, see *Configuring Referential Integrity* in the *Directory Services Developer's Guide*.

`maintainPosixGroupMembership`

If `true`, OpenIDM 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`

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

`passwordAttribute`

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

`passwordAttributeToSynchronize`

OpenIDM 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 OpenIDM, use one of the password synchronization

plugins, described in "Synchronizing User Passwords With LDAP Servers" in the *Integrator's Guide*.

`passwordDecryptionKey`

This is a legacy attribute, and its value should remain set to `null`. To configure password synchronization between an LDAP server and OpenIDM, use one of the password synchronization plugins, described in "Synchronizing User Passwords With LDAP Servers" in the *Integrator's 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 OpenIDM 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 OpenIDM, use one of the password synchronization plugins, described in "Synchronizing User Passwords With LDAP Servers" in the *Integrator's Guide*.

`uidAttribute`

Specifies the LDAP attribute that should be used as the immutable ID (`_UID_`) for the entry. For an OpenDJ resource, you should use the `entryUUID`. You can use the `DN` as the UID attribute but note that this is *not* immutable.

`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 OpenDJ 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.

2.2. 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 OpenIDM) and implicit sync (synchronization from the OpenIDM repository 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. OpenIDM 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, OpenIDM 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 OpenIDM does not attempt to replay the change back to the managed user repository. So you are effectively indicating that OpenIDM 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.

2.3. Using the Generic LDAP Connector With Active Directory

The LDAP connector provides new functionality for managing Active Directory users and groups. Among other changes, the new 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 OpenIDM 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 OpenIDM sets the `__LOCK_OUT__` to `FALSE`, the LDAP connector sets the Active Directory `lockoutTime` to `0` to unlock the account.

  If OpenIDM sets the `__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 (to force a user to change their password when they next log in), `pwdLastSet` must be set to `0`. The LDAP connector sets `pwdLastSet` to `0`, if OpenIDM sets `__PASSWORD_EXPIRED__` to `TRUE`.

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

Note

You must update your provisioner configuration to be able to use these new operational attributes. You can use this sample provisioner configuration as a guide.

2.3.1. 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, set `"ssl" : true` in the provisioner configuration and set the path to your truststore in your project's `conf/system.properties` file. For example, add the following line to that file:

```
# Set the truststore
javax.net.ssl.trustStore=/path/to/openidm/security/truststore
```

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": "<GUID=cb2f8cbc032f474c94c896e69db2feb3>",
  "mobile": null,
  "postalCode": null,
  "st": null,
  "employeeType": null,
  "objectGUID": "<GUID=cb2f8cbc032f474c94c896e69db2feb3>",
  "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": []
}
```


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.

2.3.2. 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": "<GUID=240da4e959d81547ad8629f5b2b5114d>"
}
```

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": "<GUID=f189df8a276f91478ad5055b1580cbcb>"
}
```

2.3.3. 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
```

OpenIDM 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. 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)
```

2.5. 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.

2.6. LDAP Connector Configuration

The LDAP Connector has the following configurable properties.

2.6.1. Configuration Properties

Property	Type	Default	Encrypted [a]	Required [b]
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.				
passwordAttributeToSynchronize	String	null		Sync
The name of the password attribute to synchronize when performing password synchronization.				
synchronizePasswords	boolean	false		Sync
If true, the connector will synchronize passwords. The Password Capture Plugin needs to be installed for password synchronization to work.				
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.				
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".				
passwordDecryptionKey	GuardedByteArray	null	Yes	Sync
The key to decrypt passwords with when performing password synchronization.				
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.				
credentials	GuardedString	null	Yes	No
Password for the principal.				
changeLogBlockSize	int	100		Sync
The number of change log entries to fetch per query.				
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				

Property	Type	Default	Encrypted [a]	Required [b]
"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.				
passwordDecryptionInitializationVec	GuardedByteArray	null	Yes	Sync
The initialization vector to decrypt passwords with when performing password synchronization.				
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.				
useTimestampsForSync	boolean	false		Sync
If true, the connector will use the createTimestamp and modifyTimestamp system attributes to detect changes (Create/Update) on the directory instead of native change detection mechanism (cn=changelog on OpenDJ or Update Sequence Number -USN- on Active Directory for instance). Default value is false.				
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.				
port	int	389		No
TCP/IP port number used to communicate with the LDAP server.				
vlvSortAttribute	String	uid		No
Specify the sort attribute to use for VLV indexes on the resource.				
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.				
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.				
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.				

Property	Type	Default	Encrypted [a]	Required [b]
failover	String[]	[]		No
List all servers that should be used for failover in case the preferred server fails. If the preferred server fails, JNDI will connect to the next available server in the list. List all servers in the form of "ldap://ldap.example.com:389/", which follows the standard LDAP v3 URLs described in RFC 2255. Only the host and port parts of the URL are relevant in this setting.				
ssl	boolean	false		No
Select the check box to connect to the LDAP server using SSL.				
getGroupMemberId	boolean	false		No
Specifies whether to add an extra _memberId attribute to get the group members __UID__				
referralsHandling	String	follow		No
Defines how to handle LDAP referrals. Possible values can be follow, ignore or throw.				
principal	String	null		No
The distinguished name with which to authenticate to the LDAP server.				
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.				
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.				
authType	String	simple		No
The authentication mechanism to use: Simple or SASL-GSSAPI. Defaults to "simple".				
accountObjectClasses	String[]	['top', 'person', 'organizational', '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.				
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.				
host	String	null		No

Property	Type	Default	Encrypted [a]	Required [b]
The name or IP address of the host where the LDAP server is running.				
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.				
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).				
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.				
usePagedResultControl	boolean	false		No
When enabled, the LDAP Paged Results control is preferred over the VLV control when retrieving entries.				
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.				
blockSize	int	100		No
The maximum number of entries that can be in a block when retrieving entries in blocks.				
startTLS	boolean	false		No
Specifies whether to use the startTLS operation to initiate a TLS/SSL session.				
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.				
uidAttribute	String	entryUUID		No
The name of the LDAP attribute that is mapped to the OpenICF UID attribute.				
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.				
maintainLdapGroupMembership	boolean	false		No

Property	Type	Default	Encrypted [a]	Required [b]
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.				
`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.				
`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.				

[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/provisioners/provisioner.openicf-csv.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 following excerpt shows the *required* configuration properties:

```
"configurationProperties" : {
    "csvFile" : "&{launcher.project.location}/data/hr.csv",
    "headerUid" : "uid"
},
```

`csvFile`

The path to the CSV file that is the data source for this connector.

`headerUid`

The CSV header that maps to the `uid` (or name) for each row.

Default: `uid`

The CSV file connector also supports following optional configuration properties:

`encoding`

Default: `utf-8`

`headerPassword`

The CSV header that maps to the password for each row. Use this property when password-based authentication is required.

`fieldDelimiter`

The character in the CSV file that is used to separate field values.

Default: `,`

`quoteCharacter`

The character in the CSV file that is used to encapsulate strings.

Default: `"`

`newlineString`

The character string in the CSV file that is used to terminate each line.

Default: `\n`

`syncFileRetentionCount`

The number of historical copies of the CSV file to retain when performing synchronization operations.

Default: `3`

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]
`csvFile`	`File`	`null`		Yes
Full path to the CSV file				
`headerUid`	`String`	`uid`		No
Name of the uid column as found in the CSV file				
`quoteCharacter`	`String`	`"`		No
Character used to quote fields				
`headerPassword`	`String`	`password`		No
Name of the password column as found in the CSV file				
`fieldDelimiter`	`String`	`,`		No
Character used to delimit columnar fields				
`syncFileRetentionCount`	`int`	`3`		No
Number of sync history files to retain				
`newlineString`	`String`			No
Character(s) used to terminate a line in the CSV file				

[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/provisioners/provisioner.openicf-contractordb.json`. The corresponding data definition language file is provided in `samples/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_TIMESTEMP",
       "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. 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.

4.3. Database Table Connector Configuration

The Database Table Connector has the following configurable properties.

4.3.1. Configuration Properties

Property	Type	Default	Encrypted [a]	Required [b]
`quoting`	`String`			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.				
`host`	`String`			No
Enter the name of the host on which the database is running.				
`port`	`String`			No
Enter the port number on which the database server is listening.				
`user`	`String`			No
Enter the name of the mandatory Database user with permission to access the accounts table.				
`password`	`GuardedString`	`null`	Yes	No
Enter a user account that has permission to access the accounts table.				
`database`	`String`			No
Enter the name of the database on the database server that contains the table.				

Property	Type	Default	Encrypted [a]	Required [b]
table	String			Yes
Enter the name of the table in the database that contains the accounts.				
keyColumn	String			Yes
This mandatory column value will be used as the unique identifier for rows in the table.				
passwordColumn	String			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.				
jdbcDriver	String	oracle.jdbc.driver.OracleDriver		No
Specify the JDBC Driver class name. For Oracle: oracle.jdbc.driver.OracleDriver. For MySQL: org.gjt.mm.mysql.Driver. Can be empty if datasource is provided.				
jdbcUrlTemplate	String	jdbc:oracle:thin%h:%p:%d		No
Specify the JDBC Driver Connection URL. Oracle template is jdbc:oracle:thin:@[host]:[port(1521)]:[DB]. MySQL template is jdbc:mysql://[host]:[port(3306)]/[db], for more info, read the JDBC driver documentation. Could be empty if datasource is provided.				
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
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.				
validConnectionQuery	String	null		No
Specify whether the check connection alive query should be used. If empty, the default implementation checks the connection by switching autocommit on/off. It might be more efficient to test the connection by selecting 1 from a dummy table.				
changeLogColumn	String			Sync
The change log column stores the latest change time. Providing this value the Sync capabilities are activated.				

Property	Type	Default	Encrypted [a]	Required [b]
`datasource`	`String`			No
If specified, the connector will attempt to connect using only this data source, and will ignore other specified resource parameters. For example: jdbc/SampleDataSourceName				
`jndiProperties`	`String[]`	`null`		No
Could be empty or enter the JDBC JNDI Initial context factory, context provider in a format: key = value.				
`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.				

[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.

OpenIDM includes Active Directory and Azure sample scripts for the PowerShell connector that can help you get started with this toolkit. For more information, see "*Samples That Use the PowerShell Connector Toolkit to Create Scripted Connectors*" in the *Samples Guide*.

The sample scripts illustrate the following scenarios:

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

5.1. Before You Start

To implement the scripted PowerShell connector, you need to 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, follow the instructions in "Installing and Configuring a .NET Connector Server" in the *Integrator's Guide*.
- PowerShell version 2.0 or above. The PowerShell Connector Toolkit is not bundled with OpenIDM, but is available from ForgeRock's BackStage. To install the connector, download the archive (`mspowershell-connector-1.4.3.0.zip`) and extract the `MsPowerShell.Connector.dll` to the same directory where the Connector Server (`ConnectorServerService.exe` or the legacy version `ConnectorServer.exe`) is located.

If you're running a supported version of Microsoft Windows Server, as described in "*Before You Install*" in the *Release Notes*, you should already meet these requirements.

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 that .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 OpenIDM to connect to the .NET connector server.

   To do so, copy the remote connector provisioner file from the `openidm\samples\provisioners` directory to your project's `conf\` directory, and edit the file 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 OpenIDM to Connect to the .NET Connector Server" in the *Integrator's Guide*.

3. Download the PowerShell Connector Toolkit archive (`mspowershell-connector-1.4.3.0.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 files (`ConnectorServerService.exe` and the legacy `ConnectorServer.exe`) are located.

4. OpenIDM includes PowerShell scripts in `openidm\samples` subdirectories, including `powershell2AD/` for Active Directory, and `powershell2AzureAD` for Azure AD. Copy these scripts to the host on which the .NET connector server is installed.

   The location of the scripts must be referenced in your connector configuration file, for example:

   ```
   "CreateScriptFileName" : "C:/openidm/samples/powershell2AD/tools/ADCreate.ps1",
   ...
   ```

5. Copy the sample connector configuration for the PowerShell connector from the `samples\provisioners` directory to your project's `conf` directory.

   OpenIDM 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

The sample PowerShell connector configuration files include the following configurable properties:

Property	Type	Example	Encrypted [a]	Required [b]
*operation*ScriptFileName	String	C:/openidm/AD/ADCreate.ps1, C:/openidm/samples/powershell2Azure/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.				
MaxInterpretersPoolSize	Integer	5	No	No
The maximum size of the interpreter pool.				

Property	Type	Example	Encrypted [a]	Required [b]
`MinInterpretersPoolSize`	`Integer`	`1`	No	No
The minimum size of the interpreter pool.				
`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`	`["ActiveDirector` (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				
`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 OpenIDM.

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

5.4. Testing the PowerShell Connector

Start OpenIDM 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" : "[1.4.2.0,1.5.0.0)"
  },
  "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 OpenIDM 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"
```

Chapter 6

Groovy Connector Toolkit

OpenICF provides a generic Groovy Connector Toolkit that enables you to run a Groovy script for any OpenICF 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. Specific scripts are provided within these samples, which demonstrate how the Groovy Connector Toolkit can be used. These scripts cannot be used as is in your deployment, but are a good starting point on which to base your customization.

6.1. Groovy Connector Toolkit

OpenIDM bundles the Groovy Connector Toolkit is bundled in the JAR `openidm/connectors/groovy-connector-1.4.3.0.jar`.

Sample implementations are provided in "*Samples That Use the Groovy Connector Toolkit to Create Scripted Connectors*" in the *Samples Guide*.

6.2. 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.

6.3. Scripted Groovy Connector Configuration

The Scripted Groovy Connector has the following configurable properties.

6.3.1. Operation Script Files Properties

Property	Type	Default	Encrypted [a]	Required [b]
authenticateScriptFileName	String	null		Authenticate
The name of the file used to perform the AUTHENTICATE operation.				
deleteScriptFileName	String	null		Delete
The name of the file used to perform the DELETE operation.				
schemaScriptFileName	String	null		Schema
The name of the file used to perform the SCHEMA 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.				
testScriptFileName	String	null		Test
The name of the file used to perform the TEST operation.				
updateScriptFileName	String	null		Update
The name of the file used to perform the UPDATE operation.				
searchScriptFileName	String	null		Get Search
The name of the file used to perform the SEARCH operation.				
scriptOnResourceScriptFileName	String	null		Script On Resource
The name of the file used to perform the RUNSCRIPTONRESOURCE operation.				
createScriptFileName	String	null		Create
The name of the file used to perform the CREATE 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. Groovy Engine configuration Properties

Property	Type	Default	Encrypted [a]	Required [b]
warningLevel	int	1		No
Warning Level of the compiler				
minimumRecompilationInterval	int	100		No
Sets the minimum of time after a script can be recompiled.				
scriptRoots	String[]	null		Yes
The root folder to load the scripts from. If the value is null or empty the classpath value is used.				
debug	boolean	false		No
If true, debugging code should be activated				
targetDirectory	File	null		No
Directory into which to write classes.				
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.				
classpath	String[]	[]		No
Classpath for use during compilation.				
scriptExtensions	String[]	['groovy']		No
Description is not available				
sourceEncoding	String	UTF-8		No
Encoding for source files				
scriptBaseClass	String	null		No
Base class name for scripts (must derive from Script)				
verbose	boolean	false		No
If true, the compiler should produce action information				
recompileGroovySource	boolean	false		No
If set to true recompilation is enabled				
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.				

[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.

6.3.3. Configuration Properties

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

Chapter 7

Scripted SAP Connector

The scripted 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 OpenIDM
- Synchronization of users between OpenIDM and an SAP (R/3) system

7.1. Before You Start

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.

7.2. Setting Up the SAP Connector

1. Download the SAP connector from ForgeRock's BackStage site.
2. Copy the SAP connector JAR file (`sap-connector-1.4.1.0.jar`) to the `openidm/connectors` directory:

   ```
   $ cp ~/Downloads/sap-connector-1.4.1.0.jar /path/to/openidm/connectors
   ```

3. Copy the SAP JCo libraries that correspond to your architecture 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 OpenIDM logging configuration to log messages from the SAP connector.

   By default, OpenIDM 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
samples.r3.level=FINER
samples.hr.level=FINER
samples.level=FINER
```

7.3. 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 OpenIDM to use these sample scripts, how to test the connection to the SAP HR system, and how to update user records.

7.3.1. Setting up OpenIDM 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. To test this connector, you can use the sample Groovy scripts available from the *ForgeRock Artifact Repository Browser*. You can find the source for these scripts in this location, in the `samples/` directory, as well as the `samples/hr/` subdirectory.

   `TestSAP.groovy`
   `SearchSAPHR.groovy`
   `UpdateSAPHR.groovy`
   `SchemaSAPHR.groovy`
   `EmplComm.groovy`

   Update your connector configuration to point to those scripts. The sample connector configuration assumes the following locations for the scripts (relative to the value of the `scriptRoots` property):

   ```
   "testScriptFileName" : "samples/TestSAP.groovy",
   "searchScriptFileName" : "samples/hr/SearchSAPHR.groovy",
   "updateScriptFileName" : "samples/hr/UpdateSAPHR.groovy",
   "schemaScriptFileName" : "samples/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 `samples` directory (because

it belongs to the `samples` package) and the remaining HR scripts must be under a `samples/hr` directory (because they belong to the `hr` package).

7.3.2. Testing the Connection to the SAP HR System

1. Start OpenIDM 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" : "1.4.1.0"
     },
     "displayName" : "Sap Connector",
     "ok" : true
   }
   ```

3. Retrieve a list of the existing users (with their employee number) in the SAP HR system:

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/system/saphr/employee?_queryId=query-all-ids"
{
  "result" : [ {
    "_id" : "00000010",
    "__NAME__" : "00000010"
  }, {
    "_id" : "00000069",
    "__NAME__" : "00000069"
  }, {
    "_id" : "00000070",
    "__NAME__" : "00000070"
  }
,
...
```

4. Retrieve the complete record of an employee in the SAP HR system by including the employee's ID in the URL.

   The following command retrieves the record for employee Maria Gonzales:

```
$ curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --request GET \
 "http://localhost:8080/openidm/system/saphr/employee/55099307"
{
  "_id" : "55099307",
  "PERSONAL_DATA" : {
    "PERNO" : "55099307",
    "INFOTYPE" : "0002",
    "TO_DATE" : "Fri Dec 31 00:00:00 CET 9999",
    "FROM_DATE" : "Tue Mar 30 00:00:00 CET 1954",
    "SEQNO" : "000",
    "CH_ON" : "Thu Mar 27 00:00:00 CET 2003",
    "CHANGED_BY" : "MAYROCK",
    "LAST_NAME" : "Gonzales",
    "FIRSTNAME" : "Maria",
    "NAME_FORM" : "00",
    "FORMOFADR" : "2",
    "GENDER" : "2",
    "BIRTHDATE" : "Tue Mar 30 00:00:00 CET 1954",
    "LANGU" : "D",
    "NO_O_CHLDR" : "0",
    "BIRTHYEAR" : "1954",
    "BIRTHMONTH" : "03",
    "BIRTHDAY" : "30",
    "LASTNAME_M" : "GONZALES",
    "FSTNAME_M" : "MARIA"
  }
,
...
}
```

7.3.3. 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"
```

7.4. 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 an OpenIDM reconciliation operation to detect changes on the SAP system.

7.4.1. Setting up OpenIDM 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. To test this connector, you can use the sample Groovy scripts available from the *ForgeRock Artifact Repository Browser*. You can find the source for these scripts in this location, in the `samples/` directory, as well as the `samples/r3/` subdirectory.

   `TestSAP.groovy`
   `SearchSAPR3.groovy`
   `CreateSAPR3.groovy`
   `UpdateSAPR3.groovy`
   `DeleteSAPR3.groovy`
   `SchemaSAPR3.groovy`

   Update your connector configuration to point to those scripts. The sample connector configuration assumes the following locations for the scripts (relative to the value of the `scriptRoots` property):

   ```
   "testScriptFileName" : "samples/TestSAP.groovy",
   "searchScriptFileName" : "samples/r3/SearchSAPR3.groovy",
   "createScriptFileName" : "samples/r3/CreateSAPR3.groovy",
   "updateScriptFileName" : "samples/r3/UpdateSAPR3.groovy",
   "deleteScriptFileName" : "samples/r3/DeleteSAPR3.groovy",
   "schemaScriptFileName" : "samples/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 `samples` directory (because it belongs to the `samples` package) and the R/3 scripts must be under a `samples/r3` directory (because they belong to the `r3` package).

7.4.2. Testing the Connection to the SAP R/3 System

1. Start OpenIDM 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": "1.4.1.0"
  },
  "displayName": "Sap Connector",
  "ok": true
}
```

7.4.3. Using the SAP Connector to Manage SAP R/3 Users

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

7.4.3.1. 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
}
```

7.4.3.2. 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

7.4.3.3. 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"
    }
,
...
```

7.4.3.3.1. 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)
}
```

7.4.3.4. Updating SAP User Accounts

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

7.4.3.4.1. 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"
```

7.4.3.4.2. 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"
```

7.4.3.4.3. 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"
```

7.4.3.5. 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.

7.4.3.6. 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.

7.4.3.6.1. 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.

7.4.3.6.2. 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"
}
```

7.4.3.6.3. 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.

7.4.3.7. 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.

7.4.3.7.1. 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.

7.4.3.7.2. 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"
}
```

7.4.3.7.3. 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.

7.5. 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".

7.6. 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.

7.7. SAP Connector Configuration

The SAP Connector has the following configurable properties.

7.7.1. Configuration Properties

Property	Type	Default	Encrypted [a]	Required [b]
`minimumRecompilationInterval`	`int`	`100`		No
Description is not available				
`updateScriptFileName`	`String`	`null`		Update
Description is not available				
`scriptOnResourceScriptFileName`	`String`	`null`		Script On Resource
Description is not available				
`searchScriptFileName`	`String`	`null`		Get Search
Description is not available				
`scriptBaseClass`	`String`	`null`		No
Description is not available				
`verbose`	`boolean`	`false`		No
Description is not available				
`createScriptFileName`	`String`	`null`		Create
Description is not available				
`tolerance`	`int`	`10`		No
Description is not available				
`deleteScriptFileName`	`String`	`null`		Delete
Description is not available				
`schemaScriptFileName`	`String`	`null`		Schema

Property	Type	Default	Encrypted [a]	Required [b]
Description is not available				
`customConfiguration`	`String`	`null`		No
Description is not available				
`classpath`	`String[]`	`[]`		No
Description is not available				
`recompileGroovySource`	`boolean`	`false`		No
Description is not available				
`scriptRoots`	`String[]`	`null`		Yes
Description is not available				
`customizerScriptFileName`	`String`	`null`		No
Description is not available				
`resolveUsernameScriptFileName`	`String`	`null`		Resolve Username
Description is not available				
`debug`	`boolean`	`false`		No
Description is not available				
`disabledGlobalASTTransformations`	`String[]`	`null`		No
Description is not available				
`scriptExtensions`	`String[]`	`['groovy']`		No
Description is not available				
`sourceEncoding`	`String`	`UTF-8`		No
Description is not available				
`authenticateScriptFileName`	`String`	`null`		Authenticate
Description is not available				
`warningLevel`	`int`	`1`		No
Description is not available				
`targetDirectory`	`File`	`null`		No
Description is not available				
`customSensitiveConfiguration`	`GuardedString`	`null`	Yes	No

Property	Type	Default	Encrypted [a]	Required [b]
Description is not available				
testScriptFileName	String	null		Test
Description is not available				
syncScriptFileName	String	null		Sync
Description is not available				
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.

7.7.2. Basic Configuration Properties

Property	Type	Default	Encrypted [a]	Required [b]
gwServ	String	null		Yes
SAP gateway service				
gwHost	String	null		Yes
SAP gateway host name				
asHost	String	null		Yes
The FQDN of your SAP Application Server, for example sap.example.com				
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
SAP system number				
language	String	EN		Yes
SAP Logon language				
destination	String	OPENIDM		Yes

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

7.7.3. 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.

7.7.4. Advanced Configuration Properties

Property	Type	Default	Encrypted [a]	Required [b]
`msHost`	`String`	`null`		No
Specifies the host that the message server is running on				
`msServ`	`String`	`null`		No
Name of the service where the message server can be reached				
`r3Name`	`String`	`null`		No
Specifies the name of the SAP system, used when you log in to a logon group that uses load balancing				
`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				

[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.

7.7.5. JCo Connection Pool Configuration Properties

Property	Type	Default	Encrypted [a]	Required [b]
expirationPeriod	String	60000		No
Period in ms after that the destination checks the released connections for expiration. Default is one minute				
peakLimit	String	0		No
Maximum number of active connections that can be created for a destination simultaneously. The default is 0 (unlimited).				
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.				
poolCapacity	String	1		No
Maximum number of idle connections kept open by the destination. 0 = no connection pooling. Default is 1.				
expirationTime	String	60000		No
Time in ms after that a free connection can be closed. Default is one minute.				

[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.

7.7.6. SAP Secure Network Connection Configuration Properties

Property	Type	Default	Encrypted [a]	Required [b]
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				
sncMyName	String	null		No
Specifies the connectors 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).				
sncMode	String	0		Yes
Flag used to activate SNC. Possible values are 0 (OFF) and 1 (ON).				
sncLibrary	String	null		No

Property	Type	Default	Encrypted [a]	Required [b]
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???s SNC name, for example, "p:CN=ABC, O=MyCompany, C=US". You can find the application server???s SNC name in the profile parameter snc/identity/as on the AS ABAP.				

[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.

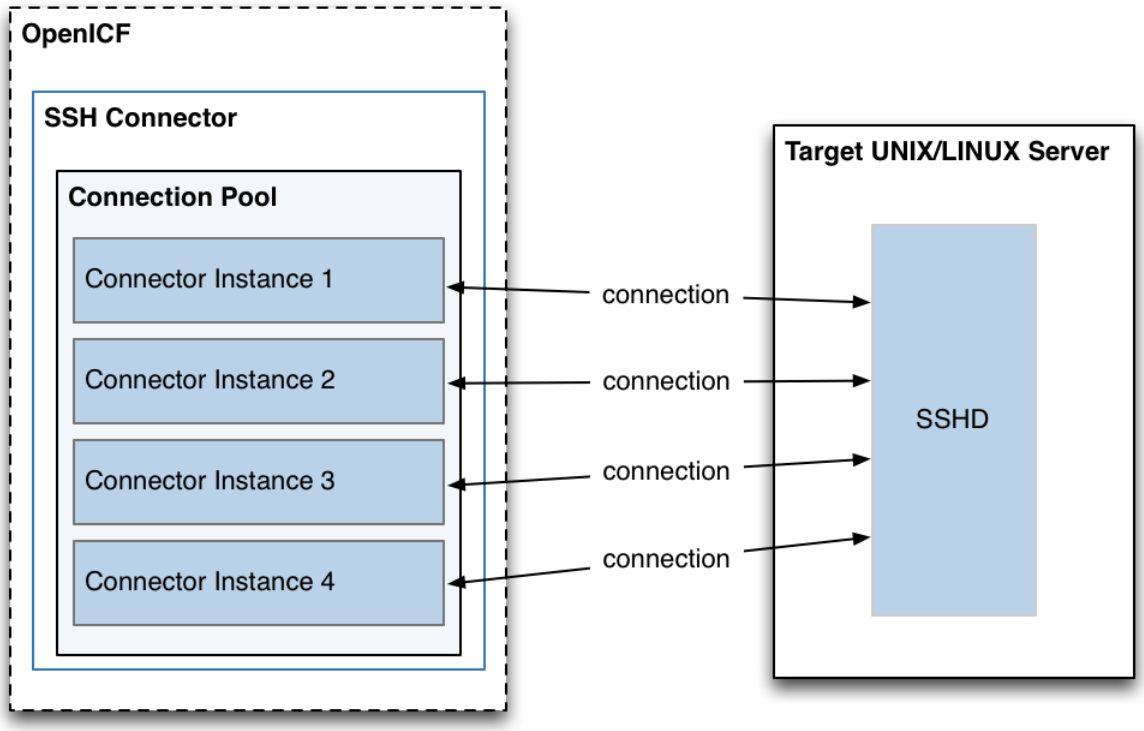
Chapter 8

Scripted SSH Connector

The scripted 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:



8.1. 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 OpenIDM 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 OpenIDM 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 OpenIDM loads the provisioner file.

8.2. Configuring the SSH Connector

OpenIDM 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 OpenIDM installation directory. The sample connector configuration expects the scripts in *`project-dir`*`/tools`, so this parameter is set to `&{launcher.project.location}/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.

8.3. 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.

8.4. SSH Connector Configuration

The SSH Connector has the following configurable properties.

8.4.1. Configuration Properties

Property	Type	Default	Encrypted [a]	Required [b]
`minimumRecompilationInterval`	`int`	`100`		No
Description is not available				
`scriptRoots`	`String[]`	`null`		Yes
Description is not available				
`customizerScriptFileName`	`String`	`null`		No
Description is not available				
`resolveUsernameScriptFileName`	`String`	`null`		Resolve Username
Description is not available				
`debug`	`boolean`	`false`		No
Description is not available				
`disabledGlobalASTTransformations`	`String[]`	`null`		No
Description is not available				
`scriptExtensions`	`String[]`	`['groovy']`		No
Description is not available				
`updateScriptFileName`	`String`	`null`		Update

Property	Type	Default	Encrypted [a]	Required [b]
Description is not available				
sourceEncoding	String	UTF-8		No
Description is not available				
searchScriptFileName	String	null		Get Search
Description is not available				
scriptOnResourceScriptFileName	String	null		Script On Resource
Description is not available				
scriptBaseClass	String	null		No
Description is not available				
verbose	boolean	false		No
Description is not available				
createScriptFileName	String	null		Create
Description is not available				
tolerance	int	10		No
Description is not available				
authenticateScriptFileName	String	null		Authenticate
Description is not available				
warningLevel	int	1		No
Description is not available				
deleteScriptFileName	String	null		Delete
Description is not available				
schemaScriptFileName	String	null		Schema
Description is not available				
customConfiguration	String	null		No
Description is not available				
targetDirectory	File	null		No
Description is not available				

Property	Type	Default	Encrypted [a]	Required [b]
classpath	String[]	[]		No
Description is not available				
testScriptFileName	String	null		Test
Description is not available				
customSensitiveConfiguration	GuardedString	null	Yes	No
Description is not available				
recompileGroovySource	boolean	false		No
Description is not available				
syncScriptFileName	String	null		Sync
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.

8.4.2. 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)				

[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

Google Apps Connector

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

9.1. Configuring the Google Apps Connector

To use this connector, you need a Google Apps account.

A sample Google Apps connector configuration file is provided in `samples/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 "*Google Sample - Connecting to Google With the Google Apps Connector*" in the *Samples Guide*.

9.2. 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.

9.3. GoogleApps Connector Configuration

The GoogleApps Connector has the following configurable properties.

9.3.1. 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.				

[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

Scripted Kerberos Connector

The scripted Kerberos connector is an implementation of the scripted 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 OpenIDM:

- `/path/to/openidm/lib/ssh-connector-1.4.1.0.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 OpenIDM. The connector is provided in `/path/to/openidm/connectors/kerberos-connector-1.4.2.0.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

10.1. 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`

10.2. Configuring the Kerberos Connector

OpenIDM provides a sample connector configuration (`provisioner.openicf-kerberos.json`) in the `/path/to/openidm/samples/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 OpenIDM 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-1.4.2.0.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.

10.3. 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.

10.4. Kerberos Connector Configuration

The Kerberos Connector has the following configurable properties.

10.4.1. Configuration Properties

Property	Type	Default	Encrypted [a]	Required [b]
`minimumRecompilationInterval`	`int`	`100`		No
Description is not available				
`scriptRoots`	`String[]`	`null`		Yes
Description is not available				
`customizerScriptFileName`	`String`	`null`		No
Description is not available				
`resolveUsernameScriptFileName`	`String`	`null`		Resolve Username
Description is not available				
`debug`	`boolean`	`false`		No
Description is not available				
`disabledGlobalASTTransformations`	`String[]`	`null`		No
Description is not available				
`scriptExtensions`	`String[]`	`['groovy']`		No
Description is not available				
`updateScriptFileName`	`String`	`null`		Update
Description is not available				
`sourceEncoding`	`String`	`UTF-8`		No
Description is not available				
`searchScriptFileName`	`String`	`null`		Get Search
Description is not available				
`scriptOnResourceScriptFileName`	`String`	`null`		Script On Resource
Description is not available				
`scriptBaseClass`	`String`	`null`		No
Description is not available				
`verbose`	`boolean`	`false`		No
Description is not available				

Property	Type	Default	Encrypted [a]	Required [b]
createScriptFileName	String	null		Create
Description is not available				
tolerance	int	10		No
Description is not available				
authenticateScriptFileName	String	null		Authenticate
Description is not available				
warningLevel	int	1		No
Description is not available				
deleteScriptFileName	String	null		Delete
Description is not available				
schemaScriptFileName	String	null		Schema
Description is not available				
customConfiguration	String	null		No
Description is not available				
targetDirectory	File	null		No
Description is not available				
classpath	String[]	[]		No
Description is not available				
testScriptFileName	String	null		Test
Description is not available				
customSensitiveConfiguration	GuardedString	null	Yes	No
Description is not available				
recompileGroovySource	boolean	false		No
Description is not available				
syncScriptFileName	String	null		Sync
Description is not available				
host	String	null		Yes
Description is not available				

Property	Type	Default	Encrypted [a]	Required [b]
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
Description is not available				

Property	Type	Default	Encrypted [a]	Required [b]
`throwOperationTimeoutException`	`boolean`	`true`		Yes
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 11

Salesforce Connector

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

The Salesforce Connector is not an OpenICF connector, but a separate OpenIDM 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 "*Connecting to Salesforce With the Salesforce Connector*" in the *Samples Guide*.

Chapter 12

Marketo Connector

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

This connector forms part of OpenIDM's support for customer data management (CDM). You can synchronize any managed user to Marketo - those who have been added directly to the OpenIDM repository, and those who have registered themselves through one of the Social Identity Providers described in "*Configuring Social ID 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 a Marketo account, a client ID, client secret, and the REST API URL for your OpenIDM service, and a custom list created in your Marketo Leads database. For information on obtaining these details from Marketo, see the Marketo documentation.

A sample connector configuration file is available as part of the CDM sample, at `/path/to/openidm/samples/cdm/conf/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. To locate the REST API endpoint URL, select Admin > Web Services in Marketo, 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://140-OCV-xxx.mktorest.com/rest`, the value of the `instance` property must be `140-OCV-xxx.mktorest.com`.

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

```
{
    "name" : "marketo",
    "displayName" : "MarketoConnector",
    "description" : "Connector used to sync users to Marketo leads",
    "version" : "[1.4.0.0,1.5.0.0)",
    "author" : "ForgeRock",
    "enabled" : true,
    ...
    "configurationProperties" : {
        "instance" : "140-OCV-xxx.mktorest.com",
        "clientId" : "1b5fxxxxxxxxxxxxxxxxxxxx6a2c",
        "clientSecret" : "19jf910703j19012790j0123i-d1",
        "leadFields" : null,
        "partitionName" : null,
        "listName" : "DecemberPromotion",
    ...
```

You can also configure the Marketo connector through the Admin UI. Select Configure > Connectors, click New Connector, and complete at least the required configuration properties, described in "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": "1.4.2.0"
    },
    "displayName": "Marketo Connector",
    "ok": true
  }
]
```

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

For additional instructions, and a complete sample that demonstrates reconciliation from OpenIDM to Marketo, see "*Customer Data Management Sample*" in the *Samples Guide*.

12.1. 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.

12.2. Marketo Connector Configuration

The Marketo Connector has the following configurable properties.

12.2.1. Groovy Engine configuration Properties

Property	Type	Default	Encrypted [a]	Required [b]
`minimumRecompilationInterval`	`int`	`100`		No
Sets the minimum of time after a script can be recompiled.				
`scriptRoots`	`String[]`	`null`		Yes
The root folder to load the scripts from. If the value is null or empty the classpath value is used.				
`debug`	`boolean`	`false`		No
If true, debugging code should be activated				
`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.				
`scriptExtensions`	`String[]`	`['groovy']`		No
Description is not available				
`sourceEncoding`	`String`	`UTF-8`		No
Encoding for source files				

Property	Type	Default	Encrypted [a]	Required [b]
scriptBaseClass	String	null		No
Base class name for scripts (must derive from Script)				
verbose	boolean	false		No
If true, the compiler should produce action information				
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.				
warningLevel	int	1		No
Warning Level of the compiler				
targetDirectory	File	null		No
Directory into which to write classes.				
classpath	String[]	[]		No
Classpath for use during compilation.				
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.

12.2.2. Operation Script Files Properties

Property	Type	Default	Encrypted [a]	Required [b]
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	UpdateMarketo.groovy		Update
The name of the file used to perform the UPDATE operation.				
searchScriptFileName	String	SearchMarketo.groovy		Get Search
The name of the file used to perform the SEARCH 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.				
createScriptFileName	String	CreateMarketo.groovy		Create
The name of the file used to perform the CREATE operation.				
authenticateScriptFileName	String	null		Authenticate
The name of the file used to perform the AUTHENTICATE operation.				
deleteScriptFileName	String	DeleteMarketo.groovy		Delete
The name of the file used to perform the DELETE 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.

12.2.3. Configuration Properties

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

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

[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

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.

The Active Directory connector will be deprecated in a future OpenICF release, and, ultimately, support for its use with OpenIDM will be discontinued. 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*".

13.1. 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 OpenIDM 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/provisioners/provisioner.openicf-ad.json`. On the OpenIDM host, copy the sample Active Directory connector configuration file to your project's `conf/` directory:

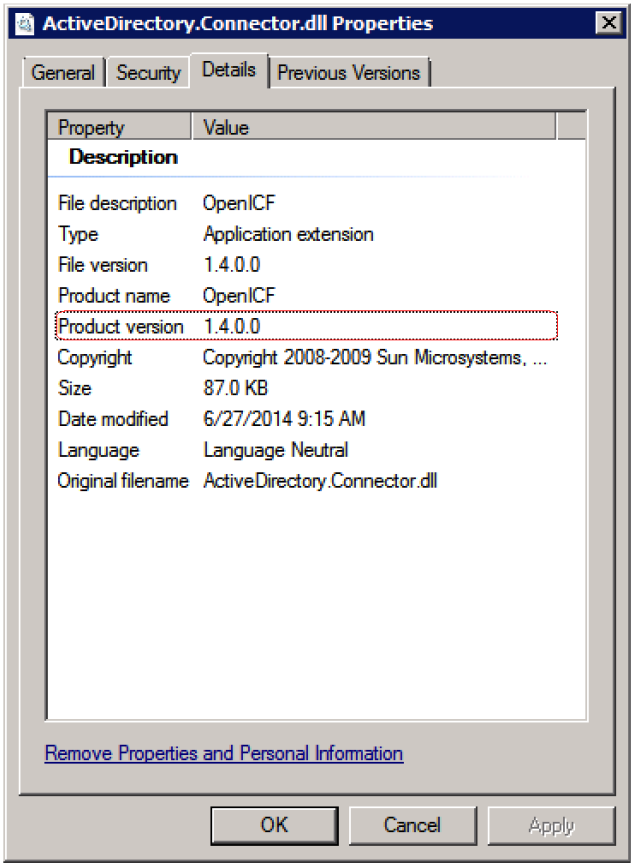
```
$ cd /path/to/openidm
$ cp samples/provisioners/provisioner.openicf-ad.json project-dir/conf/
```

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

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

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

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



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

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

The main configurable properties are as follows:

`connectorHostRef`

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

`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 OpenIDM 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 OpenIDM Admin UI, to make sure that the users in your Active Directory server were provisioned into the repository.

13.2. 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 14

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.

14.1. 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.

14.2. O365 Connector Configuration

The O365 Connector has the following configurable properties.

14.2.1. 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
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				

Property	Type	Default	Encrypted [a]	Required [b]
`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.

14.2.2. 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 15

XML File Connector

OpenIDM includes a simple XML file connector. This connector is really useful only in a demonstration context and should not be used in the general provisioning of XML data stores. In real deployments, if you need to connect to a custom XML data file, you should create your own scripted connector by using the Groovy connector toolkit.

The XML file connector is deprecated and support for its use in OpenIDM will be removed in a future release.

15.1. Configuring the XML File Connector

A sample XML connector configuration is provided in `path/to/openidm/samples/provisioners/provisioner.openicf-xml.json`. The following excerpt of the provisioner configuration shows the main configurable properties:

```
{
    "connectorRef": {
        "connectorHostRef": "#LOCAL",
        "bundleName": "org.forgerock.openicf.connectors.xml-connector",
        "bundleVersion": "[1.1.0.3,1.2.0.0)",
        "connectorName": "org.forgerock.openicf.connectors.xml.XMLConnector"
    }
}
```

The `connectorHostRef` is optional if the connector server is local.

The configuration properties for the XML file connector set the relative path to the file containing the identity data, and also the paths to the required XML schemas:

```
{
    "configurationProperties": {
        "xsdIcfFilePath" : "&{launcher.project.location}/data/resource-schema-1.xsd",
        "xsdFilePath" : "&{launcher.project.location}/data/resource-schema-extension.xsd",
        "xmlFilePath" : "&{launcher.project.location}/data/xmlConnectorData.xml"
    }
}
```

`&{launcher.project.location}` refers to the project directory of your OpenIDM instance, for example, `path/to/openidm/samples/sample1`. Note that relative paths such as these work only if your connector server runs locally. For remote connector servers, you must specify the absolute path to the schema and data files.

`xsdIcfFilePath`

References the XSD file defining schema common to all XML file resources. Do not change the schema defined in this file.

`xsdFilePath`

References custom schema defining attributes specific to your project.

`xmlFilePath`

References the XML file that contains account entries.

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.