ICF 1.5.20.30

July 7, 2025

ICF 1.5.20.30

Copyright

Trademark

Disclaimer

The information provided in Ping Identity product documentation is provided "as is" without warranty of any kind. Ping Identity disclaims all warranties, either express or implied, including the warranties of merchantability and fitness for a particular purpose. In no event shall Ping Identity or its suppliers be liable for any damages whatsoever including direct, indirect, incidental, consequential, loss of business profits or special damages, even if Ping Identity or its suppliers have been advised of the possibility of such damages. Some states do not allow the exclusion or limitation of liability for consequential or incidental damages so the foregoing limitation may not apply.

Table of Contents

Identity Connector Framework (OpenICF)

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

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

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

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

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

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

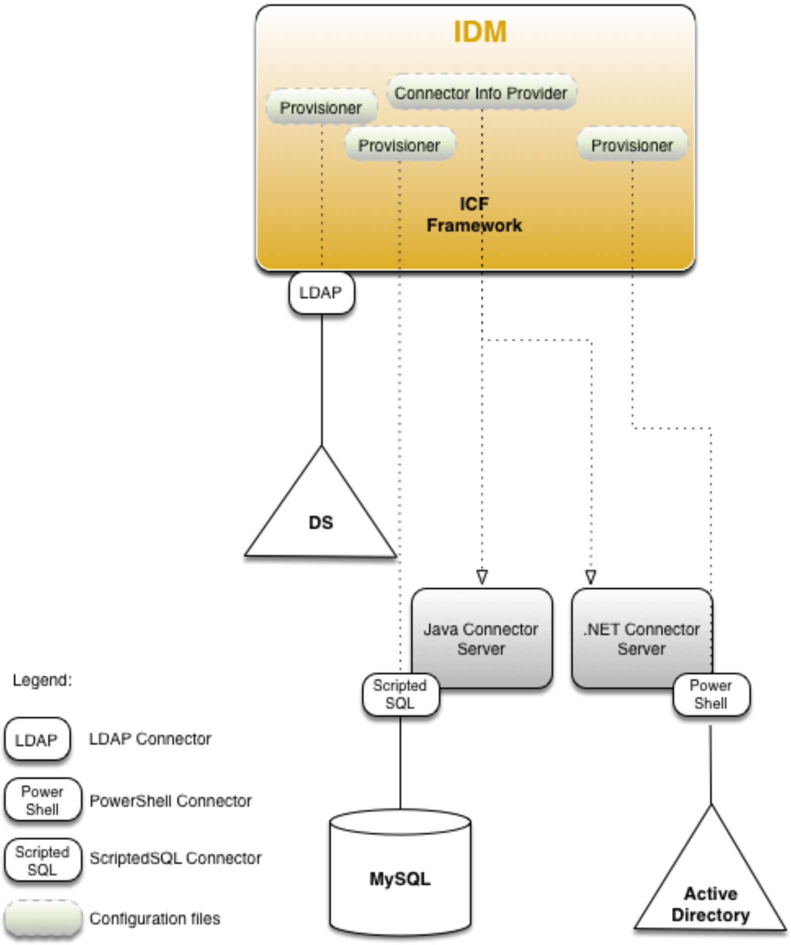


Figure 1. How IDM Uses the ICF Framework and Connectors

Tip

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

ICF configuration properties used by IDM

Use the following properties (in resolver/boot.properties) to specify how ICF should manage operations on external resources:

`openidm.icf.retry.enabled`

Specifies whether ICF operations should be retried, if network connectivity is lost. `False` by default.

`openidm.icf.retry.delaySeconds`

Delay, in seconds, between ICF retry operations. `10` by default.

`openidm.icf.retry.updates.enabled`

Specifies whether ICF update operations should be retried, if network connectivity is lost. `False` by default.

`openidm.icf.retry.maxRetries`

If `openidm.icf.retry.enabled=true`, specifies the maximum number of ICF operation retry attempts when network connectivity is lost. `12` by default.

Configure connectors

You configure connectors through the ICF provisioner service, and access them over REST at the `openidm/conf` endpoint.

Connector configurations are stored in files in your project's `conf/` directory, and are named `project-dir/conf/provisioner.openicf-name` where *name* corresponds to the name of the connector. If you are creating your own connector configuration files, *do not* include additional dash characters (`-`) in the connector name, as this can cause problems with the OSGi parser. For example, `provisioner.openicf-hrdb.json` is acceptable, and `provisioner.openicf-hr-db.json` is not.

You can create a connector configuration in the following ways:

- Start with the sample provisioner files in the `/path/to/openidm/samples/example-configurations/provisioners` directory. Learn more in the Sample Provisioner Files.
- Configure connectors in the admin UI. Sign on to the admin UI at `https://localhost:8443/admin`, then continue with the process described in Configure Connectors With the admin UI.
- Use the service that IDM exposes through the REST interface to create basic connector configuration files. Learn more in Configure Connectors Over REST.
- Use the `cli.sh` or `cli.bat` scripts to generate a basic connector configuration. Learn more in the `configureconnector` documentation.

Sample provisioner files

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

The following example shows a high-level connector configuration. The individual configuration objects are described in detail later in this section:

```
{
  "connectorRef"              : connector-ref-object,
  "producerBufferSize"        : integer,
  "poolConfigOption"          : pool-config-option-object,
  "operationTimeout"          : operation-timeout-object,
  "operationRateLimits"       : operation-rate-limits-object,
  "configurationProperties"   : configuration-properties-object,
  "syncFailureHandler"        : sync-failure-handler-object,
  "resultsHandlerConfig"      : results-handler-config-object,
  "excludeUnmodified"         : boolean, true/false,
  "objectTypes"               : object-types-object,
  "operationOptions"          : operation-options-object
}
```

Configure connectors with the admin UI

To configure connectors in the admin UI, select **Configure** > **Connector**.

If your project has an existing connector configuration (for example, if you have started IDM with one of the sample configurations), click on that connector to edit it. If you're starting with a new project, click **New Connector** to configure a new connector.

The connectors displayed on the **Connectors** page reflect the provisioner files in your project's `conf/` directory. To add a new connector configuration, you can also copy a provisioner file from the `/path/to/openidm/samples/example-configurations/provisioners` directory, then edit it to fit your deployment.

When you add a new connector, the Connector Type dropdown list reflects the connector .jar files that are in the `/path/to/openidm/connectors` directory. You can have more than one connector configuration for a specific connector type. For example, you might use the LDAP connector to set up two connector configurations—one to an Active Directory server and one to a PingDS (DS) instance.

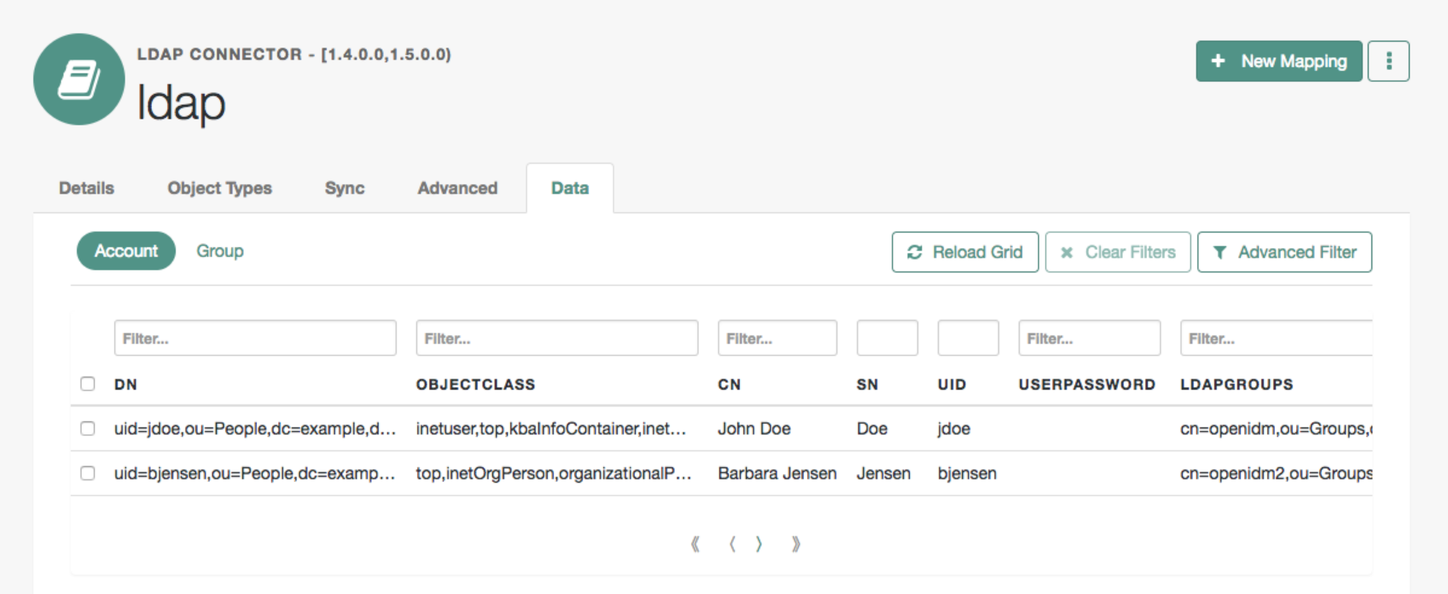
The Connector Types listed here do not include all supported connectors. The *scripted* connectors (such as scripted Groovy, scripted REST, scripted SQL, and PowerShell) are not available in the list of connector types. In general, the scripted connectors require extensive custom configuration changes, and a single HTML template to cover all possible permutations is not feasible. To add a scripted connector configuration, configure the connector over REST.

Alternatively, copy one of the example provisioner files in `/path/to/openidm/samples/example-configurations/provisioners` into your project's `conf` directory and edit the configuration directly in the provisioner file.

Additional connectors are available from the BackStage download site site. For connectors that are not bundled with IDM, the UI displays a generic template, based on the schema provided by the connector.

The tabs on the connector configuration screens correspond to the objects and properties described in the remaining sections of this chapter.

When a connector configuration is complete, and IDM is able to establish the connection to the remote resource, the Data tab displays the objects in that remote resource. For example, the following image shows the contents of a connected LDAP resource:



□	DN	OBJECTCLASS	CN	SN	UID	USERPASSWORD	LDAPGROUPS
□	uid=jdoe,ou=People,dc=example,d...	inetuser,top,kbaInfoContainer,inet...	John Doe	Doe	jdoe		cn=openidm,ou=Groups,(
□	uid=bjensen,ou=People,dc=examp...	top,inetOrgPerson,organizationalP...	Barbara Jensen	Jensen	bjensen		cn=openidm2,ou=Groups

Figure 1. Data Tab For a Connected LDAP Resource

You can search through these objects with either the Basic Filter shown in each column, or the Advanced Filter option, which lets you build many of the queries shown in Define and call data queries.

Configure connectors over REST

To create a new connector configuration over REST, follow these steps:

1. List the available connectors.
2. Generate the core configuration.
3. Add the target system properties, then connect to the target system to generate the final configuration.
4. Submit the final configuration to IDM.

This procedure walks you through creating a connector configuration over REST, for a CSV file connector.

1. List the available connectors.

   In a default IDM installation, the available connectors are installed in the `openidm/connectors` directory. If you are using a remote connector server, additional connectors might be available in the `openicf/connectors` directory on the remote server.

   Run the following command to list the available connectors:

   ```
   curl \
   --header "X-OpenIDM-Username: openidm-admin" \
   --header "X-OpenIDM-Password: openidm-admin" \
   --header "Accept-API-Version: resource=1.0" \
   --request POST \
   "http://localhost:8080/openidm/system?_action=availableConnectors"
   ```

   On a default IDM installation, this command returns the following output:

```
{
  "connectorRef": [
    {
      "displayName": "SSH Connector",
      "bundleVersion": "1.5.20.30",
      "systemType": "provisioner.openicf",
      "bundleName": "org.forgerock.openicf.connectors.ssh-connector",
      "connectorName": "org.forgerock.openicf.connectors.ssh.SSHConnector"
    },
    {
      "displayName": "ServiceNow Connector",
      "bundleVersion": "1.5.20.30",
      "systemType": "provisioner.openicf",
      "bundleName": "org.forgerock.openicf.connectors.servicenow-connector",
      "connectorName": "org.forgerock.openicf.connectors.servicenow.ServiceNowConnector"
    },
    {
      "displayName": "Scripted SQL Connector",
      "bundleVersion": "1.5.20.30",
      "systemType": "provisioner.openicf",
      "bundleName": "org.forgerock.openicf.connectors.scriptedsql-connector",
      "connectorName": "org.forgerock.openicf.connectors.scriptedsql.ScriptedSQLConnector"
    },
    {
      "displayName": "Scripted REST Connector",
      "bundleVersion": "1.5.20.30",
      "systemType": "provisioner.openicf",
      "bundleName": "org.forgerock.openicf.connectors.scriptedrest-connector",
      "connectorName": "org.forgerock.openicf.connectors.scriptedrest.ScriptedRESTConnector"
    },
    {
      "displayName": "SCIM Connector",
      "bundleVersion": "1.5.20.30",
      "systemType": "provisioner.openicf",
      "bundleName": "org.forgerock.openicf.connectors.scim-connector",
      "connectorName": "org.forgerock.openicf.connectors.scim.ScimConnector"
    },
    {
      "displayName":"Salesforce Connector",
      "bundleVersion":"1.5.20.30",
      "systemType":"provisioner.openicf",
      "bundleName":"org.forgerock.openicf.connectors.salesforce-connector",
      "connectorName":"org.forgerock.openicf.connectors.salesforce.SalesforceConnector"
    },
    {
      "displayName":"MSGraphAPI Connector",
      "bundleVersion":"1.5.20.30",
      "systemType":"provisioner.openicf",
      "bundleName":"org.forgerock.openicf.connectors.msgraphapi-connector",
      "connectorName":"org.forgerock.openicf.connectors.msgraphapi.MSGraphAPIConnector"
    },
    {
      "displayName": "MongoDB Connector",
      "bundleVersion": "1.5.20.30",
      "systemType": "provisioner.openicf",
      "bundleName": "org.forgerock.openicf.connectors.mongodb-connector",
      "connectorName": "org.forgerock.openicf.connectors.mongodb.MongoDBConnector"
    },
    {
      "displayName": "Marketo Connector",
      "bundleVersion": "1.5.20.30",
      "systemType": "provisioner.openicf",
      "bundleName": "org.forgerock.openicf.connectors.marketo-connector",
      "connectorName": "org.forgerock.openicf.connectors.marketo.MarketoConnector"
    },
    {
      "displayName": "LDAP Connector",
      "bundleVersion": "1.5.20.30",
      "systemType": "provisioner.openicf",
      "bundleName": "org.forgerock.openicf.connectors.ldap-connector",
      "connectorName": "org.identityconnectors.ldap.LdapConnector"
    },
    {
      "displayName": "Kerberos Connector",
      "bundleVersion": "1.5.20.30",
      "systemType": "provisioner.openicf",
      "bundleName": "org.forgerock.openicf.connectors.kerberos-connector",
      "connectorName": "org.forgerock.openicf.connectors.kerberos.KerberosConnector"
    },
    {
      "displayName": "Scripted Poolable Groovy Connector",
      "bundleVersion": "1.5.20.30",
      "systemType": "provisioner.openicf",
      "bundleName": "org.forgerock.openicf.connectors.groovy-connector",
      "connectorName": "org.forgerock.openicf.connectors.groovy.ScriptedPoolableConnector"
    },
    {
      "displayName": "Scripted Groovy Connector",
      "bundleVersion": "1.5.20.30",
      "systemType": "provisioner.openicf",
      "bundleName": "org.forgerock.openicf.connectors.groovy-connector",
      "connectorName": "org.forgerock.openicf.connectors.groovy.ScriptedConnector"
    },
    {
      "displayName": "GoogleApps Connector",
      "bundleVersion": "1.5.20.30",
      "systemType": "provisioner.openicf",
      "bundleName": "org.forgerock.openicf.connectors.googleapps-connector",
      "connectorName": "org.forgerock.openicf.connectors.googleapps.GoogleAppsConnector"
    },
    {
      "displayName": "Database Table Connector",
      "bundleVersion": "1.5.20.30",
      "systemType": "provisioner.openicf",
      "bundleName": "org.forgerock.openicf.connectors.databasetable-connector",
      "connectorName": "org.identityconnectors.databasetable.DatabaseTableConnector"
    },
    {
      "displayName": "CSV File Connector",
      "bundleVersion": "1.5.20.30",
      "systemType": "provisioner.openicf",
      "bundleName": "org.forgerock.openicf.connectors.csvfile-connector",
      "connectorName": "org.forgerock.openicf.csvfile.CSVFileConnector"
    },
    {
      "displayName": "Adobe Marketing Cloud Connector",
      "bundleVersion": "1.5.20.30",
      "systemType": "provisioner.openicf",
      "bundleName": "org.forgerock.openicf.connectors.adobecm-connector",
      "connectorName": "org.forgerock.openicf.acm.ACMConnector"
    }
  ]
}
```

2. Generate a core configuration.

   Locate the connector to configure from the previous step's output, and copy the JSON object to insert as the value of the `"connectorRef"` property in the `data` payload of the following command.

   This example generates a core configuration for the CSV file connector:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request POST \
--data '{
  "connectorRef": {
    "systemType": "provisioner.openicf",
    "bundleName": "org.forgerock.openicf.connectors.csvfile-connector",
    "connectorName": "org.forgerock.openicf.csvfile.CSVFileConnector",
    "displayName": "CSV File Connector",
    "bundleVersion": "1.5.20.30"
  }
}' \
"http://localhost:8080/openidm/system?_action=createCoreConfig"
```

   The command returns a connector configuration, similar to the following:

```
{
  "connectorRef": {
    "systemType": "provisioner.openicf",
    "bundleName": "org.forgerock.openicf.connectors.csvfile-connector",
    "connectorName": "org.forgerock.openicf.csvfile.CSVFileConnector",
    "displayName": "CSV File Connector",
    "bundleVersion": "1.5.20.30"
  },
  "resultsHandlerConfig": {
    "enableNormalizingResultsHandler": false,
    "enableFilteredResultsHandler": false,
    "enableCaseInsensitiveFilter": false,
    "enableAttributesToGetSearchResultsHandler": true
  },
  "operationTimeout": {
    "CREATE": -1,
    "UPDATE": -1,
    "DELETE": -1,
    "TEST": -1,
    "SCRIPT_ON_CONNECTOR": -1,
    "SCRIPT_ON_RESOURCE": -1,
    "GET": -1,
    "RESOLVEUSERNAME": -1,
    "AUTHENTICATE": -1,
    "SEARCH": -1,
    "VALIDATE": -1,
    "SYNC": -1,
    "SCHEMA": -1
  },
  "configurationProperties": {
    "headerPassword": "password",
    "spaceReplacementString": "_",
    "csvFile": null,
    "newlineString": "\n",
    "headerUid": "uid",
    "quoteCharacter": "\"",
    "escapeCharacter": "\\",
    "fieldDelimiter": ",",
    "syncFileRetentionCount": 3
  }
}
```

3. Connect to the target system to generate the final configuration.

   The configuration returned in the previous step is not functional. It does not include the required `configurationProperties` that are specific to the target system (such as the host name and port number of the target system, or the `csvFile` for a CSV file connector). It also doesn't include the complete list of `objectTypes` and `operationOptions`.

   To connect to the target system, add values for the required `configurationProperties`, and submit the updated configuration in the data payload of the following command.

   This example connects to the specified CSV file:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request POST \
--data '{
  "configurationProperties": {
    "headerPassword": "password",
    "spaceReplacementString": "_",
    "csvFile": "&{idm.instance.dir}/data/csvConnectorData.csv",
    "newlineString": "\n",
    "headerUid": "uid",
    "quoteCharacter": "\"",
    "fieldDelimiter": ",",
    "syncFileRetentionCount": 3
  },
  "connectorRef": {
    "systemType": "provisioner.openicf",
    "bundleName": "org.forgerock.openicf.connectors.csvfile-connector",
    "connectorName": "org.forgerock.openicf.csvfile.CSVFileConnector",
    "displayName": "CSV File Connector",
    "bundleVersion": "1.5.20.30"
  },
  "resultsHandlerConfig": {
    "enableNormalizingResultsHandler": true,
    "enableFilteredResultsHandler": true,
    "enableCaseInsensitiveFilter": false,
    "enableAttributesToGetSearchResultsHandler": true
  },
  "operationTimeout": {
    "CREATE": -1,
    "UPDATE": -1,
    "DELETE": -1,
    "TEST": -1,
    "SCRIPT_ON_CONNECTOR": -1,
    "SCRIPT_ON_RESOURCE": -1,
    "GET": -1,
    "RESOLVEUSERNAME": -1,
    "AUTHENTICATE": -1,
    "SEARCH": -1,
    "VALIDATE": -1,
    "SYNC": -1,
    "SCHEMA": -1
  }
}' \
"http://localhost:8080/openidm/system?_action=createFullConfig"
```

Note

The single quotes around the JSON object in the `--data` parameter prevent the command from being executed when a new line is encountered in the content. You can therefore include line feeds for readability.

With this command, IDM connects to the target resource, and attempts to read the schema, if it is available. It then iterates through the schema objects and attributes, and creates JSON representations of the supported objects and operations. The command output includes the JSON payload that you submitted, along with the `operationOptions` and `objectTypes`.

> **Important**
>
> Because IDM produces a full property set for all attributes and all object types in the schema, the resulting configuration can be very large. For an LDAP server, for example, IDM can generate a configuration containing several tens of thousands of lines. It might be useful to reduce the schema on the external resource to a minimum before you run the `createFullConfig` command.

4. When you have the final configuration, use a PUT request to add it to the IDM configuration, in the JSON payload of the following command:

   ```
   curl \
   --header "X-OpenIDM-Username: openidm-admin" \
   --header "X-OpenIDM-Password: openidm-admin" \
   --header "Accept-API-Version: resource=1.0" \
   --header "Content-Type: application/json" \
   --request PUT \
   --data '{complete-configuration}' \
   "http://localhost:8080/openidm/config/provisioner.openicf/connectorName"
   ```

   Alternatively, you can save the complete configuration in a file named `provisioner.openicf-connectorName.json`, and place the file in the `conf` directory of your project.

Connector reference properties

The following example shows a connector reference object:

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

`bundleName`

string, required

The `ConnectorBundle-Name` of the ICF connector.

`bundleVersion`

string, required

The `ConnectorBundle-Version` of the ICF connector. The value can be a single version (such as `1.4.0.0`) or a range of versions, which lets you support multiple connector versions in a single project.

You can specify a range of versions as follows:

- `[1.1.0.0,1.4.0.0]` indicates that all connector versions from 1.1 to 1.4, inclusive, are supported.
- `[1.1.0.0,1.4.0.0)` indicates that all connector versions from 1.1 to 1.4, including 1.1 but excluding 1.4, are supported.
- `(1.1.0.0,1.4.0.0]` indicates that all connector versions from 1.1 to 1.4, excluding 1.1 but including 1.4, are supported.
- `(1.1.0.0,1.4.0.0)` indicates that all connector versions from 1.1 to 1.4, exclusive, are supported.

When a range of versions is specified, IDM uses the latest connector that is available within that range. If your project requires a specific connector version, you must explicitly state the version in your connector configuration file, or constrain the range to address only the version that you need.

`connectorName`

string, required

The connector implementation class name.

`connectorHostRef`

string, optional

If the connector runs remotely, the value of this field must match the `name` field of the `RemoteConnectorServers` object in the connector server configuration file (`provisioner.openicf.connectorinfoprovider.json`). For example:

```
...
    "remoteConnectorServers" :
        [
            {
                "name" : "dotnet",
                ...
            }
        ]
...
```

If the connector runs locally, the value of this field can be one of the following:

- If the connector .jar is installed in `openidm/connectors/` , the value must be `"#LOCAL"` . This is currently the default, and recommended location.
- If the connector .jar is installed in `openidm/bundle/` (not recommended), the value must be `"osgi:service/org.forgerock.openicf.framework.api.osgi.ConnectorManager"` .

Pool configuration

Learn more about Connection pooling configuration and the connectors that use each pooling mechanism in Connector by pooling mechanism.

Operation timeouts

The operation timeout property enables you to configure timeout values per operation type. By default, no timeout is configured for any operation type. A sample configuration follows:

```
{
  "CREATE"              : -1,
  "TEST"                : -1,
  "AUTHENTICATE"        : -1,
  "SEARCH"              : -1,
  "VALIDATE"            : -1,
  "GET"                 : -1,
  "UPDATE"              : -1,
  "DELETE"              : -1,
  "SCRIPT_ON_CONNECTOR" : -1,
  "SCRIPT_ON_RESOURCE"  : -1,
  "SYNC"                : -1,
  "SCHEMA"              : -1
}
```

operation-name

Timeout in milliseconds

A value of `-1` disables the timeout.

Operation rate limits

The `operationRateLimits` property enables you to configure rate limit values per operation type. By default, no rate limit is configured for any operation type. A sample configuration follows:

```
"operationRateLimits": {
  "CREATE": {
    "requestLimit": 50,
    "requestPeriod": 500,
    "requestTimeout": 5000
  }
}
```

`operation-name`

`requestLimit`

The number of requests allowed over a period of time (`requestPeriod`). The default value is `50` requests.

`requestPeriod`

The request limit resets after this period of time (in milliseconds). The default value is `500` milliseconds.

For example, using the previous example configuration allows 50 requests in a 500 millisecond period of time.

`requestTimeout`

The amount of time (in milliseconds) before throwing an `OperationTimeoutException` for an operation. The default is `5000` milliseconds (5 seconds).

Connection configuration

The `configurationProperties` object specifies the configuration for the connection between the connector and the resource, and is therefore resource-specific.

The following example shows a configuration properties object for the default CSV sample resource connector:

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

property

Individual properties depend on the type of connector.

Synchronization failure configuration

The `syncFailureHandler` object specifies what should happen if a liveSync operation reports a failure for an operation. The following example shows a synchronization failure configuration:

```
{
    "maxRetries" : 5,
    "postRetryAction" : "logged-ignore"
}
```

`maxRetries`

positive integer or `-1`, required

The number of attempts that IDM should make to process a failed modification. A value of zero indicates that failed modifications should not be reattempted. In this case, the post retry action is executed immediately when a liveSync operation fails. A value of -1 (or omitting the `maxRetries` property, or the entire `syncFailureHandler` object) indicates that failed modifications should be retried an infinite number of times. In this case, no post retry action is executed.

`postRetryAction`

string, required

The action that should be taken if the synchronization operation fails after the specified number of attempts. The post retry action can be one of the following:

- `logged-ignore` - IDM ignores the failed modification, and logs its occurrence.
- `dead-letter-queue` - IDM saves the details of the failed modification in a table in the repository (accessible over REST at `repo/synchronisation/deadLetterQueue/provisioner-name`).

- `script` specifies a custom script that should be executed when the maximum number of retries has been reached.

Learn more in Configure the LiveSync Retry Policy.

Configure how results are handled

The `resultsHandlerConfig` object specifies how OpenICF returns results. These configuration properties do not apply to all connectors and depend on the interfaces that are implemented by each connector. For information about the interfaces that connectors support, refer to the Connector reference.

The following example shows a results handler configuration object:

```
"resultsHandlerConfig" : {
    "enableNormalizingResultsHandler" : true,
    "enableFilteredResultsHandler" : false,
    "enableCaseInsensitiveFilter" : false,
    "enableAttributesToGetSearchResultsHandler" : false
}
```

enableNormalizingResultsHandler

boolean, false by default

When this property is enabled, ICF normalizes returned attributes to ensure that they are filtered consistently. If the connector implements the attribute normalizer interface, enable the interface by setting this property to `true`. If the connector does not implement the attribute normalizer interface, the value of this property has no effect.

enableFilteredResultsHandler

boolean, false by default

Most connectors use the filtering and search capabilities of the remote connected system. In these cases, you can leave this property set to `false`. If the connector does not use the remote system's filtering and search capabilities, you *must* set this property to `true`.

All the non-scripted connectors, except for the CSV connector, use the filtering mechanism of the remote system. In the case of the CSV connector, the remote resource has no filtering mechanism, so you must set `enableFilteredResultsHandler` to `true`. For the scripted connectors, the setting will depend on how you have implemented the connector.

enableCaseInsensitiveFilter

boolean, false by default

This property applies only if `enableFilteredResultsHandler` is set to `true`. The filtered results handler is case-sensitive by default. For example, a search for `lastName = "Jensen"` will not match a stored user with `lastName : jensen`. When the filtered results handler is enabled, you can use this property to enable case-insensitive filtering. If you leave this property set to `false`, searches on that resource will be case-sensitive.

`enableAttributesToGetSearchResultsHandler`

boolean, false by default

By default, IDM determines which attributes should be retrieved in a search. If you set this property to `true`, the ICF framework removes *all* attributes from the READ/QUERY response, except for those that are specifically requested. For performance reasons, you should set this property to `false` for local connectors and to `true` for remote connectors.

Specify which attributes are updated

The `excludeUnmodified` property determines which properties are updated during synchronization. When this property is set to `true`, synchronization operations update *only* the modified properties on a target resource, rather than the whole target object. The default behavior is to include all attributes. In the sample LDAP provisioner files provided with IDM, `excludeUnmodified` is set to `true`, so unmodified attributes are excluded during update operations.

Set the supported object types

The `objectTypes` configuration specifies the object types (user, group, account, and so on) that are supported by the connector. The object names that you define here determine how the object is accessed in the URI. For example:

```
system/systemName/objectType
```

This configuration is based on the JSON Schema with the extensions described in the following section.

Attribute names that start or end with `__` are regarded as *special attributes* by OpenICF. The purpose of the special attributes in ICF is to enable someone who is developing a *new* connector to create a contract regarding how a property can be referenced, regardless of the application that is using the connector. In this way, the connector can map specific object information between an arbitrary application and the resource, without knowing how that information is referenced in the application.

These attributes have no specific meaning in the context of IDM, although some of the connectors that are bundled with IDM use these attributes. The generic LDAP connector, for example, can be used with PingDS (DS), Active Directory, OpenLDAP, and other LDAP directories. Each of these directories might use a different attribute name to represent the same type of information. For example, Active Directory uses `unicodePassword` and DS uses `userPassword` to represent the same thing, a user's password. The LDAP connector uses the special OpenICF `__PASSWORD__` attribute to abstract that difference. In the same way, the LDAP connector maps the `__NAME__` attribute to an LDAP `dn`.

The ICF `__UID__` is a special case. The `__UID__` *must not* be included in the IDM configuration or in any update or create operation. This attribute denotes the unique identity attribute of an object and IDM always maps it to the `_id` of the object.

The following excerpt shows the configuration of an `account` object type:

```
{
    "account" : {
        "$schema" : "http://json-schema.org/draft-03/schema",
        "id" : "__ACCOUNT__",
        "type" : "object",
        "nativeType" : "__ACCOUNT__",
        "absentIfEmpty" : false,
        "absentIfNull" : true,
        "properties" : {
            "name" : {
                "type" : "string",
                "nativeName" : "__NAME__",
                "nativeType" : "JAVA_TYPE_PRIMITIVE_LONG",
                "flags" : [
                    "NOT_CREATABLE",
                    "NOT_UPDATEABLE",
                    "NOT_READABLE",
                    "NOT_RETURNED_BY_DEFAULT"
                ]
            },
            "groups" : {
                "type" : "array",
                "items" : {
                    "type" : "string",
                    "nativeType" : "string"
                },
                "nativeName" : "__GROUPS__",
                "nativeType" : "string",
                "flags" : [
                    "NOT_RETURNED_BY_DEFAULT"
                ]
            },
            "givenName" : {
                "type" : "string",
                "nativeName" : "givenName",
                "nativeType" : "string"
            },
        }
    }
}
```

ICF supports an `__ALL__` object type that ensures that objects of every type are included in a synchronization operation. The primary purpose of this object type is to prevent synchronization errors when multiple changes affect more than one object type.

For example, imagine a deployment synchronizing two external systems. On system A, the administrator creates a user, `jdoe`, then adds the user to a group, `engineers`. When these changes are synchronized to system B, if the `__GROUPS__` object type is synchronized first, the synchronization will fail, because the group contains a user that does not yet exist on system B. Synchronizing the `__ALL__` object type ensures that user `jdoe` is created on the external system before he is added to the group `engineers`.

The `__ALL__` object type is assumed by default - you do not need to declare it in your provisioner configuration file. If it is not declared, the object type is named `__ALL__`. If you want to map a different name for this object type, declare it in your provisioner configuration. The following excerpt from a sample provisioner configuration uses the name `allobjects`:

```
"objectTypes": {
    "allobjects": {
        "$schema": "http://json-schema.org/draft-03/schema",
        "id": "__ALL__",
        "type": "object",
        "nativeType": "__ALL__"
    },
    ...
}
```

A liveSync operation invoked with no object type assumes an object type of `__ALL__`. For example, the following call invokes a liveSync operation on all defined object types in an LDAP system:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system/ldap?_action=liveSync"
```

> **Note**
>
> Using the `__ALL__` object type requires a mechanism to ensure the order in which synchronization changes are processed. Servers that use the `cn=changelog` mechanism to order sync changes, such as PingDS (DS), Oracle DSEE, and the legacy Sun Directory Server, cannot use the `__ALL__` object type by default. Such servers must be forced to use timestamps to order their sync changes. For these LDAP server types, set `useTimestampsForSync` to `true` in the provisioner configuration.
>
> Additionally, you can use the `timestampSyncOffset` LDAP configuration property to account for replication delays between LDAP instances. Refer to LDAP connector.
>
> LDAP servers that use timestamps rather than change logs (such as Active Directory GCs and OpenLDAP) can use the `__ALL__` object type without any additional configuration. Active Directory and Active Directory LDS, which use Update Sequence Numbers, can also use the `__ALL__` object type without additional configuration.

Add objects and properties through the UI

To add object types and properties to a connector configuration by using the admin UI, select **Configure > Connectors**. Select the connector that you want to change, then select the **Object Types** tab.

In the case of the LDAP connector, the connector reads the schema from the remote resource to determine the object types and properties that can be added to its configuration. When you select one of these object types, you can think of it as a template. Edit the basic object type, as required, to suit your deployment.

To add a property to an object type, select the **Edit** icon next to the object type, then select **Add Property**.

Specify object types on the external resource

At the object level, the `nativeType` property refers to an object type supported by a connector or external resource. For example, an LDAP connector might have object types such as `__ACCOUNT__` and `__GROUP__`.

`nativeType`

string, optional

The native ICF object type.

The value of this property must be an object type supported by the resource or the connector.

Behavior for empty attributes

The `absentIfEmpty` and `absentIfNull` object class properties enable you to specify how attributes are handled during synchronization if their values are null (for single-valued attributes) or empty (for multivalued attributes). You can set these properties per object type.

By default, these properties are set as follows:

`"absentIfEmpty" : false`

Multivalued attributes whose values are empty are included in the resource response during synchronization.

`"absentIfNull" : true`

Single-valued attributes whose values are null are removed from the resource response during synchronization.

Specify attribute types on the external resource

At the property level, `nativeType` refers to the data type of an attribute on the external resource.

`nativeType`

string, optional

The native ICF attribute type.

The following native types are supported:

```
JAVA_TYPE_BIGDECIMAL
JAVA_TYPE_BIGINTEGER
JAVA_TYPE_BYTE
JAVA_TYPE_BYTE_ARRAY
JAVA_TYPE_CHAR
JAVA_TYPE_CHARACTER
JAVA_TYPE_DATE
JAVA_TYPE_DOUBLE
JAVA_TYPE_FILE
JAVA_TYPE_FLOAT
JAVA_TYPE_GUARDEDBYTEARRAY
JAVA_TYPE_GUARDEDSTRING
JAVA_TYPE_INT
JAVA_TYPE_INTEGER
JAVA_TYPE_LONG
JAVA_TYPE_OBJECT
JAVA_TYPE_PRIMITIVE_BOOLEAN
JAVA_TYPE_PRIMITIVE_BYTE
JAVA_TYPE_PRIMITIVE_DOUBLE
JAVA_TYPE_PRIMITIVE_FLOAT
JAVA_TYPE_PRIMITIVE_LONG
JAVA_TYPE_STRING
```

> **Note**
>
> - IDM only handles JSON primitive types (`boolean` , `map` , `list` , `number` , and `string`). You must encode any non-JSON primitive types so that they can be stored as JSON.
>   As a general rule, your connector configurations should always map the property type on the external resource (`nativeType`) to a supported JSON primitive type in IDM. If you are synchronizing pre-hashed passwords, set the `nativeType` to a `JAVA_TYPE_BYTE_ARRAY` , and the IDM `type` to a `string` , for example:
>
>   ```
>   ...
>       "userPassword" : {
>           "type" : "string",
>           "nativeName" : "userPassword",
>           "nativeType" : "JAVA_TYPE_BYTE_ARRAY"
>       },
>   ...
>   ```
>
>   With this configuration, when a `userPassword` is read from the remote system, it is returned as a `Byte[]` by the connector. It is then converted to a `String` (Base64-encoded `Byte[]`) by IDM.
>   Alternatively, you can make sure that any non-JSON primitive types returned by your connector are appropriately transformed into an encoded `string` value in your mapping. For example:
>
>   ```
>   {
>       "source": "password",
>       "target": "password",
>       "transform": {
>           "type": "text/javascript",
>           "source": "source.toString();"
>       }
>   },
>   ```
>
> - The `JAVA_TYPE_DATE` property is deprecated. Functionality may be removed in a future release. This type is an alias for `string` . Any dates with this type should be formatted according to ISO 8601.

`nativeName`

string, optional

The native ICF attribute name.

`flags`

string, optional

The native ICF attribute flags. ICF supports the following attribute flags:

`MULTIVALUED`

The property can be multivalued.

For multivalued properties, if the property value type is anything other than a `string` , you *must* include an `items` property that declares the data type.

The following example shows the `entries` property of the `authentication` object in a provisioner file. The `entries` property is multivalued, and its elements are of type `object` :

```
"authentication" : {
    ...
    "properties" : {
        ...
        "entries" : {
            "type" : "object",
            "required" : false,
            "nativeName" : "entries",
            "nativeType" : "object",
                "items" : {
                    "type" : "object"
                },
            "flags" : [
                "MULTIVALUED"
            ]
        },
        ...
    },
    ...
}
```

Important

When comparing multivalued properties across systems, the *order* of the values is important. Two properties with the same values, but in different orders, will be seen as a *change* during reconciliation, regardless of whether the value has actually changed.

NOT_CREATABLE, NOT_READABLE, NOT_UPDATEABLE

In some cases, the connector might not support manipulating an attribute because the attribute can only be changed directly on the remote system. For example, if the `name` attribute of an account can only be created by Active Directory, and *never* changed by IDM, you would add `NOT_CREATABLE` and `NOT_UPDATEABLE` to the provisioner configuration for that attribute.

NOT_RETURNED_BY_DEFAULT

Some attributes, such as LDAP groups or other calculated attributes, can be expensive to read. To avoid returning these attributes in a default read of the object, unless they are explicitly requested, add the `NOT_RETURNED_BY_DEFAULT` flag to the provisioner configuration for that attribute.

You can also use this flag to prevent properties from being read by default during a synchronization operation. To synchronize changes to a target object, IDM performs an UPDATE rather than a PATCH. This causes *all* attributes that are mapped from the source to the target to be modified when the synchronization is processed (rather than only those attributes that have changed). Although the *value* of a property might not change, the property still registers an update. This behavior can be problematic for properties such as the `password`, which might have restrictions on updating with a similar value. To prevent such properties from being updated during synchronization, set the `NOT_RETURNED_BY_DEFAULT` flag, which effectively prevents the property from being read from the source during the synchronization. For example:

```
"__PASSWORD__" : {
    "type" : "string",
    "nativeName" : "__PASSWORD__",
    "nativeType" : "JAVA_TYPE_GUARDEDSTRING",
    "flags" : [
        "NOT_RETURNED_BY_DEFAULT"
    ],
    "runAsUser" : true
}
```

You can configure connectors to enable provisioning of any arbitrary property. For example, the following property definitions would enable you to provision image files, used as avatars, to `account` objects in a system resource. The first definition would work for a single photo encoded as a base64 string. The second definition would work for multiple photos encoded in the same way:

```
"attributeByteArray" : {
    "type" : "string",
    "nativeName" : "attributeByteArray",
    "nativeType" : "JAVA_TYPE_BYTE_ARRAY"
},
```

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

Note

Do not use the dash character (`-`) in property names, like `last-name` . Dashes in names make JavaScript syntax more complex. If you cannot avoid the dash, write `source['last-name']` instead of `source.last-name` in your JavaScript scripts.

Configure operation options

The `operationOptions` object enables you to deny specific operations on a resource. For example, you can use this configuration object to deny `CREATE` and `DELETE` operations on a read-only resource to avoid IDM accidentally updating the resource during a synchronization operation.

The following example defines the options for the `"SYNC"` operation:

```
"operationOptions" : {
    "SYNC" : {
        "denied" : true,
        "onDeny" : "DO_NOTHING",
        "objectFeatures" : {
            "__ACCOUNT__" : {
                "denied" : true,
                "onDeny" : "THROW_EXCEPTION",
                "operationOptionInfo" : {
                    "$schema" : "http://json-schema.org/draft-03/schema",
                    "type" : "object",
                    "properties" : {
                        "_OperationOption-float" : {
                            "type" : "number",
                            "nativeType" : "JAVA_TYPE_PRIMITIVE_FLOAT"
                        }
                    }
                }
            },
            "__GROUP__" : {
                "denied" : false,
                "onDeny" : "DO_NOTHING"
            }
        }
    },
    ...
}
```

The ICF Framework supports the following operations:

- `AUTHENTICATE`
- `CREATE`
- `DELETE`
- `GET`
- `RESOLVEUSERNAME`
- `SCHEMA`
- `SCRIPT_ON_CONNECTOR`
- `SCRIPT_ON_RESOURCE`
- `SEARCH`
- `SYNC`
- `TEST`
- `UPDATE`
- `VALIDATE`

For detailed information on these operations, refer to the ICF API documentation.

The `operationOptions` object has the following configurable properties:

`denied`

boolean, optional

This property prevents operation execution if the value is `true`.

`onDeny`

string, optional

If `denied` is `true`, then the service uses this value. Default value: `DO_NOTHING`.

- `DO_NOTHING`: On operation the service does nothing.
- `THROW_EXCEPTION`: On operation the service throws a `ForbiddenException` exception.

Remove a connector

> **Caution**
>
> If you remove a connector used in a mapping while it's part of a scheduled task, you could experience unintended consequences.

If you're removing a connector, consider the following checklist. Depending on your configuration, this list might not be comprehensive:

- Consider the remote resource. Make sure you no longer need data from that resource and that the resource no longer requires data from IDM.
- Open the `sync.json` file for your project. Delete the code block associated with the mapping.
- Review the `schedule-recon.json` file. If it contains the schedule for a single operation, delete the file or update it as a schedule for a different mapping.

When these steps are complete, you can delete the connector configuration file, typically named `provisioner-*.json`.

You can also delete the connector using the admin UI:

1. Sign on as `openidm-admin` and click **Configure** > **Connectors**.
2. On the **Connectors** page, find the connector to delete and click ⋮ > **Delete**.
3. In the **Confirm Delete** modal, click **Delete**. The admin UI automatically makes the specified changes to the noted configuration files.

External system status

After you configure an external connection, you can access those systems and their data objects using the REST interface at `http://localhost:8080/openidm/system/connectorName`. You can also test external system availability.

To list the external systems that are connected to an IDM instance, use the `test` action on the URL `http://localhost:8080/openidm/system/`. The following example shows an IDM system with two connected LDAP systems:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system?_action=test"
[
  {
    "name": "ldap",
    "enabled": true,
    "config": "config/provisioner.openicf/ldap",
    "connectorRef": {
      "bundleVersion": "[1.5.0.0,1.6.0.0)",
      "bundleName": "org.forgerock.openicf.connectors.ldap-connector",
      "connectorName": "org.identityconnectors.ldap.LdapConnector"
    },
    "displayName": "LDAP Connector",
    "objectTypes": [
      "__ALL__",
      "account",
      "group"
    ],
    "ok": true
  },
  {
    "name": "ldap2",
    "enabled": true,
    "config": "config/provisioner.openicf/ldap2",
    "connectorRef": {
      "bundleVersion": "[1.5.0.0,1.6.0.0)",
      "bundleName": "org.forgerock.openicf.connectors.ldap-connector",
      "connectorName": "org.identityconnectors.ldap.LdapConnector"
    },
    "displayName": "LDAP Connector",
    "objectTypes": [
      "__ALL__",
      "account",
      "group"
    ],
    "ok": true
  }
]
```

The status of the system is provided by the `ok` parameter. If the connection is available, the value of this parameter is `true`.

To obtain the status for a single system, include the name of the connector in the URL, for example:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system/ldap?_action=test"
{
  "name": "ldap",
  "enabled": true,
  "config": "config/provisioner.openicf/ldap",
  "connectorRef": {
    "bundleVersion": "[1.5.0.0,1.6.0.0)",
    "bundleName": "org.forgerock.openicf.connectors.ldap-connector",
    "connectorName": "org.identityconnectors.ldap.LdapConnector"
  },
  "displayName": "LDAP Connector",
  "objectTypes": [
    "__ALL__",
    "account",
    "group"
  ],
  "ok": true
}
```

If there is a problem with the connection, the `ok` parameter returns `false`, with an indication of the error. In the following example, the LDAP server named `ldap`, running on `localhost:1389`, is down:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system/ldap?_action=test"
{
  "name": "ldap",
  "enabled": true,
  "config": "config/provisioner.openicf/ldap",
  "connectorRef": {
    "bundleVersion": "[1.5.0.0,1.6.0.0)",
    "bundleName": "org.forgerock.openicf.connectors.ldap-connector",
    "connectorName": "org.identityconnectors.ldap.LdapConnector"
  },
  "displayName": "LDAP Connector",
  "objectTypes": [
    "__ALL__",
    "account",
    "group"
  ],
  "error": "javax.naming.CommunicationException: localhost:1389 [Root exception
  is java.net.ConnectException: Connection refused (Connection refused)]",
  "ok": false
}
```

To test the validity of a connector configuration, use the `testConfig` action and include the configuration in the command. For example:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request POST \
--data '{
  "configurationProperties": {
    "headerPassword": "password",
    "csvFile": "&{idm.instance.dir}/data/csvConnectorData.csv",
    "newlineString": "\n",
    "headerUid": "uid",
    "quoteCharacter": "\"",
    "fieldDelimiter": ",",
    "syncFileRetentionCount": 3
  },
  "connectorRef": {
    "systemType": "provisioner.openicf",
    "bundleName": "org.forgerock.openicf.connectors.csvfile-connector",
    "connectorName": "org.forgerock.openicf.csvfile.CSVFileConnector",
    "displayName": "CSV File Connector",
    "bundleVersion": "[1.5.0.0,1.6.0.0)"
  },
  "resultsHandlerConfig": {
    "enableNormalizingResultsHandler": true,
    "enableFilteredResultsHandler": true,
    "enableCaseInsensitiveFilter": false,
    "enableAttributesToGetSearchResultsHandler": true
  },
  "operationTimeout": {
    "CREATE": -1,
    "UPDATE": -1,
    "DELETE": -1,
    "TEST": -1,
    "SCRIPT_ON_CONNECTOR": -1,
    "SCRIPT_ON_RESOURCE": -1,
    "GET": -1,
    "RESOLVEUSERNAME": -1,
    "AUTHENTICATE": -1,
    "SEARCH": -1,
    "VALIDATE": -1,
    "SYNC": -1,
    "SCHEMA": -1
  }
 }' \
 "http://localhost:8080/openidm/system?_action=testConfig"
```

If the configuration is valid, the command returns `"ok": true`, for example:

```
{
   "ok": true
}
```

If the configuration is not valid, the command returns an error, indicating the problem with the configuration. For example, the following result is returned when the LDAP connector configuration is missing a required property (in this case, the `baseContexts` to synchronize):

```
{
  "error": "org.identityconnectors.framework.common.exceptions.ConfigurationException: The list of base
contexts cannot be empty",
  "name": "ldap",
  "ok": false
}
```

The `testConfig` action requires a running IDM instance, as it uses the REST API, but does not require an active connector instance for the connector whose configuration you want to test.

Test connector servers

You can test the availability of connector servers using the `testConnectorServers` action on the `http://localhost:8080/openidm/system` endpoint.

The following example shows a single connected RCS:

Request

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system?_action=testConnectorServers"
```

Return

```
{
  "openicf": [
    {
      "name": "rcs1",
      "type": "remoteConnectorServer",
      "ok": true
    }
  ]
}
```

ICF interfaces

The ICF framework supports the following interfaces:

> **Note**
>
> Some connectors support only a subset of these interfaces.

AttributeNormalizer

Normalize attributes to ensure consistent filtering.

Authenticate

Provides simple authentication with two parameters, presumed to be a username and password. IDM requires the connector to implement the `AuthenticateOp` interface to provide pass-through authentication.

The following connectors support pass-through authentication using the `AuthenticateOp` interface by default:

- LDAP connector
- CSV file connector
- Database Table connector
- Microsoft Graph API connector
- Scripted SQL connector

> **Note**
>
> All Scripted Groovy-based connectors are capable of pass-through authentication if the `AuthenticateScript.groovy` script is implemented, but the only default implementation is the ScriptedSQL connector. Learn more in Authenticate script and Authenticate operation.

Batch

Execute a series of operations in a single request. If a resource doesn't support batch operations, the connector won't implement the batch operation interface. The ICF framework will still support batched requests but the operations will be executed iteratively through the connector.

Connector Event

Subscribe for notification of any specified event on the target resource. This operation can be used in the context of IoT device reports, to receive notification of events such as low battery signals, inactive devices, and so on.

Create

Create an object and return its UID.

Delete

Delete an object by its UID.

Get

Get an object by its UID.

PoolableConnector

Use pools of target resources.

Resolve Username

Resolve an object to its UID based on its username.

Schema

Describe supported object types, operations, and options.

Script on Connector

Allow script execution on the connector.

Script on Resource

Allow script execution on the resource.

Search

Allow searches for resource objects.

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

Sync

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

Sync Event

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

Test

Test the connection configuration, including connecting to the resource.

Update

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

Update Attribute Values

Allows an authorized caller to update (modify or replace) attribute values on the target resource. This operation is more advanced than the `UpdateOp` operation, and provides better performance and atomicity semantics.

ICF operation options

The ICF framework supports the following predefined operation options:

> **Note**
>
> Some connectors support only a subset of these options.

Scope

An option to use with Search (in conjunction with the Container option) that specifies how far beneath the container to search. Must be one of the following values:

- `SCOPE_OBJECT`
- `SCOPE_ONE_LEVEL`
- `SCOPE_SUBTREE`

Container

An option to use with Search that specifies the container under which to perform the search. Must be of type `QualifiedUid`. Should be implemented for those object classes whose `ObjectClassInfo.isContainer()` returns true.

Run as User

An option that specifies an account under which to execute the script or operation. The specified account will appear to have performed any action that the script or operation performs.

Run with Password

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

Attributes to Get

Determines which attributes to retrieve during Search and Sync. This option overrides the default behavior, which is for the connector to return the precise set of attributes identified as returned by default in the schema for that connector.

This option allows a client application to request additional attributes that would not otherwise not be returned (generally because such attributes are more expensive for a connector to fetch and to format) or to request only a subset of the attributes that would normally be returned.

Paged Results Cookie

An option to use with Search that specifies an opaque cookie, used by the connector to track its position in the set of query results.

Paged Results Offset

An option to use with Search that specifies the index within the result set of the first result which should be returned.

Page Size

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

Sort Keys

An option to use with Search that specifies the sort keys that should be used for ordering the connector object returned by search request.

Fail on Error

This option is used with the Batch operation to specify whether the batch process should be aborted when the first error is encountered. The default behavior is to continue processing regardless of errors.

Require Serial

This option instructs the connector to execute batched requests in a serial manner, if possible. The default behavior of the Batch operation is to execute requests in parallel, for speed and efficiency. In either case the task ID must be reflected in the response for each task so that tasks can be correctly reordered.

Connection pooling configuration

Certain connectors can be pooled, while other connectors are non-poolable. Learn more about the five Pooling mechanisms used in OpenICF and understand Connectors by pooling mechanism.

Pooling mechanisms

If a connector depends on a third-party library that has its own pooling mechanism, then ICF leverages that pooling mechanism. For example, an HTTP connector uses an Apache HTTP client and a DB connector uses a Tomcat JDBC.

OpenICF uses an ICF pooling mechanism only if:

- A connector has no third-party library pooling mechanism, or
- If OpenICF can’t control the amount of connections the third-party library uses. For example, the Microsoft Graph API connector.

OpenICF uses the following pooling mechanisms to manage connections.

ICF pooling

ICF pooling maintains connector objects and is managed by the framework. This pooling mechanism improves throughput and is used by, and is only relevant to, the connectors that implement the `PoolableConnector` interface.

For an ICF-pooled connector, ICF maintains a pool of connector instances and reuses these instances for connector operations. When an operation must be run, 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 run, the connector instance is released back into the connector pool, ready to be used for a subsequent operation.

The following example shows a pool configuration option object for an ICF poolable connector:

```
{
  "maxObjects"                 : 10,
  "maxIdle"                    : 10,
  "maxWait"                    : 150000,
  "minEvictableIdleTimeMillis" : 120000,
  "minIdle"                    : 1
}
```

To configure ICF connection pooling, set the following values in the connector configuration file `poolConfigOptions` property:

`maxObjects`

The maximum number of connector instances in the pool, both idle and active. The default value is `10` instances.

`maxIdle`

The maximum number of idle connector instances in the pool. The default value is `10` idle instances.

`maxWait`

The maximum period to wait for a free connector instance to become available before failing. The default period is `150000` milliseconds, or 150 seconds.

`minEvictableIdleTimeMillis`

The minimum period to wait before evicting an idle connector instance from the pool. The default period is `120000` milliseconds, or 120 seconds.

`minIdle`

The minimum number of idle connector instances in the pool. If `minIdle=0`, then a connector pool cleaner thread will run and close expired connections.

HTTP pooling

Connectors that use HTTP pooling require an HTTP client and leverage `PoolingHttpClientConnectionManager` to manage a pool of HTTP connections. Each connector defines and supports different properties that configure the pooling connection manager.

JDBC connectors

This pooling mechanism applies to connectors interacting with a database. A Tomcat JDBC driver handles the pooling.

Connector-specific pooling

For connector-specific pooling, the connector implements the pooling and can include properties to configure the pool.

Non-poolable connectors

For non-poolable connectors, ICF creates a new instance of the connector class and uses a new or existing instance of the connector configuration to initialize the instance before the operation runs. After the operation is run, OpenICF disposes of the connector instance.

Connectors by pooling mechanism

To ensure you configure the correct pooling option for a connector, consult the following table for a list of connectors by pooling type.

Connectors by pooling mechanism

ICF	HTTP	JDBC connectors	Connector-specific	Non-poolable
Adobe Marketing Cloud	Adobe Admin Console	Database table	Amazon Web Services (AWS)	AS400
Groovy connector toolkit	Box	Oracle EBS		AWS IAM Identity Center
Kerberos	Cerner	SAP HANA Database		CSV file
LDAP	DocuSign	Scripted SQL		Duo
Microsoft Graph API	Dropbox			Google Apps
PeopleSoft	Epic			Groovy connector toolkit
PowerShell connector toolkit	Google Cloud Platform (GCP)			Multiple CSV
ServiceNow	HubSpot			SAP S/4HANA
SSH	IBM RACF			
Workday	Marketo			
	MongoDB			
	PingOne			
	SaaS REST			
	Salesforce			
	SAP			
	SAP SuccessFactors			
	SCIM			
	Scripted REST			
	Webex			

Connector reference

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

> **Tip**
>
> Any available connector works with IDM, either directly or using RCS. Advanced Identity Cloud can use any available connector through RCS.
> If you are looking for Advanced Identity Cloud applications, refer to:
>
> - Application management
> - App catalog

All connectors are available for download from Backstage, but some connectors are included in the default deployment for Advanced Identity Cloud, IDM, and RCS. The following table identifies which connectors are included in the default deployments:

Connector included in default deployment

Connector	IDM	RCS
Adobe Admin Console	✕ No	✕ No
Adobe Marketing Cloud	✓ Yes	✕ No
Amazon Web Services (AWS)	✕ No	✕ No
AWS IAM Identity Center	✕ No	✕ No
AS400	✕ No	✓ Yes
Box	✕ No	✕ No
Cerner	✕ No	✓ Yes
CSV file	✓ Yes	✓ Yes
Database table	✓ Yes	✓ Yes
DocuSign	✕ No	✕ No
Dropbox	✕ No	✕ No
Duo	✕ No	✕ No
Epic	✕ No	✓ Yes
Google Apps	✓ Yes	✕ No
Google Cloud Platform (GCP)	✕ No	✕ No

Connector	IDM	RCS
Groovy connector toolkit	✓ Yes	✓ Yes
HubSpot	✕ No	✕ No
IBM RACF	✕ No	✓ Yes
Kerberos	✓ Yes	✓ Yes
LDAP	✓ Yes	✓ Yes
Marketo	✓ Yes	✕ No
Microsoft Graph API	✓ Yes	✕ No
MongoDB	✓ Yes	✓ Yes
Multiple CSV	✕ No	✕ No
Oracle EBS	✕ No	✓ Yes
PeopleSoft	✕ No	✓ Yes
PingOne	✕ No	✕ No
PowerShell connector toolkit	✕ No	✕ No
SaaS REST	✕ No	✕ No
Salesforce	✓ Yes	✕ No
SAP	✕ No	✓ Yes
SAP HANA Database	✕ No	✓ Yes
SAP S/4HANA	✕ No	✕ No
SAP SuccessFactors	✕ No	✕ No
SCIM	✓ Yes	✓ Yes
Scripted REST	✓ Yes	✓ Yes
Scripted SQL	✓ Yes	✓ Yes
ServiceNow	✓ Yes	✕ No
SSH	✓ Yes	✓ Yes

Connector	IDM	RCS
Webex	✕ No	✕ No
Workday	✕ No	✕ No

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

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

SaaS common connectors

Software as a Service (SaaS) common connectors enable connection to cloud-based apps, data, and services. SaaS common connectors share certain code and configuration templates. When a procedure, feature, or release notes specify something as SaaS common, it applies to all SaaS common connectors.

SaaS common connectors

- Adobe Admin Console connector
- Box connector
- DocuSign connector
- Dropbox connector
- PingOne connector
- SaaS REST Connector
- Webex Connector

> **Important**
>
> SaaS common connectors version 1.5.20.29 and later support OAuth JWT Bearer flow with:
>
> - Connector framework 1.5.20.24 or later
> - RCS 1.5.20.24 or later
>
> Learn more in Changed functionality.

Scripted Groovy connectors

Scripted Groovy connectors are based on the Scripted Groovy connector toolkit and share certain code and configuration templates. When a procedure, feature, or release note specifies something as Scripted Groovy, it applies to all Scripted Groovy connectors.

Scripted Groovy connectors

- Groovy connector toolkit
- Marketo connector
- MongoDB connector
- SAP connector
- Scripted REST connector
- Scripted SQL connector
- SSH connector

Name changes for ForgeRock products

Product names changed when ForgeRock became part of Ping Identity.

The following name changes have been in effect since early 2024:

Old name	New name
ForgeRock Identity Cloud	PingOne Advanced Identity Cloud
ForgeRock Access Management	PingAM
ForgeRock Directory Services	PingDS
ForgeRock Identity Management	PingIDM
ForgeRock Identity Gateway	PingGateway

Learn more about the name changes in New names for ForgeRock products in the Knowledge Base.

Adobe Admin Console connector

> **Tip**
>
> This is a SaaS common connector.

The Adobe admin console connector allows you to manage users and groups, as well as manage user group memberships between the Adobe admin console and IDM. You need an administrator account.

Before you start

1. Create an Adobe Admin Console developer account.
2. Create a new project. Add User Management API, choose the type of authentication OAuth server-to-server

3. From the credentials tab, get the client_id, client_secret, orgId, and scope.

Install the Adobe Admin Console connector

> **Tip**
>
> To check for an Advanced Identity Cloud application for this connector, refer to:
>
> - Application management
> - App catalog

You can download any connector from Backstage, but some are included in the default deployment for Advanced Identity Cloud, IDM, or RCS. When using an included connector, you can skip installing it and move directly to configuration.

Connector included in default deployment

Connector	IDM	RCS
Adobe Admin Console	✕ No	✕ No

Download the connector .jar file from Backstage.

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

```
mv ~/Downloads/adobe-connector-1.5.20.30.jar /path/to/openidm/connectors/
```

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

Configure the Adobe Admin Console connector

Create a connector configuration using the IDM admin UI:

1. From the navigation bar, click **Configure** > **Connectors**.
2. On the **Connectors** page, click **New Connector**.
3. On the **New Connector** page, type a **Connector Name**.
4. From the **Connector Type** drop-down list, select **Adobe Admin Console Connector - 1.5.20.30**.
5. Complete the **Base Connector Details**.

> **Tip**
>
> For a list of all configuration properties, refer to Adobe Admin Console Connector Configuration

6. Click **Save**.

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

Refer to this procedure to create a connector configuration over REST.

- `Adobe User Management API Endpoint` : https://usermanagement.adobe.io/v2
- `Use Basic Auth For OAuth Token Neg` : `true` | `false`
- `Max connections` : Max size of the http connection pool used. Defaults to `10` .
- `Connection Timeout (seconds)` : Defines a timeout for the underlying http connection in seconds. Defaults to `30` .
- `Token Endpoint` : https://ims-na1.adobelogin.com/ims/token/v3
- `Client ID` : Your Client ID.
- `Client Secret` : Your Client Secret.
- `Grant type` : client_credentials
- `Scope` : openid, AdobeID, user_management_sdk
- `orgId` : Your Organization Id

> **Note**
>
> In the Scope field, the scopes must be separated by a comma.

If necessary, add or edit your object types to have these three objects with their properties:

PROPERTY NAME	TYPE	NATIVE TYPE	REQUIRED
`_id`	String	String	NO
`email`	String	String	YES
`firstname`	String	String	NO
`lastname`	String	String	NO
`status`	String	String	NO
`username`	String	String	NO
`type`	String	String	YES
`orgSpecific`	boolean	boolean	NO
`businessAccount`	boolean	boolean	NO
`country`	String	String	NO
`__GROUPS__`	Array	String	NO

PROPERTY NAME	TYPE	NATIVE TYPE	REQUIRED
`_id`	String	String	NO
`groupName`	String	String	YES
`type`	String	String	NO
`adminGroupName`	String	String	NO
`userGroupName`	String	String	NO
`memberCount`	Integer	Integer	NO
`productName`	String	String	NO
`profileGroupName`	String	String	NO

If configuring the connector over REST or through the filesystem, specify the connection details to the Adobe resource provider in the `configurationProperties` for the connector. If you are using OAuth for your connection, the minimum required properties are `scope`, `orgId`, `grantType`, `serviceUri`, `tokenEndpoint`, `clientId`, and `clientSecret`.

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

```
{
    "configurationProperties" : {
        "tokenExpiration" : null,
        "accessToken" : null,
        "serviceUri" : "https://usermanagement.adobe.io/v2",
        "login" : null,
        "password" : null,
        "authenticationMethod" : "OAUTH",
        "tokenEndpoint" : "https://ims-na1.adobelogin.com/ims/token/v3",
        "clientId" : "xxxxxxxxxxxxxxxxxx",
        "clientSecret" : "xxxxxxxxxxxxxxxxxx",
        "refreshToken" : null,
        "authToken" : null,
        "acceptSelfSignedCertificates" : false,
        "disableHostNameVerifier" : false,
        "disableHttpCompression" : false,
        "clientCertAlias" : null,
        "clientCertPassword" : null,
        "maximumConnections" : "10",
        "httpProxyHost" : null,
        "httpProxyPort" : null,
        "httpProxyUsername" : null,
        "httpProxyPassword" : null,
        "connectionTimeout" : "30",
        "grantType" : "client_credentials",
        "scope" : "openid, AdobeID, user_management_sdk",
        "authorizationTokenPrefix" : "Bearer",
        "useBasicAuthForOauthTokenNeg" : true,
        "groupReadRateLimit" : "0.09/sec",
        "userReadRateLimit" : "0.41/sec",
        "writeRateLimit" : "0.16/sec"
      }
}
```

Note

If throttling problems continue, this guide may be helpful: Improve reconciliation query performance.

Configure connection pooling

The Adobe Admin Console connector supports HTTP pooling, which can substantially improve the performance of the connector. Learn more about the basic connection pooling configuration and different pooling mechanisms described in Connection pooling configuration.

Mapping

Attributes Grid: Where the columns represent the attribute name mapped from source to target and the necessary data transformation to synchronize successfully.

SOURCE	TARGET	TRANSFORMATION SCRIPT
`id`	`userId`	N/A

SOURCE	TARGET	TRANSFORMATION SCRIPT
`email`	`_id`	N/A
`firstname`	`givenName`	N/A
`lastname`	`sn`	N/A
`type`	`type`	N/A
`status`	`status`	N/A
`username`	`username`	N/A
`country`	`country`	N/A
`orgSpecific`	`orgSpecific`	N/A
`businessAccount`	`businessAccount`	N/A
`__GROUPS__`	`__GROUPS__`	N/A

Attributes Grid: Where the columns represent the attribute name mapped from source to target and the necessary data transformation to synchronize successfully.

SOURCE	TARGET	TRANSFORMATION SCRIPT
`mail`	`email`	N/A
`givenName`	`firstname`	N/A
`sn`	`lastname`	N/A
`type`	`type`	N/A
`username`	`username`	N/A
`country`	`country`	N/A
`__GROUPS__`	`__GROUPS__`	N/A

Attributes Grid: Where the columns represent the attribute name mapped from source to target and the necessary data transformation to synchronize successfully.

SOURCE	TARGET	TRANSFORMATION SCRIPT
`groupId`	`_id`	N/A

SOURCE	TARGET	TRANSFORMATION SCRIPT
`groupname`	`groupName`	N/A
`type`	`type`	N/A
`memberCount`	`memberCount`	N/A
`adminGroupName`	`adminGroupName`	N/A
`productName`	`productName`	N/A
`licenseQuota`	`licenseQuota`	N/A
`profileGroupName`	`profileGroupName`	N/A

Attributes Grid: Where the columns represent the attribute name mapped from source to target and the necessary data transformation to synchronize successfully.

SOURCE	TARGET	TRANSFORMATION SCRIPT
`groupName`	`groupName`	N/A
`description`	`description`	N/A

Test the Adobe Admin Console connector

Test that the connector was configured correctly:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Accept-API-Version: resource=1.0' \
--request POST \
'http://localhost:8080/system/adobe?_action=test'
{
    "name": "",
    "enabled": true,
    "config": "config/provisioner.openicf/",
    "connectorRef": {
        "bundleVersion": "1.5.20.30",
        "bundleName": "org.forgerock.openicf.connectors.adobe-connector",
        "connectorName": "org.forgerock.openicf.connectors.adobe.AdobeConnector"
    },
    "displayName": "Adobe Admin Console Connector",
    "objectTypes": [
        "__GROUP__",
        "__ACCOUNT__",
        "__ALL__",
    ],
    "ok": true
}
```

User

Create user

To create a user, it is necessary to provide *at least* the `email` and `type` fields. The possible values for the `type` field are adobeID, federatedID, and enterpriseID (case insensitive). To add groups or product profiles to a user, you must use the `__GROUPS__` field. To do this, you need to provide the corresponding IDs. The country field of a set cannot be updated. If not sent, it defaults to the country of the domain name. When creating a user, the username field is initially set to be the same as the email address; however, this username field can be modified later through user profile updates:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request POST \
--data '{
    "email" : "john.doe@domain1.com",
    "type" : "adobeID",
    "firstName" : "John",
    "lastName" : "Doe",
    "lastName" : "US",
    "__GROUPS__" : [
            "groupId",
            "groupId",
    ]
}' \
'http://localhost:8080/system/adobe/__ACCOUNT__?_action=create'
{
    "_id" : "john.doe@domain1.com",
    "id" : "userID",
    "email" : "john.doe@domain1.com",
    "username" : "john.doe@domain1.com",
    "orgSpecific": true,
    "businessAccount": true,
    "firstName" : "John",
    "lastname" : "Doe",
    "type" : "adobeID",
    "__NAME__" : "john.doe@domain1.com",
    "status" : "active",
    "country" : "US",
    "__GROUPS__" : [
            "groupId"
            "groupId"
    ]
}
```

Get Users

Retrieve a list of users from Adobe Admin Console. To paginate the results, the parameter pageSize must have a value greater than 1. The size of each page is 2,000 except, for the first page, which can contain fewer results due to technical users not being retrieved. By default, all users are retrieved.

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request GET \
'http://localhost:8080/openidm/system/adobe/__ACCOUNT__?_queryFilter=true'
{
    "result": [
        {
            "_id": "email@domain1.com",
            "__GROUPS__": [
                "groupId"
            ],
            "id": "userID",
            "country": "US",
            "email": "email@domain1.com",
            "orgSpecific": true,
            "username": "email@domain1.com",
            "businessAccount": true,
            "firstname": "John",
            "__NAME__": "john.doe@domain1.com",
            "type": "adobeID",
            "status": "active",
            "lastname": "Doe"
        },
       /…
    ],
    "resultCount": 999,
    "pagedResultsCookie": null,
    "totalPagedResultsPolicy": "NONE",
    "totalPagedResults": -1,
    "remainingPagedResults": -1
}
```

Get user

Retrieve a user from Adobe Admin Console. The user email must be provided in the URI path.

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request GET \
'http://localhost:8080/openidm/system/adobe/__ACCOUNT__/USER_EMAIL'
{
    "_id" : "email@domain1.com",
    "email" : "email@domain1.com",
    "firstname" : "John",
    "lastname" : "Doe",
    "username" : "email@domain1.com",
    "type" : "adobeID",
    "status" : "active",
    "orgSpecific" : true,
    "businessAccount" : true,
    "__GROUPS__" : [
            "groupId1",
            "groupId2",
    ]
}
```

Get users type

Retrieves Adobe users only by displaying `type` and `_id` field. By default, retrieves all users:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request GET \
'http://localhost:8080/openidm/system/adobe/__ACCOUNT__?_queryFilter=true&_fields=type'
{
    "result": [
        {
            "_id" : "email1@domain.com",
            "type": "adobeID"
        {
            "_id" : "email2@domain.net",
            "type": "federatedID"
        },
        {
            "_id" : "email3@domain.org",
            "type": "enterpriseID"
        }
    ],
    "resultCount": 999,
    "pagedResultsCookie": null,
    "totalPagedResultsPolicy": "NONE",
    "totalPagedResults": -1,
    "remainingPagedResults": -1
}
```

Update user

Only enterprise or federated users can be updated. The fields that can be updated are `firstname`, `lastname`, `username`, and `__GROUPS__`. The user email must be provided in the URI path:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request PUT \
--data '{
    "firstname" : "Jonny",
    "lastname" : "Doo",
    "username" : "jonnydoo",
    "__GROUPS__" : [
            "groupId1",
            "groupId2",
    ]
}' \
'http://localhost:8080/system/adobe/__ACCOUNT__/USER_EMAIL'
{
    "_id": "john.doe@domain1.com",
    "id": "userID",
    "firstname": "Jonny",
    "username": "jonnydoo",
    "lastname": "Doo",
    "email": "john.doe@domain1.com",
    "orgSpecific": true,
    "status": "active",
    "businessAccount": true,
    "country": "US",
    "type": "federatedID",
    "__NAME__": "userjd",
    "__GROUPS__": [
        "groupId1",
        "groupId2"
    ]
}
```

Delete user

Delete a user from the Adobe organization. The user email must be provided in the URI path:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request DELETE \
'http://localhost:8080/openidm/system/adobe/__ACCOUNT__/USER_EMAIL'
{
    "_id": "john.doe@domain1.com",
    "id": "946F1E3A65DDEA2A0A495CEB@196c1e336579f87e495faa.e",
    "firstname": "John",
    "username": "userjd",
    "lastname": "Doe",
    "email": "john.doe@domain1.com",
    "orgSpecific": true,
    "status": "active",
    "businessAccount": true,
    "country": "US",
    "type": "federatedID",
    "__NAME__": "userjd",
    "__GROUPS__": [
        "groupId"
    ]
}
```

GROUPS

Create group

To create a group, it is necessary to *at least* provide `groupName` field. The `description` field is optional and is not returned; it is only visible from the Adobe web interface console:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request POST \
--data '{
    "groupName" : "group name",
    "description" : "group description"
}' \
'http://localhost:8080/openidm/system/adobe/__GROUP__?_action=create'
{
    "_id" : "groupId",
    "__NAME__" : "groupId",
}
```

Get groups

Retrieve a list of groups. To paginate the results the pageSize parameter value must be greater than 1, the size of each page is 400. By default, retrieves all users:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request GET \
'http://localhost:8080/openidm/system/adobe/__GROUP__?_queryFilter=true'
{
    "result": [
        {
            "_id" : "groupId1"
        },
        {
            "_id" : "groupId2"
        },
        {
            "_id" : "groupId3",
        },
        ...
    ],
    "resultCount": 999,
    "pagedResultsCookie": null,
    "totalPagedResultsPolicy": "NONE",
    "totalPagedResults": -1,
    "remainingPagedResults": -1
}
```

Get group

Retrieve a group, only the `_id` field can be displayed. The group id must be provided in the URI path:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request GET \
'http://localhost:8080/openidm/system/adobe/__GROUP__/GROUP_ID'
{
    "_id" : "groupId",
    "__NAME__" : "groupId"
}
```

Update a group

The field that can be updated for a group is `description`. The group description is only visible from the web interface console. The group id must be provided in the URI path:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request PUT \
--header 'If-Match: *' \
--data '{
    "description" : "New Description"
}' \
'http://localhost:8080/openidm/system/adobe/__GROUP__/GROUP_ID'
{
    "_id" : "groupId",
    "__NAME__" : "groupId",
}
```

Delete a group

The group id must be provided in the URI path:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request DELETE \
'http://localhost:8080/openidm/system/adobe/__GROUP__/GROUP_ID'
{
    "_id" : "groupId",
    "__NAME__" : "groupId"
}
```

OpenICF Interfaces Implemented by the Adobe Admin Console Connector

The Adobe Admin Console Connector implements the following OpenICF interfaces. For additional details, see ICF interfaces:

Create

Creates an object and its `uid`.

Delete

Deletes an object, referenced by its `uid`.

Schema

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

Script on Connector

Enables an application to run a script in the context of the connector.

Any script that runs on the connector has the following characteristics:

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

Search

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

Test

Tests the connector configuration.

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

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

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

Update

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

Adobe Admin Console Connector Configuration

The Adobe Admin Console Connector has the following configurable properties:

Basic Configuration Properties

Property	Type	Default	Encrypted[(1)]	Required[(2)]
`serviceUri`	`String`	`null`		✓ Yes
The service endpoint URI.				
`orgId`	`String`	`null`		✓ Yes
Your organizations unique ID, for example 12345@AdobeOrg.				
`login`	`String`	`null`		✓ Yes
The service login name.				

Property	Type	Default	Encrypted[1]	Required[2]
`groupReadRateLimit`	`String`	`null`		✓ Yes
Defines throttling for read operations either per seconds ("30/sec") or per minute ("100/min").				
`password`	`GuardedString`	`null`	🔒 Yes	✕ No
The service user password.				
`userReadRateLimit`	`String`	`null`		✓ Yes
Defines throttling for read operations either per seconds ("30/sec") or per minute ("100/min").				
`authenticationMethod`	`String`	`OAUTH`		✓ Yes
Defines which method is to be used to authenticate on the remote server. Options are BASIC (username/password), OAUTH (Client id/secret) or TOKEN (static token).				
`tokenEndpoint`	`String`	`null`		✕ No
When using OAUTH as authentication method, this property defines the endpoint where a new access token should be queried for (https://myserver.com/oauth2/token).				
`writeRateLimit`	`String`	`null`		✓ Yes
Defines throttling for write operations (create/update/delete) either per second ("30/sec") or per minute ("100/min").				
`clientId`	`String`	`null`		✓ Yes
The client identifier for OAuth2.				
`clientSecret`	`GuardedString`	`null`	🔒 Yes	✕ No
Secure client secret for OAuth2.				
`authToken`	`GuardedString`	`null`	🔒 Yes	✕ No
Static authentication token.				
`acceptSelfSignedCertificates`	`boolean`	`false`		✓ Yes
To be used for debug/test purposes. To be avoided in production.				
`disableHostNameVerifier`	`boolean`	`false`		✓ Yes
To be used for debug/test purposes. To be avoided in production.				
`disableHttpCompression`	`boolean`	`false`		✓ Yes

Property	Type	Default	Encrypted(1)	Required(2)
Content compression is enabled by default. Set this property to true to disable it.				
`clientCertAlias`	`String`	`null`		✓ Yes
If TLS Mutual Auth is needed, set this to the certificate alias from the keystore.				
`clientCertPassword`	`GuardedString`	`null`	🔒 Yes	✓ Yes
If TLS Mutual Auth is needed and the client certificate (private key) password is different from the keystore password, set this to the client private key password.				
`maximumConnections`	`Integer`	`10`		✓ Yes
Defines the max size of the HTTP connection pool used.				
`httpProxyHost`	`String`	`null`		✓ Yes
Defines the Hostname if an HTTP proxy is used between the connector and the service.				
`httpProxyPort`	`Integer`	`null`		✓ Yes
Defines the Port if an HTTP proxy is used between the connector and the service.				
`httpProxyUsername`	`String`	`null`		✓ Yes
Defines Proxy Username if an HTTP proxy is used between the connector and the service.				
`httpProxyPassword`	`GuardedString`	`null`	🔒 Yes	✓ Yes
Defines Proxy Password if an HTTP proxy is used between the connector and the service.				
`connectionTimeout`	`int`	`30`		✕ No
Defines a timeout for the underlying HTTP connection in seconds.				
`refreshToken`	`GuardedString`	`null`		✕ No
Used by the refresh_token grant type.				
`grantType`	`String`	`null`		✕ No
The OAuth2 grant type to use (client_credentials, refresh_token, or jwt_bearer).				
`scope`	`String`	`null`		✕ No
The OAuth2 scope to use.				
`authorizationTokenPrefix`	`String`	`Bearer`		✕ No

Property	Type	Default	Encrypted[1]	Required[2]
The prefix to be used in the Authorization HTTP header for Token authentication.				
`useBasicAuthForOauthTokenNeg`	`boolean`	`true`		✓ Yes
The Authentication method for refresh token (Basic Authentication or Sending the ClientId and Client Secret in the Header).				
`jwtKey`	`String`	`null`		✕ No
The JWT data structure that represents a cryptographic key.				
`jwtExpiration`	`Integer`	`null`		✕ No
Defines the JWT expiration time in seconds.				
`jwtAlgorithm`	`String`	`null`		✕ No
The Algorithm type to sign payload.				
`jwtClaims`	`Map`	`null`		✕ No
JWT Claims to be included in the payload				
`jwtPem`	`String`	`null`		✕ No
The contents of the private key of the PEM file				
`jwtCert`	`String`	`null`		✕ No
The contents of the certificate of the PEM file				
`keyAlgorithm`	`String`	`null`		✕ No
Indicates the type of key (such as RSA, DSA or EC) used to sign from the PEM.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

Adobe Marketing Cloud connector

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

> **Important**
>
> To use this connector, you need an *Adobe ID*.

Before you start

Configure a new integration on AdobeIO, as shown in the following steps. Note that these steps assume a specific version of the AdobeIO user interface. For information on the current version, refer to the corresponding Adobe documentation.

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

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

   Export IDM’s self-signed certificate as follows:

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

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

   2. Export the certificate:

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

   3. Export the unencrypted private key:

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

2. Log in to https://console.adobe.io/.

3. Click **Integrations** > **New Integration**.

4. Click **Access an API** > **Continue**.

5. Under the **Experience Cloud** item, click **Adobe Campaign** > **Continue**, then click **New integration** > **Continue**.

6. Enter a name and short description for the new integration. For example, `IDM-managed`.

7. Drag and drop your public certificate file into the **Public keys certificates** area. Alternatively, click **Select a File**, and browse to the file location.

8. Select a license, then click **Create Integration**.

9. Select **Continue to integration details** to obtain the Client Credentials required by the connector.

   You will need these details for the connector configuration.

Install the Adobe Marketing Cloud connector

> **Tip**
>
> To check for an Advanced Identity Cloud application for this connector, refer to:
>
> - Application management
> - App catalog

You can download any connector from Backstage, but some are included in the default deployment for Advanced Identity Cloud, IDM, or RCS. When using an included connector, you can skip installing it and move directly to configuration.

Connector included in default deployment

Connector	IDM	RCS
Adobe Marketing Cloud	✓ Yes	✕ No

Download the connector .jar file from Backstage.

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

```
mv ~/Downloads/adobecm-connector-1.5.20.29.jar /path/to/openidm/connectors/
```

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

Configure the Adobe Marketing Cloud connector

Create a connector configuration using the IDM admin UI:

1. From the navigation bar, click **Configure > Connectors**.

2. On the **Connectors** page, click **New Connector**.

3. On the **New Connector** page, type a **Connector Name**.

4. From the **Connector Type** drop-down list, select **Adobe Marketing Cloud Connector - 1.5.20.29**.

5. Complete the **Base Connector Details**.

> **Tip**
>
> For a list of all configuration properties, refer to Adobe Marketing Cloud Connector Configuration

6. Click **Save**.

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

Refer to this procedure to create a connector configuration over REST.

Alternatively, you can create a connector configuration file and place it in your project's `conf/` directory. IDM bundles a sample configuration file (`/path/to/openidm/samples/example-configurations/provisioners/provisioner.openicf-adobe.json`) that you can use as a starting point.

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

```
"configurationProperties" : {
    "endpoint" : "mc.adobe.io",
    "imsHost" : "ims-na1.adobelogin.com",
    "tenant" : "https://example.adobesandbox.com/",
    "apiKey" : "",
    "techAccId" : "example@techacct.adobe.com",
    "orgId" : "example@AdobeOrg",
    "clientSecret" : "CLIENT_SECRET",
    "privateKey" : "PRIVATE_KEY"
},
...
```

`endpoint`

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

`imsHost`

The Adobe Identity Management System (IMS) host. `ims-na1.adobelogin.com` by default - you should not have to change this value.

`tenant`

Your tenant (organization) name or sandbox host.

`apiKey`

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

`techAccId`

Your Technical account ID, required to generate the JWT.

`orgId`

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

`clientSecret`

The client secret assigned to your API client account.

`privateKey`

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

> **Tip**
>
> For a list of all configuration properties, refer to Adobe Marketing Cloud Connector Configuration.

Test the Adobe Marketing Cloud connector

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

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system/adobe?_action=test"
[
  {
    "name": "adobe",
    "enabled": true,
    "config": "config/provisioner.openicf/adobe",
    "connectorRef": {
      "bundleName": "org.forgerock.openicf.connectors.adobecm-connector",
      "connectorName": "org.forgerock.openicf.acm.ACMConnector",
      "bundleVersion": "[1.5.0.0,1.6.0.0)"
    },
    "displayName": "Adobe Marketing Cloud Connector",
    "objectTypes": [
      "__ALL__",
      "account"
    ],
    "ok": true
  }
]
```

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

Adobe Marketing Cloud remote connector

If you want to run this connector outside of PingOne Advanced Identity Cloud or IDM, you can configure the Adobe Marketing Cloud connector as a remote connector. Java Connectors installed remotely on a Java Connector Server function identically to those bundled locally within PingOne Advanced Identity Cloud or installed locally on IDM.

You can download the Adobe Marketing Cloud connector from here.

Refer to Remote connectors for configuring the Adobe Marketing Cloud remote connector.

Configure connection pooling

The Adobe Marketing Cloud connector uses ICF pooling to manage connections. Learn more about the different pooling mechanisms in Connectors by pooling mechanism.

OpenICF Interfaces Implemented by the Adobe Marketing Cloud Connector

The Adobe Marketing Cloud Connector implements the following OpenICF interfaces. For additional details, see ICF interfaces:

Create

Creates an object and its `uid`.

Delete

Deletes an object, referenced by its `uid`.

Schema

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

Script on Connector

Enables an application to run a script in the context of the connector.

Any script that runs on the connector has the following characteristics:

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

Search

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

Test

Tests the connector configuration.

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

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

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

Update

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

Adobe Marketing Cloud Connector Configuration

The Adobe Marketing Cloud Connector has the following configurable properties:

Basic Configuration Properties

Property	Type	Default	Encrypted[1]	Required[2]
`endpoint`	`String`	`mc.adobe.io`		✓ Yes
The Adobe IO endpoint for Marketing Cloud. mc.adobe.io by default - you should not have to change this.				
`imsHost`	`String`	`ims-na1.adobelogin.com`		✓ Yes
Adobe Identity Management System (IMS) host. ims-na1.adobelogin.com by default - you should not have to change this.				
`tenant`	`String`	`null`		✓ Yes
Your tenant (organization) name or sandbox host.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

Adobe Integration Properties

Property	Type	Default	Encrypted[1]	Required[2]
`apiKey`	`GuardedString`	`null`	🔒 Yes	✓ Yes
The API key (client ID) assigned to your API client account.				
`technicalAccountID`	`String`	`null`		✓ Yes
Your Technical account ID, required to generate the JWT.				
`organizationID`	`String`	`null`		✓ Yes
Your organizations unique ID, for example 12345@AdobeOrg.				
`clientSecret`	`GuardedString`	`null`	🔒 Yes	✓ Yes

Property	Type	Default	Encrypted[1]	Required[2]
The client secret assigned to your API client account.				
`privateKey`	`GuardedString`	`null`	🔒 Yes	✓ Yes
The private key used to sign the JWT token, corresponds to the public key certificate attached to the integration.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

AS400 connector

You can use the AS400 connector to manage and synchronize users between AS400 and IDM or Advanced Identity Cloud.

Before you start

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

Host Name

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

Username

The AS400 Organizational Admin username.

Password

The AS400 Organizational Admin password.

Is Secure

Whether to enable a secure connection to AS400.

Install the AS400 connector

> **Tip**
>
> To check for an Advanced Identity Cloud application for this connector, refer to:
>
> - Application management
> - App catalog

You can download any connector from Backstage, but some are included in the default deployment for Advanced Identity Cloud, IDM, or RCS. When using an included connector, you can skip installing it and move directly to configuration.

Connector included in default deployment

Connector	IDM	RCS
AS400	✕ No	✓ Yes

Download the connector .jar file from Backstage.

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

```
mv ~/Downloads/as400-connector-1.5.20.26.jar /path/to/openidm/connectors/
```

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

Configure the AS400 connector

Create a connector configuration using the IDM admin UI:

1. From the navigation bar, click **Configure > Connectors**.
2. On the **Connectors** page, click **New Connector**.
3. On the **New Connector** page, type a **Connector Name**.
4. From the **Connector Type** drop-down list, select **AS400 Connector - 1.5.20.26**.
5. Complete the **Base Connector Details**.

> **Tip**
>
> For a list of all configuration properties, refer to AS400 Connector Configuration

6. Click **Save**.

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

Refer to this procedure to create a connector configuration over REST.

Test the AS400 connector

Test that the configuration is correct by running the following command:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system/as400?_action=test"
{
  "name": "as400",
  "enabled": true,
  "config": "config/provisioner.openicf/as400",
  "connectorRef": {
    "bundleVersion": "[1.5.0.0,1.6.0.0)",
    "bundleName": "org.forgerock.openicf.connectors.as400-connector",
    "connectorName": "org.forgerock.openicf.connectors.as400.As400Connector"
  },
  "displayName": "AS400 Connector",
  "objectTypes": [
    "__ACCOUNT__",
    "__ALL__",
    "__GROUP__"
  ],
  "ok": true
}
```

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

AS400 remote connector

If you want to run this connector outside of PingOne Advanced Identity Cloud or IDM, you can configure the AS400 connector as a remote connector. Java Connectors installed remotely on a Java Connector Server function identically to those bundled locally within PingOne Advanced Identity Cloud or installed locally on IDM.

You can download the AS400 connector from here.

Refer to Remote connectors for configuring the AS400 remote connector.

Configure connection pooling

The AS400 connector uses a non-poolable mechanism to manage connections. Learn more about the different pooling mechanisms in Connectors by pooling mechanism.

Supported resource types

The AS400 connector supports the following resources:

ICF Native Type	AS400 Resource Type
`__ACCOUNT__`	Users
`__GROUP__`	Groups

Supported search filters

The AS400 connector supports search operations with the following filter operators and attributes:

Object Type	Operators	Attributes
`__GROUP__`	id filter	`Id`

Attributes

The AS400 connector supports the following account attributes:

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

Attribute	Description
`IsAuthCollectionActive`	Whether or not authority collection is active.
`MTCPB`	Limit capabilities.
`TEXT`	A free-form text field.
`SPCAUT`	The special access permissions for the user.
`SPCENV`	The special environment.
`DSPSGNINF`	The display sign-on information.
`PWDEXPITV`	The password expiration interval.
`PWDCHGBLK`	Whether or not to block password change.
`LCLPWDMGT`	Local password management.
`LMTDEVSSN`	Limit device session.
`KBDBUF`	Keyboard buffering.
`MAXSTG`	Maximum allowed storage.
`PTYLMT`	Highest schedule priority.
`JOBD`	Job description.
`OWNER`	The owner of the user profile.
`ACGCDE`	The accounting code.
`DOCPWD`	The document password.
`MSGQ`	The message queue.
`DLVRY`	Delivery.
`SEV`	The severity code.
`PRTDEV`	The print device.
`OUTQ`	The output queue.
`ATNPGM`	The attention program.
`SRTSEQ`	The sort sequence.

Attribute	Description
`LANGID`	The language ID.
`CNTRYID`	The country or region ID.
`CCSID`	The Coded Character Set ID.
`CHRIDCTL`	The character identifier control.
`SETJOBATR`	The local job attributes.
`LOCALE`	The locale.
`USROPT`	The user options.
`UID`	The user ID number.
`HOMEDIR`	The home directory.
`USREXPDATE`	The user’s expiration date.
`USREXPITV`	The user’s expiration interval.
`AUT`	Authority.
`EIMASSOC`	The EIM association.
`PasswordExpireDate`	The date the password expires.
`GRPPRF`	Specifies the user’s group profile name whose authority is used when there is no job-specific authority given to the user.
`SUPGRPPRF`	Specifies the user’s supplemental group profiles. Used with `GRPPRF` to determine what authority the user has when there is no job-specific authority given to the user.

Use the AS400 connector

The AS400 connector can perform the following actions:

Users

The following example creates a user with all available attributes:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json"\
--request POST \
--data "{
  "__NAME__":"BJENSEN",
  "__PASSWORD__":"ASDE1234",
  "PWDEXP":false,
  "__ENABLE__":true,
  "USRCLS":"*USER",
  "ASTLVL":"*BASIC",
  "CURLIB":"*CRTDFT",
  "INLPGM":"*NONE",
  "INLMNU":"MAIN",
  "TEXT":"TEXTFILEDVALUE",
  "SPCAUT":["*AUDIT"],
  "SPCENV":"*S36",
  "DSPSGNINF":"*YES",
  "PWDEXPITV":"323",
  "PWDCHGBLK":"93",
  "LCLPWDMGT":true,
  "LMTDEVSSN":"*NO",
  "MAXSTG":"10000",
  "PTYLMT":8,
  "JOBD":"QDFTJOBD",
  "OWNER":"*USRPRF",
  "ACGCDE":"*BLANK",
  "DOCPWD":"W12345",
  "MSGQ":"*USRPRF",
  "DLVRY":"*HOLD",
  "SEV":"50",
  "PRTDEV":"*SYSVAL",
  "OUTQ":"*DEV",
  "ATNPGM":"*ASSIST",
  "SRTSEQ":"*HEX",
  "LANGID":"ENG",
  "CCSID":"*HEX",
  "CHRIDCTL":"*DEVD",
  "SETJOBATR":["*CCSID"],
  "LOCALE":"*C",
  "USROPT":["*HLPFULL"],
  "UID":"*GEN",
  "HOMEDIR":"*USRPRF",
  "EIMASSOC":["*NOCHG"],
  "USREXPITV":99,
  "USREXPDATE":"*USREXPITV",
  "LMTCPB":"*YES",
  "CNTRYID":"*SYSVAL",
  "GRPPRF":"AZURE",
  "SUPGRPPRF":["AWS"]
}" \
"https://localhost:8443/openidm/system/As400/__ACCOUNT__?_action=create&_prettyprint=true"
{
  "_id" : "BJENSEN",
  "USROPT" : [ "*HLPFULL" ],
  "SEV" : "50",
  "USREXPITV" : 99,
  "IsAuthCollectionActive" : false,
  "HOMEDIR" : "/home/BJENSEN",
  "MAXSTG" : "10000",
  "UID" : "1277",
  "PTYLMT" : 8,
  "__NAME__" : "BJENSEN",
  "PRTDEV" : "*SYSVAL",
  "__ENABLE__" : true,
  "LMTDEVSSN" : "*NO",
  "__UID__" : "BJENSEN",
  "SRTSEQ" : "*HEX",
  "DSPSGNINF" : "*YES",
  "PWDCHGBLK" : "93",
  "GRPPRF" : "AZURE",
  "USREXPDATE" : "12/06/22",
  "CURLIB" : "*CRTDFT",
  "LMTCPB" : "*YES",
  "ASTLVL" : "*BASIC",
  "SUPGRPPRF" : [ "AWS" ],
  "MSGQ" : "/QSYS.LIB/QUSRSYS.LIB/BJENSEN.MSGQ",
  "LANGID" : "ENG",
  "CCSID" : "65535",
  "PWDEXPITV" : "323",
  "IsUserEntitlementRequired" : true,
  "TEXT" : "TEXTFILEDVALUE",
  "JOBD" : "/QSYS.LIB/QGPL.LIB/QDFTJOBD.JOBD",
  "ActionAuditLevel" : "*BASIC",
  "ObjectAuditValue" : "*NONE",
  "PasswordChangedDate" : "Mon Aug 29 05:15:20 IST 2022",
  "ATNPGM" : "/QSYS.LIB/QEZMAIN.PGM",
  "LCLPWDMGT" : true,
  "INLPGM" : "*NONE",
  "USRCLS" : "*USER",
  "SPCAUT" : [ "*AUDIT" ],
  "SETJOBATR" : [ "*CCSID" ],
  "SPCENV" : "*S36",
  "ACGCDE" : "",
  "IsPasswordNone" : false,
  "DLVRY" : "*HOLD",
  "IsAuthCollectionRepositoryExist" : false,
  "UserExpirationAction" : "*DISABLE",
  "INLMNU" : "/QSYS.LIB/%LIBL%.LIB/MAIN.MNU",
  "LOCALE" : "*C",
  "KBDBUF" : "*SYSVAL",
  "OWNER" : "*USRPRF",
  "PasswordExpireDate" : "Tue Jul 18 00:00:00 IST 2023",
  "PWDEXP" : false,
  "OUTQ" : "*DEV",
  "CNTRYID" : "*SYSVAL",
  "CHRIDCTL" : "*DEVD",
  "StorageUsed" : "12"
}
```

Note

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

- Any letter
- Any digits
- The #, $, _, and @ special characters.

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

The following example queries all users in the system:

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

The following example queries all users in the system:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/as400/__ACCOUNT__/BJENSEN?prettyprint=true"
{
  "_id" : "BJENSEN",
  "USROPT" : [ "*HLPFULL" ],
  "SEV" : "50",
  "USREXPITV" : 99,
  "IsAuthCollectionActive" : false,
  "HOMEDIR" : "/home/BJENSEN",
  "MAXSTG" : "10000",
  "UID" : "1277",
  "PTYLMT" : 8,
  "__NAME__" : "BJENSEN",
  "PRTDEV" : "*SYSVAL",
  "__ENABLE__" : true,
  "LMTDEVSSN" : "*NO",
  "__UID__" : "BJENSEN",
  "SRTSEQ" : "*HEX",
  "DSPSGNINF" : "*YES",
  "PWDCHGBLK" : "93",
  "GRPPRF" : "AZURE",
  "USREXPDATE" : "12/06/22",
  "CURLIB" : "*CRTDFT",
  "LMTCPB" : "*YES",
  "ASTLVL" : "*BASIC",
  "SUPGRPPRF" : [ "AWS" ],
  "MSGQ" : "/QSYS.LIB/QUSRSYS.LIB/BJENSEN.MSGQ",
  "LANGID" : "ENG",
  "CCSID" : "65535",
  "PWDEXPITV" : "323",
  "IsUserEntitlementRequired" : true,
  "TEXT" : "TEXTFILEDVALUE",
  "JOBD" : "/QSYS.LIB/QGPL.LIB/QDFTJOBD.JOBD",
  "ActionAuditLevel" : "*BASIC",
  "ObjectAuditValue" : "*NONE",
  "PasswordChangedDate" : "Mon Aug 29 05:15:20 IST 2022",
  "ATNPGM" : "/QSYS.LIB/QEZMAIN.PGM",
  "LCLPWDMGT" : true,
  "INLPGM" : "*NONE",
  "USRCLS" : "*USER",
  "SPCAUT" : [ "*AUDIT" ],
  "SETJOBATR" : [ "*CCSID" ],
  "SPCENV" : "*S36",
  "ACGCDE" : "",
  "IsPasswordNone" : false,
  "DLVRY" : "*HOLD",
  "IsAuthCollectionRepositoryExist" : false,
  "UserExpirationAction" : "*DISABLE",
  "INLMNU" : "/QSYS.LIB/%LIBL%.LIB/MAIN.MNU",
  "LOCALE" : "*C",
  "KBDBUF" : "*SYSVAL",
  "OWNER" : "*USRPRF",
  "PasswordExpireDate" : "Tue Jul 18 00:00:00 IST 2023",
  "PWDEXP" : false,
  "OUTQ" : "*DEV",
  "CNTRYID" : "*SYSVAL",
  "CHRIDCTL" : "*DEVD",
  "StorageUsed" : "12"
}
```

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

- `PASSWORD`
- `PWDEXP`
- `STATUS`
- `USRCLS`
- `ASTLVL`
- `CURLIB`
- `INLPGM`
- `INLMNU`
- `LMTCPB`
- `TEXT`
- `SPCAUT`
- `SPCENV`
- `DSPSGNINF`
- `PWDEXPITV`
- `PWDCHGBLK`
- `LCLPWDMGT`
- `LMTDEVSSN`
- `KBDBUF`
- `MAXSTG`
- `PTYLMT`
- `JOBD`
- `OWNER`
- `ACGCDE`

- `DOCPWD`
- `MSGQ`
- `DLVRY`
- `SEV`
- `PRTDEV`
- `OUTQ`
- `ATNPGM`
- `SRTSEQ`
- `LANGID`
- `CNTRYID`
- `CCSID`
- `CHRIDCTL`
- `SETJOBATR`
- `LOCALE`
- `USROPT`
- `UID`
- `HOMEDIR`
- `USREXPDATE`
- `USREXPITV`
- `EIMASSOC`
- `GRPPRF`
- `SUPGRPPRF`

The following request updates a user:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "Accept-API-Version: resource=1.0" \
--header "If-Match: *" \
--request PUT \
--data "{
  "__PASSWORD__":"ASDE1234",
  "PWDEXP":false,
  "__ENABLE__":true,
  "USRCLS":"*USER",
  "ASTLVL":"*BASIC",
  "CURLIB":"*CRTDFT",
  "INLPGM":"*NONE",
  "INLMNU":"MAIN",
  "TEXT":"TEXTFILEDVALUE",
  "SPCAUT":["*AUDIT"],
  "SPCENV":"*S36",
  "DSPSGNINF":"*YES",
  "PWDEXPITV":"323",
  "PWDCHGBLK":"93",
  "LCLPWDMGT":true,
  "LMTDEVSSN":"*NO",
  "MAXSTG":"10000",
  "PTYLMT":8,
  "JOBD":"QDFTJOBD",
  "OWNER":"*USRPRF",
  "ACGCDE":"*BLANK",
  "DOCPWD":"W12345",
  "MSGQ":"*USRPRF",
  "DLVRY":"*HOLD",
  "SEV":"50",
  "PRTDEV":"*SYSVAL",
  "OUTQ":"*DEV",
  "ATNPGM":"*ASSIST",
  "SRTSEQ":"*HEX",
  "LANGID":"ENG",
  "CCSID":"*HEX",
  "CHRIDCTL":"*DEVD",
  "SETJOBATR":["*CCSID"],
  "LOCALE":"*C",
  "USROPT":["*HLPFULL"],
  "UID":"*GEN",
  "HOMEDIR":"*USRPRF",
  "EIMASSOC":["*NOCHG"],
  "USREXPITV":99,
  "USREXPDATE":"*USREXPITV",
  "LMTCPB":"*YES",
  "CNTRYID":"*SYSVAL",
  "GRPPRF":"AZURE","SUPGRPPRF":["AWS"]
}" \
"https://localhost:8443/openidm/system/As400/__ACCOUNT__/BJENSEN_prettyprint=true"
{
  "_id" : "BJENSEN",
  "USROPT" : [ "*HLPFULL" ],
  "SEV" : "50",
  "USREXPITV" : 99,
  "IsAuthCollectionActive" : false,
  "HOMEDIR" : "/home/BJENSEN",
  "MAXSTG" : "10000",
  "UID" : "1277",
  "PTYLMT" : 8,
  "__NAME__" : "BJENSEN",
  "PRTDEV" : "*SYSVAL",
  "__ENABLE__" : true,
  "LMTDEVSSN" : "*NO",
  "__UID__" : "BJENSEN",
  "SRTSEQ" : "*HEX",
  "DSPSGNINF" : "*YES",
  "PWDCHGBLK" : "93",
  "GRPPRF" : "AZURE",
  "USREXPDATE" : "12/06/22",
  "CURLIB" : "*CRTDFT",
  "LMTCPB" : "*YES",
  "ASTLVL" : "*BASIC",
  "SUPGRPPRF" : [ "AWS" ],
  "MSGQ" : "/QSYS.LIB/QUSRSYS.LIB/BJENSEN.MSGQ",
  "LANGID" : "ENG",
  "CCSID" : "65535",
  "PWDEXPITV" : "323",
  "IsUserEntitlementRequired" : true,
  "TEXT" : "TEXTFILEDVALUE",
  "JOBD" : "/QSYS.LIB/QGPL.LIB/QDFTJOBD.JOBD",
  "ActionAuditLevel" : "*BASIC",
  "ObjectAuditValue" : "*NONE",
  "PasswordChangedDate" : "Mon Aug 29 05:15:20 IST 2022",
  "ATNPGM" : "/QSYS.LIB/QEZMAIN.PGM",
  "LCLPWDMGT" : true,
  "INLPGM" : "*NONE",
  "USRCLS" : "*USER",
  "SPCAUT" : [ "*AUDIT" ],
  "SETJOBATR" : [ "*CCSID" ],
  "SPCENV" : "*S36",
  "ACGCDE" : "",
  "IsPasswordNone" : false,
  "DLVRY" : "*HOLD",
  "IsAuthCollectionRepositoryExist" : false,
  "UserExpirationAction" : "*DISABLE",
  "INLMNU" : "/QSYS.LIB/%LIBL%.LIB/MAIN.MNU",
  "LOCALE" : "*C",
  "KBDBUF" : "*SYSVAL",
  "OWNER" : "*USRPRF",
  "PasswordExpireDate" : "Tue Jul 18 00:00:00 IST 2023",
  "PWDEXP" : false,
  "OUTQ" : "*DEV",
  "CNTRYID" : "*SYSVAL",
  "CHRIDCTL" : "*DEVD",
  "StorageUsed" : "12"
}
```

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

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "Accept-API-Version: resource=1.0" \
--header "If-Match: *" \
--request PUT \
--data "{
  "__PASSWORD__":"newpassword123"
}" \
"https://localhost:8443/openidm/system/as400/__ACCOUNT__/BJENSEN_prettyprint=true"
{
  "_id" : "BJENSEN",
  "USROPT" : [ "*HLPFULL" ],
  "SEV" : "50",
  ...
}
```

The following example activates a user:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "Accept-API-Version: resource=1.0" \
--header "If-Match: *" \
--request PUT \
--data "{
  "__ENABLE__": true
}
"https://localhost:8443/openidm/system/as400/__ACCOUNT__/BJENSEN_prettyprint=true"
{
  "_id" : "BJENSEN",
  ...
  "__ENABLE__": true
  ...
}
```

The following example deactivates a user:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "Accept-API-Version: resource=1.0" \
--header "If-Match: *" \
--request PUT \
--data "{"
  ""__ENABLE__": false
}" \
"https://localhost:8443/openidm/system/as400/__ACCOUNT__/BJENSEN_prettyprint=true"
{
  "_id" : "BJENSEN",
  ...
  "__ENABLE__": false
  ...
}
```

The following example deletes a user:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "Accept-API-Version: resource=1.0" \
--request DELETE \
"https://localhost:8443/openidm/system/as400/__ACCOUNT__/BJENSEN_prettyprint=true"
{
  "_id" : "BJENSEN",
  ...
}
```

Groups

The following example queries all AS400 Groups by their IDs:

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

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

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

OpenICF Interfaces Implemented by the AS400 Connector

The AS400 Connector implements the following OpenICF interfaces. For additional details, see ICF interfaces:

Create

Creates an object and its `uid`.

Delete

Deletes an object, referenced by its `uid`.

Schema

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

Script on Connector

Enables an application to run a script in the context of the connector.

Any script that runs on the connector has the following characteristics:

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

Search

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

Test

Tests the connector configuration.

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

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

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

Update

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

AS400 Connector Configuration

The AS400 Connector has the following configurable properties:

Configuration properties

Property	Type	Default	Encrypted[(1)]	Required[(2)]
`hostName`	`String`	`null`		✓ Yes
Host name or IP address of As400.				

Property	Type	Default	Encrypted[1]	Required[2]
`userName`	`String`	`null`		✓ Yes
The username to login As400.				
`password`	`GuardedString`	`null`	🔒 Yes	✓ Yes
The password to login As400.				
`isSecure`	`boolean`	`true`		✓ Yes
Enables or not secure connection to As400.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

Basic Configuration Properties

Property	Type	Default	Encrypted[1]	Required[2]
`maximumConnections`	`Integer`	`10`		✗ No
Provides the maximum connections.				
`maxLifetime`	`Integer`	`null`		✗ No
Provides the maximum life for an available connection. The default value is 86400000.				
`maxInactivity`	`Integer`	`null`		✗ No
Provides the maximum amount of inactive time before an available connection is closed. The default value is 3600000.				
`maxUseTime`	`Long`	`null`		✗ No
Provides the maximum amount of time a connection can be in use before it is closed and returned to the pool. The default value is -1 indicating that there is no limit.				
`maxUseCount`	`Integer`	`null`		✗ No
Provides the maximum number of times a connection can be used before it is replaced in the pool. The default value is -1 indicating that there is no limit.				
`isRunMaintenance`	`boolean`	`true`		✗ No
Indicates whether the maintenance thread is used to cleanup expired connections. The default is true.				
`isThreadUsed`	`boolean`	`true`		✗ No

Property	Type	Default	Encrypted[1]	Required[2]
Indicates whether threads are used in communication with the host servers and for running maintenance. The default value is true.				
`cleanupInterval`	`Long`	`null`		✕ No
Time interval for how often the maintenance daemon is run. The default value is 300000 milliseconds.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

Amazon Web Services (AWS) connector

Amazon Web Services (AWS) Identity and Access Management (IAM) is a web service for securely controlling access to AWS services. The AWS connector lets you manage and synchronize accounts between AWS and IDM managed user objects. You can also search, assign, and unassign certain other objects from AWS.

> **Important**
>
> To use this connector, you must have an AWS administrator account with proper access to AWS as described in the AWS documentation.

Before you start

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

Access Key ID

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

Secret Key ID

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

Role ARN

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

Credentials Expiration

Time (in seconds) to configure the duration in which the temporary credentials expire. Optional. Default: `3600`.

Region

The region where the AWS instance is hosted.

Parent ID

The unique identifier assigned to the parent entity (like the root account) in the AWS Organization hierarchy. Required for Organizational Unit operations.

UserName

The unique name of a user. Required specifically for retrieving inline policies associated with that user.

Install the AWS connector

> **Tip**
>
> To check for an Advanced Identity Cloud application for this connector, refer to:
>
> - Application management
> - App catalog

You can download any connector from Backstage, but some are included in the default deployment for Advanced Identity Cloud, IDM, or RCS. When using an included connector, you can skip installing it and move directly to configuration.

Connector included in default deployment

Connector	IDM	RCS
Amazon Web Services (AWS)	✕ No	✕ No

Download the connector .jar file from Backstage.

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

  ```
  mv ~/Downloads/aws-connector-1.5.20.26.jar /path/to/openidm/connectors/
  ```

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

Configure the AWS connector

Create a connector configuration using the IDM admin UI:

1. From the navigation bar, click **Configure > Connectors**.
2. On the **Connectors** page, click **New Connector**.
3. On the **New Connector** page, type a **Connector Name**.
4. From the **Connector Type** drop-down list, select **AWS Connector - 1.5.20.26**.
5. Complete the **Base Connector Details**.

> **Tip**
>
> For a list of all configuration properties, refer to AWS Connector Configuration

6. Click **Save**.

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

Refer to this procedure to create a connector configuration over REST.

Test the AWS connector

Test that the configuration is correct by running the following command:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system/aws?_action=test"
{
  "name": "aws",
  "enabled": true,
  "config": "config/provisioner.openicf/aws",
  "connectorRef": {
    "bundleVersion": "[1.5.0.0,1.6.0.0)",
    "bundleName": "org.forgerock.openicf.connectors.aws-connector",
    "connectorName": "org.forgerock.openicf.connectors.aws.AwsConnector"
  },
  "displayName": "AWS Connector",
  "objectTypes": [
    "__ACCOUNT__",
    "__GROUP__",
    "__ROLE__",
    "__MANAGEDPOLICY__",
    "__INLINEPOLICY__",
    "__SERVICECONTROLPOLICY__",
    "__ORGUNIT__"
  ],
  "ok": true
}
```

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

AWS remote connector

If you want to run this connector outside of PingOne Advanced Identity Cloud or IDM, you can configure the AWS connector as a remote connector. Java Connectors installed remotely on a Java Connector Server function identically to those bundled locally within PingOne Advanced Identity Cloud or installed locally on IDM.

You can download the AWS connector from here.

Refer to Remote connectors for configuring the AWS remote connector.

Configure connection pooling

The AWS connector uses connector-specific pooling to manage connections. Learn more about the different pooling mechanisms in Connectors by pooling mechanism.

Supported resource types

The connector maps the following ICF native types to AWS resource types:

ICF Native Type	AWS Resource Type	Naming Attribute
`__ACCOUNT__`	User	`__NAME__`
`__GROUP__`	Group	`__NAME__`
`__ROLE__`	Role	`__NAME__`
`__MANAGEDPOLICY__`	Managed Policy	`__NAME__` Maps to PolicyArn
`__INLINEPOLICY__`	Inline Policy	`__NAME__` Maps to PolicyName
`__SERVICECONTROLPOLICY__`	Service Control Policy	`__NAME__` Maps to PolicyId
`__ORGUNIT__`	Organizational Unit	`__NAME__` Maps to ParentId or Organizational Unit Name/Arn depending on context

Supported search filters

The AWS connector supports search operations with the following filter operators and attributes:

Object Type	Operator	Attributes
`__ACCOUNT__`	Equals filter	`Path`, `UserName` (`__NAME__`)
`__GROUP__`	Equals filter	`Path`, `GroupName` (`__NAME__`)
`__ROLE__`	Equals filter	`Path`, `RoleName` (`__NAME__`)
`__MANAGEDPOLICY__`	Equals filter	`Path`, `PolicyArn` (`__NAME__`)
`__INLINEPOLICY__`	Equals filter	`PolicyName` (`__NAME__`)
`__SERVICECONTROLPOLICY__`	Equals filter	`PolicyId` (`__NAME__`)

Object Type	Operator	Attributes
`__ORGUNIT__`	Equals filter	`ParentId` (`__NAME__`)

Supported attributes

The AWS connector supports the following attributes.

AWS account (user) attributes

The AWS connector supports the following AWS account attributes:

Attribute	Description
`UserName`	The name of the user. Required. Can contain up to 64 letters, digits, and the characters `+`, `=`, `,`, `.`, `@`, `_`, `-`. Must be unique within the account.
`UserId`	Auto-generated unique user ID. Read-only.
`Path`	The path for the user. Used to create a folder-like hierarchy. Default value is `/`.
`Password`	Password for the user's console login profile. Write-only.
`Arn`	Amazon Resource Names (ARNs) uniquely identify the AWS resource. Read-only.
`CreatedDate`	Date the user was created, in ISO 8601 date-time format. Read-only.
`PasswordLastUsed`	Date the user's password was last used for login. Read-only.
`PermissionBoundary`	The ARN of the policy used to set the permissions boundary for the user.

`Tags`	A list of customizable key-value pairs attached to the user. For example:
	`"Tags": [{`
	`    "Key": "Department",`
	`    "Value": "Accounting"`
	`}]`
	Learn more about Tagging AWS resources in the AWS documentation.

`Group`	List of group names the user belongs to.
`ManagedPolicy`	List of managed policy ARNs attached to the user.
`InlinePolicy`	List of inline policies embedded in the user. Each item contains `PolicyName` and `PolicyDocument`.
`Role`	List of roles assigned to the user. Each item contains `RoleName` and potentially `PolicyArn`.

AWS group attributes

Attribute	Description
`GroupName`	Name of the group. Required.
`GroupId`	Auto-generated unique group ID. Read-only.
`Arn`	Amazon Resource Name (ARN) uniquely identifies the group resource. Read-only.
`Path`	The path for the group. Used to create a folder-like hierarchy. Default value is `/`. Read-only.

AWS role attributes

Attribute	Description
`RoleName`	Name of the Role. Required.
`RoleId`	Auto-generated unique role ID. Read-only.
`Path`	The path for the role. Used to create a folder-like hierarchy. Default value is `/`. Read-only.
`Arn`	Amazon Resource Name (ARN) uniquely identifies the role resource. Read-only.
`CreateDate`	Date the role was created. Read-only.
`AssumeRolePolicyDocument`	The trust policy document associated with the role. Read-only.

AWS managed policy attributes

Attribute	Description
`PolicyArn`	The Amazon Resource Name (ARN) uniquely identifies the Managed Policy. Required for identification. Read-only.
`PolicyId`	Auto-generated unique policy ID. Read-only.
`PolicyName`	Name of the policy. Read-only.
`Path`	The path for the policy. Used to create a folder-like hierarchy. Default value is `/`. Read-only.
`CreateDate`	Date the policy was created. Read-only.
`AttachmentCount`	Number of entities (users, groups, roles) attached to the policy. Read-only.
`IsAttachable`	Whether the policy can be attached to users, groups, or roles. Read-only.
`DefaultVersionId`	The identifier for the default version of the policy. Read-only.

Attribute	Description
`PermissionsBoundaryUsageCount`	Number of entities using this policy as a permissions boundary. Read-only.
`UpdateDate`	Date the policy was last updated. Read-only.

AWS inline policy attributes

Attribute	Description
`PolicyName`	Name of the inline policy. Required.
`UserName`	Name of the user the inline policy is attached to. Required for identification.
`PolicyDocument`	The policy document.

AWS Service Control Policy (SCP) attributes

Attribute	Description
`Id`	The unique identifier (ID) of the SCP. Required for identification. Read-only.
`PolicyName`	Name of the SCP. Read-only.
`PolicySummary`	Object containing details like Arn, Type, Description, and AwsManaged status. Read-only.

AWS Organizational Unit (OU) attributes

Attribute	Description
`ParentId`	The unique identifier (ID) of the parent entity (root or OU). Required for identification. Read-only.
`OrganizationalUnits`	List of OU objects, each containing `Name` and `Arn`. Read-only.

Use the AWS connector

You can use the AWS connector to perform create, read, update, and delete (CRUD) operations on AWS IAM objects.

User account operations

Create an AWS user

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

Request

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request POST \
--data '{
  "__NAME__": "bjensen"
}' \
"http://localhost:8080/openidm/system/aws/__ACCOUNT__?_action=create"
```

Response

```
{
  "_id": "bjensen",
  "Path": "/",
  "UserId": "AIDAW3FY74V57KNBRIDU6",
  "__NAME__": "bjensen",
  "Arn": "arn:aws:iam::470686885243:user/bjensen",
  "CreatedDate": "Thu Jun 02 16:46:39 PDT 2022"
}
```

The following example creates a user with all assignable attributes:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request POST \
--data '{
  "__NAME__": "jdoe",
  "Path": "/engineering/",
  "__PASSWORD__": "P@ssw0rd123!",
  "PermissionsBoundary": "arn:aws:iam::aws:policy/PowerUserAccess",
  "Tags": [{ "Key": "Project", "Value": "Phoenix" }],
  "__GROUP__": ["developers"],
  "__MANAGEDPOLICY__": ["arn:aws:iam::aws:policy/AmazonEC2ReadOnlyAccess"],
  "__ROLE__": [{"RoleName": "EC2InstanceRole"}],
  "__INLINEPOLICY__": [{
    "PolicyName": "S3BucketAccess",
    "PolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [{
        "Effect": "Allow",
        "Action": "s3:ListBucket",
        "Resource": "arn:aws:s3:::example_bucket"
      }]
    }
  }]
}' \
"http://localhost:8080/openidm/system/aws/__ACCOUNT__?_action=create"
```

Response:

```
{
  "_id": "jdoe",
  "CreatedDate": "Fri May 02 13:00:00 PDT 2025",
  "Arn": "arn:aws:iam::123456789012:user/engineering/jdoe",
  "__INLINEPOLICY__": [ { "PolicyName": "S3BucketAccess" } ],
  "__NAME__": "jdoe",
  "__GROUP__": [ "developers" ],
  "Path": "/engineering/",
  "__ROLE__": [ { "RoleName": "EC2InstanceRole" } ],
  "PermissionsBoundary": "arn:aws:iam::aws:policy/PowerUserAccess",
  "__MANAGEDPOLICY__": [ "arn:aws:iam::aws:policy/AmazonEC2ReadOnlyAccess" ],
  "Tags": [ { "Project": "Phoenix" } ],
  "UserId": "AIDACKCEVSQ6C2EXAMPLE"
}
```

ⓘ **Note**

- You must specify at least `__NAME__` when creating a user.
- Usernames can be up to 64 characters long and include letters, digits, and `+ = , . @ _ -`.
- Assigning roles (`__ROLE__`) during user creation is informational in IAM; roles are assumed, not directly assigned like groups or policies. The connector reflects attached policies for consistency but doesn't perform role assignment in the AWS sense.

Update an AWS user

Modify an existing user with a PUT request. Include all attributes you want the user to have; attributes not included in the PUT request might be removed or reset depending on the target system behavior (often equivalent to PATCH for specific fields like Tags, Group, Policy, Role additions/removals).

Modifiable attributes:

- `__NAME__` (Requires specifying the old ID in the URL)
- `__PASSWORD__` (Use PATCH for password changes)
- `Path`
- `PermissionsBoundary`
- `Tags`
- `__GROUP__`
- `__MANAGEDPOLICY__`
- `__INLINEPOLICY__`
- `__ROLE__` (Reference the note in Create an AWS user)

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

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "If-Match:*" \
--request PUT \
--data '{
  "__NAME__": "bjensen",
  "Tags": [{
    "Key": "Project",
    "Value": "Meteor"
  }]
}' \
"http://localhost:8080/openidm/system/aws/__ACCOUNT__/bjensen"
{
  "_id": "bjensen",
  "Path": "/",
  "UserId": "AIDAW3FY74V57KNBRIDU6",
  "__NAME__": "bjensen",
  "Arn": "arn:aws:iam::470686885243:user/bjensen",
  "CreatedDate": "Thu Jun 02 16:46:39 PDT 2022",
  "Tags": [
    {
      "Project": "Meteor"
    }
  ]
}
```

Assign other objects to a user

Use PATCH or PUT to add groups, managed policies, inline policies, or roles to a user.

Example using PATCH to add a group and a managed policy:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "If-Match:*" \
--request PATCH \
--data '[
  {"operation": "add", "field": "__GROUP__", "value": ["qa-team"]},
  {"operation": "add", "field": "__MANAGEDPOLICY__", "value": ["arn:aws:iam::aws:policy/
AmazonS3ReadOnlyAccess"]}
]' \
"http://localhost:8080/openidm/system/aws/__ACCOUNT__/jdoe"
```

Unassign other objects from a user

Use PATCH or PUT to remove groups, managed policies, inline policies, or roles from a user.

Example using PATCH to remove a group and an inline policy:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "If-Match:*" \
--request PATCH \
--data '[
  {"operation": "remove", "field": "__GROUP__", "value": ["frontend-devs"]},
  {"operation": "remove", "field": "__INLINEPOLICY__", "value": [{"PolicyName": "S3BucketAccess"}]}
]' \
"http://localhost:8080/openidm/system/aws/__ACCOUNT__/jdoe"
```

Query AWS users

The following example queries all AWS users:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/aws/__ACCOUNT__?_queryId=query-all-ids"
{
  "result": [
    {
      "_id": "bjensen"
    },
    {
      "_id": "frank@example.com"
    },
    {
      "_id": "testFR4User"
    },
    {
      "_id": "testFR5User"
    },
    {
      "_id": "testFR6User"
    }
  ],
  …
}
```

The following command queries a specific user by their ID:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/aws/__ACCOUNT__/bjensen"
{
  "_id": "bjensen",
  "Path": "/",
  "UserId": "AIDAW3FY74V57KNBRIDU6",
  "__NAME__": "bjensen",
  "Arn": "arn:aws:iam::470686885243:user/bjensen",
  "CreatedDate": "Thu Jun 02 16:46:39 PDT 2022",
  "Tags": [
    {
      "Project": "Meteor"
    }
  ]
}
```

Reset an AWS user account password

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "if-Match:*" \
--request PATCH \
--data '[{
  "operation": "add",
  "field": "__PASSWORD__",
  "value": "Passw0rd@123!"
}]' \
"http://localhost:8080/openidm/system/aws/__ACCOUNT__/bjensen"
{
  "_id": "bjensen",
  "Path": "/",
  "UserId": "AIDAW3FY74V57KNBRIDU6",
  "__NAME__": "bjensen",
  "Arn": "arn:aws:iam::470686885243:user/bjensen",
  "CreatedDate": "Thu Jun 02 16:46:39 PDT 2022",
  "Tags": [
    {
      "Project": "Meteor"
    }
  ]
}
```

Note

While the `__PASSWORD__` field is not returned in the response, the user’s password is updated.

Delete an AWS user account

Use a DELETE request to remove a user from AWS IAM.

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request DELETE \
"http://localhost:8080/openidm/system/aws/__ACCOUNT__/bjensen"
{
  "_id": "bjensen",
  "Path": "/",
  "UserId": "AIDAW3FY74V57KNBRIDU6",
  "__NAME__": "bjensen",
  "Arn": "arn:aws:iam::470686885243:user/bjensen",
  "CreatedDate": "Thu Jun 02 16:46:39 PDT 2022",
  "Tags": [
    {
      "Project": "Meteor"
    }
  ]
}
```

Other object type operations

A similar query pattern applies to groups, roles, managed policies, inline policies, service control policies, and organizational units using their respective object types (`GROUP` , `ROLE` , and so on.) in the request URL. For example, `_queryFilter=True` to return all applicable objects, and using the specific object ID to return a particular object.

Query AWS Groups

Query all groups:

Request

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--request GET \
"http://localhost:8080/openidm/system/aws/__GROUP__?_queryFilter=True"
```

Response

```
{
  "result": [
    {
      "_id": "forge",
      "Path": "/",
      "__NAME__": "forge",
      "GroupId": "AGPAW3FY74V5TAMVGJTDO",
      "GroupName": "forge",
      "Arn": "arn:aws:iam::470686885243:group/forge"
    },
    {
      "_id": "IAMAdministrator",
      "Path": "/",
      "__NAME__": "IAMAdministrator",
      "GroupId": "AGPAW3FY74V5XKCZVOQI5",
      "GroupName": "IAMAdministrator",
      "Arn": "arn:aws:iam::470686885243:group/IAMAdministrator"
    },
    {
      "_id": "SuperUser",
      "Path": "/",
      "__NAME__": "SuperUser",
      "GroupId": "AGPAW3FY74V5XANUBMNXT",
      "GroupName": "SuperUser",
      "Arn": "arn:aws:iam::470686885243:group/SuperUser"
    },
    {
      "_id": "TempGroup",
      "Path": "/",
      "__NAME__": "TempGroup",
      "GroupId": "AGPAW3FY74V5RBM7LKG5S",
      "GroupName": "TempGroup",
      "Arn": "arn:aws:iam::470686885243:group/TempGroup"
    },
    {
      "_id": "Windows_Access",
      "Path": "/",
      "__NAME__": "Windows_Access",
      "GroupId": "AGPAW3FY74V57Z7SG3GRY",
      "GroupName": "Windows_Access",
      "Arn": "arn:aws:iam::470686885243:group/Windows_Access"
    }
  ],
  ...
}
```

Query a specific group:

Request

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--request GET \
"http://localhost:8080/openidm/system/aws/__GROUP__/developers"
```

Response

```
{
  "_id": "developers",
  "Path": "/",
  "__NAME__": "developers",
  "GroupId": "AGPACKCEVSQ6C2EXAMPLE",
  "GroupName": "developers",
  "Arn": "arn:aws:iam::123456789012:group/developers"
}
```

Query AWS Roles

Query all roles:

Request

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--request GET \
"http://localhost:8080/openidm/system/aws/__ROLE__?_queryFilter=True"
```

Response

```
{
  "result": [
    {
      "_id": "Adminrole",
      "CreatedDate": "Fri Mar 08 13:24:10 IST 2024",
      "AssumeRolePolicyDocument":
"%7B%22Version%22%3A%222012-10-17%22%2C%22Statement%22%3A%5B%7B%22Effect%22%3A%22Allow%22%2C%22Principal%22%3A%7B%22A
WS%22%3A%22arn%3Aaws%3Aiam%3A%3A470686885243%3Aroot%22%7D%2C%22Action%22%3A%22sts%3AAssumeRole%22%2C%22Condition%22%3
A%7B%7D%7D%5D%7D",
      "__NAME__": "Adminrole",
      "Path": "/",
      "RoleArn": "arn:aws:iam::470686885243:role/Adminrole",
      "RoleName": "Adminrole",
      "RoleId": "AROAW3FY74V5XMWBZPK5U"
    },
    {
      "_id": "aws-quicksight-secretsmanager-role-v0",
      "CreatedDate": "Fri Jan 26 23:37:52 IST 2024",
      "AssumeRolePolicyDocument":
"%7B%22Version%22%3A%222012-10-17%22%2C%22Statement%22%3A%5B%7B%22Effect%22%3A%22Allow%22%2C%22Principal%22%3A%7B%22S
ervice%22%3A%22quicksight.amazonaws.com%22%7D%2C%22Action%22%3A%22sts%3AAssumeRole%22%7D%5D%7D",
      "__NAME__": "aws-quicksight-secretsmanager-role-v0",
      "Path": "/service-role/",
      "RoleArn": "arn:aws:iam::470686885243:role/service-role/aws-quicksight-secretsmanager-role-v0",
      "RoleName": "aws-quicksight-secretsmanager-role-v0",
      "RoleId": "AROAW3FY74V54P5FRC3ZC"
    },
    ...
  ]
}
```

Query a specific role:

Request

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--request GET \
"http://localhost:8080/openidm/system/aws/__ROLE__/AWSTokenRole"
```

Response

```
{
  "_id": "AWSTokenRole",
  "CreatedDate": "Mon Mar 28 19:23:45 IST 2022",
  "AssumeRolePolicyDocument": "%7B%22Version%22%3A%222012-10-
17%22%2C%22Statement%22%3A%5B%7B%22Effect%22%3A%22Allow%22%2C%22Principal%22%3A%7B%22AWS%22%3A%22arn%3Aaws%3Aiam%3A%3
A470686885243%3Aroot%22%7D%2C%22Action%22%3A%22sts%3AAssumeRole%22%2C%22Condition%22%3A%7B%7D%7D%5D%7D",
  "__NAME__": "AWSTokenRole",
  "Path": "/",
  "RoleArn": "arn:aws:iam::470686885243:role/AWSTokenRole",
  "RoleName": "AWSTokenRole",
  "RoleId": "AROAW3FY74V54K33FGL7Z"
}
```

Query AWS Managed Policies

Query all managed policies:

Request

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--request GET \
"http://localhost:8080/openidm/system/aws/__MANAGEDPOLICY__?_queryFilter=True"
```

Response

```
{
  "result": [
    { "_id": "arn:aws:iam::aws:policy/AdministratorAccess", ... },
    { "_id": "arn:aws:iam::aws:policy/PowerUserAccess", ... },
    { "_id": "arn:aws:iam::aws:policy/AmazonEC2ReadOnlyAccess", ... },
    ...
  ],
  ...
}
```

Query a specific managed policy using ARN as the ID:

Request

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--request GET \
"http://localhost:8080/openidm/system/aws/__MANAGEDPOLICY__/arn:aws:iam::aws:policy/
AmazonEC2ReadOnlyAccess"
```

Response

```
{
  "_id": "arn:aws:iam::aws:policy/AmazonEC2ReadOnlyAccess",
  "UpdateDate": "...",
  "PolicyArn": "arn:aws:iam::aws:policy/AmazonEC2ReadOnlyAccess",
  "AttachmentCount": "5",
  "CreatedDate": "...",
  "PermissionsBoundaryUsageCount": "0",
  "__NAME__": "AmazonEC2ReadOnlyAccess",
  "PolicyName": "AmazonEC2ReadOnlyAccess",
  "IsAttachable": "true",
  "Path": "/",
  "DefaultVersionId": "v15",
  "PolicyId": "ANPACKCEVSQ6C2EXAMPLE"
}
```

Query AWS Inline Policies

Query all inline policies:

Request

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--request GET \
"http://localhost:8080/openidm/system/aws/__INLINEPOLICY__?_queryFilter=True"
```

Response

```
{
  "result": [
    {
      "_id": "Demo_Inline",
      "Username": "Enduser",
      "PolicyDocument":
"%7B%20%09%22Version%22%3A%20%222012-10-17%22%2C%20%09%22Statement%22%3A%20%5B%20%09%09%7B%20%09%09%09%22Sid%22%3A%20
%22VisualEditor0%22%2C%20%09%09%09%22Effect%22%3A%20%22Allow%22%2C%20%09%09%09%22Action%22%3A%20%5B%20%09%09%09%09%22
iam%3AGenerateCredentialReport%22%2C%20%09%09%09%09%22iam%3AGetAccountPasswordPolicy%22%2C%20%09%09%09%09%22iam%3AUpd
ateCloudFrontPublicKey%22%2C%20%09%09%09%09%22iam%3AGetServiceLastAccessedDetailsWithEntities%22%2C%20%09%09%09%09%22
iam%3AListServerCertificates%22%2C%20%09%09%09%09%22iam%3ASetSTSRegionalEndpointStatus%22%2C%20%09%09%09%09%22iam%3AG
etServiceLastAccessedDetails%22%2C%20%09%09%09%09%22iam%3AListVirtualMFADevices%22%2C%20%09%09%09%09%22iam%3AGetOrgan
izationsAccessReport%22%2C%20%09%09%09%09%22iam%3ASetSecurityTokenServicePreferences%22%2C%20%09%09%09%09%22iam%3AUpd
ateAccountName%22%2C%20%09%09%09%09%22iam%3ASimulateCustomPolicy%22%2C%20%09%09%09%09%22iam%3AGetAccountEmailAddress%
22%2C%20%09%09%09%09%22iam%3AGetCloudFrontPublicKey%22%2C%20%09%09%09%09%22iam%3ACreateAccountAlias%22%2C%20%09%09%09
%09%22iam%3AUpdateAccountEmailAddress%22%2C%20%09%09%09%09%22iam%3AGetAccountAuthorizationDetails%22%2C%20%09%09%09%0
9%22iam%3ADeleteCloudFrontPublicKey%22%2C%20%09%09%09%09%22iam%3ADeleteAccountAlias%22%2C%20%09%09%09%09%22iam%3AGetC
redentialReport%22%2C%20%09%09%09%09%22iam%3AListPolicies%22%2C%20%09%09%09%09%22iam%3ADeleteAccountPasswordPolicy%22
%2C%20%09%09%09%09%22iam%3AListSAMLProviders%22%2C%20%09%09%09%09%22iam%3AListCloudFrontPublicKeys%22%2C%20%09%09%09%
09%22iam%3AListRoles%22%2C%20%09%09%09%09%22iam%3AListInstanceProfiles%22%2C%20%09%09%09%09%22iam%3AUploadCloudFrontP
ublicKey%22%2C%20%09%09%09%09%22iam%3AGetContextKeysForCustomPolicy%22%2C%20%09%09%09%09%22iam%3AUpdateAccountPasswor
dPolicy%22%2C%20%09%09%09%09%22iam%3AListOpenIDConnectProviders%22%2C%20%09%09%09%09%22iam%3AGetAccountName%22%2C%20%
09%09%09%09%22iam%3AListAccountAliases%22%2C%20%09%09%09%09%22iam%3AListUsers%22%2C%20%09%09%09%09%22iam%3AListGroups
%22%2C%20%09%09%09%09%22iam%3AListSTSRegionalEndpointsStatus%22%2C%20%09%09%09%09%22iam%3AGetAccountSummary%22%20%09%
09%09%5D%2C%20%09%09%09%22Resource%22%3A%20%22%2A%22%20%09%09%7D%2C%20%09%09%7B%20%09%09%09%22Sid%22%3A%20%22VisualEd
itor1%22%2C%20%09%09%09%22Effect%22%3A%20%22Allow%22%2C%20%09%09%09%22Action%22%3A%20%22iam%3A%2A%22%2C%20%09%09%09%2
2Resource%22%3A%20%5B%20%09%09%09%09%22arn%3Aaws%3Aiam%3A%3A470686885243%3Auser%2F%2A%22%2C%20%09%09%09%09%22arn%3Aaw
s%3Aiam%3A%3A470686885243%3Aaccess-report%2F%2A%22%2C%20%09%09%09%09%22arn%3Aaws%3Aiam%3A%3A470686885243%3Aoidc-
provider%2F%2A%22%2C%20%09%09%09%09%22arn%3Aaws%3Aiam%3A%3A470686885243%3Apolicy%2F%2A%22%2C%20%09%09%09%09%22arn%3Aa
ws%3Aiam%3A%3A470686885243%3Amfa%2F%2A%22%2C%20%09%09%09%09%22arn%3Aaws%3Aiam%3A%3A470686885243%3Ainstance-
profile%2F%2A%22%2C%20%09%09%09%09%22arn%3Aaws%3Aiam%3A%3A470686885243%3Asms-
mfa%2F%2A%22%2C%20%09%09%09%09%22arn%3Aaws%3Aiam%3A%3A470686885243%3Agroup%2F%2A%22%2C%20%09%09%09%09%22arn%3Aaws%3Ai
am%3A%3A470686885243%3Asaml-
provider%2F%2A%22%2C%20%09%09%09%09%22arn%3Aaws%3Aiam%3A%3A470686885243%3Arole%2F%2A%22%2C%20%09%09%09%09%22arn%3Aaws
%3Aiam%3A%3A470686885243%3Aserver-certificate%2F%2A%22%20%09%09%09%5D%20%09%09%7D%20%09%5D%20%7D",
      "PolicyName": "Demo_Inline",
      "__NAME__": "Demo_Inline"
    },
    {
      "_id": "inline_example",
      "Username": "Enduser",
      "PolicyDocument":
"%7B%0A%09%22Version%22%3A%20%222012-10-17%22%2C%0A%09%22Statement%22%3A%20%5B%0A%09%09%7B%0A%09%09%09%22Sid%22%3A%20
%22VisualEditor0%22%2C%0A%09%09%09%22Effect%22%3A%20%22Allow%22%2C%0A%09%09%09%22Action%22%3A%20%22iam%3A%2A%22%2C%0A
%09%09%09%22Resource%22%3A%20%22%2A%22%0A%09%09%7D%0A%09%5D%0A%7D",
      "PolicyName": "inline_example",
      "__NAME__": "inline_example"
    },
    {
      "_id": "Test_Inline_Policy",
      "Username": "Enduser",
      "PolicyDocument":
"%7B%0A%09%22Version%22%3A%20%222012-10-17%22%2C%0A%09%22Statement%22%3A%20%5B%0A%09%09%7B%0A%09%09%09%22Sid%22%3A%20
%22VisualEditor0%22%2C%0A%09%09%09%22Effect%22%3A%20%22Allow%22%2C%0A%09%09%09%22Action%22%3A%20%22iam%3A%2A%22%2C%0A
%09%09%09%22Resource%22%3A%20%22%2A%22%0A%09%09%7D%0A%09%5D%0A%7D",
      "PolicyName": "Test_Inline_Policy",
      "__NAME__": "Test_Inline_Policy"
    }
  ],
  ...
}
```

Query a specific inline policy:

Request

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--request GET \
"http://localhost:8080/openidm/system/aws/__INLINEPOLICY__/Demo_Inline"
```

Response

```
{
  "_id": "Demo_Inline",
  "Username": "Enduser",
  "PolicyDocument":
"%7B%20%09%22Version%22%3A%20%222012-10-17%22%2C%20%09%22Statement%22%3A%20%5B%20%09%09%7B%20%09%09%09%22Sid%22%3A%20
%22VisualEditor0%22%2C%20%09%09%09%22Effect%22%3A%20%22Allow%22%2C%20%09%09%09%22Action%22%3A%20%5B%20%09%09%09%09%22
iam%3AGenerateCredentialReport%22%2C%20%09%09%09%09%22iam%3AGetAccountPasswordPolicy%22%2C%20%09%09%09%09%22iam%3AUpd
ateCloudFrontPublicKey%22%2C%20%09%09%09%09%22iam%3AGetServiceLastAccessedDetailsWithEntities%22%2C%20%09%09%09%09%22
iam%3AListServerCertificates%22%2C%20%09%09%09%09%22iam%3ASetSTSRegionalEndpointStatus%22%2C%20%09%09%09%09%22iam%3AG
etServiceLastAccessedDetails%22%2C%20%09%09%09%09%22iam%3AListVirtualMFADevices%22%2C%20%09%09%09%09%22iam%3AGetOrgan
izationsAccessReport%22%2C%20%09%09%09%09%22iam%3ASetSecurityTokenServicePreferences%22%2C%20%09%09%09%09%22iam%3AUpd
ateAccountName%22%2C%20%09%09%09%09%22iam%3ASimulateCustomPolicy%22%2C%20%09%09%09%09%22iam%3AGetAccountEmailAddress%
22%2C%20%09%09%09%09%22iam%3AGetCloudFrontPublicKey%22%2C%20%09%09%09%09%22iam%3ACreateAccountAlias%22%2C%20%09%09%09
%09%22iam%3AUpdateAccountEmailAddress%22%2C%20%09%09%09%09%22iam%3AGetAccountAuthorizationDetails%22%2C%20%09%09%09%0
9%22iam%3ADeleteCloudFrontPublicKey%22%2C%20%09%09%09%09%22iam%3ADeleteAccountAlias%22%2C%20%09%09%09%09%22iam%3AGetC
redentialReport%22%2C%20%09%09%09%09%22iam%3AListPolicies%22%2C%20%09%09%09%09%22iam%3ADeleteAccountPasswordPolicy%22
%2C%20%09%09%09%09%22iam%3AListSAMLProviders%22%2C%20%09%09%09%09%22iam%3AListCloudFrontPublicKeys%22%2C%20%09%09%09%
09%22iam%3AListRoles%22%2C%20%09%09%09%09%22iam%3AListInstanceProfiles%22%2C%20%09%09%09%09%22iam%3AUploadCloudFrontP
ublicKey%22%2C%20%09%09%09%09%22iam%3AGetContextKeysForCustomPolicy%22%2C%20%09%09%09%09%22iam%3AUpdateAccountPasswor
dPolicy%22%2C%20%09%09%09%09%22iam%3AListOpenIDConnectProviders%22%2C%20%09%09%09%09%22iam%3AGetAccountName%22%2C%20%
09%09%09%09%22iam%3AListAccountAliases%22%2C%20%09%09%09%09%22iam%3AListUsers%22%2C%20%09%09%09%09%22iam%3AListGroups
%22%2C%20%09%09%09%09%22iam%3AListSTSRegionalEndpointsStatus%22%2C%20%09%09%09%09%22iam%3AGetAccountSummary%22%20%09%
09%09%5D%2C%20%09%09%09%22Resource%22%3A%20%22%2A%22%20%09%09%7D%2C%20%09%09%7B%20%09%09%09%22Sid%22%3A%20%22VisualEd
itor1%22%2C%20%09%09%09%22Effect%22%3A%20%22Allow%22%2C%20%09%09%09%22Action%22%3A%20%22iam%3A%2A%22%2C%20%09%09%09%2
2Resource%22%3A%20%5B%20%09%09%09%09%22arn%3Aaws%3Aiam%3A%3A470686885243%3Auser%2F%2A%22%2C%20%09%09%09%09%22arn%3Aaw
s%3Aiam%3A%3A470686885243%3Aaccess-report%2F%2A%22%2C%20%09%09%09%09%22arn%3Aaws%3Aiam%3A%3A470686885243%3Aoidc-
provider%2F%2A%22%2C%20%09%09%09%09%22arn%3Aaws%3Aiam%3A%3A470686885243%3Apolicy%2F%2A%22%2C%20%09%09%09%09%22arn%3Aa
ws%3Aiam%3A%3A470686885243%3Amfa%2F%2A%22%2C%20%09%09%09%09%22arn%3Aaws%3Aiam%3A%3A470686885243%3Ainstance-
profile%2F%2A%22%2C%20%09%09%09%09%22arn%3Aaws%3Aiam%3A%3A470686885243%3Asms-
mfa%2F%2A%22%2C%20%09%09%09%09%22arn%3Aaws%3Aiam%3A%3A470686885243%3Agroup%2F%2A%22%2C%20%09%09%09%09%22arn%3Aaws%3Ai
am%3A%3A470686885243%3Asaml-
provider%2F%2A%22%2C%20%09%09%09%09%22arn%3Aaws%3Aiam%3A%3A470686885243%3Arole%2F%2A%22%2C%20%09%09%09%09%22arn%3Aaws
%3Aiam%3A%3A470686885243%3Aserver-certificate%2F%2A%22%20%09%09%09%5D%20%09%09%7D%20%09%5D%20%7D",
  "PolicyName": "Demo_Inline",
  "__NAME__": "Demo_Inline"
}
```

Query AWS Service Control Policies (SCPs)

Query all SCPs:

Request

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--request GET \
"http://localhost:8080/openidm/system/aws/__SERVICECONTROLPOLICY__?_queryFilter=True"
```

Response

```
{
  "result": [
    {
      "_id": "p-FullAWSAccess",
      "PolicyName": "FullAWSAccess",
      "__NAME__": "FullAWSAccess",
      "Id": "p-FullAWSAccess",
      "PolicySummary": [
        {
          "Type": "SERVICE_CONTROL_POLICY",
          "Description": "",
          "Arn": "arn:aws:organizations::470686885243:policy/o-r7bvsqr1wd/service_control_policy/p-pcmxrekp",
          "AwsManaged": "false"
        }
      ]
    },
    {
      "_id": "p-pcmxrekp",
      "PolicyName": "Sandbox SCP",
      "__NAME__": "Sandbox SCP",
      "Id": "p-pcmxrekp",
      "PolicySummary": [
        {
          "Type": "SERVICE_CONTROL_POLICY",
          "Description": "",
          "Arn": "arn:aws:organizations::470686885243:policy/o-r7bvsqr1wd/service_control_policy/p-pcmxrekp",
          "AwsManaged": "false"
        }
      ]
    }
  ],
  ...
}
```

Query a specific SCP:

Request

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--request GET \
"http://localhost:8080/openidm/system/aws/__SERVICECONTROLPOLICY__/p-DenyHighRiskActions"
```

Response

```
{
  "_id": "p-pcmxrekp",
  "PolicyName": "Sandbox SCP",
  "__NAME__": "Sandbox SCP",
  "Id": "p-pcmxrekp",
  "PolicySummary": [
    {
      "Type": "SERVICE_CONTROL_POLICY",
      "Description": "",
      "Arn": "arn:aws:organizations::470686885243:policy/o-r7bvsqr1wd/service_control_policy/p-pcmxrekp",
      "AwsManaged": "false"
    }
  ]
}
```

Query AWS organizational units

Query all organizational units:

Request

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--request GET \
"http://localhost:8080/openidm/system/aws/__ORGUNIT__?_queryFilter=True"
```

Response

```
{
  "result": [
    {
      "_id": "ou-2g8u-y0g6eo9k",
      "__NAME__": "ORGTEST",
      "ParentId": "ou-2g8u-y0g6eo9k"
    },
    {
      "_id": "ou-2g8u-jvpza68y",
      "OrganizationalUnits": [
        {
          "Arn": "arn:aws:organizations::470686885243:ou/o-r7bvsqr1wd/ou-2g8u-kgsw9s1e",
          "Name": "1-Sandboxchild"
        }
      ],
      "__NAME__": "Sandbox",
      "ParentId": "ou-2g8u-jvpza68y"
    },
    {
      "_id": "ou-2g8u-mfus8u4b",
      "__NAME__": "Tempexample",
      "ParentId": "ou-2g8u-mfus8u4b"
    },
    {
      "_id": "ou-2g8u-b3z1vwel",
      "__NAME__": "TestOrganization",
      "ParentId": "ou-2g8u-b3z1vwel"
    }
  ],
  ...
}
```

Query a specific organizational unit:

Request

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--request GET \
"http://localhost:8080/openidm/system/aws/__ORGUNIT__/ou-2g8u-jvpza68y"
```

Response

```
{
  "_id": "ou-2g8u-jvpza68y",
  "OrganizationalUnits": [
    {
      "Arn": "arn:aws:organizations::470686885243:ou/o-r7bvsqr1wd/ou-2g8u-kgsw9s1e",
      "Name": "1-Sandboxchild"
    }
  ],
  "__NAME__": "Sandbox",
  "ParentId": "ou-2g8u-jvpza68y"
}
```

OpenICF Interfaces Implemented by the AWS Connector

The AWS Connector implements the following OpenICF interfaces. For additional details, see ICF 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.

AWS Connector Configuration

The AWS Connector has the following configurable properties:

Basic Configuration Properties

Property	Type	Default	Encrypted[1]	Required[2]
`accessKeyId`	`String`	`null`		✓ Yes
Provides the Access Key ID to access the AWS IAM Service API.				
`secretKey`	`GuardedString`	`null`	🔒 Yes	✓ Yes
Provides the Secret Key ID to access the AWS IAM Service API.				
`roleArn`	`String`	`null`		✓ Yes
Provides the Amazon Resource Name specifying the Role.				
`region`	`String`	`null`		✕ No
Provides the Regions.				
`pageSize`	`int`	`100`		✕ No
Provides the Page Size.				
`credentialsExpiration`	`int`	`3600`		✕ No
Provides the temporary credentials expiration time in seconds.				
`parentId`	`String`	`null`		✕ No
Provides the Parent ID to access the Organization Service.				

Property	Type	Default	Encrypted[1]	Required[2]
`userName`	`String`	`null`		✕ No
Provides the UserName to access the Inline policy of a User.				
`proxyHost`	`String`	`null`		✕ No
Provides the ProxyHost.				
`proxyPort`	`Integer`	`null`		✕ No
Provides the ProxyPort.				
`proxyUsername`	`String`	`null`		✕ No
Provides the Proxy Username.				
`proxyPassword`	`GuardedString`	`null`		✕ No
Provides the Proxy Password.				
`connectionTimeout`	`Integer`	`10000`		✕ No
Provides the Maximum Connection Timeout in milliseconds.				
`maxConnections`	`Integer`	`10`		✕ No
Provides the number of Maximum Connections.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

AWS IAM Identity Center connector

The AWS IAM Identity Center connector allows you to manage users and groups, as well as manage user group memberships between the AWS IAM identity center and IDM. You need an administrator account.

Before you start

Before you configure the connector, log in to your AWS administrator account in the web console and obtain the following data to be able to connect: `accessKey` , `secretKey` , `identityStoreId` , `region` , and `roleArn` .

Install the AWS IAM Identity Center connector

> **Tip**
>
> To check for an Advanced Identity Cloud application for this connector, refer to:
>
> - Application management
> - App catalog

You can download any connector from Backstage, but some are included in the default deployment for Advanced Identity Cloud, IDM, or RCS. When using an included connector, you can skip installing it and move directly to configuration.

Connector included in default deployment

Connector	IDM	RCS
AWS IAM Identity Center	✕ No	✕ No

Download the connector .jar file from Backstage.

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

```
mv ~/Downloads/awsiam-connector-1.5.20.28.jar /path/to/openidm/connectors/
```

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

Configure the AWS IAM Identity Center connector

Create a connector configuration using the IDM admin UI:

1. From the navigation bar, click **Configure > Connectors**.
2. On the **Connectors** page, click **New Connector**.
3. On the **New Connector** page, type a **Connector Name**.
4. From the **Connector Type** drop-down list, select **AWS IAM Identity Center Connector - 1.5.20.28**.
5. Complete the **Base Connector Details**.

   > **Tip**
   >
   > For a list of all configuration properties, refer to AWS IAM Identity Center Connector Configuration

6. Click **Save**.

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

Refer to this procedure to create a connector configuration over REST.

- `Access Key ID` : The access key ID is a globally unique IAM user identifier to access the AWS service API.
- `Secret Key ID` : The secret key is a password to access the AWS service API.

- `Role ARN` : Amazon Resource Name (ARN) for the role which has IAM Full Access permissions.
- `Session Name` : A name used to uniquely identify a user session within the identity service.
- `Credentials Expiration Time` : Time (in seconds) to configure the duration in which the temporary credentials would expire. The time must be between 900 and 3600 seconds.
- `Region` : The region where the AWS instance is hosted.
- `Identity Store ID` : Unique identifier associated with an identity store used by AWS IAM Identity Center.
- `Max connections` : Max size of the http connection pool used. Optional.
- `Connection Timeout (seconds)` : Defines a timeout for the http connection in seconds. Optional.
- `ProxyHost` : Proxy server host. Optional.
- `ProxyPort` : Proxy server port number. Optional.
- `ReadRateLimit` : Limits the request rate for read operations. The recommended rate is 20/sec.
- `WriteRateLimit` : Limits the request rate for write operations. The recommended rate is 10/sec.

If necessary, add or edit your object types to have these three objects with their properties:

PROPERTY NAME	TYPE	NATIVE TYPE	REQUIRED
`_id`	String	String	NO
`__NAME__`	String	String	YES
`name`	Object	Object	YES
`displayName`	String	String	YES
`userType`	String	String	NO
`profileUrl`	String	String	NO
`title`	String	String	NO
`preferredLanguage`	String	String	NO
`locale`	String	String	NO
`nickName`	String	String	NO
`timezone`	String	String	NO
`emails`	Array	Object	NO
`phoneNumbers`	Array	Object	NO

PROPERTY NAME	TYPE	NATIVE TYPE	REQUIRED
`addresses`	Array	Object	NO
`externalIds`	Array	Object	NO
`__GROUPS__`	Array	String	NO

PROPERTY NAME	TYPE	NATIVE TYPE	REQUIRED
`_id`	String	String	NO
`__NAME__`	String	String	YES
`description`	String	String	NO
`externalIds`	Array	Object	NO

Note

The __NAME__ field represents the username for users and the groupName for groups.

If configuring the connector over REST or through the filesystem, specify the connection details to the AWS IAM Identity Center resource provider in the `configurationProperties` for the connector. The minimum required properties are `accessKey`, `secretKey`, `roleArn`, `roleSessionName`, `region`, and `identityStoreId`.

```
{
  "configurationProperties": {
    "accessKey": "ACCEES_KEY",
    "secretKey": "xxxxxxxxxxxx",
    "roleArn": "arn:aws:iam::000000000:role/USERNAME_ROLE",
    "roleSessionName": "SESSION_NAME",
    "region": "us-east-2",
    "identityStoreId": "d-0a010101e0",
    "sessionExpirationTime": 3600,
    "proxyHost": null,
    "proxyPort": null,
    "proxyUsername": null,
    "proxyPassword": null,
    "connectionTimeout": null,
    "maxConnections": null,
    "readRateLimit": "20/sec",
    "writeRateLimit": "10/sec"
  }
}
```

Note

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

Configure connection pooling

The AWS IAM Identity Center connector uses a non-poolable mechanism to manage connections. Learn more about the different pooling mechanisms in Connectors by pooling mechanism.

Mapping

Attributes mapping table where the columns represent the attribute name mapped from source to target and the necessary data transformation to synchronize successfully.

SOURCE	TARGET	TRANSFORMATION SCRIPT
`_id`	`_id`	N/A
`__NAME__`	`userName`	N/A
`displayName`	`displayName`	N/A
`timezone`	`timezone`	N/A
`nickname`	`nickname`	N/A
`title`	`title`	N/A
`locale`	`locale`	N/A
`preferredLanguage`	`preferredLanguage`	N/A
`profileUrl`	`profileUrl`	N/A
`userType`	`userType`	N/A
`name`	`name`	N/A
`phoneNumbers`	`phoneNumbers`	N/A
`addresses`	`addresses`	N/A
`emails`	`emails`	N/A
`externalIds`	`externalIds`	N/A
`__GROUPS__`	`groups`	N/A

Attributes mapping table where the columns represent the attribute name mapped from source to target and the necessary data transformation to synchronize successfully.

SOURCE	TARGET	TRANSFORMATION SCRIPT
`userName`	`__NAME__`	N/A
`displayName`	`displayName`	N/A
`timezone`	`timezone`	N/A
`nickname`	`nickname`	N/A
`title`	`title`	N/A
`locale`	`locale`	N/A
`preferredLanguage`	`preferredLanguage`	N/A
`profileUrl`	`profileUrl`	N/A
`userType`	`userType`	N/A
`name`	`name`	N/A
`phoneNumbers`	`phoneNumbers`	N/A
`addresses`	`addresses`	N/A
`emails`	`emails`	N/A
`__GROUPS__`	`groups`	N/A

Attributes mapping table where the columns represent the attribute name mapped from source to target and the necessary data transformation to synchronize successfully.

SOURCE	TARGET	TRANSFORMATION SCRIPT
`_id`	`_id`	N/A
`__NAME__`	`groupName`	N/A
`description`	`description`	N/A
`externalIds`	`externalIds`	N/A

Attributes mapping table where the columns represent the attribute name mapped from source to target and the necessary data transformation to synchronize successfully.

SOURCE	TARGET	TRANSFORMATION SCRIPT
`__NAME__`	`groupName`	N/A
`description`	`description`	N/A

Test the AWS IAM Identity Center connector

Test that the connector was configured correctly:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Accept-API-Version: resource=1.0' \
--request POST \
'http://localhost:8080/system/awsiam?_action=test'
{
    "name": "awsiam",
    "enabled": true,
    "config": "config/provisioner.openicf/awsiam",
    "connectorRef": {
    "bundleVersion": "1.5.20.28",
    "bundleName": "org.forgerock.openicf.connectors.awsiam-connector",
    "connectorName": "org.forgerock.openicf.connectors.awsiam.AwsIamConnector"
    },
    "displayName": "AWS IAM IC Connector",
    "objectTypes": [
    "__ACCOUNT__",
    "__ALL__",
    "__GROUP__"
    ],
    "ok": true
}
```

Use the AWS IAM Identity Center connector

User

Create user

To create a user in AWS IAM Identity Center, you must provide *at least* the `__NAME__`, `name` (givenName and familyName) and `displayName` fields.

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request POST \
--data '{
    "__NAME__": "JohnDoe",
    "displayName": "John Doe",
    "locale": "US",
    "nickName": "JonnyDoe",
    "timezone": "UTC",
    "title": "Engineer",
    "profileUrl": "https://www.profile.com/jdoe",
    "userType": "USER",
    "preferredLanguage": "us-US",
    "name": {
        "givenName": "John",
        "middleName": "Michael",
        "familyName": "Doe",
        "honorificPrefix": "Sr.",
        "honorificSufix": "PhD",
        "formatted": "Sr. John Michael Doe, PhD"
    },
    "addresses": {
        "type": "home",
        "streetAddress": "123 Main St",
        "locality": "Springfield",
        "region": "IL",
        "postalCode": "62701",
        "country": "USA",
        "primary": true,
        "formatted": "123 Main St, Springfield, IL 62701, USA"
    },
    "emails": {
        "type": "home",
        "value": "johndoe@example.com",
        "primary": true
    },
    "phoneNumbers": {
        "type": "mobile",
        "value": "+0101010101",
        "primary": true
    },
    "__GROUPS__": [
        "groupId1",
        "groupId2",
    ]
}' \
'http://localhost:8080/system/awsiam/__ACCOUNT__?_action=create'
{
    "_id" : " "userId",
    "__NAME__": "JohnDoe",
    "displayName": "John Doe",
    "locale": "US",
    "nickName": "JonnyDoe",
    "timezone": "UTC",
    "title": "Engineer",
    "profileUrl": "https://www.profile.com/jdoe",
    "userType": "USER",
    "preferredLanguage": "us-US",
    "name": {
        "givenName": "John",
        "middleName": "Michael",
        "familyName": "Doe",
        "honorificPrefix": "Sr.",
        "honorificSufix": "PhD",
        "formatted": "Sr. John Michael Doe, PhD"
    },
    "addresses": {
        "type": "home",
        "streetAddress": "123 Main St",
        "locality": "Springfield",
        "region": "IL",
        "postalCode": "62701",
        "country": "USA",
        "primary": true,
        "formatted": "123 Main St, Springfield, IL 62701, USA"
    },
    "emails": {
        "type": "home",
        "value": "johndoe@example.com",
        "primary": true
    },
    "phoneNumbers": {
        "type": "mobile",
        "value": "+0101010101",
        "primary": true
    },
    "__GROUPS__": [
        "groupId1",
        "groupId2",
    ]
}
```

Get Users

Return all users from AWS IAM Identity Center.

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request GET \
'http://localhost:8080/openidm/system/awsiam/__ACCOUNT__?_queryFilter=true'
{
    "result": [
        {
            "_id": "",
            "__NAME__": "jdoe",
            "displayName": "John Doe",
            "name": {
                "givenName": "John",
                "middleName": "Michael",
                "familyName": "Doe",
            },
            "addresses": [].
            "emails": [],
            "phoneNumbers": [],
            "__GROUPS__": [
                "groupId1",
                "groupId2"
            ]
        },
    ...
        {
            "_id": "",
            "__NAME__": "jdoe",
            "displayName": "John Doe",
            "name": {
                "givenName": "John",
                "middleName": "Michael",
                "familyName": "Doe",
            },
            "addresses": [].
            "emails": [],
            "phoneNumbers": [],
            "__GROUPS__": [
                "groupId1",
                "groupId2"
            ]
        },
    ],
    "resultCount": 999,
    "pagedResultsCookie": null,
    "totalPagedResultsPolicy": "NONE",
    "totalPagedResults": -1,
    "remainingPagedResults": -1
}
```

> **Note**
>
> To paginate the results, the maximum value of `_pageSize` is 100.

Get user

Return a user from AWS IAM Identity Center. The user ID must be provided in the URI path:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request GET \
'http://localhost:8080/openidm/system/awsiam/__ACCOUNT__/USER_ID'
{
    "_id" : " "userId",
    "__NAME__": "jdoe",
    "displayName": "John Doe",
    "locale": "en-US",
    "nickname": "Johnny",
    "timezone": "America/New_York",
    "title": "Software Engineer",
    "profileUrl": "https://www.profile.com/jdoe",
    "userType": "employee",
    "preferredLanguage": "en",
    "name": {
        "givenName": "John",
        "middleName": "Michael",
        "familyName": "Doe",
        "honorificPrefix": "Sr.",
        "honorificSufix": "PhD",
        "formatted": "Sr. John Michael Doe, PhD"
    },
    "addresses": {
        "type": "home",
        "streetAddress": "123 Main St",
        "locality": "Springfield",
        "region": "IL",
        "postalCode": "62701",
        "country": "USA",
        "primary": true,
        "formatted": "123 Main St, Springfield, IL 62701, USA"
    },
    "emails": {
        "type": "work",
        "value": "john.doe@example.com",
        "primary": true
    },
    "phoneNumbers": {
        "type": "mobile",
        "value": "+0101010101",
        "primary": true
    },
    "__GROUPS__": [
        "groupId1",
        "groupId2"
    ]
}
```

Get user by filter

Return a user from AWS IAM Identity Center:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request GET \
'http://localhost:8080/openidm/system/awsiam/__ACCOUNT___queryFilter=__NAME__%20eq%20"name"'
{
    "_id" : " "userId",
    "__NAME__": "jdoe",
    "displayName": "John Doe",
    "locale": "en-US",
    "nickname": "Johnny",
    "timezone": "America/New_York",
    "title": "Software Engineer",
    "profileUrl": "https://www.profile.com/jdoe",
    "userType": "employee",
    "preferredLanguage": "en",
    "name": {
        "givenName": "John",
        "middleName": "Michael",
        "familyName": "Doe",
        "honorificPrefix": "Sr.",
        "honorificSufix": "PhD",
        "formatted": "Sr. John Michael Doe, PhD"
    },
    "addresses": {
        "type": "home",
        "streetAddress": "123 Main St",
        "locality": "Springfield",
        "region": "IL",
        "postalCode": "62701",
        "country": "USA",
        "primary": true,
        "formatted": "123 Main St, Springfield, IL 62701, USA"
    },
    "emails": {
        "type": "work",
        "value": "john.doe@example.com",
        "primary": true
    },
    "phoneNumbers": {
        "type": "mobile",
        "value": "+0101010101",
        "primary": true
    },
    "__GROUPS__": [
        "groupId1",
        "groupId2"
    ]
}
```

Note

The __NAME__ field only supports the equal filter.

Get users IDs

Return all users from AWS IAM Identity Center displaying only the `_id` field:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request GET \
'http://localhost:8080/openidm/system/awsiam/__ACCOUNT__?_queryId=query-all-ids'
{
    "result": [
        {
            "_id": "userID1"
        },
        ...
        {
            "_id": userID2"
        }
    ],
    "resultCount": 999,
    "pagedResultsCookie": null,
    "totalPagedResultsPolicy": "NONE",
    "totalPagedResults": -1,
    "remainingPagedResults": -1
}
```

Update user

Update a user in AWS IAM Identity Center. The user ID must be provided in the URI path:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request PUT \
--data '{
    "__NAME__": "JonnyDoe",
    "displayName": "Jonny Doe",
    "locale": "US",
    "nickName": "JonnyDoe",
    "timezone": "UTC",
    "title": "",
    "profileUrl": "https://www.profile.com/jonnydoe",
    "userType": "USER",
    "preferredLanguage": "us-US",
    "name": {
        "givenName": "Jonny",
        "middleName": "Michael",
        "familyName": "Doe",
        "honorificPrefix": "Jr.",
        "honorificSufix": "PhD",
        "formatted": "Jr. John Michael Doe, PhD"
    },
    "addresses": {
        "type": "home",
        "streetAddress": "123 Main St",
        "locality": "Springfield",
        "region": "IL",
        "postalCode": "60999",
        "country": "US",
        "primary": true,
        "formatted": "123 Main St, Springfield, IL 62701, USA"
    },
    "emails": {
        "type": "home",
        "value": "johndoe@example.com",
        "primary": true
    },
    "phoneNumbers": {
        "type": "home",
        "value": "505050",
        "primary": true
    },
    "__GROUPS__": [
        "groupID1",
        "groupID2",
    ]
}' \
'http://localhost:8080/system/awsiam/__ACCOUNT__/USER_ID'
{
    "_id" : "userId",
    "__NAME__" : "JonnyDoe",
    "displayName" : "Jonny Doe",
    "locale" : "US",
    "nickName" : "JonnyDoe",
    "timezone" : "UTC",
    "title" : "",
    "profileUrl" : "https://www.profile.com/jonnydoe",
    "userType" : "USER",
    "preferredLanguage" : "us-US",
    "name" : {
        "givenName" : "Jonny",
        "middleName" : "middleName",
        "familyName" : "Doe",
        "honorificPrefix" : "Jr",
        "honorificSufix" : "PhD",
        "formatted" : "Jr. John Doe, PhD"
    },
    "addresses": {
        "type": "home",
        "streetAddress": "123 Main St",
        "locality": "Springfield",
        "region": "IL",
        "postalCode": "60999",
        "country": "US",
        "primary": true,
        "formatted": "123 Main St, Springfield, IL 62701, USA"
    },
    "emails" : {
        "type" : "home",
        "value" : "johndoe@example.com",
        "primary" : true
    },
    "phoneNumbers" : {
        "type" : "home",
        "value" : "505050",
        "primary" : true
    },
    "__GROUPS__" : [
        "groupID1",
        "groupID2",
    ]
}
```

Delete user

Delete a user in the AWS IAM Identity Center. The user ID must be provided in the URI path:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request DELETE \
'http://localhost:8080/openidm/system/awsiam/__ACCOUNT__/USER_ID'
{
    "_id" : "userId",
    "__NAME__" : "JohnDoe",
    "displayName" : "John Doe",
    "locale" : "US",
    "nickName" : "JonnyDoe",
    "timezone" : "UTC",
    "title" : "",
    "profileUrl" : "www.example.doe",
    "userType" : "USER",
    "preferredLanguage" : "us-US",
    "name" : {
        "givenName" : "John",
        "middleName" : "middleName",
        "familyName" : "Doe",
        "honorificPrefix" : "Sr",
        "honorificSufix" : "PhD",
        "formatted" : "Sr. John Doe, PhD"
    },
    "addresses" : {
        "type" : "home",
        "streetAddress" : "false street",
        "locality" : "springfield",
        "region" : "north",
        "postalCode" : "0000",
        "country" : "US",
        "primary" : false,
        "formatted" : "no"
    },
    "emails" : {
        "type" : "home",
        "value" : "testeruser@example.com",
        "primary" : true
    },
    "phoneNumbers" : {
        "type" : "home",
        "value" : "505050",
        "primary" : true
    },
    "__GROUPS__" : [
        "groupID1",
        "groupID2",
    ]
}
```

GROUPS

Create group

To create a group in AWS IAM Identity Center, it is necessary to *at least* provide the `__NAME__` field. The `description` field is optional:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request POST \
--data '{
    "__NAME__" : "New Group",
    "description" : "Some description"
}' \
'http://localhost:8080/openidm/system/awsiam/__GROUP__?_action=create'
{
    "_id": "groupId",
    "description": "description",
    "__NAME__": "New Group",
    "externalIds": []
}
```

Get groups

Return all groups from AWS IAM Identity Center.

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request GET \
'http://localhost:8080/openidm/system/awsiam/__GROUP__?_queryFilter=true'
{
    "result": [
        {
            "_id": "groupId1",
            "__NAME__": "Display name group 1",
            "description": "description",
            "externalIds": []
        },
        ...
        {
            "_id": "groupId99",
            "__NAME__": "Display name group 99",
            "description": "description",
            "externalIds": []
        }
    ],
    "resultCount": 99,
    "pagedResultsCookie": null,
    "totalPagedResultsPolicy": "NONE",
    "totalPagedResults": -1,
    "remainingPagedResults": -1
}
```

Note

To paginate the results, the maximum value of `_pageSize` is 100.

Get groups IDs

Return all groups from AWS IAM Identity Center displaying only the `_id` field:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request GET \
'http://localhost:8080/openidm/system/awsiam/__GROUP__?_queryId=query-all-ids'
{
    "result": [
        {
            "_id": "groupID1",
        },
        ...
        {
            "_id": "groupID99",
        }
    ],
    "resultCount": 99,
    "pagedResultsCookie": null,
    "totalPagedResultsPolicy": "NONE",
    "totalPagedResults": -1,
    "remainingPagedResults": -1
}
```

Get group

Return a group from AWS IAM Identity Center. The group ID must be provided in the URI path:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request GET \
'http://localhost:8080/openidm/system/awsiam/__GROUP__/GROUP_ID'
{
    "_id": "groupId",
    "description": "Some description",
    "__NAME__": "Group Name",
    "externalIds": []
}
```

Get group by filter

Return a group from AWS IAM Identity Center:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request GET \
'http://localhost:8080/openidm/system/awsiam/__GROUP___queryFilter=__NAME__%20eq%20"username"'
{
    "_id": "groupId",
    "description": "Some description",
    "__NAME__": "Group Name",
    "externalIds": []
}
```

Note

The __NAME__ field only supports the equal filter.

Update a group

Update a group in AWS IAM Identity Center. The group ID must be provided in the URI path:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request PUT \
--header 'If-Match: *' \
--data '{
    "__NAME__" : "New DisplayName",
    "description" : "New Description"
}' \
'http://localhost:8080/openidm/system/awsiam/__GROUP__/GROUP_ID'
{
    "_id": "groupId",
    "description": "New description",
    "__NAME__": "New DisplayName",
    "externalIds": []
}
```

Delete a group

Delete a group in AWS IAM Identity Center. The group ID must be provided in the URI path:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request DELETE \
'http://localhost:8080/openidm/system/awsiam/__GROUP__/GROUP_ID'
{
    "_id": "groupId",
    "description": "description",
    "__NAME__": "deleted group",
    "externalIds": []
}
```

OpenICF Interfaces Implemented by the AWS IAM Identity Center Connector

The AWS IAM Identity Center Connector implements the following OpenICF interfaces. For additional details, see ICF 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.

AWS IAM Identity Center Connector Configuration

The AWS IAM Identity Center Connector has the following configurable properties:

Basic Configuration Properties

Property	Type	Default	Encrypted(1)	Required(2)
`accessKey`	`String`	`null`		✓ Yes
Provides the Access Key ID to access the AWS IAM IC Service API.				
`secretKey`	`GuardedString`	`null`	🔒 Yes	✓ Yes
Provides the Secret Key ID to access the AWS IAM IC Service API.				
`roleArn`	`String`	`null`		✓ Yes
Provides the Amazon Resource Name specifying the Role.				
`roleSessionName`	`String`	`null`		✓ Yes
Temporary name for the role session.				
`region`	`String`	`null`		✓ Yes
Provides the Regions.				
`identityStoreId`	`String`	`null`		✓ Yes
Provides the identity store ID for the user and group store.				
`sessionExpirationTime`	`Integer`	`3600`		✓ Yes
Provides the temporary Session expiration time in seconds.				

Property	Type	Default	Encrypted[1]	Required[2]
`proxyHost`	`String`	`null`		✕ No
Provides the Proxy Host.				
`proxyPort`	`String`	`null`		✕ No
Provides the Proxy Port.				
`proxyUsername`	`String`	`null`		✕ No
Provides the Proxy Username.				
`proxyPassword`	`GuardedString`	`null`	🔒 Yes	✕ No
Provides the Proxy Password.				
`connectionTimeout`	`Integer`	`null`		✕ No
Provides the Maximum Connection Timeout in seconds.				
`maxConnections`	`Integer`	`null`		✕ No
Provides the number of Maximum Connections.				
`readRateLimit`	`String`	`null`		✓ Yes
Defines throttling for read operations either per seconds ("30/sec") or per minute ("100/min").				
`writeRateLimit`	`String`	`null`		✓ Yes
Defines throttling for write operations (create/update/delete) either per second ("30/sec") or per minute ("100/min").				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

Box connector

> **Tip**
>
> This is a SaaS common connector.

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

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

Before you start

The instructions in this guide assume you have a Box Administrator Account and you have created and authorized a Custom Application, as described in the Box Documentation. Before you configure the connector, log in to your administrator account and note the following information:

- Client ID
- Client Secret
- Authentication Method
- Grant Type
- Subject Type
- Subject ID
- Service Uri
- Token Endpoint

Install the Box connector

> **Tip**
>
> To check for an Advanced Identity Cloud application for this connector, refer to:
>
> - Application management
> - App catalog

You can download any connector from Backstage, but some are included in the default deployment for Advanced Identity Cloud, IDM, or RCS. When using an included connector, you can skip installing it and move directly to configuration.

Connector included in default deployment

Connector	IDM	RCS
Box	✕ No	✕ No

Download the connector .jar file from Backstage.

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

```
mv ~/Downloads/box-connector-1.5.20.30.jar /path/to/openidm/connectors/
```

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

Configure the Box connector

Create a connector configuration using the IDM admin UI:

1. From the navigation bar, click **Configure** > **Connectors**.
2. On the **Connectors** page, click **New Connector**.
3. On the **New Connector** page, type a **Connector Name**.
4. From the **Connector Type** drop-down list, select **Box Connector - 1.5.20.30**.
5. Complete the **Base Connector Details**.

> **Tip**
>
> For a list of all configuration properties, refer to Box Connector Configuration

6. Click **Save**.

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

Refer to this procedure to create a connector configuration over REST.

The following excerpt shows sample configuration properties:

```
"configurationProperties": {
  "serviceUri" : "_CHANGEME_",
  "tokenEndpoint" : "_CHANGEME_",
  "clientId" : "_CHANGEME_",
  "clientSecret" : "_CHANGEME_",
  "acceptSelfSignedCertificates" : true,
  "disableHostNameVerifier" : true,
  "authenticationMethod" : "_CHANGEME_",
  "grantType" : "_CHANGEME_",
  "useBasicAuthForOauthTokenNeg" : false,
  "boxSubjectType" : "_CHANGEME_",
  "boxSubjectId" : "_CHANGEME_"
  "rateLimit": "_CHANGEME_"
}
```

`serviceUri`

The Box API hostname. In most cases it should be `https://api.box.com/2.0`.

`tokenEndpoint`

URL to obtain `access tokens` and `refresh tokens`. In most cases it should be `https://api.box.com/oauth2/token`.

`clientId`

The Box Application `Client ID`. To locate this value, log in to your Box account and go to **Dev Console** > **Box Developer** > **My Apps** > **Select the app** > **Configuration** > **OAuth 2.0 Credentials** > **Client ID**.

`clientSecret`

The Box Application `Secret Key`. To locate this value, log in to your Box account and go to **Dev Console** > **Box Developer** > **My Apps** > **Select the app** > **Configuration** > **OAuth 2.0 Credentials** > **Client Secret**.

`acceptSelfSignedCertificates`

The `acceptSelfSignedCertificates` option enables Box Sync to connect to Box servers that present self-signed digital certificates.

`disableHostNameVerifier`

The `disableHostNameVerifier` is a configuration option used to disable host name verification on `HTTPS` connections.

`grantType`

Parameter used within the `OAuth 2.0 authorization` flow to specify the type of grant being used to obtain an access token. The only value supported is `client_credentials`.

`boxSubjectType`

The Box Application `SubjectType`. `User` or `Enterprise`, according to availability. To locate this value, log in to your Box account and go to **Dev Console** > **Box Developer** > **My Apps** > **Select the app** > **General Settings** > **UserID / EnterpriseID**.

`boxSubjectId`

The Box Application `User ID` or `Enterprise ID`. It must match with the selected `boxSubjectType`. To locate this value, log in to your Box account and go to **Dev Console** > **Box Developer** > **My Apps** > **Select the app** > **General Settings** > **UserID / EnterpriseID**.

`rateLimit`

Limits how many requests the connector makes over a certain period of time. The default value is 1000 requests per minute (`1000/min`) as described in the Box Documentation. Additional examples: `997/min` or `600/sec`.

> **Note**
>
> If throttling problems occur, this guide can be helpful:
> Improve reconciliation query performance

Test the Box connector

Test that the configuration is correct by running the following command:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request POST \
'http://localhost:8080/openidm/system/box?_action=test'
{
  "name": box,
  "enabled": true,
  "config": "config/provisioner.openicf/box",
  "connectorRef": {
    "bundleVersion": "[1.5.0.0,1.6.0.0)",
    "bundleName": "org.forgerock.openicf.connectors.box-connector",
    "connectorName": "org.forgerock.openicf.connectors.box.BoxConnector"
  },
  "displayName": "org.forgerock.openicf.connectors.box.BoxConnector",
  "objectTypes": [
    "__ACCOUNT__",
    "__ALL__",
    "__GROUP__"
  ],
  "ok": true
}
```

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

Box remote connector

If you want to run this connector outside of PingOne Advanced Identity Cloud or IDM, you can configure the Box connector as a remote connector. Java Connectors installed remotely on a Java Connector Server function identically to those bundled locally within PingOne Advanced Identity Cloud or installed locally on IDM.

You can download the Box connector from here.

Refer to Remote connectors for configuring the Box remote connector.

Configure connection pooling

The Box connector supports HTTP pooling, which can substantially improve the performance of the connector. Learn more about the basic connection pooling configuration and different pooling mechanisms described in Connection pooling configuration.

Use the Box connector

You can use the Box connector to perform the following actions on a Box account.

Users

Create a Box user

This example creates a Box user with the minimum required attributes.

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \ \
--header 'Content-Type: application/json' \
--request POST \
--data '{
  "__NAME__": "Jane Doe",
  "login": "janeDoe@example.com"
}' \
'http://localhost:8080/openidm/system/Box/__ACCOUNT__'
{
  "_id": "34383152830",
  "hostname": "https://app.box.com/",
  "__NAME__": "Jane Doe",
  "is_exempt_from_device_limits": "false",
  "type": "user",
  "job_title": "",
  "space_amount": "1.000000456753152E15",
  "phone": "",
  "status": "active",
  "enterprise": [
    {
      "type": "enterprise",
      "id": "1162568706",
      "name": "testing"
    }
  ],
  "can_see_managed_users": "true",
  "is_external_collab_restricted": "false",
  "external_app_user_id": null,
  "is_exempt_from_login_verification": "false",
  "is_sync_enabled": "true",
  "groups": [],
  "max_upload_size": "5.36870912E10",
  "language": "en",
  "login": "janedoe@nexter.net",
  "avatar_url": "https://app.box.com/api/avatar/large/34721687671",
  "role": "user",
  "address": "",
  "is_platform_access_only": "false"
}
```

> **Note**
>
> When you create a new user, you must specify at least the `login` and `__NAME__` attributes. The `login` attribute is typically the user's email address.

Create a Box full user

This example creates a Box full user.

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request POST \
--data '{
  "__NAME__": "Miguel Benitez",
  "is_exempt_from_device_limits": "false",
  "type": "user",
  "job_title": "Designer",
  "space_amount": "1.000000456753152E15",
  "phone": "578945621",
  "status": "active",
  "enterprise": [
    {
      "type": "enterprise",
      "name": "testing"
    }
  ],
  "can_see_managed_users": "true",
  "is_external_collab_restricted": "false",
  "external_app_user_id": null,
  "is_exempt_from_login_verification": "false",
  "is_sync_enabled": "true",
  "groups": ["20013904637", "20013904699"],
  "max_upload_size": "5.36870912E10",
  "language": "en",
  "login": "someone@example.com",
  "avatar_url": "https://app.box.com/api/avatar/large/34721853021",
  "role": "user",
  "address": "San Carlos Buenos Aires",
  "is_platform_access_only": "false"
}' \
'http://localhost:8080/openidm/system/Box/__ACCOUNT__'
{
  "_id": "34721853021",
  "hostname": "https://app.box.com/",
  "__NAME__": "Miguel Benitez",
  "is_exempt_from_device_limits": "false",
  "type": "user",
  "job_title": "Designer",
  "space_amount": "1.000000456753152E15",
  "phone": "578945621",
  "status": "active",
  "enterprise": [
    {
      "type": "enterprise",
      "id": "1162568706",
      "name": "testing"
    }
  ],
  "can_see_managed_users": "true",
  "is_external_collab_restricted": "false",
  "external_app_user_id": null,
  "is_exempt_from_login_verification": "false",
  "is_sync_enabled": "true",
  "groups": [
    "20013904637",
    "20013904699"
  ],
  "max_upload_size": "5.36870912E10",
  "language": "en",
  "login": "someone@example.com",
  "avatar_url": "https://app.box.com/api/avatar/large/34721853021",
  "role": "user",
  "address": "San Carlos, Buenos Aires",
  "is_platform_access_only": "false"
}
```

Note

Attribute limitations:

- `job_title` : Max length 100.
- `phone` : Max length 100.
- `address` : Max length 255.
- `language` : The language of the user, formatted in a modified version of the ISO 639-1 format.
- `role` : The user’s enterprise role. Value is `coadmin` or `user` .
- `status` : Value is one of `active` , `inactive` , `cannot_delete_edit` , `cannot_delete_edit_upload` .
- `space_amount` : (int64) The user’s total available space in bytes. Set this to -1 to indicate unlimited storage.
- `timezone` : The user’s timezone. Example: "Africa/Bujumbura".

List all Box users

This example queries all Box users:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request GET \
'http://localhost:8080/openidm/system/Box/__ACCOUNT__?_queryFilter=True'
{
  "result": [
    {
      "_id": "34383152830",
      "hostname": "https://app.box.com/",
      "__NAME__": "Jane Doe",
      "is_exempt_from_device_limits": "false",
      "type": "user",
      "job_title": "",
      "space_amount": "1.000000456753152E15",
      "phone": "",
      "status": "active",
      "enterprise": [
        {
          "type": "enterprise",
          "id": "1162568706",
          "name": "testing"
        }
      ],
      "can_see_managed_users": "true",
      "is_external_collab_restricted": "false",
      "external_app_user_id": null,
      "is_exempt_from_login_verification": "false",
      "is_sync_enabled": "true",
      "groups": [],
      "max_upload_size": "5.36870912E10",
      "language": "en",
      "login": "janedoe@nexter.net",
      "avatar_url": "https://app.box.com/api/avatar/large/34383152830",
      "role": "user",
      "address": "",
      "is_platform_access_only": "false"
    },
    ...
    {
      "_id": "34721853021",
      "hostname": "https://app.box.com/",
      "__NAME__": "Miguel Benitez",
      "is_exempt_from_device_limits": "false",
      "type": "user",
      "job_title": "Designer",
      "space_amount": "1.000000456753152E15",
      "phone": "578945621",
      "status": "active",
      "enterprise": [
        {
          "type": "enterprise",
          "id": "1162568706",
          "name": "testing"
        }
      ],
      "can_see_managed_users": "true",
      "is_external_collab_restricted": "false",
      "external_app_user_id": null,
      "is_exempt_from_login_verification": "false",
      "is_sync_enabled": "true",
      "groups": ["20013904637", "20013904699"],
      "max_upload_size": "5.36870912E10",
      "language": "en",
      "login": "someone@example.com",
      "avatar_url": "https://app.box.com/api/avatar/large/34721853021",
      "role": "user",
      "address": "San Carlos, Buenos Aires",
      "is_platform_access_only": "false"
    }
  ],
  "resultCount": 10,
  "pagedResultsCookie": "eyJ0eXBlIjoiaWQiLCJkaXIiOiJuZXh0IiwidGFpbCI6IjM0NzIxODUzMDIxIn0",
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

Note

- For `pagedResultsCookie`, the last page returned is empty.
- `startsWith` for `__NAME__` is the only filter available.

List all Box user IDs

This example queries all Box users by their IDs:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request GET \
'http://localhost:8080/openidm/system/Box/__ACCOUNT__?_queryId=query-all-ids'
{
  "result": [
    {
      "_id": "32996521506"
    },
    ...
    {
      "_id": "34721853021"
    }
  ],
  "resultCount": 10,
  "pagedResultsCookie": "eyJ0eXBlIjoiaWQiLCJkaXIiOiJuZXh0IiwidGFpbCI6IjM0NzIxODUzMDIxIn0",
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

Get Box user

The following command queries a specific Box user by its ID:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request GET \
'http://localhost:8080/openidm/system/Box/__ACCOUNT__/34721853021'
{
  "_id": "34721853021",
  "hostname": "https://app.box.com/",
  "__NAME__": "Miguel Benitez",
  "is_exempt_from_device_limits": "false",
  "type": "user",
  "job_title": "",
  "space_amount": "1.000000456753152E15",
  "phone": "578945621",
  "status": "active",
  "enterprise": [
    {
      "type": "enterprise",
      "id": "1162568706",
      "name": "testing"
    }
  ],
  "can_see_managed_users": "true",
  "is_external_collab_restricted": "false",
  "external_app_user_id": null,
  "is_exempt_from_login_verification": "false",
  "is_sync_enabled": "true",
  "groups": [
    "20013904637",
    "20013904699"
  ],
  "max_upload_size": "5.36870912E10",
  "language": "en",
  "login": "someone@example.com",
  "avatar_url": "https://app.box.com/api/avatar/large/34721853021",
  "role": "user",
  "address": "San Carlos, Buenos Aires",
  "is_platform_access_only": "false"
}
```

Update a Box user

The following command updates a specific Box user by its ID:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--header "If-Match: *" \
--request PUT \
--data '{
  "__NAME__": "Miguel Benitez",
  "is_exempt_from_device_limits": "false",
  "type": "user",
  "job_title": "Web Developer",
  "space_amount": "1.000000456753152E15",
  "phone": "1157199024",
  "status": "active",
  "enterprise": [
    {
      "type": "enterprise",
      "id": "1162568706",
      "name": "testing"
    }
  ],
  "can_see_managed_users": "true",
  "is_external_collab_restricted": "false",
  "external_app_user_id": null,
  "is_exempt_from_login_verification": "false",
  "is_sync_enabled": "true",
  "groups": [agregarle el grupo que esta arriba],
  "max_upload_size": "5.36870912E10",
  "language": "en",
  "avatar_url": "https://app.box.com/api/avatar/large/34721853021",
  "role": "user",
  "address": "Puerto La Cruz P.R",
  "is_platform_access_only": "false"
}' \
'http://localhost:8080/openidm/system/Box/__ACCOUNT__/34721853021'
{
  "_id": "34721853021",
  "hostname": "https://app.box.com/",
  "__NAME__": "Miguel Benitez",
  "is_exempt_from_device_limits": "false",
  "type": "user",
  "job_title": "Web Developer",
  "space_amount": "9.99999999999999E14",
  "phone": "1157199024",
  "status": "active",
  "enterprise": [
    {
      "type": "enterprise",
      "id": "1162568706",
      "name": "testing"
    }
  ],
  "can_see_managed_users": "true",
  "is_external_collab_restricted": "false",
  "external_app_user_id": null,
  "is_exempt_from_login_verification": "false",
  "is_sync_enabled": "true",
  "groups": [
    "20013904637",
    "20013904699"
  ],
  "max_upload_size": "5.36870912E10",
  "language": "en",
  "login": "someone@example.com",
  "avatar_url": "https://app.box.com/api/avatar/large/34721853021",
  "role": "user",
  "address": "Puerto La Cruz, P.R",
  "is_platform_access_only": "false"
}
```

Note

If the target user’s email is not confirmed, you can’t change the primary login email address.

Delete a Box user

The following example deletes a Box user:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--header "If-Match: *" \
--request DELETE \
'http://localhost:8080/openidm/system/Box/__ACCOUNT__/34721853021'
{
  "_id": "34721853021",
  "is_platform_access_only": "false",
  "is_sync_enabled": "true",
  "avatar_url": "https://app.box.com/api/avatar/large/34740572881",
  "hostname": "https://app.box.com/",
  "external_app_user_id": null,
  "is_exempt_from_login_verification": "false",
  "groups": [
    "20013904637",
    "20013904699"
  ],
  "type": "user",
  "enterprise": [
    {
      "type": "enterprise",
      "id": "1162568706",
      "name": "testing"
    }
  ],
  "max_upload_size": "5.36870912E10",
  "__NAME__": "Miguel Benitez",
  "space_amount": "9.99999999999999E14",
  "language": "en",
  "is_external_collab_restricted": "false",
  "address": "Puerto La Cruz, P.R",
  "can_see_managed_users": "true",
  "job_title": "Web Developer",
  "is_exempt_from_device_limits": "false",
  "status": "active",
  "role": "user",
  "phone": "1157199024",
  "login": "someone@example.com"
}
```

> **Note**
>
> The response returns the user object before deletion.

GROUPS

Create a Box group

This example creates a Box group:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request POST \
--data '{
  "type": "group",
  "description": "Support Group - as imported from Active Directory",
  "external_sync_identifier": "AD:123456",
  "group_type": "managed_group",
  "invitability_level": "admins_only",
  "member_viewability_level": "admins_only",
  "__NAME__": "Support",
  "permissions": {
    "can_invite_as_collaborator": true
  },
  "provenance": "Active Directory"
}' \
'http://localhost:8080/openidm/system/Box/__GROUP__'
{
  "_id": "20147818911",
  "provenance": "Active Directory",
  "description": "Support Group - as imported from Active Directory",
  "group_type": "managed_group",
  "invitability_level": "admins_only",
  "member_viewability_level": "admins_only",
  "__NAME__": "Support",
  "type": "group",
  "external_sync_identifier": "AD:123456"
}
```

Query all Box groups

This example queries all Box groups:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request GET \
'http://localhost:8080/openidm/system/Box/__GROUP__?_queryFilter=True'
{
  "result": [
    {
      "_id": "20005069402",
      "provenance": null,
      "description": "generic_description",
      "group_type": "managed_group",
      "invitability_level": "all_managed_users",
      "member_viewability_level": "all_managed_users",
      "__NAME__": "A_20240611161336713",
      "type": "group",
      "external_sync_identifier": null
    },
    ...
    {
      "_id": "20147818911",
      "provenance": "Active Directory",
      "description": "Support Group - as imported from Active Directory",
      "group_type": "managed_group",
      "invitability_level": "admins_only",
      "member_viewability_level": "admins_only",
      "__NAME__": "Support",
      "type": "group",
      "external_sync_identifier": "AD:123456"
    }
  ],
  "resultCount": 22,
  "pagedResultsCookie": null,
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

Get a Box group

This example gets a Box group by its ID:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request GET \
'http://localhost:8080/openidm/system/Box/__GROUP__/20147818911'
{
  "_id": "20147818911",
  "provenance": "Active Directory",
  "description": "Support Group - as imported from Active Directory",
  "group_type": "managed_group",
  "invitability_level": "admins_only",
  "member_viewability_level": "admins_only",
  "__NAME__": "Support",
  "type": "group",
  "external_sync_identifier": "AD:123456"
}
```

Update a Box group

This example updates a Box group by its ID:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--header "If-Match: *" \
--request PUT \
--data '{
  "type": "group",
  "description": "Support Group - as imported from Active Directory",
  "external_sync_identifier": "AD:123456",
  "group_type": "managed_group",
  "invitability_level": "admins_only",
  "member_viewability_level": "admins_only",
  "name": "Support",
  "permissions": {
    "can_invite_as_collaborator": true
  },
  "provenance": "Active Directory"
}' \
'http://localhost:8080/openidm/system/Box/__GROUP__/20147818911'
{
  "_id": "20147818911",
  "provenance": "Active Directory",
  "description": "Support Group - as imported from Active Directory",
  "group_type": "managed_group",
  "invitability_level": "admins_only",
  "member_viewability_level": "admins_only",
  "__NAME__": "Support",
  "type": "group",
  "external_sync_identifier": "AD:123456"
}
```

Note

Other fields you can update are:

- `invitability_level` : Specifies who can invite the group to collaborate on folders. Available values:
  - `admins_only`
  - `admins_and_members`
  - `all_managed_users`
- `member_viewability_level` : Specifies who can see the members of the group. Available values:
  - `admins_only`
  - `admins_and_members`
  - `all_managed_users`
- `provenance` : Max length 255

Delete a Box group

This example deletes a Box group by its ID:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--header "If-Match: *" \
--request DELETE \
'http://localhost:8080/openidm/system/Box/__GROUP__/20147818911'
{
  "_id": "20147818911",
  "provenance": "Active Directory",
  "description": "Support Group - as imported from Active Directory",
  "group_type": "managed_group",
  "invitability_level": "admins_only",
  "member_viewability_level": "admins_only",
  "__NAME__": "Support",
  "type": "group",
  "external_sync_identifier": "AD:123456"
}
```

Note

The response returns the group object before deletion.

Mapping

Attributes Grid: Where the columns represent the attribute name mapped from source to target and the necessary data transformation to synchronize successfully.

SOURCE	TARGET	TRANSFORMATION SCRIPT
`id`	`userId`	N/A
`__NAME__`	`UserName`	N/A
`enterprise`	`enterprise`	N/A
`external_app_user_id`	`external_app_user_id`	N/A
`login`	`mail`	N/A
`type`	`type`	N/A
`address`	`address`	N/A
`avatar_url`	`avatar_url`	N/A
`can_see_managed_users`	`can_see_managed_users`	N/A
`hostname`	`hostname`	N/A

SOURCE	TARGET	TRANSFORMATION SCRIPT
`is_exempt_from_device_limits`	`is_exempt_from_device_limits`	N/A
`is_exempt_from_login_verification`	`is_exempt_from_login_verification`	N/A
`job_title`	`job_title`	N/A
`phone`	`phone`	N/A
`space_amount`	`space_amounts`	N/A
`max_upload_size`	`max_upload_size`	N/A
`language`	`language`	N/A
`status`	`status`	N/A
`memberof`	`memberof`	N/A
`is_sync_enabled`	`is_sync_enabled`	N/A
`is_external_collab_restricted`	`is_external_collab_restricted`	N/A
`is_platform_access_only`	`is_platform_access_only`	N/A
`role`	`role`	N/A

Attributes Grid: Where the columns represent the attribute name mapped from source to target and the necessary data transformation to synchronize successfully.

SOURCE	TARGET	TRANSFORMATION SCRIPT
`userId`	`id`	N/A
`UserName`	`__NAME__`	N/A
`enterprise`	`enterprise`	N/A
`external_app_user_id`	`external_app_user_id`	N/A
`mail`	`login`	N/A
`type`	`type`	N/A
`address`	`address`	N/A
`avatar_url`	`avatar_url`	N/A

`can_see_managed_users`	`can_see_managed_users`	N/A
`hostname`	`hostname`	N/A
`is_exempt_from_device_limits`	`is_exempt_from_device_limits`	N/A
`is_exempt_from_login_verification`	`is_exempt_from_login_verification`	N/A
`job_title`	`job_title`	N/A
`phone`	`phone`	N/A
`space_amount`	`space_amounts`	N/A
`max_upload_size`	`max_upload_size`	N/A
`language`	`language`	N/A
`status`	`status`	N/A
`memberof`	`memberof`	N/A
`is_sync_enabled`	`is_sync_enabled`	N/A
`is_external_collab_restricted`	`is_external_collab_restricted`	N/A
`is_platform_access_only`	`is_platform_access_only`	N/A
`role`	`role`	N/A

Attributes Grid: Where the columns represent the attribute name mapped from source to target and the necessary data transformation to synchronize successfully.

SOURCE	TARGET	TRANSFORMATION SCRIPT
`_id`	`_id`	N/A
`__NAME__`	`groupName`	N/A
`group_type`	`group_type`	N/A
`invitability_level`	`invitability_level`	N/A
`permissions`	`can_invite_as_collaborator`	`source.can_invite_as_collaborator`
`external_sync_identifier`	`external_sync_identifier`	N/A
`provenance`	`provenance`	N/A

SOURCE	TARGET	TRANSFORMATION SCRIPT
`description`	`description`	N/A
`member_viewability_level`	`member_viewability_level`	N/A
`type`	`type`	N/A

Attributes Grid: Where the columns represent the attribute name mapped from source to target and the necessary data transformation to synchronize successfully.

SOURCE	TARGET	TRANSFORMATION SCRIPT
`groupName`	`__NAME__`	N/A
`group_type`	`group_type`	N/A
`_id`	`_id`	N/A
`invitability_level`	`invitability_level`	N/A
`can_invite_as_collaborator`	`permissions`	`source.can_invite_as_collaborator`
`external_sync_identifier`	`external_sync_identifier`	N/A
`provenance`	`provenance`	N/A
`description`	`description`	N/A
`member_viewability_level`	`member_viewability_level`	N/A
`type`	`type`	N/A

OpenICF Interfaces Implemented by the Box.com Connector

The Box.com Connector implements the following OpenICF interfaces. For additional details, see ICF 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.

Box.com Connector Configuration

The Box.com Connector has the following configurable properties:

Basic Configuration Properties

Property	Type	Default	Encrypted[1]	Required[2]
`serviceUri`	`String`	`null`		✓ Yes
The service endpoint URI.				
`boxSubjectType`	`String`	`null`		✓ Yes
Description is not available				

Property	Type	Default	Encrypted[1]	Required[2]
`login`	`String`	`null`		✓ Yes
The service login name.				
`boxSubjectId`	`String`	`null`		✓ Yes
Description is not available				
`password`	`GuardedString`	`null`	🔒 Yes	✗ No
The service user password.				
`rateLimit`	`String`	`null`		✓ Yes
Description is not available				
`authenticationMethod`	`String`	`OAUTH`		✓ Yes
Defines which method is to be used to authenticate on the remote server. Options are BASIC (username/password), OAUTH (Client id/secret) or TOKEN (static token).				
`tokenEndpoint`	`String`	`null`		✗ No
When using OAUTH as authentication method, this property defines the endpoint where a new access token should be queried for (https://myserver.com/oauth2/token).				
`clientId`	`String`	`null`		✓ Yes
Client Id of the application registered at Box.com.				
`clientSecret`	`GuardedString`	`null`	🔒 Yes	✗ No
Client Secret of the application registered at Box.com.				
`authToken`	`GuardedString`	`null`	🔒 Yes	✗ No
Static authentication token.				
`acceptSelfSignedCertificates`	`boolean`	`false`		✓ Yes
To be used for debug/test purposes. To be avoided in production.				
`disableHostNameVerifier`	`boolean`	`false`		✓ Yes
To be used for debug/test purposes. To be avoided in production.				
`disableHttpCompression`	`boolean`	`false`		✓ Yes

Property	Type	Default	Encrypted[1]	Required[2]
Content compression is enabled by default. Set this property to true to disable it.				
`clientCertAlias`	`String`	`null`		✓ Yes
If TLS Mutual Auth is needed, set this to the certificate alias from the keystore.				
`clientCertPassword`	`GuardedString`	`null`	🔒 Yes	✓ Yes
If TLS Mutual Auth is needed and the client certificate (private key) password is different from the keystore password, set this to the client private key password.				
`maximumConnections`	`Integer`	`10`		✓ Yes
Defines the max size of the HTTP connection pool used.				
`httpProxyHost`	`String`	`null`		✓ Yes
Defines the Hostname if an HTTP proxy is used between the connector and the service.				
`httpProxyPort`	`Integer`	`null`		✓ Yes
Defines the Port if an HTTP proxy is used between the connector and the service.				
`httpProxyUsername`	`String`	`null`		✓ Yes
Defines Proxy Username if an HTTP proxy is used between the connector and the service.				
`httpProxyPassword`	`GuardedString`	`null`	🔒 Yes	✓ Yes
Defines Proxy Password if an HTTP proxy is used between the connector and the service.				
`connectionTimeout`	`int`	`30`		✗ No
Defines a timeout for the underlying HTTP connection in seconds.				
`refreshToken`	`GuardedString`	`null`		✗ No
A refresh token retrieved in the final leg of OAuth 2. In most cases these are valid for 60 days, or until used.				
`grantType`	`String`	`null`		✗ No
The OAuth2 grant type to use (client_credentials, refresh_token, or jwt_bearer).				
`scope`	`String`	`null`		✗ No
The OAuth2 scope to use.				
`authorizationTokenPrefix`	`String`	`Bearer`		✗ No

Property	Type	Default	Encrypted[1]	Required[2]
The prefix to be used in the Authorization HTTP header for Token authentication.				
`useBasicAuthForOauthTokenNeg`	`boolean`	`true`		✓ Yes
The Authentication method for refresh token (Basic Authentication or Sending the ClientId and Client Secret in the Header).				
`jwtKey`	`String`	`null`		✕ No
The JWT data structure that represents a cryptographic key.				
`jwtExpiration`	`Integer`	`null`		✕ No
Defines the JWT expiration time in seconds.				
`jwtAlgorithm`	`String`	`null`		✕ No
The Algorithm type to sign payload.				
`jwtClaims`	`Map`	`null`		✕ No
JWT Claims to be included in the payload				
`jwtPem`	`String`	`null`		✕ No
The contents of the private key of the PEM file				
`jwtCert`	`String`	`null`		✕ No
The contents of the certificate of the PEM file				
`keyAlgorithm`	`String`	`null`		✕ No
Indicates the type of key (such as RSA, DSA or EC) used to sign from the PEM.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

Cerner connector

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

Before you start

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

Bearer token

The bearer token associated with your system account.

Tenant

Your Cerner tenant ID.

Region

The Cerner Cloud region where the tenant resides.

Install the Cerner connector

> **Tip**
>
> To check for an Advanced Identity Cloud application for this connector, refer to:
>
> - Application management
> - App catalog

You can download any connector from Backstage, but some are included in the default deployment for Advanced Identity Cloud, IDM, or RCS. When using an included connector, you can skip installing it and move directly to configuration.

Connector included in default deployment

Connector	IDM	RCS
Cerner	✗ No	✓ Yes

Download the connector .jar file from Backstage.

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

```
mv ~/Downloads/cerner-connector-1.5.20.26.jar /path/to/openidm/connectors/
```

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

Configure the Cerner connector

Create a connector configuration using the IDM admin UI:

1. From the navigation bar, click **Configure > Connectors**.

2. On the **Connectors** page, click **New Connector**.
3. On the **New Connector** page, type a **Connector Name**.
4. From the **Connector Type** drop-down list, select **Cerner Connector - 1.5.20.26**.
5. Complete the **Base Connector Details**.

> **Tip**
>
> For a list of all configuration properties, refer to Cerner Connector Configuration

6. Click **Save**.

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

Refer to this procedure to create a connector configuration over REST.

Test the Cerner connector

Test that the configuration is correct by running the following command:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system/Cerner?_action=test"
{
  "name": "Cerner",
  "enabled": true,
  "config": "config/provisioner.openicf/Cerner",
  "connectorRef": {
    "bundleVersion": "[1.5.0.0,1.6.0.0)",
    "bundleName": "org.forgerock.openicf.connectors.cerner-connector",
    "connectorName": "org.forgerock.openicf.connectors.cerner.CernerConnector"
  },
  "displayName": "Cerner Connector",
  "objectTypes": [
    "__ORGANIZATION__",
    "__ACCOUNT__",
    "__ORGANIZATIONGROUP__",
    "__ALL__",
    "__PERSONNELGROUP__"
  ],
  "ok": true
}
```

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

Cerner remote connector

If you want to run this connector outside of PingOne Advanced Identity Cloud or IDM, you can configure the Cerner connector as a remote connector. Java Connectors installed remotely on a Java Connector Server function identically to those bundled locally within PingOne Advanced Identity Cloud or installed locally on IDM.

You can download the Cerner connector from here.

Refer to Remote connectors for configuring the Cerner remote connector.

Configure connection pooling

The Cerner connector supports HTTP pooling, which can substantially improve the performance of the connector. Learn more about the basic connection pooling configuration and different pooling mechanisms described in Connection pooling configuration.

Use the Cerner connector

Supported object types

Connector resource	Cerner resource type
`__ACCOUNT__`	Personnel
`__ORGANIZATION__`	Organization
`__PERSONNELGROUP__`	Personnel Group
`__ORGANIZATIONGROUP__`	Organization Group

Attribute	Notes
`__NAME__`	The user's name, in a `FAMILY, GIVEN` format. Required.
`birthDate`	Must be in `YYYY-MM-DD` format.
`gender`	Accepted values are `MALE`, `FEMALE`, `OTHER`, `UNKNOWN`.
`given`	The user's first name. Required.
`family`	The user's last name. Required.

<table>
<tr><th>Attribute</th><th>Notes</th></tr>
<tr><td><code>name</code></td><td><table>
<tr><td><code>given</code></td><td></td></tr>
<tr><td><code>middle</code></td><td></td></tr>
<tr><td><code>family</code></td><td></td></tr>
<tr><td><code>suffix</code></td><td></td></tr>
<tr><td><code>prefix</code></td><td></td></tr>
</table></td></tr>
<tr><td><code>addresses</code></td><td><table>
<tr><td><code>postalCode</code></td><td></td></tr>
<tr><td><code>country</code></td><td></td></tr>
<tr><td><code>use</code></td><td>Accepted values are <code>HOME</code> or <code>WORK</code>.</td></tr>
<tr><td><code>city</code></td><td></td></tr>
<tr><td><code>state</code></td><td></td></tr>
<tr><td><code>lines</code></td><td>The street portion of the address.</td></tr>
</table></td></tr>
<tr><td><code>aliasType</code></td><td>Accepted values are: <code>SPI</code>, <code>TAX</code>, <code>SL</code>, <code>EXTERNAL</code>, <code>UPIN</code>, <code>USER</code>, or <code>UNKNOWN</code>. Required.</td></tr>
<tr><td><code>aliasValue</code></td><td></td></tr>
<tr><td><code>aliasSystem</code></td><td></td></tr>
<tr><td><code>sourceIdentifiers</code></td><td><table>
<tr><td><code>id</code></td><td></td></tr>
<tr><td><code>dataPartitionId</code></td><td></td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Attribute</th><th>Notes</th></tr>
<tr><td><code>qualifications</code></td><td>
<table>
<tr><td><code>issuer</code></td><td></td></tr>
<tr><td><code>code</code></td><td>Qualification code such as <code>MD</code> or <code>PhD</code>.
Accepted values are: <code>AA</code>, <code>AAS</code>, <code>ABA</code>, <code>AE</code>, <code>AS</code>, <code>BA</code>, <code>BBA</code>, <code>BE</code>, <code>BFA</code>, <code>BN</code>, <code>BS</code>, <code>BSL</code>, <code>BSN</code>, <code>BT</code>, <code>CANP</code>, <code>CER</code>, <code>CMA</code>, <code>CNM</code>, <code>CNP</code>, <code>CNS</code>, <code>CPNP</code>, <code>CRN</code>, <code>CTR</code>, <code>DBA</code>, <code>DED</code>, <code>DIP</code>, <code>DO</code>, <code>EMT</code>, <code>EMTP</code>, <code>FPNP</code>, <code>HS</code>, <code>JD</code>, <code>MA</code>, <code>MBA</code>, <code>MCE</code>, <code>MD</code>, <code>MDA</code>, <code>MDI</code>, <code>ME</code>, <code>MED</code>, <code>MEE</code>, <code>MFA</code>, <code>MME</code>, <code>MS</code>, <code>MSL</code>, <code>MSN</code>, <code>MT</code>, <code>MTH</code>, <code>NG</code>, <code>NP</code>, <code>PA</code>, <code>PHD</code>, <code>PHE</code>, <code>PNS</code>, <code>PN</code>, <code>PharmD</code>, <code>RMA</code>, <code>RN</code>, <code>RPH</code>, <code>SEC</code>, or <code>TS</code>.</td></tr>
<tr><td><code>start</code></td><td>The first date and time that the qualification is valid, in a <code>YYYY-MM-DDThh:mm:ssZ</code> date format.</td></tr>
<tr><td><code>end</code></td><td>The date and time that the qualification expires, in a <code>YYYY-MM-DDThh:mm:ssZ</code> date format.</td></tr>
</table>
</td></tr>
<tr><td><code>telecoms</code></td><td>
<table>
<tr><td><code>system</code></td><td>Accepted values are <code>PHONE</code>, <code>EMAIL</code>, or <code>OTHER</code>.</td></tr>
<tr><td><code>value</code></td><td></td></tr>
</table>
</td></tr>
<tr><td><code>languages</code></td><td>For a list of valid language tags, refer to the <i>Internet Assigned Numbers Authority</i> (IANA) language subtag registry.</td></tr>
</table>

<table>
<tr><th>Attribute</th><th>Notes</th></tr>
<tr><td><code>__NAME__</code></td><td>The name of the organization. This corresponds to <code>aliasValue</code>, <code>aliasSystem</code>, comma separated. Required.</td></tr>
<tr><td><code>name</code></td><td>The name of the organization. Required.</td></tr>
<tr><td><code>aliasType</code></td><td>Alias types related to the organization. <code>DEA</code>, <code>TAX</code>, <code>SOI</code>, and <code>NPI</code> are supported for queries. Organizations with <code>NPI</code> and <code>DEA</code> cannot be created or updated.</td></tr>
<tr><td><code>telecoms</code></td><td>
<table>
<tr><td><code>system</code></td><td>Accepted values are <code>PHONE</code>, <code>EMAIL</code>, or <code>OTHER</code>.</td></tr>
<tr><td><code>value</code></td><td></td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Attribute</th><th>Notes</th></tr>
<tr><td><code>addresses</code></td><td><table>
<tr><td><code>postalCode</code></td><td></td></tr>
<tr><td><code>country</code></td><td></td></tr>
<tr><td><code>text</code></td><td>Formatted display text of the address.</td></tr>
<tr><td><code>city</code></td><td></td></tr>
<tr><td><code>state</code></td><td></td></tr>
<tr><td><code>lines</code></td><td>The street portion of the address.</td></tr>
</table></td></tr>
<tr><td><code>aliases</code></td><td><table>
<tr><td><code>type</code></td><td>Types of alias for the organization.</td></tr>
<tr><td><code>system</code></td><td></td></tr>
<tr><td><code>value</code></td><td></td></tr>
</table></td></tr>
<tr><td><code>languages</code></td><td>For a list of valid language tags, refer to the <i>Internet Assigned Numbers Authority</i> (IANA) language subtag registry.</td></tr>
<tr><td><code>coverageAreaPostalCodes</code></td><td>The postal codes indicating the area of coverage provided by the organization.</td></tr>
<tr><td><code>sourceIdentifiers</code></td><td><table>
<tr><td><code>id</code></td><td></td></tr>
<tr><td><code>dataPartitionId</code></td><td></td></tr>
</table></td></tr>
</table>

Attribute	Notes
`__NAME__`	A comma-separated name for the personnel group.
`mnemonic`	The mnemonic determines the function of the personnel group.
`mnemonicType`	The type of the personnel group mnemonic. Usually either `SINGLETON` or `MULTIVALUED`.
`name`	The name of the personnel group.

Attribute	Notes

`aliases`	`type`
	`system`
	`value`

`aliasType`	The type of alias. Requires `aliasValue` and `aliasSystem`.
`aliasSystem`	The source of the alias value. Requires `aliasType` and `aliasValue`.
`aliasValue`	The unique identifier of alias. Requires `aliasType` and `aliasSystem`.

Attribute	Notes
`__NAME__`	A comma-separated name for the organization group.
`organizationId`	A list of organization IDs that are members of the organization group.
`name`	The name of the organization group.

`aliases`	`type`
	`system`
	`value`

`aliasType`	The type of alias. Requires `aliasValue` and `aliasSystem`.
`aliasSystem`	The source of the alias value. Requires `aliasType` and `aliasValue`.
`aliasValue`	The unique identifier of alias. Requires `aliasType` and `aliasSystem`.

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

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

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

Note

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

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

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

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

The following example queries all Cerner users:

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

Note

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

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

The following command queries a specific user by their ID:

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

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

The following example deletes a Cerner account:

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

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

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "If-Match:*" \
--request PUT \
--data '{
  "given": "Barbara",
  "family": "Jensen",
  "aliasType": "USER",
  "__NAME__": "Jensen, Barbara",
  "name": {
    "middle": "Simone"
  },
  "personnelGroupId": [
        "8636d4c3-de7c-4f8a-828b-b709d6bfd636"
  ]
}' \
"http://localhost:8080/openidm/system/Cerner/__ACCOUNT__/5170a9cd-e501-4cbf-a1bf-9e6d293362c6"
{
  "_id": "5170a9cd-e501-4cbf-a1bf-9e6d293362c6",
  "formattedName": "Barbara Simone Jensen",
  "__NAME__": "Jensen,Barbara",
  "aliasValue": "Jensen",
  "family": "Jensen",
  "updatedAt": "2022-10-25T23:50:31Z",
  "aliasType": "USER",
  "given": "Barbara",
  "organizationId": [],
  "aliasSystem": "Barbara",
  "name": {
    "given": "Barbara",
    "middle": "Simone",
    "family": "Jensen",
    "formatted": "Barbara Simone Jensen"
  },
  "languages": [],
  "id": "5170a9cd-e501-4cbf-a1bf-9e6d293362c6",
  "isManual": true,
  "personnelGroupId": [
    "8636d4c3-de7c-4f8a-828b-b709d6bfd636"
  ],
  "aliases": {
    "type": "USER",
    "value": "Jensen",
    "system": "Barbara"
  },
  "createdAt": "2022-04-29T22:54:08Z"
}
```

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

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

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

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

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

OpenICF Interfaces Implemented by the Cerner Connector

The Cerner Connector implements the following OpenICF interfaces. For additional details, see ICF interfaces:

Create

Creates an object and its `uid`.

Delete

Deletes an object, referenced by its `uid`.

Schema

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

Script on Connector

Enables an application to run a script in the context of the connector.

Any script that runs on the connector has the following characteristics:

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

Search

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

Test

Tests the connector configuration.

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

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

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

Update

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

Cerner Connector Configuration

The Cerner Connector has the following configurable properties:

Configuration properties

Property	Type	Default	Encrypted[(1)]	Required[(2)]
`bearerToken`	`GuardedString`	`null`	🔒 Yes	✓ Yes
Provides the bearer token to authorize Cerner.				
`tenant`	`String`	`playground`		✕ No
Provides the tenant to authorize Cerner.				
`region`	`String`	`us-1`		✕ No
Provides the region to authorize Cerner.				

Property	Type	Default	Encrypted(1)	Required(2)
`maximumConnections`	`Integer`	`10`		✕ No
Provides the maximum connections.				
`connectionTimeout`	`Integer`	`300`		✕ No
Provides the maximum connection timeout in seconds.				
`httpProxyHost`	`String`	`null`		✓ Yes
Provides the Proxy Host.				
`httpProxyPort`	`Integer`	`null`		✓ Yes
Provides the Proxy Port.				
`httpProxyUsername`	`String`	`null`		✓ Yes
Provides the Proxy Username.				
`httpProxyPassword`	`GuardedString`	`null`	🔒 Yes	✓ Yes
Provides the Proxy Password.				

(1) Whether the property value is considered confidential, and is therefore encrypted in IDM.

(2) A list of operations in this column indicates that the property is required for those operations.

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.

> ⚠ **Warning**
>
> This connector does not verify CSV data before attempting a synchronization. You must ensure that your CSV file is complete and properly formed before using the connector.
> Do *not* remove or replace CSV files that are the source or target of an active scheduled reconciliation or synchronization operation.

Install the CSV file connector

> **Tip**
>
> To check for an Advanced Identity Cloud application for this connector, refer to:
>
> - Application management
> - App catalog

You can download any connector from Backstage, but some are included in the default deployment for Advanced Identity Cloud, IDM, or RCS. When using an included connector, you can skip installing it and move directly to configuration.

Connector included in default deployment

Connector	IDM	RCS
CSV file	✓ Yes	✓ Yes

Download the connector .jar file from Backstage.

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

```
mv ~/Downloads/csvfile-connector-1.5.20.30.jar /path/to/openidm/connectors/
```

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

Configure the CSV file connector

Create a connector configuration using the IDM admin UI:

1. From the navigation bar, click **Configure** > **Connectors**.
2. On the **Connectors** page, click **New Connector**.
3. On the **New Connector** page, type a **Connector Name**.
4. From the **Connector Type** drop-down list, select **CSV file Connector - 1.5.20.30**.
5. Complete the **Base Connector Details**.

   > **Tip**
   >
   > For a list of all configuration properties, refer to CSV file Connector Configuration

6. Click **Save**.

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

Refer to this procedure to create a connector configuration over REST.

Alternatively, use the sample CSV file connector configuration in `openidm/samples/example-configurations/provisioners/provisioner.openicf-csvfile.json` as a basis for your configuration.

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

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

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

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

For a list of all configuration properties for this connector, refer to Configuration Properties.

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

CSV file remote connector

If you want to run this connector outside of PingOne Advanced Identity Cloud or IDM, you can configure the CSV file connector as a remote connector. Java Connectors installed remotely on a Java Connector Server function identically to those bundled locally within PingOne Advanced Identity Cloud or installed locally on IDM.

You can download the CSV file connector from here.

Refer to Remote connectors for configuring the CSV file remote connector.

Configure connection pooling

The CSV file connector uses a non-poolable mechanism to manage connections. Learn more about the different pooling mechanisms in Connectors by pooling mechanism.

OpenICF Interfaces Implemented by the CSV File Connector

The CSV File Connector implements the following OpenICF interfaces. For additional details, see ICF interfaces:

Authenticate

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

Batch

Execute a series of operations in a single request.

Create

Creates an object and its `uid`.

Delete

Deletes an object, referenced by its `uid`.

Resolve Username

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

Schema

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

Script on Connector

Enables an application to run a script in the context of the connector.

Any script that runs on the connector has the following characteristics:

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

Search

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

Sync

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

Test

Tests the connector configuration.

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

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

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

Update

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

CSV File Connector Configuration

The CSV File Connector has the following configurable properties:

Configuration properties

Property	Type	Default	Encrypted[1]	Required[2]
`headerPassword`	`String`	`password`		✕ No
The CSV header that maps to the password for each row. Use this property when password-based authentication is required.				
`spaceReplacementString`	`String`	`_`		✕ No
The character(s) used to replace spaces within column names.				
`csvFile`	`File`	`null`		✓ Yes
The full path to the CSV file that is the data source for this connector.				
`newlineString`	`String`	`\n`		✕ No
The character string in the CSV file that is used to terminate each line.				
`headerUid`	`String`	`uid`		✕ No
The CSV header that maps to the uid (or name) for each row.				
`quoteCharacter`	`String`	`"`		✕ No
The character in the CSV file that is used to encapsulate strings.				
`escapeCharacter`	`String`	`\`		✕ No
The character in the CSV file that is used to escape characters.				
`fieldDelimiter`	`String`	`,`		✕ No
The character in the CSV file that is used to separate field values.				
`syncFileRetentionCount`	`int`	`3`		✕ No
The number of historical copies of the CSV file to retain when performing synchronization operations.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

Database Table connector

The Database Table connector lets you provision to a single table in a JDBC database.

Install the Database Table connector

Tip

To check for an Advanced Identity Cloud application for this connector, refer to:

- Application management
- App catalog

You can download any connector from Backstage, but some are included in the default deployment for Advanced Identity Cloud, IDM, or RCS. When using an included connector, you can skip installing it and move directly to configuration.

Connector included in default deployment

Connector	IDM	RCS
Database table	✓ Yes	✓ Yes

Download the connector .jar file from Backstage.

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

```
mv ~/Downloads/databasetable-connector-1.5.20.30.jar /path/to/openidm/connectors/
```

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

Configure the Database Table connector

Create a connector configuration using the IDM admin UI:

1. From the navigation bar, click **Configure > Connectors**.
2. On the **Connectors** page, click **New Connector**.
3. On the **New Connector** page, type a **Connector Name**.
4. From the **Connector Type** drop-down list, select **Database Table Connector - 1.5.20.30**.
5. Complete the **Base Connector Details**.

> **Tip**
>
> For a list of all configuration properties, refer to Database Table Connector Configuration

6. Click **Save**.

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

Refer to this procedure to create a connector configuration over REST.

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

The following excerpt shows a sample Database Table connector configuration:

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

The mandatory configurable properties are as follows:

`url`

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

`driverClassName`

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

`table`

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

`keyColumn`

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

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

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

Database Table remote connector

If you want to run this connector outside of PingOne Advanced Identity Cloud or IDM, you can configure the Database Table connector as a remote connector. Java Connectors installed remotely on a Java Connector Server function identically to those bundled locally within PingOne Advanced Identity Cloud or installed locally on IDM.

You can download the Database Table connector from here.

Refer to Remote connectors for configuring the Database Table remote connector.

Configure connection pooling

The Database Table connector embeds the Apache Tomcat 9 JDBC Connection Pool. Learn more about the different pooling mechanisms in Connectors by pooling mechanism.

Implementation specifics

- To use this connector for liveSync, add a changelog type column to the database and provide the name of this column in the `changeLogColumn` property. Note that the Database Table connector supports liveSync for create and update operations only. To detect deletes in the database you must run a full reconciliation.
- For PATCH requests, a connector can potentially add, remove, or replace an attribute value. The Database Table connector does not implement the add or remove operations, so a PATCH request always replaces the entire attribute value with the new value.
- The Database Table connector supports paged reconciliation queries *only* for the following databases:
  - MySQL
  - PostgreSQL
  - Oracle Database 11gR2 and later versions

    - Microsoft SQL Server 2012 and later versions

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

  For more information about configuring paged reconciliation queries, refer to Paging Reconciliation Query Results.

- If your database does not support precise (nanosecond) timestamps, you can use the `inclusiveSync` configuration property to ensure that modified entries are not missed in liveSync operations. If `inclusiveSync` is set to `true`, the connector synchronizes all entries whose change timestamp is greater than or equal to the `syncToken`. Be aware that if you set this property to `true`, the activity log creates a new entry *every time* liveSync occurs, even if entries are changed. This can lead to rapid growth of the activity audit log.

OpenICF Interfaces Implemented by the Database Table Connector

The Database Table Connector implements the following OpenICF interfaces. For additional details, see ICF interfaces:

Authenticate

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

Create

Creates an object and its `uid`.

Delete

Deletes an object, referenced by its `uid`.

Resolve Username

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

Schema

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

Script on Connector

Enables an application to run a script in the context of the connector.

Any script that runs on the connector has the following characteristics:

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

Search

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

Sync

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

Test

Tests the connector configuration.

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

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

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

Update

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

Database Table Connector Configuration

The Database Table Connector has the following configurable properties:

Configuration properties

Property	Type	Default	Encrypted[1]	Required[2]
`connectionProperties`	`String`	`null`		✕ No
The connection properties that will be sent to our JDBC driver when establishing new connections. Format of the string must be [propertyName=property;]* NOTE - The "user" and "password" properties will be passed explicitly, so they do not need to be included here. The default value is null.				
`propagateInterruptState`	`boolean`	`false`		✕ No
Set this to true to propagate the interrupt state for a thread that has been interrupted (not clearing the interrupt state). Default value is false for backwards compatibility.				
`useDisposableConnectionFacade`	`boolean`	`true`		✕ No
Set this to true if you wish to put a facade on your connection so that it cannot be reused after it has been closed. This prevents a thread holding on to a reference of a connection it has already called closed on, to execute queries on it.				

Property	Type	Default	Encrypted[1]	Required[2]
`defaultCatalog`	`String`	`null`		✕ No
The default catalog of connections created by this pool.				
`validationInterval`	`long`	`3000`		✕ No
To avoid excess validation, run validation at most at this frequency (in milliseconds). If a connection is due for validation, but was validated within this interval, it will not be validated again. The default value is 3000 (3 seconds).				
`ignoreExceptionOnPreLoad`	`boolean`	`false`		✕ No
Flag whether ignore error of connection creation while initializing the pool. Set to true if you want to ignore error of connection creation while initializing the pool. Set to false if you want to fail the initialization of the pool by throwing exception.				
`jmxEnabled`	`boolean`	`true`		✕ No
Register the pool with JMX or not. The default value is true.				
`commitOnReturn`	`boolean`	`false`		✕ No
If autoCommit==false then the pool can complete the transaction by calling commit on the connection as it is returned to the pool If rollbackOnReturn==true then this attribute is ignored. Default value is false.				
`logAbandoned`	`boolean`	`false`		✕ No
Flag to log stack traces for application code which abandoned a Connection. Logging of abandoned Connections adds overhead for every Connection borrow because a stack trace has to be generated. The default value is false.				
`maxIdle`	`int`	`100`		✕ No
The maximum number of connections that should be kept in the pool at all times. Idle connections are checked periodically (if enabled) and connections that have been idle for longer than minEvictableIdleTimeMillis are released. The default value is derived from maxActive:100. (Also see testWhileIdle).				
`testWhileIdle`	`boolean`	`false`		✕ No
The indication of whether objects will be validated by the idle object evictor (if any). If an object fails to validate, it will be dropped from the pool. NOTE - for a true value to have any effect, the validationQuery parameter must be set to a non-null string. The default value is false and this property has to be set in order for the pool cleaner/test thread is to run (also see timeBetweenEvictionRunsMillis).				
`removeAbandoned`	`boolean`	`false`		✕ No
Flag to remove abandoned connections if they exceed the removeAbandonedTimeout. If set to true a connection is considered abandoned and eligible for removal if it has been in use longer than the removeAbandonedTimeout Setting this to true can recover db connections from applications that fail to close a connection. See also logAbandoned The default value is false.				

Property	Type	Default	Encrypted[1]	Required[2]
`abandonWhenPercentageFull`	`int`	`0`		✕ No
Connections that have been abandoned (timed out) wont get closed and reported up unless the number of connections in use are above the percentage defined by abandonWhenPercentageFull. The value should be between 0-100. The default value is 0, which implies that connections are eligible for closure as soon as removeAbandonedTimeout has been reached.				
`minIdle`	`int`	`10`		✕ No
The minimum number of established connections that should be kept in the pool at all times. The connection pool can shrink below this number if validation queries fail. The default value is derived from initialSize:10. (Also see testWhileIdle).				
`defaultReadOnly`	`Boolean`	`null`		✕ No
The default read-only state of connections created by this pool. If not set then the setReadOnly method will not be called. (Some drivers dont support read only mode, ex: Informix)				
`maxWait`	`int`	`30000`		✕ No
The maximum number of milliseconds that the pool will wait (when there are no available connections) for a connection to be returned before throwing an exception. Default value is 30000 (30 seconds).				
`logValidationErrors`	`boolean`	`false`		✕ No
Set this to true to log errors during the validation phase to the log file. If set to true, errors will be logged as SEVERE. Default value is false for backwards compatibility.				
`driverClassName`	`String`	`null`		✕ No
The fully qualified Java class name of the JDBC driver to be used. The driver has to be accessible from the same classloader as tomcat-jdbc.jar.				
`name`	`String`	`Tomcat Connection Pool[1-544298878]`		✕ No
Returns the name of the connection pool. By default a JVM unique random name is assigned.				
`useStatementFacade`	`boolean`	`true`		✕ No
Returns true if this connection pool is configured to wrap statements in order to enable equals() and hashCode() methods to be called on the closed statements if any statement proxy is set.				
`initSQL`	`String`	`null`		✕ No
A custom query to be run when a connection is first created. The default value is null.				
`validationQueryTimeout`	`int`	`-1`		✕ No

Property	Type	Default	Encrypted[1]	Required[2]
The timeout in seconds before a connection validation queries fail. This works by calling java.test_sample.Statement.setQueryTimeout(seconds) on the statement that executes the validationQuery. The pool itself doesnt timeout the query, it is still up to the JDBC driver to enforce query timeouts. A value less than or equal to zero will disable this feature. The default value is -1.				
`validationQuery`	`String`	`null`		✕ No
The SQL query that will be used to validate connections from this pool before returning them to the caller. If specified, this query does not have to return any data, it just cant throw a SQLException. The default value is null. Example values are SELECT 1(mysql), select 1 from dual(oracle), SELECT 1(MS Sql Server).				
`rollbackOnReturn`	`boolean`	`false`		✕ No
If autoCommit==false then the pool can terminate the transaction by calling rollback on the connection as it is returned to the pool Default value is false.				
`alternateUsernameAllowed`	`boolean`	`false`		✕ No
By default, the jdbc-pool will ignore the DataSource.getConnection(username,password) call, and simply return a previously pooled connection under the globally configured properties username and password, for performance reasons. The pool can however be configured to allow use of different credentials each time a connection is requested. To enable the functionality described in the DataSource.getConnection(username,password) call, simply set the property alternateUsernameAllowed to true. Should you request a connection with the credentials user1/password1 and the connection was previously connected using different user2/password2, the connection will be closed, and reopened with the requested credentials. This way, the pool size is still managed on a global level, and not on a per schema level.				
`validatorClassName`	`String`	`null`		✕ No
The name of a class which implements the org.apache.tomcat.jdbc.pool.Validator interface and provides a no-arg constructor (may be implicit). If specified, the class will be used to create a Validator instance which is then used instead of any validation query to validate connections. The default value is null. An example value is com.mycompany.project.SimpleValidator.				
`suspectTimeout`	`int`	`0`		✕ No
Timeout value in seconds. Similar to to the removeAbandonedTimeout value but instead of treating the connection as abandoned, and potentially closing the connection, this simply logs the warning if logAbandoned is set to true. If this value is equal or less than 0, no suspect checking will be performed. Suspect checking only takes place if the timeout value is larger than 0 and the connection was not abandoned or if abandon check is disabled. If a connection is suspect a WARN message gets logged and a JMX notification gets sent once.				
`useEquals`	`boolean`	`true`		✕ No
Set to true if you wish the ProxyConnection class to use String.equals and set to false when you wish to use == when comparing method names. This property does not apply to added interceptors as those are configured individually. The default value is true.				

Property	Type	Default	Encrypted[1]	Required[2]
`removeAbandonedTimeout`	`int`	`60`		✕ No
Timeout in seconds before an abandoned(in use) connection can be removed. The default value is 60 (60 seconds). The value should be set to the longest running query your applications might have.				
`defaultAutoCommit`	`Boolean`	`null`		✕ No
The default auto-commit state of connections created by this pool. If not set, default is JDBC driver default (If not set then the setAutoCommit method will not be called).				
`testOnConnect`	`boolean`	`false`		✕ No
Returns true if we should run the validation query when connecting to the database for the first time on a connection. Normally this is always set to false, unless one wants to use the validationQuery as an init query.				
`jdbcInterceptors`	`String`	`null`		✕ No
A semicolon separated list of classnames extending org.apache.tomcat.jdbc.pool.JdbcInterceptor class. See Configuring JDBC interceptors below for more detailed description of syntaz and examples. These interceptors will be inserted as an interceptor into the chain of operations on a java.test_sample.Connection object. The default value is null.				
`initialSize`	`int`	`10`		✕ No
The initial number of connections that are created when the pool is started. Default value is 10.				
`defaultTransactionIsolation`	`int`	`-1`		✕ No
The default TransactionIsolation state of connections created by this pool. One of the following: NONE, READ_COMMITTED, READ_UNCOMMITTED, REPEATABLE_READ, SERIALIZABLE If not set, the method will not be called and it defaults to the JDBC driver.				
`numTestsPerEvictionRun`	`int`	`0`		✕ No
Property not used in tomcat-jdbc-pool.				
`url`	`String`	`null`		✕ No
The URL used to connect to the database.				
`testOnBorrow`	`boolean`	`false`		✕ No
The indication of whether objects will be validated before being borrowed from the pool. If the object fails to validate, it will be dropped from the pool, and we will attempt to borrow another. NOTE - for a true value to have any effect, the validationQuery parameter must be set to a non-null string. In order to have a more efficient validation, see validationInterval. Default value is false.				
`fairQueue`	`boolean`	`true`		✕ No

Property	Type	Default	Encrypted[1]	Required[2]
Set to true if you wish that calls to getConnection should be treated fairly in a true FIFO fashion. This uses the org.apache.tomcat.jdbc.pool.FairBlockingQueue implementation for the list of the idle connections. The default value is true. This flag is required when you want to use asynchronous connection retrieval. Setting this flag ensures that threads receive connections in the order they arrive. During performance tests, there is a very large difference in how locks and lock waiting is implemented. When fairQueue=true there is a decision making process based on what operating system the system is running. If the system is running on Linux (property os.name=Linux. To disable this Linux specific behavior and still use the fair queue, simply add the property org.apache.tomcat.jdbc.pool.FairBlockingQueue.ignoreOS=true to your system properties before the connection pool classes are loaded.				
`accessToUnderlyingConnectionAllowed`	`boolean`	`true`		✕ No
Property not used. Access can be achieved by calling unwrap on the pooled connection. see javax.test_sample.DataSource interface, or call getConnection through reflection or cast the object as javax.test_sample.PooledConnection				
`maxAge`	`long`	`0`		✕ No
Time in milliseconds to keep this connection. When a connection is returned to the pool, the pool will check to see if the now - time-when-connected > maxAge has been reached, and if so, it closes the connection rather than returning it to the pool. The default value is 0, which implies that connections will be left open and no age check will be done upon returning the connection to the pool.				
`minEvictableIdleTimeMillis`	`int`	`60000`		✕ No
The minimum amount of time an object may sit idle in the pool before it is eligible for eviction. The default value is 60000 (60 seconds).				
`timeBetweenEvictionRunsMillis`	`int`	`5000`		✕ No
The number of milliseconds to sleep between runs of the idle connection validation/cleaner thread. This value should not be set under 1 second. It dictates how often we check for idle, abandoned connections, and how often we validate idle connections. The default value is 5000 (5 seconds).				
`testOnReturn`	`boolean`	`false`		✕ No
The indication of whether objects will be validated before being returned to the pool. NOTE - for a true value to have any effect, the validationQuery parameter must be set to a non-null string. The default value is false.				
`useLock`	`boolean`	`false`		✕ No
Return true if a lock should be used when operations are performed on the connection object. Should be set to false unless you plan to have a background thread of your own doing idle and abandon checking such as JMX clients. If the pool sweeper is enabled, then the lock will automatically be used regardless of this setting.				
`maxActive`	`int`	`100`		✕ No
The maximum number of active connections that can be allocated from this pool at the same time. The default value is 100.				

Property	Type	Default	Encrypted[1]	Required[2]
`username`	`String`	`null`		✕ No
The connection username to be passed to our JDBC driver to establish a connection. Note that method DataSource.getConnection(username,password) by default will not use credentials passed into the method, but will use the ones configured here. See alternateUsernameAllowed property for more details.				
`table`	`String`	`TABLE_NAME`		✓ Yes
Enter the name of the table in the database that contains the accounts.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

Basic Configuration Properties

Property	Type	Default	Encrypted[1]	Required[2]
`password`	`String`	`null`	🔒 Yes	✓ Yes
The connection password to be passed to the JDBC driver to establish a connection. Note that method DataSource.getConnection(username,password) by default will not use credentials passed into the method, but will use the ones configured here. See alternateUsernameAllowed property for more details.				
`quoting`	`String`	`NONE`		✕ No
Select whether database column names for this resource should be quoted, and the quoting characters. By default, database column names are not quoted (None). For other selections (Single, Double, Back, or Brackets), column names will appear between single quotes, double quotes, back quotes, or brackets in the SQL generated to access the database.				
`keyColumn`	`String`	`KEY_COLUMN`		✓ Yes
This mandatory column value will be used as the unique identifier for rows in the table.				
`passwordColumn`	`String`	`null`		✕ No
Enter the name of the column in the table that will hold the password values. If empty, no validation is done on resources and passwords.				
`disablePaging`	`boolean`	`false`		✓ Yes
If true, optional paging in a query will be ignored by the connector. Defaults to false.				
`enableEmptyString`	`boolean`	`false`		✕ No

Property	Type	Default	Encrypted[1]	Required[2]
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.				
`changeLogColumn`	`String`	`null`		• Sync
The change log column stores the latest change time. Providing this value the Sync capabilities are activated.				
`suppressPassword`	`boolean`	`true`		✕ No
If set to true then the password will not be returned. Never. Even though it is explicitly requested. If set to false then the password will be returned if it is explicitly requested.				
`inclusiveSync`	`boolean`	`false`		✕ No
If true, the SyncOp will query for ChangeLogColumn >= syncToken. One record will always be returned from the database in this case and be handled by the connector. If set to false, the SyncOp will query for ChangeLogColumn > syncToken. Defaults to false.				
`returnGeneratedKeys`	`boolean`	`true`		✕ No
If set to true then the JDBC Statement.RETURN_GENERATED_KEYS is used for prepared statement. It may need to be set to false with Oracle JDBC driver for create() operation. Defaults to tue.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

DocuSign connector

Tip

This is a SaaS common connector.

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

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

Important

Only applicable to connector version 1.5.20.21 and lower.
For a complete example that includes the configuration required to synchronize users with this connector, refer to Synchronize data between IDM and DocuSign.

Before you start

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

- Secret Key
- API Account ID
- Integration Key
- Account Base URI
- Hourly Limit
- Burst Limit

Install the DocuSign connector

Tip

To check for an Advanced Identity Cloud application for this connector, refer to:

- Application management
- App catalog

You can download any connector from Backstage, but some are included in the default deployment for Advanced Identity Cloud, IDM, or RCS. When using an included connector, you can skip installing it and move directly to configuration.

Connector included in default deployment

Connector	IDM	RCS
DocuSign	✕ No	✕ No

Download the connector .jar file from Backstage.

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

  ```
  mv ~/Downloads/docusign-connector-1.5.20.30.jar /path/to/openidm/connectors/
  ```

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

Configure the DocuSign connector

Create a connector configuration using the IDM admin UI:

1. From the navigation bar, click **Configure > Connectors**.
2. On the **Connectors** page, click **New Connector**.
3. On the **New Connector** page, type a **Connector Name**.
4. From the **Connector Type** drop-down list, select **DocuSign Connector - 1.5.20.30**.
5. Complete the **Base Connector Details**.

   > **Tip**
   >
   > For a list of all configuration properties, refer to DocuSign Connector Configuration

6. Click **Save**.

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

Refer to this procedure to create a connector configuration over REST.

Alternatively, configure the connector with a configuration file. IDM provides a sample connector configuration file in the `/path/to/openidm/samples/example-configurations/provisioners` directory. Copy this sample file (`provisioner.openicf-docusign.json`) to your project's `conf` directory.

The following excerpt shows sample configuration properties:

```
"configurationProperties": {
    "serviceUri" : "_CHANGEME_",
    "clientId" : "_CHANGEME_",
    "clientSecret" : "_CHANGEME_",
    "account" : "_CHANGEME_",
    "hourRateLimit" : "_CHANGEME_",
    "burstRateLimit" : "_CHANGEME_",
    ...
}
```

`serviceUri`

The Docusign API hostname, in the most of the cases labeled `Account Base URI`, for example `https://demo.docusign.net`.

`clientId`

The Docusign Application `Integration Key`. You can locate this ID under **Integrations** > **Apps and Keys** > **Apps and Integration Keys** when you log in to your DocuSign account

`clientSecret`

The Docusign Application `Secret Key`. You can locate this ID under **Integrations** > **Apps and Keys** > **Apps and Integration Keys** > **Actions** > **Edit** > **Authentication** when you log in to your DocuSign account.

`account`

The `API Account ID`. You can locate this account ID under **Integrations** > **Apps and Keys** > **My Account Information** when you log in to your DocuSign account.

`hourRateLimit`

The `Hourly Limit`. You can locate this value under **Integrations** > **API Usage Center** > **API Limit** when you log in to your DocuSign account.

`burstRateLimit`

The `Burst Limit`. You can locate this value under **Integrations** > **API Usage Center** > **API Limit** when you log in to your DocuSign account.

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

> **Note**
>
> If throttling problems continue, this guide may be helpful: Improve reconciliation query performance

Test the DocuSign connector

Test that the configuration is correct by running the following command:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--request POST \
"http://localhost:8080/openidm/system/docusign?_action=test"
{
  "name": "docusign",
  "enabled": true,
  "config": "config/provisioner.openicf/docusign",
  "connectorRef": {
    "bundleVersion": "[1.5.0.0,1.6.0.0)",
    "bundleName": "org.forgerock.openicf.connectors.docusign-connector",
    "connectorName": "org.forgerock.openicf.connectors.docusign.DocuSignConnector"
  },
  "displayName": "DocuSign Connector",
  "objectTypes": [
    "userSignature",
    "signingGroup",
    "__GROUP__",
    "__ALL__",
    "contact",
    "permissionProfile",
    "__ACCOUNT__"
  ],
  "ok": true
}
```

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

DocuSign remote connector

If you want to run this connector outside of PingOne Advanced Identity Cloud or IDM, you can configure the DocuSign connector as a remote connector. Java Connectors installed remotely on a Java Connector Server function identically to those bundled locally within PingOne Advanced Identity Cloud or installed locally on IDM.

You can download the DocuSign connector from here.

Refer to Remote connectors for configuring the DocuSign remote connector.

Configure connection pooling

The DocuSign connector supports HTTP pooling, which can substantially improve the performance of the connector. Learn more about the basic connection pooling configuration and different pooling mechanisms described in Connection pooling configuration.

Use the DocuSign Connector

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

Users

Create a DocuSign User

This example creates a user with the minimum required attributes.

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request POST \
--data '{
  "userName": "Carlos Garcia",
  "email": "cgarcia@example.com"
}' \
"http://localhost:8080/openidm/system/docusign/__ACCOUNT__?_action=create"
{
  "_id": "10a0c285-1976-4fb0-8955-875d281e50fd",
  "permissionProfileId": "",
  "createdDateTime": "2024-02-09T16:29:20.9270000Z",
  "__GROUP__": [
    "12580253"
  ],
  "sendActivationOnInvalidLogin": "false",
  "lastName": "Garcia",
  "isAdmin": "False",
  "jobTitle": "",
  "uri": "/users/10a0c285-1976-4fb0-8955-875d281e50fd",
  "permissionProfileName": "",
  "enableConnectForUser": "false",
  "email": "cgarcia@example.com",
  "userStatus": "ActivationSent",
  "workAddress": {
    "address1": "",
    "address2": "",
    "city": "",
    "stateOrProvince": "",
    "postalCode": "",
    "phone": "",
    "country": ""
  },
  "initialsImageUri": "/users/10a0c285-1976-4fb0-8955-875d281e50fd/signatures/
9ca8008b-31cd-42a3-89f6-6a9ccd88f25f/initials_image",
  "userId": "10a0c285-1976-4fb0-8955-875d281e50fd",
  "firstName": "Carlos",
  "__NAME__": "Carlos Garcia",
  "userSettings": {
    "canManageAccount": "false",
    "accountManagementGranular": {
      "canManageUsers": "false",
      "canManageAdmins": "false",
      "canManageSharing": "false",
      "canManageEnvelopeTransfer": "false",
      "canManageAccountSettings": "false",
      "canManageReporting": "false",
      "canManageAccountSecuritySettings": "false",
      "canManageSigningGroups": "false",
      "canManageDocumentRetention": "false",
      "canManageConnect": "false",
      "canViewUsers": "false",
      "canManageGroupsButNotUsers": "false",
      "canManageStamps": "false",
      "canManageJointAgreements": "false"
    },
    "canSendEnvelope": "false",
    "canSendAPIRequests": "false",
    "apiAccountWideAccess": "false",
    "enableVaulting": "false",
    "vaultingMode": "none",
    "enableTransactionPoint": "true",
    "enableSequentialSigningAPI": "true",
    "enableSequentialSigningUI": "true",
    "enableDSPro": "false",
    "powerFormMode": "user",
    "allowPowerFormsAdminToAccessAllPowerFormEnvelope": "false",
    "canEditSharedAddressbook": "use_private_and_shared",
    "manageClickwrapsMode": "none",
    "enableSignOnPaperOverride": "false",
    "enableSignerAttachments": "true",
    "allowSendOnBehalfOf": "false",
    "canManageTemplates": "none",
    "allowEnvelopeTransferTo": "false",
    "allowRecipientLanguageSelection": "true",
    "apiCanExportAC": "false",
    "bulkSend": "false",
    "canChargeAccount": "true",
    "canManageDistributor": "false",
    "canSignEnvelope": "true",
    "newSendUI": "false",
    "recipientViewedNotification": "false",
    "templateActiveCreation": "false",
    "templateApplyNotify": "true",
    "templateAutoMatching": "true",
    "templateMatchingSensitivity": "80",
    "templatePageLevelMatching": "false",
    "transactionPointSiteNameURL": "",
    "transactionPointUserName": "",
    "timezoneOffset": "tz_66_pacific",
    "timezoneMask": "",
    "timezoneDST": "",
    "modifiedBy": "",
    "modifiedPage": "update membership settings by acm sync job",
    "modifiedDate": "2/9/2024 5:02:47 pm",
    "adminOnly": "false",
    "selfSignedRecipientEmailDocument": "include_link",
    "signerEmailNotifications": {
      "envelopeActivation": "true",
      "envelopeComplete": "true",
      "carbonCopyNotification": "true",
      "certifiedDeliveryNotification": "true",
      "envelopeDeclined": "true",
      "envelopeVoided": "true",
      "envelopeCorrected": "true",
      "reassignedSigner": "true",
      "purgeDocuments": "true",
      "faxReceived": "true",
      "documentMarkupActivation": "true",
      "agentNotification": "true",
      "offlineSigningFailed": "true",
      "whenSigningGroupMember": "true",
      "commentsReceiveAll": "true",
      "commentsOnlyPrivateAndMention": "true"
    },
    "senderEmailNotifications": {
      "envelopeComplete": "true",
      "changedSigner": "true",
      "senderEnvelopeDeclined": "true",
      "withdrawnConsent": "true",
      "recipientViewed": "true",
      "deliveryFailed": "true",
      "offlineSigningFailed": "true",
      "purgeDocuments": "true",
      "commentsReceiveAll": "true",
      "commentsOnlyPrivateAndMention": "true",
      "powerformResponsesLimitNotificationEmail": "true",
      "clickwrapResponsesLimitNotificationEmail": "true"
    },
    "localePolicy": {
      "cultureName": "en",
      "nameFormat": "first_middle_last",
      "initialFormat": "first1last1",
      "addressFormat": "en_us",
      "dateFormat": "mdyyyy",
      "timeFormat": "none",
      "calendarType": "gregorian",
      "timeZone": "tz_66_pacific",
      "currencyPositiveFormat": "csym_1_comma_234_comma_567_period_89",
      "currencyNegativeFormat": "opar_csym_1_comma_234_comma_567_period_89_cpar",
      "effectiveNameFormat": "first_middle_last",
      "effectiveInitialFormat": "first1last1",
      "effectiveAddressFormat": "en_us",
      "effectiveDateFormat": "longformat",
      "effectiveTimeFormat": "hhmm",
      "effectiveCalendarType": "gregorian",
      "effectiveTimeZone": "tz_66_pacific",
      "effectiveCurrencyPositiveFormat": "csym_1_comma_234_comma_567_period_89",
      "effectiveCurrencyNegativeFormat": "opar_csym_1_comma_234_comma_567_period_89_cpar",
      "currencyCode": "usd",
      "effectiveCurrencyCode": "usd",
      "signDateFormat": "mdyyyy",
      "signTimeFormat": "none"
    },
    "locale": "en",
    "canLockEnvelopes": "true",
    "canUseScratchpad": "true",
    "canCreateWorkspaces": "false",
    "isWorkspaceParticipant": "false",
    "allowEmailChange": "true",
    "allowPasswordChange": "true",
    "federatedStatus": "none",
    "allowSupplementalDocuments": "true",
    "supplementalDocumentsMustView": "true",
    "supplementalDocumentsMustAccept": "true",
    "supplementalDocumentsMustRead": "true",
    "canManageOrganization": "false",
    "expressSendOnly": "false",
    "supplementalDocumentIncludeInDownload": "false",
    "disableDocumentUpload": "false",
    "disableOtherActions": "false",
    "useAccountServerForPasswordChange": "true",
    "isCommentsParticipant": "false",
    "useAccountServerForEmailChange": "true",
    "allowEsealRecipients": "false",
    "sealIdentifiers": [],
    "agreedToComments": "false",
    "canUseSmartContracts": "false",
    "canSendEnvelopesViaSMS": "false",
    "webForms": "none",
    "allowedOrchestrationAccess": "none"
  },
  "userType": "CompanyUser",
  "userName": "Carlos Garcia",
  "groupList": [
    {
      "groupId": "12580253",
      "groupName": "Everyone",
      "groupType": "everyoneGroup"
    }
  ]
}
```

Note

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

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

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

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

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

Create a DocuSign Full User

This example creates a Docusign Full User.

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request POST \
--data '{
  "userType": "CompanyUser",
  "lastName": "User3",
  "email": "showing3@example.com",
  "permissionProfileId": "",
  "permissionProfileName": "",
  "sendActivationOnInvalidLogin": "false",
  "userName": "showing3",
  "groupList": [
    {
      "groupId": "14261467"
    }
  ],
  "firstName": "showing3",
  "isAdmin": "False",
  "enableConnectForUser": "false",
  "jobTitle": "hired",
  "userSettings":
    {
      "canManageAccount": "false",
      "accountManagementGranular": {
        "canManageUsers": "false",
        "canManageAdmins": "false",
        "canManageSharing": "false",
        "canManageEnvelopeTransfer": "false",
        "canManageAccountSettings": "false",
        "canManageReporting": "false",
        "canManageAccountSecuritySettings": "false",
        "canManageSigningGroups": "false",
        "canManageDocumentRetention": "false",
        "canManageConnect": "false",
        "canViewUsers": "false",
        "canManageGroupsButNotUsers": "false",
        "canManageStamps": "false",
        "canManageJointAgreements": "false"
      },
      "canSendEnvelope": "false",
      "canSendAPIRequests": "false",
      "apiAccountWideAccess": "false",
      "enableVaulting": "false",
      "vaultingMode": "none",
      "enableTransactionPoint": "true",
      "enableSequentialSigningAPI": "true",
      "enableSequentialSigningUI": "true",
      "enableDSPro": "false",
      "powerFormMode": "user",
      "allowPowerFormsAdminToAccessAllPowerFormEnvelope": "false",
      "canEditSharedAddressbook": "use_private_and_shared",
      "manageClickwrapsMode": "none",
      "enableSignOnPaperOverride": "false",
      "enableSignerAttachments": "true",
      "allowSendOnBehalfOf": "false",
      "canManageTemplates": "none",
      "allowEnvelopeTransferTo": "false",
      "allowRecipientLanguageSelection": "true",
      "apiCanExportAC": "false",
      "bulkSend": "false",
      "canChargeAccount": "true",
      "canManageDistributor": "false",
      "canSignEnvelope": "true",
      "newSendUI": "false",
      "recipientViewedNotification": "false",
      "templateActiveCreation": "false",
      "templateApplyNotify": "true",
      "templateAutoMatching": "true",
      "templateMatchingSensitivity": "80",
      "templatePageLevelMatching": "false",
      "transactionPointSiteNameURL": "",
      "transactionPointUserName": "",
      "timezoneOffset": "tz_66_pacific",
      "timezoneMask": "",
      "timezoneDST": "",
      "modifiedBy": "",
      "modifiedPage": "",
      "modifiedDate": "1/9/2024 4:04:55 pm",
      "adminOnly": "false",
      "selfSignedRecipientEmailDocument": "include_link",
      "signerEmailNotifications": {
        "envelopeActivation": "true",
        "envelopeComplete": "true",
        "carbonCopyNotification": "true",
        "certifiedDeliveryNotification": "true",
        "envelopeDeclined": "true",
        "envelopeVoided": "true",
        "envelopeCorrected": "true",
        "reassignedSigner": "true",
        "purgeDocuments": "true",
        "faxReceived": "true",
        "documentMarkupActivation": "true",
        "agentNotification": "true",
        "offlineSigningFailed": "true",
        "whenSigningGroupMember": "true",
        "commentsReceiveAll": "true",
        "commentsOnlyPrivateAndMention": "true"
      },
      "senderEmailNotifications": {
        "envelopeComplete": "true",
        "changedSigner": "true",
        "senderEnvelopeDeclined": "true",
        "withdrawnConsent": "true",
        "recipientViewed": "true",
        "deliveryFailed": "true",
        "offlineSigningFailed": "true",
        "purgeDocuments": "true",
        "commentsReceiveAll": "true",
        "commentsOnlyPrivateAndMention": "true",
        "powerformResponsesLimitNotificationEmail": "true",
        "clickwrapResponsesLimitNotificationEmail": "true"
      },
      "localePolicy": {
        "cultureName": "en",
        "nameFormat": "first_middle_last",
        "initialFormat": "first1last1",
        "addressFormat": "en_us",
        "dateFormat": "mdyyyy",
        "timeFormat": "none",
        "calendarType": "gregorian",
        "timeZone": "tz_66_pacific",
        "currencyPositiveFormat": "csym_1_comma_234_comma_567_period_89",
        "currencyNegativeFormat": "opar_csym_1_comma_234_comma_567_period_89_cpar",
        "effectiveNameFormat": "first_middle_last",
        "effectiveInitialFormat": "first1last1",
        "effectiveAddressFormat": "en_us",
        "effectiveDateFormat": "longformat",
        "effectiveTimeFormat": "hhmm",
        "effectiveCalendarType": "gregorian",
        "effectiveTimeZone": "tz_66_pacific",
        "effectiveCurrencyPositiveFormat": "csym_1_comma_234_comma_567_period_89",
        "effectiveCurrencyNegativeFormat": "opar_csym_1_comma_234_comma_567_period_89_cpar",
        "currencyCode": "usd",
        "effectiveCurrencyCode": "usd",
        "signDateFormat": "mdyyyy",
        "signTimeFormat": "none"
      },
      "locale": "en",
      "canLockEnvelopes": "true",
      "canUseScratchpad": "true",
      "canCreateWorkspaces": "false",
      "isWorkspaceParticipant": "false",
      "allowEmailChange": "true",
      "allowPasswordChange": "true",
      "federatedStatus": "none",
      "allowSupplementalDocuments": "true",
      "supplementalDocumentsMustView": "true",
      "supplementalDocumentsMustAccept": "true",
      "supplementalDocumentsMustRead": "true",
      "canManageOrganization": "false",
      "expressSendOnly": "false",
      "supplementalDocumentIncludeInDownload": "false",
      "disableDocumentUpload": "false",
      "disableOtherActions": "false",
      "useAccountServerForPasswordChange": "true",
      "isCommentsParticipant": "false",
      "useAccountServerForEmailChange": "true",
      "allowEsealRecipients": "false",
      "sealIdentifiers": [],
      "agreedToComments": "false",
      "canUseSmartContracts": "false",
      "canSendEnvelopesViaSMS": "false",
      "webForms": "none",
      "allowedOrchestrationAccess": "none"
    }
}' \
"http://localhost:8080/openidm/system/docusign/__ACCOUNT__?_action=create"
{
  "_id": "10a0c285-1976-4fb0-8955-875d281e50fd",
  "permissionProfileId": "",
  "signatureImageUri": "/users/10a0c285-1976-4fb0-8955-875d281e50fd/signatures/
9ca8008b-31cd-42a3-89f6-6a9ccd88f25f/signature_image",
  "createdDateTime": "2024-02-09T16:29:20.9270000Z",
  "__GROUP__": [
    "12580253",
    "14261467"
  ],
  "sendActivationOnInvalidLogin": "false",
  "lastName": "updated",
  "isAdmin": "False",
  "jobTitle": "hired",
  "uri": "/users/10a0c285-1976-4fb0-8955-875d281e50fd",
  "permissionProfileName": "",
  "enableConnectForUser": "false",
  "email": "showing3@example.com",
  "userStatus": "ActivationSent",
  "workAddress": {
    "address1": "",
    "address2": "",
    "city": "",
    "stateOrProvince": "",
    "postalCode": "",
    "phone": "",
    "country": ""
  },
  "initialsImageUri": "/users/10a0c285-1976-4fb0-8955-875d281e50fd/signatures/
9ca8008b-31cd-42a3-89f6-6a9ccd88f25f/initials_image",
  "userId": "10a0c285-1976-4fb0-8955-875d281e50fd",
  "firstName": "showing3",
  "__NAME__": "showing3 updated",
  "userSettings": {
    "canManageAccount": "false",
    "accountManagementGranular": {
      "canManageUsers": "false",
      "canManageAdmins": "false",
      "canManageSharing": "false",
      "canManageEnvelopeTransfer": "false",
      "canManageAccountSettings": "false",
      "canManageReporting": "false",
      "canManageAccountSecuritySettings": "false",
      "canManageSigningGroups": "false",
      "canManageDocumentRetention": "false",
      "canManageConnect": "false",
      "canViewUsers": "false",
      "canManageGroupsButNotUsers": "false",
      "canManageStamps": "false",
      "canManageJointAgreements": "false"
    },
    "canSendEnvelope": "false",
    "canSendAPIRequests": "false",
    "apiAccountWideAccess": "false",
    "enableVaulting": "false",
    "vaultingMode": "none",
    "enableTransactionPoint": "true",
    "enableSequentialSigningAPI": "true",
    "enableSequentialSigningUI": "true",
    "enableDSPro": "false",
    "powerFormMode": "user",
    "allowPowerFormsAdminToAccessAllPowerFormEnvelope": "false",
    "canEditSharedAddressbook": "use_private_and_shared",
    "manageClickwrapsMode": "none",
    "enableSignOnPaperOverride": "false",
    "enableSignerAttachments": "true",
    "allowSendOnBehalfOf": "false",
    "canManageTemplates": "none",
    "allowEnvelopeTransferTo": "false",
    "allowRecipientLanguageSelection": "true",
    "apiCanExportAC": "false",
    "bulkSend": "false",
    "canChargeAccount": "true",
    "canManageDistributor": "false",
    "canSignEnvelope": "true",
    "newSendUI": "false",
    "recipientViewedNotification": "false",
    "templateActiveCreation": "false",
    "templateApplyNotify": "true",
    "templateAutoMatching": "true",
    "templateMatchingSensitivity": "80",
    "templatePageLevelMatching": "false",
    "transactionPointSiteNameURL": "",
    "transactionPointUserName": "",
    "timezoneOffset": "tz_66_pacific",
    "timezoneMask": "",
    "timezoneDST": "",
    "modifiedBy": "",
    "modifiedPage": "update membership settings by acm sync job",
    "modifiedDate": "2/9/2024 5:02:47 pm",
    "adminOnly": "false",
    "selfSignedRecipientEmailDocument": "include_link",
    "signerEmailNotifications": {
      "envelopeActivation": "true",
      "envelopeComplete": "true",
      "carbonCopyNotification": "true",
      "certifiedDeliveryNotification": "true",
      "envelopeDeclined": "true",
      "envelopeVoided": "true",
      "envelopeCorrected": "true",
      "reassignedSigner": "true",
      "purgeDocuments": "true",
      "faxReceived": "true",
      "documentMarkupActivation": "true",
      "agentNotification": "true",
        "offlineSigningFailed": "true",
        "whenSigningGroupMember": "true",
        "commentsReceiveAll": "true",
        "commentsOnlyPrivateAndMention": "true"
      },
      "senderEmailNotifications": {
        "envelopeComplete": "true",
        "changedSigner": "true",
        "senderEnvelopeDeclined": "true",
        "withdrawnConsent": "true",
        "recipientViewed": "true",
        "deliveryFailed": "true",
        "offlineSigningFailed": "true",
        "purgeDocuments": "true",
        "commentsReceiveAll": "true",
        "commentsOnlyPrivateAndMention": "true",
        "powerformResponsesLimitNotificationEmail": "true",
        "clickwrapResponsesLimitNotificationEmail": "true"
      },
      "localePolicy": {
        "cultureName": "en",
        "nameFormat": "first_middle_last",
        "initialFormat": "first1last1",
        "addressFormat": "en_us",
        "dateFormat": "mdyyyy",
        "timeFormat": "none",
        "calendarType": "gregorian",
        "timeZone": "tz_66_pacific",
        "currencyPositiveFormat": "csym_1_comma_234_comma_567_period_89",
        "currencyNegativeFormat": "opar_csym_1_comma_234_comma_567_period_89_cpar",
        "effectiveNameFormat": "first_middle_last",
        "effectiveInitialFormat": "first1last1",
        "effectiveAddressFormat": "en_us",
        "effectiveDateFormat": "longformat",
        "effectiveTimeFormat": "hhmm",
        "effectiveCalendarType": "gregorian",
        "effectiveTimeZone": "tz_66_pacific",
        "effectiveCurrencyPositiveFormat": "csym_1_comma_234_comma_567_period_89",
        "effectiveCurrencyNegativeFormat": "opar_csym_1_comma_234_comma_567_period_89_cpar",
        "currencyCode": "usd",
        "effectiveCurrencyCode": "usd",
        "signDateFormat": "mdyyyy",
        "signTimeFormat": "none"
      },
      "locale": "en",
      "canLockEnvelopes": "true",
      "canUseScratchpad": "true",
      "canCreateWorkspaces": "false",
      "isWorkspaceParticipant": "false",
      "allowEmailChange": "true",
      "allowPasswordChange": "true",
      "federatedStatus": "none",
      "allowSupplementalDocuments": "true",
      "supplementalDocumentsMustView": "true",
      "supplementalDocumentsMustAccept": "true",
    "supplementalDocumentsMustRead": "true",
    "canManageOrganization": "false",
    "expressSendOnly": "false",
    "supplementalDocumentIncludeInDownload": "false",
    "disableDocumentUpload": "false",
    "disableOtherActions": "false",
    "useAccountServerForPasswordChange": "true",
    "isCommentsParticipant": "false",
    "useAccountServerForEmailChange": "true",
    "allowEsealRecipients": "false",
    "sealIdentifiers": [],
    "agreedToComments": "false",
    "canUseSmartContracts": "false",
    "canSendEnvelopesViaSMS": "false",
    "webForms": "none",
    "allowedOrchestrationAccess": "none"
  },
  "userType": "CompanyUser",
  "userName": "showing3 updated",
  "groupList": [
    {
      "groupId": "12580253",
      "groupName": "Everyone",
      "groupType": "everyoneGroup"
    },
    {
      "groupId": "14261467",
      "groupName": "GLHFVAV_docusign_group",
      "groupType": "customGroup"
    }
  ]
}
```

Query DocuSign User

This example queries all DocuSign users:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--request GET \
"http://localhost:8080/openidm/system/docusign/__ACCOUNT__?_queryFilter=True"
{
  "result": [
    {
      "_id": "bb590b6d-8774-49c4-89fd-821115b8fc00",
      ...
      "userName": "A_20240206164614656_DOCUSINGTEST",
      "createdDateTime": "2024-02-06T19:46:15.4770000Z",
      "userId": "bb590b6d-8774-49c4-89fd-821115b8fc00",
      "permissionProfileName": "",
      "userStatus": "Active",
      "uri": "/users/bb590b6d-8774-49c4-89fd-821115b8fc00"
    },
    {
      "_id": "410d0a11-1923-4e78-b543-1898de76c728",
      ...
      "userName": "A_20240206164614656_DOCUSINGTEST",
      "createdDateTime": "2024-02-08T19:57:28.1270000Z",
      "userId": "410d0a11-1923-4e78-b543-1898de76c728",
      "permissionProfileName": "DocuSign Viewer",
      "userStatus": "Closed",
      "uri": "/users/410d0a11-1923-4e78-b543-1898de76c728"
    },
    ...
    {
      "_id": "451664c9-6425-4595-b823-9493063734fe",
      ...
      "userName": "A_20240206170333281_DOCUSINGTEST",
      "createdDateTime": "2024-02-21T15:03:56.2000000Z",
      "userId": "451664c9-6425-4595-b823-9493063734fe",
      "permissionProfileName": "",
      "userStatus": "ActivationSent",
      "uri": "/users/451664c9-6425-4595-b823-9493063734fe"
    }
  ],
  "resultCount": 50,
  "pagedResultsCookie": "4xj8Qq7Gkd",
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

`userStatus` can be filtered by a comma-separated list of the following:

- ActivationRequired
- ActivationSent
- Active

- Closed
- Disabled

Note

- Some filters won’t work properly unless an `userStatus` is specified.
- The maximum page size is 100.

Query IDs DocuSign User

This example queries all DocuSign users by their IDs:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--request GET \
"http://localhost:8080/openidm/system/docusign/__ACCOUNT__?_queryId=query-all-ids"
{
  "result": [
    {
      "_id": "bc9f0464-808a-4703-b4c2-c1e6a77f0c3a"
    },
    {
      "_id": "dc1c6940-1de7-4434-a91e-1407424cac91"
    },
    {
      "_id": "94be4fed-cfd7-47d5-9fcc-813405084f17"
    }
  ],
  "resultCount": 3,
  "pagedResultsCookie": "fXE265UhNb",
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

Get DocuSign User

The following command queries a specific user by its ID:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--request GET \
"http://localhost:8080/openidm/system/docusign/__ACCOUNT__/dc1c6940-1de7-4434-a91e-1407424cac91"
{
  "_id": "83947d26-2e69-40a5-892c-7a7a2600d530",
  "userId": "83947d26-2e69-40a5-892c-7a7a2600d530",
  "permissionProfileName": "",
  "createdDateTime": "2024-02-07T12:54:17.3000000Z",
  "email": "IQNREMU@foo.bar",
  "userSettings": {
    "canManageAccount": "false",
    "accountManagementGranular": {
      "canManageUsers": "false",
      "canManageAdmins": "false",
      "canManageSharing": "false",
      "canManageEnvelopeTransfer": "false",
      "canManageAccountSettings": "false",
      "canManageReporting": "false",
      "canManageAccountSecuritySettings": "false",
      "canManageSigningGroups": "false",
      "canManageDocumentRetention": "false",
      "canManageConnect": "false",
      "canViewUsers": "false",
      "canManageGroupsButNotUsers": "false",
      "canManageStamps": "false",
      "canManageJointAgreements": "false"
    },
    "canSendEnvelope": "false",
    "canSendAPIRequests": "false",
    "apiAccountWideAccess": "false",
    "enableVaulting": "false",
    "vaultingMode": "none",
    "enableTransactionPoint": "true",
    "enableSequentialSigningAPI": "true",
    "enableSequentialSigningUI": "true",
    "enableDSPro": "false",
    "powerFormMode": "user",
    "allowPowerFormsAdminToAccessAllPowerFormEnvelope": "false",
    "canEditSharedAddressbook": "use_private_and_shared",
    "manageClickwrapsMode": "none",
    "enableSignOnPaperOverride": "false",
    "enableSignerAttachments": "true",
    "allowSendOnBehalfOf": "false",
    "canManageTemplates": "none",
    "allowEnvelopeTransferTo": "false",
    "allowRecipientLanguageSelection": "true",
    "apiCanExportAC": "false",
    "bulkSend": "false",
    "canChargeAccount": "true",
    "canManageDistributor": "false",
    "canSignEnvelope": "true",
    "newSendUI": "false",
      "recipientViewedNotification": "false",
      "templateActiveCreation": "false",
      "templateApplyNotify": "true",
      "templateAutoMatching": "true",
      "templateMatchingSensitivity": "80",
      "templatePageLevelMatching": "false",
      "transactionPointSiteNameURL": "",
      "transactionPointUserName": "",
      "timezoneOffset": "tz_66_pacific",
      "timezoneMask": "",
      "timezoneDST": "",
      "modifiedBy": "",
      "modifiedPage": "update membership settings by acm sync job",
      "modifiedDate": "2/7/2024 1:02:42 pm",
      "adminOnly": "false",
      "selfSignedRecipientEmailDocument": "include_link",
      "signerEmailNotifications": {
        "envelopeActivation": "true",
        "envelopeComplete": "true",
        "carbonCopyNotification": "true",
        "certifiedDeliveryNotification": "true",
        "envelopeDeclined": "true",
        "envelopeVoided": "true",
        "envelopeCorrected": "true",
        "reassignedSigner": "true",
        "purgeDocuments": "true",
        "faxReceived": "true",
        "documentMarkupActivation": "true",
        "agentNotification": "true",
        "offlineSigningFailed": "true",
        "whenSigningGroupMember": "true",
        "commentsReceiveAll": "true",
        "commentsOnlyPrivateAndMention": "true"
      },
      "senderEmailNotifications": {
        "envelopeComplete": "true",
        "changedSigner": "true",
        "senderEnvelopeDeclined": "true",
        "withdrawnConsent": "true",
        "recipientViewed": "true",
        "deliveryFailed": "true",
        "offlineSigningFailed": "true",
        "purgeDocuments": "true",
        "commentsReceiveAll": "true",
        "commentsOnlyPrivateAndMention": "true",
        "powerformResponsesLimitNotificationEmail": "true",
        "clickwrapResponsesLimitNotificationEmail": "true"
      },
      "localePolicy": {
        "cultureName": "en",
        "nameFormat": "first_middle_last",
        "initialFormat": "first1last1",
        "addressFormat": "en_us",
        "dateFormat": "mdyyyy",
        "timeFormat": "none",
      "calendarType": "gregorian",
      "timeZone": "tz_66_pacific",
      "currencyPositiveFormat": "csym_1_comma_234_comma_567_period_89",
      "currencyNegativeFormat": "opar_csym_1_comma_234_comma_567_period_89_cpar",
      "effectiveNameFormat": "first_middle_last",
      "effectiveInitialFormat": "first1last1",
      "effectiveAddressFormat": "en_us",
      "effectiveDateFormat": "longformat",
      "effectiveTimeFormat": "hhmm",
      "effectiveCalendarType": "gregorian",
      "effectiveTimeZone": "tz_66_pacific",
      "effectiveCurrencyPositiveFormat": "csym_1_comma_234_comma_567_period_89",
      "effectiveCurrencyNegativeFormat": "opar_csym_1_comma_234_comma_567_period_89_cpar",
      "currencyCode": "usd",
      "effectiveCurrencyCode": "usd",
      "signDateFormat": "mdyyyy",
      "signTimeFormat": "none"
    },
    "locale": "en",
    "canLockEnvelopes": "true",
    "canUseScratchpad": "true",
    "canCreateWorkspaces": "false",
    "isWorkspaceParticipant": "false",
    "allowEmailChange": "true",
    "allowPasswordChange": "true",
    "federatedStatus": "none",
    "allowSupplementalDocuments": "true",
    "supplementalDocumentsMustView": "true",
    "supplementalDocumentsMustAccept": "true",
    "supplementalDocumentsMustRead": "true",
    "canManageOrganization": "false",
    "expressSendOnly": "false",
    "supplementalDocumentIncludeInDownload": "false",
    "disableDocumentUpload": "false",
    "disableOtherActions": "false",
    "useAccountServerForPasswordChange": "true",
    "isCommentsParticipant": "false",
    "useAccountServerForEmailChange": "true",
    "allowEsealRecipients": "false",
    "sealIdentifiers": [],
    "agreedToComments": "false",
    "canUseSmartContracts": "false",
    "canSendEnvelopesViaSMS": "false",
    "webForms": "none",
    "allowedOrchestrationAccess": "none"
  },
  "uri": "/users/83947d26-2e69-40a5-892c-7a7a2600d530",
  "enableConnectForUser": "false",
  "__GROUP__": [
    "12580253"
  ],
  "jobTitle": "",
  "userStatus": "ActivationSent",
  "firstName": "IQNREMU",
  "sendActivationOnInvalidLogin": "false",
  "signatureImageUri": "/users/83947d26-2e69-40a5-892c-7a7a2600d530/signatures/
9ca8008b-31cd-42a3-89f6-6a9ccd88f25f/signature_image",
  "userType": "CompanyUser",
  "userName": "IQNREMU_DOCUSINGTEST",
  "workAddress": {
    "address1": "",
    "address2": "",
    "city": "",
    "stateOrProvince": "",
    "postalCode": "",
    "phone": "",
    "country": ""
  },
  "initialsImageUri": "/users/83947d26-2e69-40a5-892c-7a7a2600d530/signatures/
9ca8008b-31cd-42a3-89f6-6a9ccd88f25f/initials_image",
  "permissionProfileId": "",
  "lastName": "DOCUSINGTEST",
  "groupList": [
    {
      "groupId": "12580253",
      "groupName": "Everyone",
      "groupType": "everyoneGroup"
    }
  ],
  "__NAME__": "IQNREMU_DOCUSINGTEST",
  "isAdmin": "False"
}
```

Update a DocuSign User

⚠ **Warning**

Sometimes, Docusign returns a successful response with an "updated user", but this is not always the case. It is recommended to check the user with a Get operation after a modification.

ⓘ **Note**

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

Close a DocuSign User

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

The following example closes a DocuSign user:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--request DELETE \
"http://localhost:8080/openidm/system/docusign/__ACCOUNT__/dc1c6940-1de7-4434-a91e-1407424cac91"
{
  "_id": "83947d26-2e69-40a5-892c-7a7a2600d530",
  "userId": "83947d26-2e69-40a5-892c-7a7a2600d530",
  "permissionProfileName": "",
  "createdDateTime": "2024-02-07T12:54:17.3000000Z",
  "email": "IQNREMU@foo.bar",
  "userSettings": {
    "canManageAccount": "false",
    ...
    "allowedOrchestrationAccess": "none"
  },
  "uri": "/users/83947d26-2e69-40a5-892c-7a7a2600d530",
  "enableConnectForUser": "false",
  "__GROUP__": [
    "12580253"
  ],
  "jobTitle": "",
  "userStatus": "Closed",
  "firstName": "IQNREMU",
  "sendActivationOnInvalidLogin": "false",
  "signatureImageUri": "/users/83947d26-2e69-40a5-892c-7a7a2600d530/signatures/
9ca8008b-31cd-42a3-89f6-6a9ccd88f25f/signature_image",
  "userType": "CompanyUser",
  "userName": "IQNREMU_DOCUSINGTEST",
  "workAddress": {
    "address1": "",
    "address2": "",
    "city": "",
    "stateOrProvince": "",
    "postalCode": "",
    "phone": "",
    "country": ""
  },
  "initialsImageUri": "/users/83947d26-2e69-40a5-892c-7a7a2600d530/signatures/
9ca8008b-31cd-42a3-89f6-6a9ccd88f25f/initials_image",
  "permissionProfileId": "",
  "lastName": "DOCUSINGTEST",
  "groupList": [
    {
      "groupId": "12580253",
      "groupName": "Everyone",
      "groupType": "everyoneGroup"
    }
  ],
  "__NAME__": "IQNREMU_DOCUSINGTEST",
  "isAdmin": "False"
}
```

Note

A `Closed` account:

- Remains in the DocuSign repository.
- Can still be queried by its ID.
- Cannot be updated.

GROUPS

Create a DocuSign Group

This example creates a DocuSign group:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request POST \
--data '{
  "groupName": "testing group 4",
  "permissionProfileId": "1597"
}' \
"http://localhost:8080/openidm/system/docusign/__GROUP__?_action=create"
{
  "_id": "14756690",
  "groupName": "testing group 4",
  "usersCount": "0",
  "__NAME__": "testing group 4",
  "groupId": "14756690",
  "groupType": "customGroup",
  "permissionProfileId": "1597"
}
```

Get a DocuSign Group

This example gets a DocuSign Group by its ID:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--request GET \
"http://localhost:8080/openidm/system/docusign/__GROUP__/14756690"
{
  "_id": "14756690",
  "groupName": "testing group 4",
  "usersCount": "0",
  "__NAME__": "testing group 4",
  "groupId": "14756690",
  "groupType": "customGroup",
  "permissionProfileId": "1597"
}
```

Query DocuSign Groups

This example queries all DocuSign groups:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--request GET \
"http://localhost:8080/openidm/system/docusign/__GROUP__?_queryFilter=True"
{
  "result": [
    {
      "_id": "14189207",
      "groupName": "ABZAECX_docusign_group",
      "__NAME__": "ABZAECX_docusign_group",
      "groupType": "customGroup",
      "groupId": "14189207",
      "usersCount": "0"
    },
    {
      "_id": "12580252",
      "groupName": "Administrators",
      "__NAME__": "Administrators",
      "groupType": "adminGroup",
      "groupId": "12580252",
      "usersCount": "2"
    },
    ...
    {
      "_id": "14188985",
      "groupName": "AOMJYHH_docusign_group",
      "__NAME__": "AOMJYHH_docusign_group",
      "groupType": "customGroup",
      "groupId": "14188985",
       "usersCount": "0"
    }
  ],
  "resultCount": 17,
  "pagedResultsCookie": "9eoQJIWWyP",
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

List All ID’s of DocuSign Groups

This example queries all DocuSign groups by their IDs:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--request GET \
"http://localhost:8080/openidm/system/docusign/__GROUP__?_queryId=query-all-ids"
{
  "result": [
    {
      "_id": "14189207"
    },
    {
      "_id": "12580252"
    },
    ...
    {
      "_id": "14189460"
    }
  ],
  "resultCount": 17,
  "pagedResultsCookie": "9eoQJIWWyP",
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

Update a DocuSign Group

This example updates a DocuSign Group by its ID:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "If-Match: *" \
--request PUT \
--data '{
  "groupName": "newGroupName",
  "permissionProfileId": "3108"
}' \
"http://localhost:8080/openidm/system/docusign/__GROUP__/14756690"
{
  "_id": "14756690",
  "groupType": "customGroup",
  "groupId": "14756690",
  "groupName": "newGroupName",
  "__NAME__": "newGroupName",
  "usersCount": "0",
  "permissionProfileId": "3108"
}
```

Delete a DocuSign Group

This example deletes a DocuSign Group by its ID:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--request DELETE \
"http://localhost:8080/openidm/system/docusign/__GROUP__/14756690"
{
  "_id": "14756690",
  "groupType": "customGroup",
  "groupId": "14261434",
  "groupName": "newGroupName",
  "__NAME__": "newGroupName",
  "usersCount": "0",
  "permissionProfileId": "3108"
}
```

> **Note**
>
> The response of the example is the group data before deletion.

SIGNING GROUPS

Create a DocuSign Signing Group

This example creates a DocuSign Signing Group:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request POST \
--data '{
  "groupEmail": "example@mail.com",
  "groupType": "sharedSigningGroup",
  "groupName": "SigningGroup Name"
}' \
"http://localhost:8080/openidm/system/docusign/signingGroup?_action=create"
{
  "_id": "950719",
  "groupEmail": "example@mail.com",
  "createdBy": "Carlos Garcia",
  "created": "2024-02-25T23:54:47.0000000Z",
  "modifiedBy": "Carlos Garcia",
  "signingGroupId": "950719",
  "groupName": "SigningGroup Name",
  "groupType": "sharedSigningGroup",
  "__NAME__": "SigningGroup Name",
  "modified": "2024-02-25T23:54:47.0000000Z"
}
```

Note

The only valid value for `groupType` is sharedSigningGroup.

Get a DocuSign Signing Group

This example gets a DocuSign Signing Group by its ID:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--request GET \
"http://localhost:8080/openidm/system/docusign/signingGroup/946554"
{
  "_id": "946554",
  "createdBy": "Carlos Garcia",
  "created": "2024-02-07T15:26:28.0000000Z",
  "modifiedBy": "Carlos Garcia",
  "signingGroupId": "946554",
  "groupName": "EZGHHNE_DOCUSINGSigningGroup",
  "groupType": "sharedSigningGroup",
  "__NAME__": "EZGHHNE_DOCUSINGSigningGroup",
  "modified": "2024-02-07T15:26:28.0000000Z"
}
```

Query DocuSign Signing Groups

This example queries all DocuSign Signing Groups:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--request GET \
"http://localhost:8080/openidm/system/docusign/signingGroup?_queryFilter=True"
{
  "result": [
    {
      "_id": "946554",
      "createdBy": "Carlos Garcia",
      "created": "2024-02-07T15:26:28.0000000Z",
      "modifiedBy": "Carlos Garcia",
      "signingGroupId": "946554",
      "groupName": "EZGHHNE_DOCUSINGSigningGroup",
      "groupType": "sharedSigningGroup",
      "__NAME__": "EZGHHNE_DOCUSINGSigningGroup",
      "modified": "2024-02-07T15:26:28.0000000Z"
    },
    {
      "_id": "946555",
      "createdBy": "Carlos Garcia",
      "created": "2024-02-07T15:26:28.0000000Z",
      "modifiedBy": "Carlos Garcia",
      "signingGroupId": "946555",
      "groupName": "LPEFDWL_DOCUSINGSigningGroup",
      "groupType": "sharedSigningGroup",
      "__NAME__": "LPEFDWL_DOCUSINGSigningGroup",
      "modified": "2024-02-07T15:26:28.0000000Z"
    },
    ...
    {
      "_id": "946556",
      "createdBy": "Carlos Garcia",
      "created": "2024-02-07T15:26:29.0000000Z",
      "modifiedBy": "Carlos Garcia",
      "signingGroupId": "946556",
      "groupName": "HQYGJAA_DOCUSINGSigningGroup",
      "groupType": "sharedSigningGroup",
      "__NAME__": "HQYGJAA_DOCUSINGSigningGroup",
      "modified": "2024-02-07T15:26:29.0000000Z"
    }
  ],
  "resultCount": 159,
  "pagedResultsCookie": "FgIJ0KDqSv",
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

Note

Querying a non-existent Signing Group will lead to a 409 response.

Update a DocuSign Signing Group

This example updates a DocuSign Signing Group by its ID:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "If-Match: *" \
--request PUT \
--data '{
  "groupEmail": "example@mail.com",
  "groupType": "sharedSigningGroup",
  "groupName": "New group name"
}' \
"http://localhost:8080/openidm/system/docusign/signingGroup/946554"
{
  "_id": "946554",
  "groupEmail": "example@mail.com",
  "createdBy": "Carlos Garcia",
  "created": "2024-02-07T15:26:28.0000000Z",
  "modifiedBy": "Carlos Garcia",
  "signingGroupId": "946554",
  "groupName": "New group name",
  "groupType": "sharedSigningGroup",
  "__NAME__": "SigningGroup Test 6 changed",
  "modified": "2024-02-26T02:21:02.0000000Z"
}
```

Delete a DocuSign Signing Group

This example deletes a DocuSign Signing Group by its ID:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--request DELETE \
"http://localhost:8080/openidm/system/docusign/signingGroup/946554"
{
  "_id": "946554",
  "groupEmail": "example@mail.com",
  "createdBy": "Carlos Garcia",
  "created": "2024-02-07T15:26:28.0000000Z",
  "modifiedBy": "Carlos Garcia",
  "signingGroupId": "946554",
  "groupName": "New group name",
  "groupType": "sharedSigningGroup",
  "__NAME__": "SigningGroup Test 6 changed",
  "modified": "2024-02-26T02:21:02.0000000Z"
}
```

CONTACTS

Create a DocuSign Contact

This example creates a DocuSign Contact:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request POST \
--data '{
  "shared": "False",
  "organization": "contact co",
  "emails": [
    "contacttesting5@mail15.com"
  ],
  "contactPhoneNumbers": [
    {
      "phoneNumber": "+12223331177",
      "phoneType": "mobile"   (1)
    },
    {
      "phoneNumber": "+12213332255",
      "phoneType": "home"  (1)
    }
  ],
  "name": "contact testing5"
}' \
"http://localhost:8080/openidm/system/docusign/contact?_action=create"
{
  "_id": "04468c26-ce0e-4484-9057-86881ed2d329",
  "shared": "False",
  "organization": "contact co",
  "contactId": "04468c26-ce0e-4484-9057-86881ed2d329",
  "__NAME__": "contact testing5",
  "emails": [
    "contacttesting5@mail15.com"
  ],
  "contactPhoneNumbers": [
    {
      "phoneNumber": "+12223331177",
      "phoneType": "mobile"
    },
    {
      "phoneNumber": "+12213332255",
      "phoneType": "home"
    }
  ],
  "name": "contact testing5"
}
```

Note

1 phoneType valid values are:

- home
- mobile
- work
- other
- fax

Get a DocuSign Contact

This example gets a DocuSign Contact by its ID:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--request GET \
"http://localhost:8080/openidm/system/docusign/contact/dac27654-b3aa-4b25-b1fb-dd2b34091809"
{
  "_id": "dac27654-b3aa-4b25-b1fb-dd2b34091809",
  "shared": "False",
  "contactId": "dac27654-b3aa-4b25-b1fb-dd2b34091809",
  "__NAME__": "A_20240220153034606_DOCUSINGContact",
  "contactPhoneNumbers": [
    {
      "phoneNumber": "+122234976",
      "phoneType": "mobile"
    },
    {
      "phoneNumber": "+122564976",
      "phoneType": "home"
    }
  ],
  "organization": "contact co",
  "emails": [
    "A_20240220153034606@foo.bar"
  ],
  "name": "A_20240220153034606_DOCUSINGContact"
}
```

Query DocuSign Contacts

This example queries all DocuSign contacts:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--request GET \
"http://localhost:8080/openidm/system/docusign/contact?_queryFilter=True"
{
  "result": [
    {
      "_id": "4702c98b-e781-4528-bf76-dcba838bdf64",
      "shared": "False",
      "contactId": "4702c98b-e781-4528-bf76-dcba838bdf64",
      "__NAME__": "A_20240219163557433_DOCUSINGContact",
      "organization": "contact co",
      "emails": [
        "A_20240219163557433@foo.bar"
      ],
      "name": "A_20240219163557433_DOCUSINGContact"
    },
    {
      "_id": "dac27654-b3aa-4b25-b1fb-dd2b34091809",
      "shared": "False",
      "contactId": "dac27654-b3aa-4b25-b1fb-dd2b34091809",
      "__NAME__": "A_20240220153034606_DOCUSINGContact",
      "organization": "contact co",
      "emails": [
        "A_20240220153034606@foo.bar"
      ],
      "name": "A_20240220153034606_DOCUSINGContact"
    },
    ...
    {
      "_id": "84d3fc99-f566-433f-b291-974e3daed207",
      "shared": "False",
      "contactId": "84d3fc99-f566-433f-b291-974e3daed207",
      "__NAME__": "A_20240220181425723_DOCUSINGContact",
      "organization": "contact co",
      "emails": [
        "A_20240220181425723@foo.bar"
      ],
      "name": "A_20240220181425723_DOCUSINGContact"
    }
  ],
  "resultCount": 20,
  "pagedResultsCookie": "Zu0tReY0z1",
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

Update a DocuSign Contact

This example updates a DocuSign Contact by its ID:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "If-Match: *" \
--request PUT \
--data '{
  "shared": "False",
  "contactPhoneNumbers": [
    {
      "phoneNumber": "+122234888",
      "phoneType": "mobile"
    },
    {
      "phoneNumber": "+122564999",
      "phoneType": "home"
    }
  ],
  "organization": "New Org",
  "emails": [
    "newemail@foo.bar"
  ],
  "name": "New Contact Name"
}' \
"http://localhost:8080/openidm/system/docusign/contact/dac27654-b3aa-4b25-b1fb-dd2b34091809"
{
  "_id": "dac27654-b3aa-4b25-b1fb-dd2b34091809",
  "shared": "False",
  "contactId": "dac27654-b3aa-4b25-b1fb-dd2b34091809",
  "__NAME__": "New Contact Name",
  "contactPhoneNumbers": [
    {
      "phoneNumber": "+122234888",
      "phoneType": "mobile"
    },
    {
      "phoneNumber": "+122564999",
      "phoneType": "home"
    }
  ],
  "organization": "New Org",
  "emails": [
    "newemail@foo.bar"
  ],
  "name": "New Contact Name"
}
```

Delete a DocuSign Contact

This example deletes a DocuSign Contact by its ID:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--request DELETE \
"http://localhost:8080/openidm/system/docusign/contact/dac27654-b3aa-4b25-b1fb-dd2b34091809"
{
  "_id": "dac27654-b3aa-4b25-b1fb-dd2b34091809",
  "shared": "False",
  "contactId": "dac27654-b3aa-4b25-b1fb-dd2b34091809",
  "__NAME__": "New Contact Name",
  "contactPhoneNumbers": [
    {
      "phoneNumber": "+122234888",
      "phoneType": "mobile"
    },
    {
      "phoneNumber": "+122564999",
      "phoneType": "home"
    }
  ],
  "organization": "New Org",
  "emails": [
    "newemail@foo.bar"
  ],
  "name": "New Contact Name"
}
```

PERMISSION PROFILE

Get a DocuSign Permission Profile

This example gets a DocuSign Permission Profile by its ID:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--request GET \
"http://localhost:8080/openidm/system/docusign/permissionProfile/17560262"
{
  "_id": "17560262",
  "permissionProfileId": "17560262",
  "modifiedByUsername": "",
  "settings": {
    "allowAccountManagement": "false",
    "useNewDocuSignExperienceInterface": "optional",
    "canCreateWorkspaces": "true",
    "allowBulkSending": "false",
    "allowEnvelopeSending": "true",
    "allowESealRecipients": "false",
    "allowSignerAttachments": "true",
    "allowTaggingInSendAndCorrect": "true",
    "allowWetSigningOverride": "true",
    "allowedAddressBookAccess": "usePersonalAndShared",
    "allowedClickwrapsAccess": "none",
    "allowedTemplateAccess": "use",
    "enableRecipientViewingNotifications": "true",
    "enableSequentialSigningInterface": "true",
    "receiveCompletedSelfSignedDocumentsAsEmailLinks": "false",
    "signingUiVersion": "v2",
    "useNewSendingInterface": "true",
    "allowSupplementalDocuments": "true",
    "supplementalDocumentsMustView": "true",
    "supplementalDocumentsMustAccept": "true",
    "supplementalDocumentsMustRead": "true",
    "disableDocumentUpload": "false",
    "disableOtherActions": "false",
    "allowAutoTagging": "false",
    "enableKeyTermsSuggestionsByDocumentType": "false",
    "allowApiAccess": "false",
    "allowApiAccessToAccount": "false",
    "allowApiSendingOnBehalfOfOthers": "false",
    "allowApiSequentialSigning": "true",
    "enableApiRequestLogging": "false",
    "allowTransactions": "false",
    "canCreateTransaction": "false",
    "canDeleteTransaction": "false",
    "canDeleteDocumentsInTransaction": "false",
    "allowDocuSignDesktopClient": "false",
    "allowSendersToSetRecipientEmailLanguage": "true",
    "allowVaulting": "false",
    "allowedToBeEnvelopeTransferRecipient": "false",
    "enableTransactionPointIntegration": "false",
    "powerFormRole": "user",
    "allowPowerFormsAdminToAccessAllPowerFormEnvelopes": "false",
    "vaultingMode": "none",
    "canSendEnvelopesViaSMS": "true",
    "webForms": "none",
    "allowedOrchestrationAccess": "none"
  },
  "__NAME__": "DocuSign Sender",
  "permissionProfileName": "DocuSign Sender",
  "modifiedDateTime": "2023-12-22T14:38:29.6070000Z"
}
```

Query DocuSign Permission Profiles

This example queries all DocuSign Permission Profiles:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--request GET \
"http://localhost:8080/openidm/system/docusign/permissionProfile?_queryFilter=True"
{
  "result": [
    {
      "_id": "17560261",
      "permissionProfileId": "17560261",
      "modifiedByUsername": "",
      "__NAME__": "Account Administrator",
      "permissionProfileName": "Account Administrator",
      "modifiedDateTime": "2023-12-22T14:38:29.6030000Z"
    },
    {
      "_id": "17560262",
      "permissionProfileId": "17560262",
      "modifiedByUsername": "",
      "__NAME__": "DocuSign Sender",
      "permissionProfileName": "DocuSign Sender",
      "modifiedDateTime": "2023-12-22T14:38:29.6070000Z"
    },
    {
      "_id": "17560263",
      "permissionProfileId": "17560263",
      "modifiedByUsername": "",
      "__NAME__": "DocuSign Viewer",
      "permissionProfileName": "DocuSign Viewer",
      "modifiedDateTime": "2023-12-22T14:38:29.6100000Z"
    },
    {
      "_id": "20022036",
      "permissionProfileId": "20022036",
      "modifiedByUsername": "Carlos Garcia",
      "__NAME__": "API Admin",
      "permissionProfileName": "API Admin",
      "modifiedDateTime": "2024-02-07T15:38:03.1770000Z"
    }
  ],
  "resultCount": 4,
  "pagedResultsCookie": null,
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

OpenICF Interfaces Implemented by the DocuSign Connector

The DocuSign Connector implements the following OpenICF interfaces. For additional details, see ICF interfaces:

Create

Creates an object and its `uid`.

Delete

Deletes an object, referenced by its `uid`.

Schema

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

Script on Connector

Enables an application to run a script in the context of the connector.

Any script that runs on the connector has the following characteristics:

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

Search

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

Test

Tests the connector configuration.

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

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

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

Update

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

DocuSign Connector Configuration

The DocuSign Connector has the following configurable properties:

Basic Configuration Properties

Property	Type	Default	Encrypted[1]	Required[2]
`serviceUri`	`String`	`null`		✓ Yes
The service endpoint URI.				
`account`	`String`	`account`		✓ Yes
Description is not available				
`login`	`String`	`null`		✓ Yes
The service login name.				
`hourRateLimit`	`String`	`null`		✓ Yes
Description is not available				
`burstRateLimit`	`String`	`null`		✓ Yes
Description is not available				
`password`	`GuardedString`	`null`	🔒 Yes	✕ No
The service user password.				
`authenticationMethod`	`String`	`OAUTH`		✓ Yes
Defines which method is to be used to authenticate on the remote server. Options are BASIC (username/password), OAUTH (Client id/secret) or TOKEN (static token).				
`tokenEndpoint`	`String`	`null`		✕ No
When using OAUTH as authentication method, this property defines the endpoint where a new access token should be queried for (https://myserver.com/oauth2/token).				
`clientId`	`String`	`null`		✓ Yes
The client identifier for OAuth2.				
`clientSecret`	`GuardedString`	`null`	🔒 Yes	✕ No
Secure client secret for OAuth2.				
`authToken`	`GuardedString`	`null`	🔒 Yes	✕ No
Static authentication token.				

Property	Type	Default	Encrypted[1]	Required[2]
`acceptSelfSignedCertificates`	`boolean`	`false`		✓ Yes
To be used for debug/test purposes. To be avoided in production.				
`disableHostNameVerifier`	`boolean`	`false`		✓ Yes
To be used for debug/test purposes. To be avoided in production.				
`disableHttpCompression`	`boolean`	`false`		✓ Yes
Content compression is enabled by default. Set this property to true to disable it.				
`clientCertAlias`	`String`	`null`		✓ Yes
If TLS Mutual Auth is needed, set this to the certificate alias from the keystore.				
`clientCertPassword`	`GuardedString`	`null`	🔒 Yes	✓ Yes
If TLS Mutual Auth is needed and the client certificate (private key) password is different from the keystore password, set this to the client private key password.				
`maximumConnections`	`Integer`	`10`		✓ Yes
Defines the max size of the HTTP connection pool used.				
`httpProxyHost`	`String`	`null`		✓ Yes
Defines the Hostname if an HTTP proxy is used between the connector and the service.				
`httpProxyPort`	`Integer`	`null`		✓ Yes
Defines the Port if an HTTP proxy is used between the connector and the service.				
`httpProxyUsername`	`String`	`null`		✓ Yes
Defines Proxy Username if an HTTP proxy is used between the connector and the service.				
`httpProxyPassword`	`GuardedString`	`null`	🔒 Yes	✓ Yes
Defines Proxy Password if an HTTP proxy is used between the connector and the service.				
`connectionTimeout`	`int`	`30`		✕ No
Defines a timeout for the underlying HTTP connection in seconds.				
`refreshToken`	`GuardedString`	`null`		✕ No
Used by the refresh_token grant type.				

Property	Type	Default	Encrypted[1]	Required[2]
`grantType`	`String`	`null`		✕ No
The OAuth2 grant type to use (client_credentials, refresh_token, or jwt_bearer).				
`scope`	`String`	`null`		✕ No
The OAuth2 scope to use.				
`authorizationTokenPrefix`	`String`	`Bearer`		✕ No
The prefix to be used in the Authorization HTTP header for Token authentication.				
`useBasicAuthForOauthTokenNeg`	`boolean`	`true`		✓ Yes
The Authentication method for refresh token (Basic Authentication or Sending the ClientId and Client Secret in the Header).				
`jwtKey`	`String`	`null`		✕ No
The JWT data structure that represents a cryptographic key.				
`jwtExpiration`	`Integer`	`null`		✕ No
Defines the JWT expiration time in seconds.				
`jwtAlgorithm`	`String`	`null`		✕ No
The Algorithm type to sign payload.				
`jwtClaims`	`Map`	`null`		✕ No
JWT Claims to be included in the payload				
`jwtPem`	`String`	`null`		✕ No
The contents of the private key of the PEM file				
`jwtCert`	`String`	`null`		✕ No
The contents of the certificate of the PEM file				
`keyAlgorithm`	`String`	`null`		✕ No
Indicates the type of key (such as RSA, DSA or EC) used to sign from the PEM.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

Dropbox connector

Tip

This is a SaaS common connector.

Before you start

1. Create a Dropbox developer account.
2. Create a new application with full Dropbox access. Add the required read and write permissions for team, member, and group. Remember to save the app key and app secret.
3. Get a refresh token.

Install the Dropbox connector

Tip

To check for an Advanced Identity Cloud application for this connector, refer to:

- Application management
- App catalog

You can download any connector from Backstage, but some are included in the default deployment for Advanced Identity Cloud, IDM, or RCS. When using an included connector, you can skip installing it and move directly to configuration.

Connector included in default deployment

Connector	IDM	RCS
Dropbox	✕ No	✕ No

Download the connector .jar file from Backstage.

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

```
mv ~/Downloads/dropbox-connector-1.5.20.30.jar /path/to/openidm/connectors/
```

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

Configure the Dropbox connector

Create a connector configuration using the IDM admin UI:

1. From the navigation bar, click **Configure > Connectors**.
2. On the **Connectors** page, click **New Connector**.
3. On the **New Connector** page, type a **Connector Name**.
4. From the **Connector Type** drop-down list, select **Dropbox Connector - 1.5.20.30**.
5. Complete the **Base Connector Details**.

> **Tip**
>
> For a list of all configuration properties, refer to Dropbox Connector Configuration

6. Click **Save**.

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

Refer to this procedure to create a connector configuration over REST.

- `Dropbox Endpoint` : https://api.dropboxapi.com/2
- `Use Basic Auth For OAuth Token Neg` : `true` | `false`
- `Max connections` : max size of the http connection pool used. Defaults to `10`.
- `Connection Timeout (seconds)` : Defines a timeout for the underlying http connection in seconds. Defaults to `30`.
- `Token Endpoint` : https://api.dropboxapi.com/oauth2/token
- `Client ID` : Your Client ID.
- `Client Secret` : Your Client Secret.
- `Refresh Token` : Your Refresh Token.

If necessary, add or edit your object types to have these three objects with their properties:

PROPERTY NAME	TYPE	NATIVE TYPE	REQUIRED
`team_member_id`	String	String	NO
`given_name`	String	String	NO
`surname`	String	String	NO
`email`	String	String	YES
`member_status`	String	String	NO
`membership_type`	String	String	NO

PROPERTY NAME	TYPE	NATIVE TYPE	REQUIRED
`role_id`	String	String	NO
`email_verified`	Boolean	Boolean	NO
`invited_on`	String	String	NO
`groups`	Array	Object	NO
`secondary_emails`	Array	String	NO
`abbreviated_name`	String	String	NO
`joined_on`	String	String	NO

PROPERTY NAME	TYPE	NATIVE TYPE	REQUIRED
`group_id`	String	String	NO
`group_name`	String	String	YES
`group_external_id`	String	String	YES
`members`	Array	Object	NO
`group_management_type`	String	String	NO
`member_count`	Integer	Integer	NO

PROPERTY NAME	TYPE	NATIVE TYPE	REQUIRED
`name`	String	String	NO
`description`	String	String	NO
`_id`	String	String	NO

If configuring the connector over REST or through the filesystem, specify the connection details to the Dropbox resource provider in the `configurationProperties` for the connector. If you are using OAuth for your connection, the minimum required properties are `serviceUri`, `tokenEndpoint`, `refreshToken`, `clientId`, and `clientSecret`.

```
{
    "configurationProperties" : {
        "tokenExpiration" : null,
        "accessToken" : null,
        "serviceUri" : "https://api.dropboxapi.com/2",
        "login" : null,
        "password" : null,
        "authenticationMethod" : "OAUTH",
        "tokenEndpoint" : "https://api.dropboxapi.com/oauth2/token",
        "clientId" : "k3..........5g",
        "clientSecret" : "xxxxxxxxxxxxxxxxxx",
        "refreshToken" : "xxxxxxxxxxxxxxxxxx",
        "authToken" : null,
        "acceptSelfSignedCertificates" : false,
        "disableHostNameVerifier" : false,
        "disableHttpCompression" : false,
        "clientCertAlias" : null,
        "clientCertPassword" : null,
        "maximumConnections" : "10",
        "httpProxyHost" : null,
        "httpProxyPort" : null,
        "httpProxyUsername" : null,
        "httpProxyPassword" : null,
        "connectionTimeout" : "30",
        "grantType" : null,
        "scope" : null,
        "authorizationTokenPrefix" : "Bearer",
        "useBasicAuthForOauthTokenNeg" : true
    }
}
```

Note

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

Configure connection pooling

The Dropbox connector supports HTTP pooling, which can substantially improve the performance of the connector. Learn more about the basic connection pooling configuration and different pooling mechanisms described in Connection pooling configuration.

Mapping

Attributes Grid

SOURCE	TARGET	TRANSFORMATION SCRIPT
`team_member_id`	`teamMemberId`	N/A
`email`	`mail`	N/A
`surname`	`sn`	N/A

SOURCE	TARGET	TRANSFORMATION SCRIPT
`given_name`	`givenName`	N/A
`member_status`	`memberStatus`	N/A
`membership_type`	`membershipType`	N/A
`email_verified`	`emailVerified`	N/A
`groups`	`groups`	N/A
`secondary_emails`	`secondaryEmails`	N/A
`member_status`	`memberStatus`	N/A

Association>Association Rules>Correlation Queries

- Link Qualifier: default
- Any of the following fields: `mail`

Attributes Grid

SOURCE	TARGET	TRANSFORMATION SCRIPT
`givenName`	`given_name`	N/A
`sn`	`surname`	N/A
`mail`	`email`	N/A
`roleId`	`role_id`	N/A

Association>Association Rules>Correlation Queries

- Link Qualifier: default
- Any of the following fields: `email`

> **Note**
>
> When a member is added or removed from a group, it is necessary to perform a reconciliation from dropbox members to IDM members to keep the members of a group up to date.

Attributes Grid

SOURCE	TARGET	TRANSFORMATION SCRIPT
`group_name`	`groupName`	N/A

SOURCE	TARGET	TRANSFORMATION SCRIPT
`members`	`members`	N/A
`member_count`	`memberCount`	N/A

`group_external_id`	`groupExternalId`	`var result = source.group_external_id;`
		`if(result == undefined){`
		`result = source.group_id;`
		`}`
		`result;`

`group_management_type`	`groupManagementType`	N/A

Association>Association Rules>Correlation Queries

- Link Qualifier: default
- Any of the following fields: `groupExternalId`

Attributes Grid

SOURCE	TARGET	TRANSFORMATION SCRIPT
`groupName`	`group_name`	N/A
`members`	`members`	N/A
`groupManagementType`	`group_management_type`	N/A
`groupExternalId`	`group_external_id`	N/A

Association>Association Rules>Correlation Queries

- Link Qualifier: default
- Any of the following fields: `group_id`, `group_external_id`

Test the Dropbox connector

Test that the connector was configured correctly:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Accept-API-Version: resource=1.0' \
--request POST  'http://localhost:8080/system/Dropbox?_action=test'
{
    "name" : "Dropbox",
    "enabled" : true,
    "config" : "config/provisioner.openicf/Dropbox",
    "connectorRef" : {
        "bundleVersion" : [1.5.0.0,1.6.0.0),
        "bundleName" : "org.forgerock.openicf.connectors.dropbox-connector",
        "connectorName" : "org.forgerock.openicf.connectors.dropbox.DropboxConnector"
    },
    "displayName" : "Dropbox Connector",
    "objectTypes" : [
        "__GROUP__",
        "role",
        "__ACCOUNT__",
        "__ALL__"
    ],
    "ok" : true
}
```

MEMBERS

Invite a member

To invite a member, it is necessary to *at least* provide the `email` field. The fields `given_name`, `surname`, `role_id` are not required, but it is advisable to fill them in because these fields cannot be modified later. To get the list of roles go here:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Accept-API-Version: resource=1.0' \
--header 'Content-Type: application/json' \
--data '{
    "email" : "NEW.MEMBER@EMAIL.COM",
    "given_name" : "GIVENNAME",
    "surname" : "SURNAME",
    "role_id" : "ROLE ID"
}'
--request POST 'http://localhost:8080/system/Dropbox/__ACCOUNT__?_action=create'
{
    "_id" : "TEAM_MEMBER_ID",
    "role_id" : "ROLE_ID",
    "email" : "NEW.MEMBER@EMAIL.COM",
    ...
}
```

Get members

Retrieve a list of team members ids from Dropbox. The limit of results and the default value are 1000:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET 'http://localhost:8080/openidm/system/Dropbox/__ACCOUNT__?_queryId=query-all-ids'
{
    "result": [
        {
            "_id" : "001"
        },
        {
            "_id" : "002"
        },
        {
            "_id" : "003"
        },
        ...
    ]
}
```

Get member

Retrieve a team member from Dropbox. The team member id must be provided in the URI path:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET 'http://localhost:8080/openidm/system/Dropbox/__ACCOUNT__/TEAM_MEMBER_ID'
{
    "_id" : "TEAM_MEMBER_ID",
    "groups" : [
        {
            "access_type" : "member",
            "group_id" : "GROUP_ID"
        }
    ],
    "email" : "test@email.com",
    "abbreviated_name" : "gs",
    "joined_on" : "2023-01-01T00:00:00Z",
    "surname" : "surname",
    "member_status" : "active",
    "email_verified" : true,
    "given_name" : "givenname",
    "membership_type" : "full",
    "role_id" : "ROLE_ID"
}
```

Get member email

Retrieve a team member in Dropbox filtering by `email` field. The team member id must be provided in the URI path:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET 'http://localhost:8080/openidm/system/Dropbox/__ACCOUNT__/TEAM_MEMBER_ID?_fields=email'
{
    "_id" : "TEAM_MEMBER_ID",
    "email" : "test@email.com"
}
```

Delete member

Delete a member from a team. The team member id must be provided in the URI path:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Accept-API-Version: resource=1.0' \
--request DELETE 'http://localhost:8080/openidm/system/Dropbox/__ACCOUNT__/TEAM_MEMBER_ID'
{
    "_id" : "TEAM_MEMBER_ID",
    "email" : "deleted.member@email.com",
    ...
}
```

GROUPS

Create a new empty group

Group name must be provided *at least* when creating a new group. `group_management_type` can be `user_managed` or `company_managed`, default is `company_managed`.

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--data '{
    "group_name" : "GROUP NAME",
    "group_management_type" : "company_managed",
    "group_external_id" : "GROUP EXTERNAL ID"
}'
--request POST 'http://localhost:8080/openidm/system/Dropbox/__GROUP__?_action=create' \
{
    "_id" : "_id",
    "group_id" : "GROUP_ID",
    "member_count" : 0,
    "group_name" : "GROUP NAME",
    "group_external_id" : "GROUP EXTERNAL ID",
    "group_management_type" : "company_managed",
    "members" : []
}
```

Get groups

Retrieves a list of groups showing only the ids. The limit of results and the default value are 1000:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET 'http://localhost:8080/openidm/system/Dropbox/__GROUP__?_queryId=query-all-ids'
{
    "result": [
        {
            "_id" : "001"
        },
        {
            "_id" : "002"
        },
        {
            "_id" : "003",
        },
        ...
    ]
}
```

Update a group

> **Note**
>
> The default group in Dropbox cannot be updated.

The fields that can be updated for a group are `group_name`, `group_external_id`, `group_management_type` (`company_managed` or `user_managed`). To add a member to a group, you need to provide the type of access and the team member id; to delete a member, delete the object. The group id must be provided in the URI path:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'If-Match: *' \
--header 'Content-Type: application/json' \
--data '{
    "group_name" : "new name",
    "group_management_type" : "company_managed",
    "group_external_id" : "new external id",
    "members" : [
        {
            "access_type" : "member",
            "team_member_id" : "TEAM_MEMBER_ID"
        }
    ]
}
--request PUT 'http://localhost:8080/openidm/system/Dropbox/__GROUP__/GROUP_ID'
{
    "group_id" : "GROUP_ID",
    "group_external_id" : "new extenal id",
    "group_name" : "new name",
    "member_count" : 1,
    "group_management_type" : "company_managed",
    "members" : [
        {
            "access_type" : "member",
            "team_member_id" : "TEAM_MEMBER_ID"
        }
    ]
}
```

Delete a group

The group id must be provided in the URI path:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Accept-API-Version: resource=1.0' \
--request DELETE 'http://localhost:8080/openidm/system/Dropbox/__GROUP__/GROUP_ID'
{
    "_id" : "GROUP_ID",
    "group_name" : "group name",
    ...
}
```

ROLES

Get available roles from Dropbox members.

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET 'http://localhost:8080/openidm/system/Dropbox/role/?_queryFilter=true'
{
    "result" : [
        {
            "_id" : "ROLE_ID",
            "name" : "Name role",
            "description" : "Description role"
        },
        ...
    ]
}
```

OpenICF Interfaces Implemented by the Dropbox Connector

The Dropbox Connector implements the following OpenICF interfaces. For additional details, see ICF 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.

Dropbox Connector Configuration

The Dropbox Connector has the following configurable properties:

Basic Configuration Properties

Property	Type	Default	Encrypted[(1)]	Required[(2)]
`serviceUri`	`String`	`null`		✓ Yes
The Dropbox endpoint URI.				
`login`	`String`	`null`		✓ Yes
The Dropbox login name.				
`password`	`GuardedString`	`null`	🔒 Yes	✕ No
The Dropbox user password.				
`authenticationMethod`	`String`	`OAUTH`		✓ Yes
Defines which method is to be used to authenticate on the remote server. Options are BASIC (username/password), OAUTH (Client id/secret) or TOKEN (static token).				
`tokenEndpoint`	`String`	`null`		✕ No
When using OAUTH as authentication method, this property defines the endpoint where a new access token should be queried for (https://myserver.com/oauth2/token).				
`clientId`	`String`	`null`		✓ Yes
The client identifier for OAuth2.				
`clientSecret`	`GuardedString`	`null`	🔒 Yes	✕ No

Property	Type	Default	Encrypted[1]	Required[2]
Secure client secret for OAuth2.				
`authToken`	`GuardedString`	`null`	🔒 Yes	✗ No
Static authentication token.				
`acceptSelfSignedCertificates`	`boolean`	`false`		✓ Yes
To be used for debug/test purposes. To be avoided in production.				
`disableHostNameVerifier`	`boolean`	`false`		✓ Yes
To be used for debug/test purposes. To be avoided in production.				
`disableHttpCompression`	`boolean`	`false`		✓ Yes
Content compression is enabled by default. Set this property to true to disable it.				
`clientCertAlias`	`String`	`null`		✓ Yes
If TLS Mutual Auth is needed, set this to the certificate alias from the keystore.				
`clientCertPassword`	`GuardedString`	`null`	🔒 Yes	✓ Yes
If TLS Mutual Auth is needed and the client certificate (private key) password is different from the keystore password, set this to the client private key password.				
`maximumConnections`	`Integer`	`10`		✓ Yes
Defines the max size of the HTTP connection pool used.				
`httpProxyHost`	`String`	`null`		✓ Yes
Defines the Hostname if an HTTP proxy is used between the connector and the service.				
`httpProxyPort`	`Integer`	`null`		✓ Yes
Defines the Port if an HTTP proxy is used between the connector and the service.				
`httpProxyUsername`	`String`	`null`		✓ Yes
Defines Proxy Username if an HTTP proxy is used between the connector and the service.				
`httpProxyPassword`	`GuardedString`	`null`	🔒 Yes	✓ Yes
Defines Proxy Password if an HTTP proxy is used between the connector and the service.				
`connectionTimeout`	`int`	`30`		✗ No

Property	Type	Default	Encrypted[1]	Required[2]
Defines a timeout for the underlying HTTP connection in seconds.				
`refreshToken`	`GuardedString`	`null`		✕ No
Used by the refresh_token grant type.				
`grantType`	`String`	`null`		✕ No
The OAuth2 grant type to use (client_credentials, refresh_token, or jwt_bearer).				
`scope`	`String`	`null`		✕ No
The OAuth2 scope to use.				
`authorizationTokenPrefix`	`String`	`Bearer`		✕ No
The prefix to be used in the Authorization HTTP header for Token authentication.				
`useBasicAuthForOauthTokenNeg`	`boolean`	`true`		✓ Yes
The Authentication method for refresh token (Basic Authentication or Sending the ClientId and Client Secret in the Header).				
`jwtKey`	`String`	`null`		✕ No
The JWT data structure that represents a cryptographic key.				
`jwtExpiration`	`Integer`	`null`		✕ No
Defines the JWT expiration time in seconds.				
`jwtAlgorithm`	`String`	`null`		✕ No
The Algorithm type to sign payload.				
`jwtClaims`	`Map`	`null`		✕ No
JWT Claims to be included in the payload				
`jwtPem`	`String`	`null`		✕ No
The contents of the private key of the PEM file				
`jwtCert`	`String`	`null`		✕ No
The contents of the certificate of the PEM file				
`keyAlgorithm`	`String`	`null`		✕ No

Property	Type	Default	Encrypted[1]	Required[2]
Indicates the type of key (such as RSA, DSA or EC) used to sign from the PEM.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

Duo connector

The duo connector lets you manage Duo service accounts and synchronize accounts and groups between Duo and the IDM or Advanced Identity Cloud managed user repository.

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

Before you start

The instructions in this guide assume you have a Duo Administrator Account and you have created and authorized a Custom Application, as described in the Duo Documentation.

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

1. Navigate to `Applications`.
2. Click `Protect an Application`.
3. Locate the entry for `Admin API` in the applications list.
4. Click `Protect` to the far-right to configure the application and make note of the following:
    - Integration Key
    - Secret Key
    - API hostname

Install the Duo connector

> **Tip**
>
> To check for an Advanced Identity Cloud application for this connector, refer to:
>
> - Application management
> - App catalog

You can download any connector from Backstage, but some are included in the default deployment for Advanced Identity Cloud, IDM, or RCS. When using an included connector, you can skip installing it and move directly to configuration.

Connector included in default deployment

Connector	IDM	RCS
Duo	✕ No	✕ No

Download the connector .jar file from Backstage.

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

  ```
  mv ~/Downloads/duo-connector-1.5.20.29.jar /path/to/openidm/connectors/
  ```

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

Configure the Duo connector

Create a connector configuration using the IDM admin UI:

1. From the navigation bar, click **Configure > Connectors**.
2. On the **Connectors** page, click **New Connector**.
3. On the **New Connector** page, type a **Connector Name**.
4. From the **Connector Type** drop-down list, select **Duo Connector - 1.5.20.29**.
5. Complete the **Base Connector Details**.

   > **Tip**
   >
   > For a list of all configuration properties, refer to Duo Connector Configuration

6. Click **Save**.

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

Refer to this procedure to create a connector configuration over REST.

The following excerpt shows sample configuration properties:

```
"configurationProperties": {
  "host" : "_CHANGEME_",
  "integrationKey" : "_CHANGEME_",
  "secretKey" : "_CHANGEME_"
}
```

`Integration Key`

The Duo Application `Integration Key`.

`Secret Key`

The Duo Application `Secret Key`.

`API hostname`

The Duo API `hostname`.

Test the Duo connector

Test that the configuration is correct by running the following command:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request POST \
'http://localhost:8080/openidm/system/duo?_action=test'
{
  "name": duo,
  "enabled": true,
  "config": "config/provisioner.openicf/Duo",
  "connectorRef": {
    "bundleVersion": "[1.5.0.0,1.6.0.0)",
    "bundleName": "org.forgerock.openicf.connectors.duo-connector",
    "connectorName": "org.forgerock.openicf.connectors.duo.DuoConnector"
  },
  "displayName": "org.forgerock.openicf.connectors.duo.DuoConnector",
  "objectTypes": [
    "__ACCOUNT__",
    "__ALL__",
    "__GROUP__"
  ],
  "ok": true
}
```

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

Duo remote connector

If you want to run this connector outside of PingOne Advanced Identity Cloud or IDM, you can configure the Duo connector as a remote connector. Java Connectors installed remotely on a Java Connector Server function identically to those bundled locally within PingOne Advanced Identity Cloud or installed locally on IDM.

You can download the Duo connector from here.

Refer to Remote connectors for configuring the Duo remote connector.

Configure connection pooling

The Duo connector uses a non-poolable mechanism to manage connections. Learn more about the different pooling mechanisms in Connectors by pooling mechanism.

Use the Duo connector

You can use the Duo connector to perform the following actions on a Duo account.

Accounts: User

Create a Duo user

This example creates a Duo user with the minimum required attributes.

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request POST \
--data '{
  "userType": "user",
  "__NAME__": "Jhon Doe",
  "email": "jhon_doe@example.com",
}' \
"http://localhost:8080/openidm/system/Duo/__ACCOUNT__?_action=create"
{
  "_id": "DUAAA11BB1C1D1EE1UUU",
  "tokens": [],
  "phones": [],
  "notes": "",
  "email": "jhon_doe@example.com",
  "__GROUPS__": [],
  "__ENABLE__": True,
  "realname": "",
  "userType": "user",
  "__NAME__": "Jhon Doe",
  "alias1": null,
  "alias2": null,
  "alias3": null,
  "alias4": null,
  "last_login": null
}
```

ⓘ **Note**

When you create a new user, you must specify at least the `__NAME__`, `email` and `userType` attributes. Valid `userType` values are: `user` and `admin`.

Create a Duo full user

This example creates a Duo full user.

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request POST \
--data '{
{
  "__GROUPS__": [
    "DGAA00BB0C0D0EE0GGG"
  ],
  "tokens": [
    "DHAAA00BB0C0D0EE0TTT"
  ],
  "phones": [
    "+12125551212"
  ],
  "notes": "example",
  "email": "jane_doe@example.com",
  "__ENABLE__": True,
  "realname": "Jane Doe",
  "alias1": "Jane",
  "alias2": null,
  "alias3": "Doe",
  "alias4": null,
  "userType": "user",
  "__NAME__": "Jane Doe"
}' \
"http://localhost:8080/openidm/system/Duo/__ACCOUNT__?_action=create"
{
  "_id": "DUAAA00BB0C0D0EE0UUU",
  "realname": "Jane Doe",
  "userType": "user",
  "phones": [
    "+12125551212"
  ],
  "__GROUPS__": [
    "DGAA00BB0C0D0EE0GGG"
  ],
  "tokens": [
    "DHAAA00BB0C0D0EE0TTT"
  ],
  "__NAME__": "Jane Doe",
  "notes": "example",
  "__ENABLE__": True,
  "email": "jane_doe@example.com",
  "alias1": "jane",
  "alias2": null,
  "alias3": "doe,
  "alias4": null,
  "last_login": null"
}
```

ⓘ **Note**

Because phone numbers are automatically created before being assigned to a user, the connector deletes unused phone numbers when you perform a user update or delete. This prevents the accumulation of junk data.
`alias` is not case-sensitive. Repeated values will be ignored.
Any attribute with a `null` value will be ignored.

List all Duo users

This example queries all Duo users:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--request GET \
"http://localhost:8080/openidm/system/Duo/__ACCOUNT__?_queryFilter=true"
{
  "result": [
    {
      "_id": "DUAAA00BB0C0D0EE0UUU",
      "realname": "Jane Doe",
      "userType": "user",
      "phones": [
        "+12125551212"
      ],
      "__GROUPS__": [],
      "tokens": [],
      "__NAME__": "jane doe",
      "notes": "example",
      "__ENABLE__": True,
      "email": "jane_doe@example.com",
      "last_login": null,
      "alias1": "jane",
      "alias2": null,
      "alias4": null,
      "alias3": "doe"
    },
    ...
    {
      "_id": "DUAAA11BB1C1D1EE1UUU",
      "realname": "",
      "userType": "user",
      "phones": [],
      "__GROUPS__": [],
      "tokens": [],
      "notes": "",
      "__NAME__": "Jhon Doe",
      "__ENABLE__": True,
      "email": "jhon_doe@example.com",
      "last_login": null,
      "alias1": null,
      "alias2": null,
      "alias3": null,
      "alias4": null
    }
  ],
  "resultCount": 96,
  "pagedResultsCookie": null,
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

Get Duo user

This example queries a specific Duo user by its ID:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request GET \
'http://localhost:8080/openidm/system/Duo/__ACCOUNT__/DUAAA11BB1C1D1EE1UUU'
{
  "_id": "DUAAA11BB1C1D1EE1UUU",
  "realname": "",
  "userType": "user",
  "phones": [],
  "__GROUPS__": [],
  "tokens": [],
  "__NAME__": "Jhon Doe",
  "notes": "",
  "__ENABLE__": True,
  "email": "jhon_doe@example.com",
  "last_login": null,
  "alias1": null,
  "alias2": null,
  "alias3": null,
  "alias4": null
}
```

Update a Duo user

This example updates a specific Duo user by its ID:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "If-Match: *" \
--request POST \
--data '{
  "alias1": "Jhon",
  "alias4": "",
  "notes": "example note",
  "email": "jhon_doe@example.com",
  "__GROUPS__": [],
  "__ENABLE__": True,
  "realname": "Jhon Doe",
  "__NAME__": "Jhon Doe"
}' \
'http://localhost:8080/openidm/system/Duo/__ACCOUNT__/DUAAA11BB1C1D1EE1UUU'
{
  "_id": "DUAAA11BB1C1D1EE1UUU",
  "tokens": [],
  "phones": [
    "+12125551212"
  ],
  "notes": "example note",
  "email": "jhon_doe@example.com",
  "__GROUPS__": [],
  "__ENABLE__": True,
  "realname": "Jhon Doe",
  "userType": "user",
  "__NAME__": "Jhon Doe",
  "last_login": null,
  "alias1": "Jhon",
  "alias2": "Doe",
  "alias3": null,
  "alias4": null
}
```

Delete a Duo user

This example deletes a Duo user:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--header "If-Match: *" \
--request DELETE \
'http://localhost:8080/openidm/system/Duo/__ACCOUNT__/DUAAA11BB1C1D1EE1UUU'
{
  "_id": "DUAAA11BB1C1D1EE1UUU",
  "realname": "Jhon Doe",
  "userType": "user",
  "phones": [
    "+12125551212"
  ],
  "__GROUPS__": [],
  "tokens": [],
  "__NAME__": "Jhon Doe",
  "notes": "Tests note",
  "__ENABLE__": True,
  "email": "jhon_doe@example.com",
  "last_login": null,
  "alias1": "Jhon",
  "alias2": "Doe",
  "alias3": null,
  "alias4": null
}
```

Note

The response returns the account object before deletion.

Accounts: Admin

Create a Duo admin

This example creates a Duo admin.

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request POST \
--data '{
  "role": "Owner",
  "phones": [
    "+12125551212"
  ],
  "email": "jhon_doe@example.com",
  "realname": "Jhon Doe",
  "userType": "admin",
  "__NAME__": "Jhon Doe",
  "restricted_by_admin_units": false
}' \
"http://localhost:8080/openidm/system/Duo/__ACCOUNT__?_action=create"
{
  "_id": "DEAAA11BB1C1D1EE1UUU",
  "phones": [
    "+12125551212"
  ],
  "email": "jhon_doe@example.com",
  "__ENABLE__": False,
  "restricted_by_admin_units": "false",
  "userType": "admin",
  "__NAME__": "Jhon Doe",
  "last_login": null,
  "role": "Owner"
}
```

When you create a new Admin, you must specify at least the `__NAME__` and `email` attributes.

`role` value must be one of the following values:

- Owner
- Administrator
- Application Manager
- User Manager
- Security Analyst
- Help Desk
- Billing
- Phishing Manager
- Read-only

The role names are case-sensitive. Defaults to "Owner" if not specified.

`phones`: Phone number for the new administrator. Limited to one. It cannot be removed once assigned due to API limitations. Formatted in the E.164 standard (`+17345551212`). If no leading plus sign is provided, then it is assumed to be a United States number and an implicit `+1` country code is prepended. Dashes and spaces are ignored. If this parameter is specified, it cannot be empty.

`restricted_by_admin_units`: Is this administrator restricted by an administrative unit assignment? Either `true` or `false`. Defaults to `false` if not specified. Must be set to `true` in order to add the admin to an administrative unit using the API. Note that attempting to set to `true` for admins with the `Owner` role results in a failure response.

`send_email`: Optional. If set to `1`, the activation link and an introductory message will be emailed to the new administrator. If set to `0`, no email is sent, and the link is returned to the API method's caller only. Defaults to `0`.

`token_id`: Optional. The `token_id` of the hardware token to associate with the administrator.

`valid_days`: Optional. Number of days before the activation link expires. Defaults to `7`, the maximum allowed value is `31`.

`__ENABLE__` will remain `False` until the admin's account is validated. In the meantime, `__ENABLE__` cannot be changed by the connector.

Get Duo Admin

This example queries a specific Duo admin by its ID:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request GET \
'http://localhost:8080/openidm/system/Duo/__ACCOUNT__/DEAAA11BB1C1D1EE1UUU'
{
  "_id": "DEAAA11BB1C1D1EE1UUU",
  "phones": [
    "+12125551212"
  ],
  "email": "jhon_doe@example.com",
  "__ENABLE__": True,
  "restricted_by_admin_units": "false",
  "userType": "admin",
  "__NAME__": "Jhon Doe",
  "last_login": null,
  "role": "Owner"
}
```

Update a Duo Admin

This example updates a specific Duo admin by its ID:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "If-Match: *" \
--request POST \
--data '{
  "role": "Read-Only",
  "__ENABLE__": True,
}' \
'http://localhost:8080/openidm/system/Duo/__ACCOUNT__/DEAAA11BB1C1D1EE1UUU'
{
  "_id": "DEAAA11BB1C1D1EE1UUU",
  "phones": [
    "+12125551212"
  ],
  "email": "jhon_doe@example.com",
  "__ENABLE__": True,
  "restricted_by_admin_units": "false",
  "userType": "admin",
  "__NAME__": "Jhon Doe",
  "last_login": null,
  "role": "Read-Only"
}
```

GROUPS

Create a Duo group

This example creates a Duo group:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request POST \
--data '{
  "desc": "Generic Description",
  "__ENABLE__": "True,
  "__NAME__": "Group 1"
}' \
'http://localhost:8080/openidm/system/Duo/__GROUP__'
{
  "_id": "DGAA00BB0C0D0EE0GGG",
  "desc": "Generic Description",
  "__ENABLE__": True,
  "__NAME__": "Group 1"
}
```

Query all Duo groups

This example queries all Duo groups:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request GET \
'http://localhost:8080/openidm/system/Duo/__GROUP__?_queryFilter=True'
{
  "result": [
    {
      "_id": "DGAA00BB0C0D0EE0GGG",
      "desc": "Generic Description",
      "__ENABLE__": True,
      "__NAME__": "Testing Group"
    },
    ...
    {
      "_id": "DGAA11BB1C1D1EE1GGG",
      "desc": "Working Group",
      "__ENABLE__": True,
      "__NAME__": "Working Group"
    }
  ],
  "resultCount": 56,
  "pagedResultsCookie": null,
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

Note

The maximum number of records returned is 100 per page.

Get a Duo group

This example gets a Duo group by its ID:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request GET \
'http://localhost:8080/openidm/system/Duo/__GROUP__/DGAA00BB0C0D0EE0GGG'
{
  "_id": "DGAA00BB0C0D0EE0GGG",
  "desc": "Generic Description",
  "__ENABLE__": True,
  "__NAME__": "Group 1"
}
```

Update a Duo group

This example updates a Duo group by its ID:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--header "If-Match: *" \
--request PUT \
--data '{
  "desc": "New tests",
  "__NAME__": "New Group tests",
}' \
'http://localhost:8080/openidm/system/Duo/__GROUP__/DGAA00BB0C0D0EE0GGG'
{
  "_id": "DGAA00BB0C0D0EE0GGG",
  "__NAME__": "New Group tests",
  "desc": "New tests",
  "__ENABLE__": True
}
```

Delete a Duo group

This example deletes a Duo group by its ID:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request DELETE \
'http://localhost:8080/openidm/system/Duo/__GROUP__/DGAA00BB0C0D0EE0GGG'
{
  "_id": "DGAA00BB0C0D0EE0GGG",
  "__NAME__": "New Group tests",
  "desc": "New tests",
  "__ENABLE__": True
}
```

> **Note**
>
> The response returns the group object before deletion.

Mapping

Attributes Grid: Where the columns represent the attribute name mapped from source to target and the necessary data transformation to synchronize successfully.

SOURCE	TARGET	TRANSFORMATION SCRIPT
`_id`	`user_id`	N/A
`__NAME__`	`username`	N/A
`alias1`	`alias1`	N/A
`alias2`	`alias2`	N/A
`alias3`	`alias3`	N/A
`alias4`	`alias4`	N/A
`aliases`	`aliases`	N/A
`realname`	`realname`	N/A
`email`	`email`	N/A
`notes`	`notes`	N/A
`phones`	`phones`	N/A
`__ENABLE__`	`status`	N/A
`u2f_tokens`	`u2f_tokens`	N/A
`enable_auto_prompt`	`enable_auto_prompt`	N/A
`external_id`	`external_id`	N/A
`__GROUPS__`	`groups`	N/A
`is_enrolled`	`is_enrolled`	N/A
`last_directory_sync`	`last_directory_sync`	N/A
`lockout_reason`	`lockout_reason`	N/A
`tokens`	`tokens`	N/A
`webauthncredentials`	`webauthncredentials`	N/A

Attributes Grid: Where the columns represent the attribute name mapped from source to target and the necessary data transformation to synchronize successfully.

SOURCE	TARGET	TRANSFORMATION SCRIPT
`user_id`	`_id`	N/A
`username`	`__NAME__`	N/A
`alias1`	`alias1`	N/A
`alias2`	`alias2`	N/A
`alias3`	`alias3`	N/A
`alias4`	`alias4`	N/A
`aliases`	`aliases`	N/A
`realname`	`realname`	N/A
`email`	`email`	N/A
`notes`	`notes`	N/A
`phones`	`phones`	N/A
`status`	`__ENABLE__`	N/A
`u2f_tokens`	`u2f_tokens`	N/A
`enable_auto_prompt`	`enable_auto_prompt`	N/A
`external_id`	`external_id`	N/A
`groups`	`__GROUPS__`	N/A
`is_enrolled`	`is_enrolled`	N/A
`last_directory_sync`	`last_directory_sync`	N/A
`lockout_reason`	`lockout_reason`	N/A
`tokens`	`tokens`	N/A
`webauthncredentials`	`webauthncredentials`	N/A

Attributes Grid: Where the columns represent the attribute name mapped from source to target and the necessary data transformation to synchronize successfully.

SOURCE	TARGET	TRANSFORMATION SCRIPT
`group_id`	`group_id`	N/A
`__NAME__`	`name`	N/A
`desc`	`desc`	N/A
`__ENABLE__`	`status`	N/A

Attributes Grid: Where the columns represent the attribute name mapped from source to target and the necessary data transformation to synchronize successfully.

SOURCE	TARGET	TRANSFORMATION SCRIPT
`group_id`	`group_id`	N/A
`name`	`__NAME__`	N/A
`desc`	`desc`	N/A
`status`	`__ENABLE__`	N/A

OpenICF Interfaces Implemented by the Duo Connector

The Duo Connector implements the following OpenICF interfaces. For additional details, see ICF 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.

Duo Connector Configuration

The Duo Connector has the following configurable properties:

Basic Configuration Properties

Property	Type	Default	Encrypted[1]	Required[2]
`host`	`String`	`null`		✓ Yes
URL API Host (api-xxxxxx.duosecurity.com)				
`integrationKey`	`String`	`null`		✓ Yes
App Integration Key				
`secretKey`	`GuardedString`	`null`	🔒 Yes	✕ No
App Secret Key				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

Epic connector

Epic is a healthcare-related service that handles medical records. The Epic connector enables customers to manage Epic EMP user and SER provider accounts along with their entitlements (Template and SubTemplate) using Epic APIs. For more information, refer to the Personnel Management and Demographics API documentation available at https://open.epic.com.

The Epic connector supports the following features:

- Account management
  - Manage Epic EMP records as users
  - Create, update, delete
  - Enable, disable, unblock
- Provider management
  - Manage SER records as linked provider
  - Create, update
  - Enable, disable
- Other object types (only search operations)
  - Manage Epic Linked Template
  - Manage Epic Linked SubTemplates
  - Manage Epic InBasketClassifications
  - Manage Epic Login Department
  - Manage Epic Groups

Important

- Supported Epic versions: May 2019, August 2020, May 2020, February 2020
- The Epic connector supports EMP user and SER provider records.
- An Epic administrator account on the Epic system you want to connect to is required for this connector to work.

Note

Contact your Epic Admin to obtain the Active Linkable Templates, Login Departments, Active Linkable Sub Templates, InBasketClassifications, and Groups.

Before you start

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

- Client ID

  A valid Epic Client ID with access to Epic's personnel management and demographics (user) web services.

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

> **Note**
>
> To generate your private key:
>
> 1. Generate and download an RSA key pair.
> 2. Run the following command to convert the RSA private key to PKCS8 format:
>
> ```
> openssl pkcs8 -topk8 -nocrypt -in privatekey.pem -out epic_pkcs8_private_key.pem
> ```
>
> 3. After generating the private key in PKCS8 format, remove `-----BEGIN PRIVATE KEY-----` and `-----END PRIVATE KEY-----` from the generated PKCS8 private key file.
> 4. Remove any escape characters such as `\n` or `\r`.

- REST endpoint (Optional)
- SOAP endpoint (Optional)
- Max records (Optional)
- User template file path (Optional)
- User sub template file path (Optional)
- In Basket classification file path (Optional)
- Group file path (Optional)

> **Note**
>
> Contact your Epic Admin to obtain the Active Linkable Templates, Login Departments, Active Linkable Sub Templates, InBasketClassifications, and Groups.

> **Note**
>
> The **user template**, **user sub template**, **in basket**, and **group** file paths are local paths that are accessible to the IDM or RCS instance.

Additional connector requirements:

- Epic's SOAP 1.1 version of web services enabled and accessible.
- Epic system's Personnel management and demographics (user) web services enabled for access.

- Valid Epic web services credentials.
- A list of Templates and SubTemplates.

  Epic does not provide APIs to programmatically query for Templates and SubTemplates. You must export separate CSV files for use with the Epic connector.
- Contact Epic to enable the following APIs:

`GetImportDataLog`	Used to get provider response log
`ImportData (2019)`	Used to create/update provider
`InactivateUser`	Used to Disable user
`GetRecords`	Used to get all employee/provider records

- Unlocked user attributes

  If you need any of the following attributes unlocked, you must request them from the Epic Data Courier team:

  - `displayLicense`
  - `recipientType`
  - `isMedsAuth`
  - `isOrdersAuth`
  - `communicationPreference`
  - `taxId`
  - `isInPatientOrderProvider`
  - `isOutPatientOrderProvider`

Install the Epic connector

> **Tip**
>
> To check for an Advanced Identity Cloud application for this connector, refer to:
>
> - Application management
> - App catalog

You can download any connector from Backstage, but some are included in the default deployment for Advanced Identity Cloud, IDM, or RCS. When using an included connector, you can skip installing it and move directly to configuration.

Connector included in default deployment

Connector	IDM	RCS
Epic	✕ No	✓ Yes

Download the connector .jar file from Backstage.

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

```
mv ~/Downloads/epic-connector-1.5.20.29.jar /path/to/openidm/connectors/
```

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

Configure the Epic connector

Create a connector configuration using the IDM admin UI:

1. From the navigation bar, click **Configure > Connectors**.
2. On the **Connectors** page, click **New Connector**.
3. On the **New Connector** page, type a **Connector Name**.
4. From the **Connector Type** drop-down list, select **Epic Connector - 1.5.20.29**.
5. Complete the **Base Connector Details**.

> **Tip**
>
> For a list of all configuration properties, refer to Epic Connector Configuration

6. Click **Save**.

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

Refer to this procedure to create a connector configuration over REST.

Test the Epic connector

Test that the configuration is correct by running the following command:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system/Epic?_action=test"
{
  "name": "Epic",
  "enabled": true,
  "config": "config/provisioner.openicf/Epic",
  "connectorRef": {
    "bundleVersion": "[1.5.0.0,1.6.0.0)",
    "bundleName": "org.forgerock.openicf.connectors.epic-connector",
    "connectorName": "org.forgerock.openicf.connectors.epic.EpicConnector"
  },
  "displayName": "Epic Connector",
  "objectTypes": [
    "__ACCOUNT__",
    "__ALL__"
  ],
  "ok": true
}
```

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

Epic remote connector

If you want to run this connector outside of PingOne Advanced Identity Cloud or IDM, you can configure the Epic connector as a remote connector. Java Connectors installed remotely on a Java Connector Server function identically to those bundled locally within PingOne Advanced Identity Cloud or installed locally on IDM.

You can download the Epic connector from here.

Refer to Remote connectors for configuring the Epic remote connector.

Configure connection pooling

The Epic connector supports HTTP pooling, which can substantially improve the performance of the connector. Learn more about the basic connection pooling configuration and different pooling mechanisms described in Connection pooling configuration.

Supported resource types

The Epic connector supports the following resource types:

Epic connector supported resource types

ICF Native Type	Epic Resource Type	Naming Attribute
`__ACCOUNT__`	User	__NAME__
`Department`	Department	__NAME__

ICF Native Type	Epic Resource Type	Naming Attribute
`Provider`	Linked Provider	__NAME__
`UserTemplate`	User Template	__NAME__
`UserSubTemplate`	User Sub Template	__NAME__
`InBasketClassifications`	In Basket Classifications	__NAME__
`__GROUP__`	Groups	__NAME__

Supported search filters

The Epic connector supports Search operations with the following filter operators and attributes:

Supported Operators and Filter Attributes With Epic Searches

Object Type	Operators	Attributes
`__ACCOUNT__`	Id filter	Id
`Department`	Id filter	Id
`Provider`	Id filter	Id
`UserTemplate`	Id filter	Id
`UserSubTemplate`	Id filter	Id
`InBasketClassifications`	Id filter	Id
`__GROUP__`	Id filter	Id

Attributes

The Epic connector supports the following attributes:

Account attributes

Attribute	Description
Name	The name of the user
ContactDate	Date to use for the newly created contact
IsActive	Whether the user record should be set to active or inactive

Attribute	Description
UserID	External identifier for the user
UserIDType	Type of identifier provided in UserID
UserComplexName	Individual pieces of a user’s name which includes Academic Title, Father Name, First Name, Given Name initials, Last Name, Last Name Prefix, Primary Title, Spouse Last Name, Spouse Last Name First, Spouse Prefix, Suffix
ContactComment	Contact comment that will appear for the initial contact
SystemLoginID	New value for the user’s system login
LDAPOverrideID	A string that can be provided to identify the user to the LDAP server in place of the SystemLogin
UserAlias	New value for the user’s alias
UserPhotoPath	A URL or file path identifying the location of a picture to show for this user
Sex	The sex of the user
ReportGrouper1	New value for the user’s first freetext report grouper item
ReportGrouper2	New value for the user’s second free-text report grouper item
ReportGrouper3	New value for the user’s third freetext report grouper item
Notes	Free text notes about the user
UsersManagers	The user’s managers
PrimaryManager	The primary manager of this user
BlockStatus	A user’s block status which includes IsBlocked, BlockReason, BlockComment
DefaultLoginDepartmentID	The ID and ID type of the default login department
CategoryReportGrouper6	A list of values for the user’s sixth category report grouper item, which does not have an Epic-defined meaning
InBasketClassifications	List of new values for a user In Basket classifications
UserSubtemplateIDs	Sub template of the user
DefaultTemplateID	Default and Linked template of the user
Group	Group the user belongs to

Attribute	Description
Provider	The user’s provider id
LinkedProviderID	Links a provider’s schedule to a user
Department	Department the user belongs to
UserTemplate	Template of the User

InBasketClassifications attributes

> **Note**
>
> Contact your Epic Admin to obtain the Active Linkable Templates, Login Departments, Active Linkable Sub Templates, InBasketClassifications, and Groups.

Attribute	Description
Value	The ID of the InBasketClassifications
Title	Description of the InBasketClassifications

UserSubtemplateID attributes

> **Note**
>
> Contact your Epic Admin to obtain the Active Linkable Templates, Login Departments, Active Linkable Sub Templates, InBasketClassifications, and Groups.

Attribute	Description
User Subtemplate ID	The ID of the Linked Sub-template
User Subtemplate Name	The Name of the Linked Sub-template

Group attributes

> **Note**
>
> Contact your Epic Admin to obtain the Active Linkable Templates, Login Departments, Active Linkable Sub Templates, InBasketClassifications, and Groups.

Attribute	Description
Value	The ID of the Group

Attribute	Description
Title	Display name of the Group
Type	Type of the Group

Provider attributes

Attribute	Description
ExternalID	The ID of the Provider
Name	The name of the Provider
Title	Title of the Provider
NPI ID	NPI ID of the Provider
Provider Type	Type of the Provider
Specialty	Specialty of the Provider
Practice Name	Practice Name of the Provider
Street Address	Street Address of the Provider
Phone	Phone of the Provider
Specialties	The specialties of the Provider
Doctors Degree	The qualification of the Provider

Department attributes

> **Note**
>
> Contact your Epic Admin to obtain the Active Linkable Templates, Login Departments, Active Linkable Sub Templates, InBasketClassifications, and Groups.

Attribute	Description
ExternalID	The ID of the Department
Name	The name of the Department
Center	Center of the Department
Specialty	Specialty of the Department

Attribute	Description
Location	Location of the Department
Service Area	Service Area of the Department

UserTemplate attributes

> **Note**
>
> Contact your Epic Admin to obtain the Active Linkable Templates, Login Departments, Active Linkable Sub Templates, InBasketClassifications, and Groups.

Attribute	Description
User Template ID	The ID of the LinkedTemplate
User Template Name	The Name of the LinkedTemplate

Use the Epic connector

> **Important**
>
> The Epic connector supports EMP user and SER provider records.

The Epic connector can perform the following actions:

Users

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

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json"\
--request POST \
--data '{
  "UserID": "8675309",
  "__NAME__": "Walter, Taylor"
}'\
"http://localhost:8080/openidm/system/Epic/__ACCOUNT__?_action=create"
{
  "_id": "8675309",
  "UserComplexName": {
    "FirstName": "Taylor",
    "GivenNameInitials": "",
    "MiddleName": "",
    "LastName": "Walter",
    "LastNamePrefix": "",
    "SpouseLastName": "",
    "SpousePrefix": "",
    "Suffix": "",
    "AcademicTitle": "",
    "PrimaryTitle": "",
    "SpouseLastNameFirst": false,
    "FatherName": "",
    "GrandfatherName": ""
  },
  "BlockStatus": {
    "IsBlocked": false,
    "Reason": "",
    "Comment": ""
  },
  "__GROUP__": [],
  "ContactComment": "Initial contact created via web service",
  "UserID": "8675309",
  "__NAME__": "WALTER, TAYLOR",
  "UsersManagers": [],
  "InBasketClassifications": [],
  "__ENABLE__": true
}
```

Note

- When you create a new user, you must specify *at least* `UserID` and `__NAME__` .
- The maximum length of `__NAME__` is 100 characters. The format for userName is LAST, FIRST MI format.
- The maximum number of characters for `UserID` is 15.

The following example queries all Epic users:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/Epic/__ACCOUNT__?_queryId=query-all-ids"
{
  "result": [
    {
      "_id": "dsully"
    },
    {
      "_id": "999999999"
    },
    {
      "_id": "admin@ACECompany.com"
    },
    {
      "_id": "extuser320"
    },
    {
      "_id": "Achong"
    },
    {
      "_id": "dsewell"
    },
    {
      "_id": "8675309"
    },
    {
      "_id": "atestuser"
    },
    {
      "_id": "Amaraphornc"
    },
    {
      "_id": "jocampo"
    }
  ],
  "resultCount": 10,
  "pagedResultsCookie": null,
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

The following command queries a specific user by their ID:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/Epic/__ACCOUNT__/8675309"
{
  "_id": "8675309",
  "UserComplexName": {
    "FirstName": "Taylor",
    "GivenNameInitials": "",
    "MiddleName": "",
    "LastName": "Walter",
    "LastNamePrefix": "",
    "SpouseLastName": "",
    "SpousePrefix": "",
    "Suffix": "",
    "AcademicTitle": "",
    "PrimaryTitle": "",
    "SpouseLastNameFirst": false,
    "FatherName": "",
    "GrandfatherName": ""
  },
  "BlockStatus": {
    "IsBlocked": false,
    "Reason": "",
    "Comment": ""
  },
  "__GROUP__": [],
  "ContactComment": "Initial contact created via web service",
  "UserID": "8675309",
  "__NAME__": "WALTER, TAYLOR",
  "UsersManagers": [],
  "InBasketClassifications": [],
  "__ENABLE__": true
}
```

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

- `__ENABLE__`
- `__GROUP__`
- `__NAME__` (Required)
- `__PASSWORD__`
- `UserID`
- `UserIDType`
- `UserAlias`
- `UserPhotoPath`

- `Sex`
- `Notes`
- `Provider`
- `LinkedProviderID`
- `Department`
- `ContactComment`
- `ContactDate`
- `SystemLoginID`
- `LDAPOverrideID`
- `DefaultLoginDepartmentID`
- `ReportGrouper1`
- `ReportGrouper2`
- `ReportGrouper3`
- `CategoryReportGrouper`
- `InBasketClassifications`
- `UsersManagers`
- `PrimaryManager`
- `DefaultTemplateID`
- `UserTemplate`
- `UserSubtemplateIDs`
- `UserComplexName` - `UserComplexName` has the following sub-attributes:
  - `FirstName`
  - `GivenNameInitials`
  - `MiddleName`
  - `LastName`
  - `LastNamePrefix`
  - `SpouseLastName`
  - `SpousePrefix`
  - `SpouseLastNameFirst`
  - `Suffix`

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

- `IsActive`
- `BlockStatus` - `BlockStatus` has the following sub-attributes:
  - `IsBlocked`
  - `BlockStatus.Comment`
- `__UID__`
- `__NAME__`
- `__UID__`
- `__NAME__`
- `Type`
- `__UID__`
- `__NAME__`
- `__UID__`
- `__NAME__`
- `__NAME__`
- `__UID__`
- `ExternalID`
- `Title`
- `NPI ID`
- `Provider Type`
- `Specialty`
- `Practice Name`
- `Street Address`
- `Phone`

For example, to add a `Suffix` to a user:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "if-Match:*" \
--request PUT \
--data '{
  "__NAME__": "Walter, Taylor",
  "UserComplexName": {
    "Suffix": "Junior"
  }
}' \
"http://localhost:8080/openidm/system/Epic/__ACCOUNT__/8675309"
{
  "_id": "8675309",
  "UserComplexName": {
    "FirstName": "Taylor",
    "GivenNameInitials": "",
    "MiddleName": "",
    "LastName": "Walter",
    "LastNamePrefix": "",
    "SpouseLastName": "",
    "SpousePrefix": "",
    "Suffix": "Jr.",
    "AcademicTitle": "",
    "PrimaryTitle": "",
    "SpouseLastNameFirst": false,
    "FatherName": "",
    "GrandfatherName": ""
  },
  "BlockStatus": {
    "IsBlocked": false,
    "Reason": "",
    "Comment": ""
  },
  "__GROUP__": [],
  "UserID": "8675309",
  "__NAME__": "WALTER, TAYLOR JR.",
  "UsersManagers": [],
  "InBasketClassifications": [],
  "__ENABLE__": true
}
```

To reset the password for Epic user account, you can use the connector to change a user’s password.

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "if-Match:*" \
--request PUT \
--data '{
  "__PASSWORD__": "Passw0rd@123!",
  "__NAME__": "Walter, Taylor"
}' \
"http://localhost:8080/openidm/system/Epic/__ACCOUNT__/8675309"
{
  "_id": "8675309",
  "UserComplexName": {
    "FirstName": "Taylor",
    "GivenNameInitials": "",
    "MiddleName": "",
    "LastName": "Walter",
    "LastNamePrefix": "",
    "SpouseLastName": "",
    "SpousePrefix": "",
    "Suffix": "",
    "AcademicTitle": "",
    "PrimaryTitle": "",
    "SpouseLastNameFirst": false,
    "FatherName": "",
    "GrandfatherName": ""
  },
  "BlockStatus": {
    "IsBlocked": false,
    "Reason": "",
    "Comment": ""
  },
  "__GROUP__": [],
  "UserID": "8675309",
  "__NAME__": "WALTER, TAYLOR",
  "UsersManagers": [],
  "InBasketClassifications": [],
  "__ENABLE__": true
}
```

The following example activates a user with the minimum required attributes, and updates their name:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "if-Match:*" \
--request PUT \
--data '{
  "__NAME__": "Walter, Taylorupdate",
  "__ENABLE__": "true"
}' \
"http://localhost:8080/openidm/system/Epic/__ACCOUNT__/8675309"
{
  "_id": "8675309",
  "UserComplexName": {
    "FirstName": "Taylorupdate",
    "GivenNameInitials": "",
    "MiddleName": "",
    "LastName": "Walter",
    "LastNamePrefix": "",
    "SpouseLastName": "",
    "SpousePrefix": "",
    "Suffix": "",
    "AcademicTitle": "",
    "PrimaryTitle": "",
    "SpouseLastNameFirst": false,
    "FatherName": "",
    "GrandfatherName": ""
  },
  "BlockStatus": {
    "IsBlocked": false,
    "Reason": "",
    "Comment": ""
  },
  "__GROUP__": [],
  "UserID": "8675309",
  "__NAME__": "WALTER, TAYLORUPDATE",
  "UsersManagers": [],
  "InBasketClassifications": [],
  "__ENABLE__": true
}
```

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

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

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

Note

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

The following example deletes an Epic account:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request DELETE \
"http://localhost:8080/openidm/system/Epic/__ACCOUNT__/8675309"
{
  "_id": "8675309",
  "UserComplexName": {
    "FirstName": "Taylor",
    "GivenNameInitials": "",
    "MiddleName": "",
    "LastName": "Walter",
    "LastNamePrefix": "",
    "SpouseLastName": "",
    "SpousePrefix": "",
    "Suffix": "",
    "AcademicTitle": "",
    "PrimaryTitle": "",
    "SpouseLastNameFirst": false,
    "FatherName": "",
    "GrandfatherName": ""
  },
  "BlockStatus": {
    "IsBlocked": false,
    "Reason": "",
    "Comment": ""
  },
  "__GROUP__": [],
  "UserID": "8675309",
  "__NAME__": "WALTER, TAYLOR",
  "UsersManagers": [],
  "InBasketClassifications": [],
  "__ENABLE__": false
}
```

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

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/Epic/__ACCOUNT__/8675309"
{
  "code": 404,
  "reason": "Not Found",
  "message": "Object 8675309 not found on system/Epic/__ACCOUNT__"
}
```

Other objects

> **Note**
>
> Contact your Epic Admin to obtain the Active Linkable Templates, Login Departments, Active Linkable Sub Templates, InBasketClassifications, and Groups.

All supported resources can be queried, such as Department, Provider, Template, Sub Template, InBasketClassifications, and Groups:

Query all departments

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/Epic/Department?_queryId=query-all-ids"
{
  "result": [
    {
      "_id": "40"
    },
    {
      "_id": "56"
    },
    {
      "_id": "71"
    },
    {
      "_id": "77"
    },
    {
      "_id": "58"
    },
    [ ... ]
    {
      "_id": "46"
    },
    {
      "_id": "10120160"
    },
    {
      "_id": "1002020"
    },
    {
      "_id": "31"
    },
    {
      "_id": "83"
    },
    {
      "_id": "115"
    }
  ],
  "resultCount": 548,
  "pagedResultsCookie": null,
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

Query all providers

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/Epic/Provider?_queryId=query-all-ids"
{
  "result": [
    {
      "_id": "116"
    },
    {
      "_id": "E3087"
    },
    {
      "_id": "E4000"
    },
    {
      "_id": "E4913"
    },
    {
      "_id": "E5335"
    },
    {
      "_id": "E4716"
    },
    {
      "_id": "E5370"
    },
    [ ... ]
    {
      "_id": "E4001"
    },
    {
      "_id": "E4002"
    },
    {
      "_id": "E5137"
    },
    {
      "_id": "E5199"
    },
    {
      "_id": "E4003"
    },
    {
      "_id": "E4694"
    },
    {
      "_id": "E4004"
    },
    {
      "_id": "E4005"
    },
    {
      "_id": "E5019"
    },
    {
      "_id": "E4843"
    }
  ],
  "resultCount": 2560,
  "pagedResultsCookie": null,
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

Query a specific provider

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/Epic/Provider/E4716"
{
  "_id": "E4716",
  "Specialty": "Family Medicine",
  "__UID__": "E4716",
  "Provider Type": "Physician",
  "__NAME__": "WELLHIVE, PROVIDER"
}
```

Query all user templates

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/Epic/UserTemplate?_queryId=query-all-ids"
{
  "result": [
    {
      "_id": "T00004"
    },
    {
      "_id": "T00024"
    },
    {
      "_id": "T00033"
    },
    {
      "_id": "T00038"
    },
    {
      "_id": "T00076"
    },
    {
      "_id": "T00077"
    },
    {
      "_id": "T00078"
    },
    {
      "_id": "T00088"
    },
    {
      "_id": "T00089"
    },
    {
      "_id": "T00090"
    },
    {
      "_id": "T1000601"
    },
    {
      "_id": "T1002020"
    },
    {
      "_id": "T1020101"
    },
    {
      "_id": "T1020102"
    },
    [ ... ]
    {
      "_id": "T8888001"
    },
    {
      "_id": "T8889901"
    },
    {
      "_id": "T9998001"
    }
  ],
  "resultCount": 431,
  "pagedResultsCookie": null,
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

Query a specific user template

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/Epic/UserTemplate/T8888001"
{
  "_id": "T8888001",
  "__UID__": "T8888001",
  "__NAME__": "RESEARCH ADMINISTRATOR TEMPLATE"
}
```

Query all user sub templates

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/Epic/UserSubTemplate?_queryId=query-all-ids"
{
  "result": [
    {
      "_id": "ST00007"
    },
    {
      "_id": "ST00030"
    },
    {
      "_id": "ST10200"
    },
    {
      "_id": "ST10201"
    },
    {
      "_id": "ST10202"
    },
    {
      "_id": "ST10203"
    },
    {
      "_id": "ST10204"
    },
    [ ... ]
    {
      "_id": "ST10401"
    },
    {
      "_id": "ST10402"
    },
    {
      "_id": "ST10700"
    },
    {
      "_id": "ST107001"
    },
    {
      "_id": "T5080002"
    },
    {
      "_id": "T99901"
    },
    {
      "_id": "TCVREPSUB"
    }
  ],
  "resultCount": 91,
  "pagedResultsCookie": null,
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

Query a specific user sub template

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/Epic/UserSubTemplate/T00007"
{
  "_id": "T00007",
  "__NAME__": "EXCEL MEDICAL",
  "__UID__": "T00007"
}
```

Query all In Basket classifications

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/Epic/InBasketClassifications?_queryId=query-all-ids"
{
  "result": [
    {
      "_id": "1"
    },
    {
      "_id": "2"
    },
    {
      "_id": "15"
    },
    {
      "_id": "29"
    },
    {
      "_id": "30"
    },
    {
      "_id": "31"
    },
    {
      "_id": "84"
    },
    {
      "_id": "85"
    },
    {
      "_id": "100"
    },
    {
      "_id": "140"
    },
    {
      "_id": "141"
    },
    {
      "_id": "212"
    }
  ],
  "resultCount": 12,
  "pagedResultsCookie": null,
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

Query a specific In Basket classification

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/Epic/InBasketClassifications/140"
{
  "_id": "140",
  "__NAME__": "Model AP Pt Clinical Msg Pool",
  "__UID__": "140"
}
```

Query all groups

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/Epic/__GROUP__?_queryId=query-all-ids"
{
  "result": [
    {
      "_id": "1"
    },
    {
      "_id": "2"
    },
    {
      "_id": "3"
    },
    {
      "_id": "4"
    },
    {
      "_id": "5"
    },
    {
      "_id": "6"
    },
    {
      "_id": "7"
    },
    {
      "_id": "1000"
    },
    {
      "_id": "1001"
    },
    {
      "_id": "1002"
    },
    {
      "_id": "1003"
    },
    {
      "_id": "1004"
    },
    {
      "_id": "1005"
    },
    {
      "_id": "1006"
    }
  ],
  "resultCount": 14,
  "pagedResultsCookie": null,
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

Query a specific group

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/Epic/__GROUP__/1000"
{
  "_id": "1000",
  "__NAME__": "Community",
  "__UID__": "1000",
  "Type": "Community",
  "Undefined3": "Customer~Epic Customer"
}
```

OpenICF Interfaces Implemented by the Epic Connector

The Epic Connector implements the following OpenICF interfaces. For additional details, see ICF 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.

Epic Connector Configuration

The Epic Connector has the following configurable properties:

Basic Configuration Properties

Property	Type	Default	Encrypted(1)	Required(2)
`clientId`	`String`	`null`		✓ Yes
Provides the Client ID to authorize the Epic APIs.				
`privateKey`	`GuardedString`	`null`	🔒 Yes	✓ Yes
Provides the Private key in pkcs8 format.				
`userName`	`String`	`null`		✓ Yes
Provides the Username required for Connection.				
`password`	`GuardedString`	`null`	🔒 Yes	✓ Yes
Provides the Password required for Connection.				
`userTemplatesFilePath`	`String`	`null`		✕ No
Provides the location of User Template file.				

Property	Type	Default	Encrypted[1]	Required[2]
`subTemplatesFilePath`	`String`	`null`		✕ No
Provides the location of User Subtemplate file.				
`inBasketFilePath`	`String`	`null`		✕ No
Provides the location of In Basket Classifications File.				
`groupsFilePath`	`String`	`null`		✕ No
Provides the location of Group File.				
`maxRecords`	`int`	`50`		✕ No
Provides the Maximum records for search operation.				
`maxConnections`	`Integer`	`10`		✕ No
Provides the Maximum connections.				
`connectionTimeout`	`int`	`600`		✕ No
Provides the Maximum Connection Timeout in seconds.				
`httpProxyHost`	`String`	`null`		✕ No
Provides the HTTP Proxy Host.				
`httpProxyPort`	`Integer`	`null`		✕ No
Provides the HTTP Proxy Port.				
`httpProxyUsername`	`String`	`null`		✕ No
Provides the HTTP Proxy Username.				
`httpProxyPassword`	`GuardedString`	`null`	🔒 Yes	✕ No
Provides the HTTP Proxy Password.				
`restEndpoint`	`String`	`https://vendorservices.epic.com/Interconnect-amcurprd-personnel-oauth/`		✕ No

Property	Type	Default	Encrypted[1]	Required[2]
The HTTP URL for the REST End point (https://myserver.com/service/).				
`soapEndpoint`	`String`	`https://vendorservices.epic.com/Interconnect-amcurprd-personnel-username/`		✕ No
The HTTP URL for the SOAP End point (https://myserver.com/service/).				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

Google Apps connector

The Google Apps connector is bundled with IDM and Advanced Identity Cloud. A sample connector configuration is also included with IDM. The Google Apps connector lets you interact with Google's web applications.

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

Google project requirements

Use of the Google Apps connector requires a properly configured Google project. The basic steps for configuring a Google project should be used as an outline, as the specific options, menus, and features may have changed.

1. Log in to the Google Apps Developers Console and update your existing project or create a new project.

2. Enable the following APIs for your project:

   - Admin SDK API
   - Enterprise License Manager API

   > **Important**
   >
   > Failure to enable the specified APIs results in the following depending on the Google Apps connector version:
   >
   > - For version 1.5.20.19 and earlier, connector requests hang indefinitely with no error message.
   > - For version 1.5.20.20 and later, the connector logs an error.

3. Set up an OAuth2 Client.

The Google Apps connector uses OAuth2 to authorize the connection to the Google service:

1. In the Google Apps Developers Console, select **Credentials** > **Create Credentials** > **OAuth client ID**.
2. Click **Configure Consent Screen**, select **Internal**, and click **Create**.
3. On the OAuth consent screen, enter an **Application name**; for example, `RCS`, and click **Save**.

   This name displays for all applications registered in this project.
4. Select **Credentials** > **Create Credentials** > **OAuth client ID** > **Web application**, and enter information in the following fields:

   Authorized JavaScript origins

   The URI that clients use to access your application. The default URI is `https://localhost:8443`.

   > **Note**
   >
   > The URI that you enter here must be the same you use to access RCS.

   Authorized redirect URIs

   The OAuth redirect URI, `https://localhost:8443/admin/oauth.html` by default.
5. Click **Create**.
6. On the **OAuth client created** pop-up, make a note of your **Client ID** and **Client Secret**.

4. Add RCS to the **Trusted Apps** list:

   1. Log in to the Google Admin Console.
   2. From the top left menu, select **Security** > **API Controls**.
   3. Select **MANAGE THIRD-PARTY APP ACCESS**, click **Change Access**, and change the RCS app settings to **Trusted**.

Install the Google Apps connector

> **Tip**
>
> To check for an Advanced Identity Cloud application for this connector, refer to:
>
> - Application management
> - App catalog

You can download any connector from Backstage, but some are included in the default deployment for Advanced Identity Cloud, IDM, or RCS. When using an included connector, you can skip installing it and move directly to configuration.

Connector included in default deployment

Connector	IDM	RCS
Google Apps	✓ Yes	✕ No

Download the connector .jar file from Backstage.

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

```
mv ~/Downloads/googleapps-connector-1.5.20.30.jar /path/to/openidm/connectors/
```

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

Configure the Google Apps connector

Create a connector configuration using the IDM admin UI:

1. From the navigation bar, click **Configure > Connectors**.
2. On the **Connectors** page, click **New Connector**.
3. On the **New Connector** page, type a **Connector Name**.
4. From the **Connector Type** drop-down list, select **Google Apps Connector - 1.5.20.30**.
5. Complete the **Base Connector Details**.

   > **Tip**
   >
   > For a list of all configuration properties, refer to Google Apps Connector Configuration

   A **Sign in with Google** page displays.
6. Log in.

   After you log in, Google requests access for the project.
7. Click **Allow**.

   > **Note**
   >
   > If you click **Deny**, you must return to the **Connector Configuration > Details** tab in the admin UI, and save your changes again.

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

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

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

This excerpt shows a sample Google Apps connector configuration. The default location of the connector .jar file is `openidm/connectors`. Therefore, the value of the `connectorHostRef` property must be `"#LOCAL"`:

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

The required configuration properties are as follows:

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

`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, refer to 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, refer to the following section of RFC 6749: Refresh Tokens.

For a sample Google Apps configuration that includes OAuth 2-based entries for `configurationProperties`, refer to Synchronize accounts with the Google Apps connector.

Test the Google Apps connector

You can test the configuration is correct by running the following command:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system/googleapps?_action=test"
{
  "name": "googleapps",
  "enabled": true,
  "config": "config/provisioner.openicf/googleapps",
  "connectorRef": {
    "bundleVersion": "[1.5.0.0,1.6.0.0)",
    "bundleName": "org.forgerock.openicf.connectors.googleapps-connector",
    "connectorName": "org.forgerock.openicf.connectors.googleapps.GoogleAppsConnector"
  },
  "displayName": "GoogleApps Connector",
  "objectTypes": [
    "Role",
    "OrgUnit",
    "LicenseAssignment",
    "__GROUP__",
    "__ALL__",
    "RoleAssignment",
    "Privilege",
    "__ACCOUNT__",
    "Member"
  ],
  "ok": true
}
```

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

Google Apps remote connector

If you want to run this connector outside of PingOne Advanced Identity Cloud or IDM, you can configure the Google Apps connector as a remote connector. Java Connectors installed remotely on a Java Connector Server function identically to those bundled locally within PingOne Advanced Identity Cloud or installed locally on IDM.

You can download the Google Apps connector from here.

Refer to Remote connectors for configuring the Google Apps remote connector.

Configure connection pooling

The Google Apps connector uses a non-poolable mechanism to manage connections. Learn more about the different pooling mechanisms in Connectors by pooling mechanism.

Use the Google Apps connector with a proxy server

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

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

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

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

Supported resource types

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

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

Supported resource types with the Google Apps connector

ICF Native Type	Google Resource Type	Naming Attribute
__ACCOUNT__	user	primaryEmail
__GROUP__	group	email
Member	member	{groupKey}/email
OrgUnit	orgUnit	{parentOrgUnitPath}/__NAME__
LicenseAssignment	licenseAssignment	{productId}/sku/{skuId}/user/{primaryEmail}
Role	role	{roleId}
RoleAssignment	roleassignment	{roleAssignmentId}
Privilege	privilege	

Supported user attributes

The Google Apps connector supports the following user resource attributes:

Attribute	Description
`addresses`	An array of addresses for the user account.
`agreedToTerms`	Whether the user has agreed to the Terms of Service.
`aliases`	An array of aliases for the user account.
`archived`	Whether the user account is archived.
`changePasswordAtNextLogin`	Whether the user must change their password at next login.
`creationTime`	The user creation time.
`customerId`	An ID used to identify the customer.
`customSchemas`	Define custom fields for user accounts.
`deletionTime`	The user deletion time.
`emails`	This attribute is managed using other attributes (`primaryEmail` , `__SECONDARY_EMAILS__` , `aliases` , `nonEditableAliases`).
`etag`	The ETag of the user account. Read-only attribute.
`externalIds`	An array of external IDs for the user account.
`hashFunction`	The hash function for the user's password.
`id`	The unique ID for the user. `id` can be used as a user request URL's `userKey` .
`ims`	An array of instant messenger accounts for the user.
`includeInGlobalAddressList`	Whether to include the user in the global address list.
`ipWhitelisted`	Whether the user's IP is allowlisted.
`isAdmin`	Whether the user is an admin.
`isDelegatedAdmin`	Whether the user is a delegated administrator.
`isEnforcedIn2Sv`	Whether the user is enforcing two-step verification.
`isEnrolledIn2Sv`	Whether the user is enrolled in two-step verification.
`isMailboxSetup`	Whether the user's mailbox is set up.
`kind`	The kind of user, typically `admin#directory#user` . Read-only attribute.

Attribute	Description
`languages`	An array of the user's language preferences.
`lastLoginTime`	The last time the user logged in.
`name`	An object containing the `fullName` , `givenName` and `familyName` attributes.
`nonEditableAliases`	An array of non-editable aliases for the user account.
`organizations`	An array of organizations for the user account.
`orgUnitPath`	The full path of the parent organization associated with the user. If the parent organization is top-level, it is represented as a forward slash (`/`).
`password`	The user's password.
`phones`	An array of phone numbers for the user account.
`primaryEmail`	The user's primary email address.
`recoveryEmail`	The user's recovery email address.
`recoveryPhone`	The user's recovery phone number.
`relations`	An array of relations for the user account.
`__SECONDARY_EMAIL__` (Deprecated)	Do not use this attribute. **Important** As of version 1.5.20.21, `__SECONDARY_EMAIL__` is deprecated. Use the newer attribute `__SECONDARY_EMAILS__`. These two attributes are mutually exclusive.
`__SECONDARY_EMAILS__`	An array of the user's email addresses, excluding their primary email address and all editable and non-editable aliases.
`suspended`	Whether the user is suspended.
`suspensionReason`	The reason the user account is suspended.
`thumbnailPhotoUrl`	The url to a user's thumbnail photo.

Functional limitations

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

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

Although the update is processed correctly, there is a significant delay from Google, and IDM sends its GET request to return the object before the update has taken effect. This behavior has no impact on the success of the update.

- The connector does not implement the ICF Sync operation so you cannot use the connector for liveSync of supported Google Apps resources to IDM managed objects.
- The connector does not implement the Authenticate operation so you cannot use the connector to perform pass-through authentication between IDM and a Google Apps domain. You can also not use this connector to perform password Change operations (as opposed to password Reset) because the connector cannot authenticate on behalf of the end user.
- Support for Filters when performing Search operations is limited to those attributes described in Supported search filters.
- Google Apps creates a new user alias each time the `primaryEmail` address associated with the User object is modified.
- For PATCH requests, a connector can potentially add, remove, or replace an attribute value. The Google Apps connector does not implement the add or remove operations, so a PATCH request always replaces the entire attribute value with the new value.

Supported search filters

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

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

Supported Operators and Filter Attributes With Google Apps Searches

Object Type	Operators	Attributes
__ACCOUNT__	And, Contains[1], StartsWith, Equals	name, email, givenName, familyName, orgUnitPath, im, externalId, address, addressPoBox, addressExtended, addressStreet, addressLocality, addressRegion, addressPostalCode, addressCountry, orgName, orgTitle, orgDepartment, orgDescription, orgCostCenter
__GROUP__	Contains[1], Equals	customer (Equals only), userKey (Equals only), _MEMBERS_ (Contains only)
Member	And, Equals	groupKey, memberKey (And only)
OrgUnit	StartsWith	orgUnitPath
LicenseAssignment	N/A	
Role	N/A	
RoleAssignment	N/A	

Object Type	Operators	Attributes
Privilege	N/A	

[1] "Contains" looks for a whole word match, in the given order. For example, an API request with `_queryFilter=givenName+co+'Ana'` matches users with `givenName` values of "Ana" and "Ana Lucia" but not "Anabelle". A multi-word query for `_queryFilter=givenName+co+'Ana Lucia'` would match values of "Ana Lucia Evans" and "Ana Lucia Ball" but not "Lucia Ana".

Product licenses

The Google Apps connector can query all available licenses and return details of individual product licenses.

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
"http://localhost:8080/openidm/system/googleapps/License?_queryFilter=true"
{
  "result": [
    {
      "_id": "Google-Apps/1010020027",
      "__NAME__": "Google Workspace Business Starter",
      "productId": "Google-Apps",
      "productName": "Google Workspace",
      "skuId": "1010020027",
      "skuName": "Google Workspace Business Starter"
    },
    {
      "_id": "Google-Drive-storage/Google-Drive-storage-20GB",
      "__NAME__": "Google-Drive-storage-20GB",
      "productId": "Google-Drive-storage",
      "productName": "Google-Drive-storage",
      "skuId": "Google-Drive-storage-20GB",
      "skuName": "Google-Drive-storage-20GB"
    }
  ],
  "resultCount": 2,
  "pagedResultsCookie": null,
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

To read the details of a license, perform a GET request on the endpoint:

```
system/googleapps/License/{PRODUCT_ID}/{SKU_ID}
```

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
"http://localhost:8080/openidm/system/googleapps/License/Google-Drive-storage/Google-Drive-storage-20GB"
{
  "_id": "Google-Drive-storage/Google-Drive-storage-20GB",
  "__NAME__": "Google-Drive-storage-20GB",
  "productId": "Google-Drive-storage",
  "productName": "Google-Drive-storage",
  "skuId": "Google-Drive-storage-20GB",
  "skuName": "Google-Drive-storage-20GB"
}
```

OpenICF Interfaces Implemented by the GoogleApps Connector

The GoogleApps Connector implements the following OpenICF interfaces. For additional details, see ICF interfaces:

Create

Creates an object and its `uid`.

Delete

Deletes an object, referenced by its `uid`.

Schema

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

Script on Connector

Enables an application to run a script in the context of the connector.

Any script that runs on the connector has the following characteristics:

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

Search

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

Test

Tests the connector configuration.

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

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

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

Update

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

GoogleApps Connector Configuration

The GoogleApps Connector has the following configurable properties:

Basic Configuration Properties

Property	Type	Default	Encrypted[1]	Required[2]
`domain`	`String`	`null`		✓ Yes
Internet domain name. See https://support.google.com/a/answer/177483?hl=en.				
`clientId`	`String`	`null`		✓ Yes
Client identifier issued to the client during the registration process.				
`clientSecret`	`GuardedString`	`null`	🔒 Yes	✓ Yes
Client secret issued to the client during the registration process.				
`refreshToken`	`GuardedString`	`null`	🔒 Yes	✓ Yes
The refresh token allows you to get a new access token that is good for another hour. Refresh tokens never expire, they can only be revoked by the user or programatically by your app.				
`proxyHost`	`String`	`null`		✓ Yes
Defines an HTTP proxy host to use with the connection (example: "myproxy.home.com").				
`proxyPort`	`int`	`8080`		✓ Yes
Defines an HTTP proxy port to use with the connection (defaults to 8080).				
`validateCertificate`	`boolean`	`true`		✓ Yes

Property	Type	Default	Encrypted[1]	Required[2]
Specifies whether the validation of the server certificate from the locally stored truststore is enabled. (defaults to true).				
`usersMaxResults`	`int`	`100`		✕ No
Maximum number of Users to return. Acceptable values are 1 to 500, inclusive.				
`groupsMaxResults`	`int`	`200`		✕ No
Maximum number of Groups to return. Acceptable values are 1 to 200, inclusive.				
`membersMaxResults`	`int`	`200`		✕ No
Maximum number of Members to return. Acceptable values are greater than 1.				
`listProductMaxResults`	`long`	`100`		✕ No
Maximum number of Licenses to return. Acceptable values are 1 to 1000, inclusive.				
`listProductAndSkuMaxResults`	`long`	`100`		✕ No
Maximum number of Licenses to return. Acceptable values are 1 to 1000, inclusive.				

`availableLicenses`	`String[]`	`['101005/1010050001', '101001/1010010001', '101031/1010310010', '101034/1010340002', '101038/1010380002', '101034/1010340001', '101038/1010380003', '101034/1010340004', '101034/1010340003', '101034/1010340006', 'Google-Apps/Google-Apps-For-Business', '101034/1010340005', 'Google-Vault/Google-Vault', 'Google-Apps/1010020031', 'Google-Apps/1010020030', 'Google-Apps/1010060003', 'Google-Apps/1010060005', 'Google-Apps/Google-Apps-Unlimited', 'Google-Apps/1010020029', 'Google-Apps/Google-Apps-Lite', '101031/1010310003', '101033/1010330002', '101033/1010330004', 'Google-`		✕ No

Property	Type	Default	Encrypted[1]	Required[2]
		`Apps/Google-Apps-For-Education', '101031/1010310002', '101033/1010330003', 'Google-Apps/1010020026', '101031/1010310007', 'Google-Apps/1010020025', '101031/1010310008', 'Google-Apps/1010020028', 'Google-Apps/Google-Apps-For-Postini', '101031/1010310005', 'Google-Apps/1010020027', '101031/1010310006', '101031/1010310009', 'Google-Vault/Google-Vault-Former-Employee', '101038/1010370001', 'Google-Apps/1010020020', 'Google-Apps/1010060001']`		
All Google Licenses that will be queried when requesting licenses assigned to a user. The format of the license is ProductId/SkuId (e.g. Google-Apps/101002002).				
`roleMaxResults`	`int`	`100`		✕ No
Maximum number of Licenses to return. Acceptable values are 1 to 100, inclusive.				
`roleAssignmentMaxResults`	`int`	`100`		✕ No

Property	Type	Default	Encrypted[1]	Required[2]
Maximum number of Licenses to return. Acceptable values are 1 to 100, inclusive.				
`passwordHashAlgorithm`	`String`	`null`		✕ No
Indicates the Crypt(3) hash algorithm that the Identity system should use to hash the password for transport. Only supports `crypt` functions and the algorithms: `DEScrypt`, `MD5crypt`, `SHA-256crypt`, and `SHA-512crypt`. A blank value indicates that the system will not hash passwords.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

Google Cloud Platform connector

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

Before you start

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

Domain name

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

Private key

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

Service account

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

Admin user

The GCP administrator username.

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

Install the GCP connector

> **Tip**
>
> To check for an Advanced Identity Cloud application for this connector, refer to:
>
> - Application management
> - App catalog

You can download any connector from Backstage, but some are included in the default deployment for Advanced Identity Cloud, IDM, or RCS. When using an included connector, you can skip installing it and move directly to configuration.

Connector included in default deployment

Connector	IDM	RCS
Google Cloud Platform (GCP)	✕ No	✕ No

Download the connector .jar file from Backstage.

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

```
mv ~/Downloads/gcp-connector-1.5.20.29.jar /path/to/openidm/connectors/
```

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

Configure the GCP connector

Create a connector configuration using the IDM admin UI:

1. From the navigation bar, click **Configure > Connectors**.
2. On the **Connectors** page, click **New Connector**.
3. On the **New Connector** page, type a **Connector Name**.
4. From the **Connector Type** drop-down list, select **GCP Connector - 1.5.20.29**.
5. Complete the **Base Connector Details**.

   > **Tip**
   >
   > For a list of all configuration properties, refer to GCP Connector Configuration

6. Click **Save**.

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

Refer to this procedure to create a connector configuration over REST.

Test the GCP connector

You can test the configuration is correct by running the following command:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system/gcp?_action=test"
{
  "name": "gcp",
  "enabled": true,
  "config": "config/provisioner.openicf/gcp",
  "connectorRef": {
    "bundleVersion": "[1.5.0.0,1.6.0.0)",
    "bundleName": "org.forgerock.openicf.connectors.gcp-connector",
    "connectorName": "org.forgerock.openicf.connectors.gcp.GcpConnector"
  },
  "displayName": "GCP Connector",
  "objectTypes": [
    "__ACCOUNT__",
    "__ALL__"
  ],
  "ok": true
}
```

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

GCP remote connector

If you want to run this connector outside of PingOne Advanced Identity Cloud or IDM, you can configure the GCP connector as a remote connector. Java Connectors installed remotely on a Java Connector Server function identically to those bundled locally within PingOne Advanced Identity Cloud or installed locally on IDM.

You can download the GCP connector from here.

Refer to Remote connectors for configuring the GCP remote connector.

Configure connection pooling

The GCP connector supports HTTP pooling, which can substantially improve the performance of the connector. Learn more about the basic connection pooling configuration and different pooling mechanisms described in Connection pooling configuration.

Use the GCP connector

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

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

<table>
<tr><th>Attribute</th><th>Description</th></tr>
<tr><td>

`phones`

</td><td>

A list of phone numbers associated with the user account. For example:

```
"phones": [
    {
        "customType": "custom",
        "type": "custom",
        "value": "+1 888 555 2312",
        "primary": false
    }
],
```

</td></tr>
<tr><td>

`relations`

</td><td>

A list of the user’s relationships to other users. For example:

```
"relations": [
    {
        "customType": "Cousin",
        "type": "custom",
        "value": "Bob Jensen"
    }
]
```

</td></tr>
<tr><td>

`externalIds`

</td><td>

A list of external IDs for the user, such as employee or network IDs. For example:

```
"externalIds": [
    {
        "customType": "employee",
        "type": "custom",
        "value": "12345"
    }
],
```

</td></tr>
</table>

For a full list of attributes on GCP user accounts, refer to the GCP documentation.

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

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

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

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

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

- `primaryEmail`
- `__PASSWORD__`
- `givenName`
- `familyName`
- `organizations`
- `addresses`
- `emails`
- `externalIds`
- `relations`
- `phones`

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

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

Note

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

The following example queries all GCP users:

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

The following command queries a specific user by their ID:

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

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "if-Match:*" \
--request PATCH \
--data '[{
  "operation": "add",
  "field": "__PASSWORD__",
  "value": "Passw0rd@123!"
}]' \
"http://localhost:8080/openidm/system/gcp/__ACCOUNT__/115637914640083360831"
{
  "_id": "115637914640083360831",
  "givenName": "Barbara",
  "__UID__": "115637914640083360831",
  "phones": [
    {
      "value": "+1 888 555 2312",
      "type": "mobile"
    }
  ],
  "__NAME__": "bjensen@example.com",
  "familyName": "Jensen",
  "__ENABLE__": false,
  "emails": [
    {
      "address": "bjensen@example.com",
      "primary": true
    },
    {
      "address": "bjensen@example.com.test-google-a.com"
    }
  ]
}
```

Note

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

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

The following example deletes a GCP account:

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

OpenICF Interfaces Implemented by the GCP Connector

The GCP Connector implements the following OpenICF interfaces. For additional details, see ICF interfaces:

Create

Creates an object and its `uid`.

Delete

Deletes an object, referenced by its `uid`.

Schema

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

Script on Connector

Enables an application to run a script in the context of the connector.

Any script that runs on the connector has the following characteristics:

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

Search

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

Test

Tests the connector configuration.

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

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

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

Update

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

GCP Connector Configuration

The GCP Connector has the following configurable properties:

Configuration properties

Property	Type	Default	Encrypted(1)	Required(2)
`domainName`	`String`	`null`		✓ Yes
Provides the domain name for GCP.				
`privateKey`	`GuardedString`	`null`	🔒 Yes	✓ Yes
Provides private key to authenticate GCP.				
`serviceAccount`	`String`	`null`		✓ Yes
Provides service account for fetching users from GCP.				

Property	Type	Default	Encrypted[1]	Required[2]
`adminUser`	`String`	`null`		✓ Yes
Provides admin user for fetching users from GCP.				
`maxResults`	`int`	`50`		✕ No
Provides user max results for fetching users from GCP.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

Basic Configuration Properties

Property	Type	Default	Encrypted[1]	Required[2]
`httpProxyHost`	`String`	`null`		✕ No
Provides the HTTP proxy host.				
`httpProxyPort`	`Integer`	`null`		✕ No
Provides the HTTP proxy port.				
`httpProxyUsername`	`String`	`null`		✕ No
Provides the HTTP proxy username.				
`httpProxyPassword`	`GuardedString`	`null`	🔒 Yes	✕ No
Provides the HTTP Proxy password.				
`connectionTimeout`	`Integer`	`300`		✕ No
Provides the maximum connection timeout in seconds.				
`maximumConnections`	`Integer`	`10`		✕ No
Provides the maximum connections.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

Groovy connector toolkit

The generic Groovy connector toolkit runs a Groovy script for any ICF operation, such as search, update, create, and others, on any external resource.

The Groovy connector toolkit is not a complete connector in the traditional sense. Instead, it is a framework where you must write your own Groovy scripts to address your implementation requirements.

Configure scripted Groovy connectors

You cannot configure a scripted Groovy connector through the UI. Configure the connector over REST, as described in Configure Connectors Over REST.

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

Samples describes a number of scripted connector implementations. The scripts provided with these samples demonstrate how the Groovy connector toolkit can be used. These scripts cannot be used as is in your deployment but are a good starting point to base your customization. For information about writing your own scripts, refer to Scripted connectors with Groovy.

Groovy connection pooling

The Groovy connector toolkit doesn't use a specific pooling mechanism but lets you define your own pooling type: poolable or non-poolable. Learn more in Selecting a scripted connector implementation.

Use custom properties in scripts

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

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

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

Debug Groovy scripts

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

For instructions on how to use this facility, refer to the KnowledgeBase article How do I add logging to Groovy scripts in IDM.

Run scripts through the connector

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

runScriptOnConnector

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

configuration

A handler to the connector’s configuration object.

options

A handler to the Operation Options.

operation

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

log

A handler to the connector’s log.

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

```
"systemActions" : [
    {
        "scriptId" : "MyScript",
        "actions" : [
            {
                "systemType" : ".*ScriptedConnector",
                "actionType" : "groovy",
                "actionFile" : "path/to/<script-name>.groovy"
            }
        ]
    }
]
```

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

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

Note

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

Running `MyScript` will return:

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

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

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

You can also call it through the IDM script engine. Note that the system can accept arbitrary parameters, as demonstrated here:

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

runScriptOnResource

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

Define these properties in your provisioner file as follows:

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

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

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

Script compilation and caching

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

Implemented interfaces

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

OpenICF Interfaces Implemented by the Scripted Groovy Connector

The Scripted Groovy Connector implements the following OpenICF interfaces. For additional details, see ICF interfaces:

Authenticate

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

Create

Creates an object and its `uid`.

Delete

Deletes an object, referenced by its `uid`.

Resolve Username

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

Schema

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

Script on Connector

Enables an application to run a script in the context of the connector.

Any script that runs on the connector has the following characteristics:

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

Script on Resource

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

Search

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

Sync

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

Test

Tests the connector configuration.

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

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

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

Update

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

OpenICF Interfaces Implemented by the Scripted Poolable Groovy Connector

The Scripted Poolable Groovy Connector implements the following OpenICF interfaces. For additional details, see ICF interfaces:

Authenticate

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

Create

Creates an object and its `uid`.

Delete

Deletes an object, referenced by its `uid`.

Resolve Username

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

Schema

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

Script on Connector

Enables an application to run a script in the context of the connector.

Any script that runs on the connector has the following characteristics:

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

Script on Resource

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

Search

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

Sync

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

Test

Tests the connector configuration.

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

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

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

Update

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

Configuration properties

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

Scripted Groovy Connector Configuration

The Scripted Groovy Connector has the following configurable properties:

Groovy Engine configuration

Property	Type	Default	Encrypted[(1)]	Required[(2)]
`scriptRoots`	`String[]`	`null`		✓ Yes
The root folder to load the scripts from. If the value is null or empty the classpath value is used.				
`classpath`	`String[]`	`[]`		✕ No
Classpath for use during compilation.				
`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.				
`minimumRecompilationInterval`	`int`	`100`		✕ No
Sets the minimum of time after a script can be recompiled.				
`recompileGroovySource`	`boolean`	`false`		✕ No
If set to true recompilation is enabled.				

Property	Type	Default	Encrypted[1]	Required[2]
`scriptBaseClass`	`String`	`null`		✕ No
Base class name for scripts (must derive from Script).				
`scriptExtensions`	`String[]`	`['groovy']`		✕ No
Gets the extensions used to find groovy files.				
`sourceEncoding`	`String`	`UTF-8`		✕ No
Encoding for source files.				
`targetDirectory`	`File`	`null`		✕ No
Directory into which to write classes.				
`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.				
`verbose`	`boolean`	`false`		✕ No
If true, the compiler should produce action information.				
`warningLevel`	`int`	`1`		✕ No
Warning Level of the compiler.				
`customConfiguration`	`String`	`null`		✕ No
Custom Configuration script for Groovy ConfigSlurper.				
`customSensitiveConfiguration`	`GuardedString`	`null`	🔒 Yes	✕ No
Custom Sensitive Configuration script for Groovy ConfigSlurper.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

Operation Script Files

Property	Type	Default	Encrypted[1]	Required[2]
`authenticateScriptFileName`	`String`	`null`		• Authenticate

Property	Type	Default	Encrypted[(1)]	Required[(2)]
The name of the file used to perform the AUTHENTICATE operation.				
`createScriptFileName`	`String`	`null`		• Create
The name of the file used to perform the CREATE operation.				
`customizerScriptFileName`	`String`	`null`		✕ No
The script used to customize some function of the connector. Read the documentation for more details.				
`deleteScriptFileName`	`String`	`null`		• Delete
The name of the file used to perform the DELETE operation.				
`resolveUsernameScriptFileName`	`String`	`null`		• Resolve Username
The name of the file used to perform the RESOLVE_USERNAME operation.				
`schemaScriptFileName`	`String`	`null`		• Schema
The name of the file used to perform the SCHEMA operation.				
`scriptOnResourceScriptFileName`	`String`	`null`		• Script on Resource
The name of the file used to perform the RUNSCRIPTONRESOURCE operation.				
`searchScriptFileName`	`String`	`null`		• Read • Search
The name of the file used to perform the SEARCH operation.				
`syncScriptFileName`	`String`	`null`		• Sync
The name of the file used to perform the SYNC operation.				

Property	Type	Default	Encrypted[1]	Required[2]
`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.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

Scripted Poolable Groovy Connector Configuration

The Scripted Poolable Groovy Connector has the following configurable properties:

Groovy Engine configuration

Property	Type	Default	Encrypted[1]	Required[2]
`scriptRoots`	`String[]`	`null`		✓ Yes
The root folder to load the scripts from. If the value is null or empty the classpath value is used.				
`classpath`	`String[]`	`[]`		✕ No
Classpath for use during compilation.				
`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.				
`minimumRecompilationInterval`	`int`	`100`		✕ No
Sets the minimum of time after a script can be recompiled.				
`recompileGroovySource`	`boolean`	`false`		✕ No
If set to true recompilation is enabled.				

Property	Type	Default	Encrypted[1]	Required[2]
`scriptBaseClass`	`String`	`null`		✕ No
Base class name for scripts (must derive from Script).				
`scriptExtensions`	`String[]`	`['groovy']`		✕ No
Gets the extensions used to find groovy files.				
`sourceEncoding`	`String`	`UTF-8`		✕ No
Encoding for source files.				
`targetDirectory`	`File`	`null`		✕ No
Directory into which to write classes.				
`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.				
`verbose`	`boolean`	`false`		✕ No
If true, the compiler should produce action information.				
`warningLevel`	`int`	`1`		✕ No
Warning Level of the compiler.				
`customConfiguration`	`String`	`null`		✕ No
Custom Configuration script for Groovy ConfigSlurper.				
`customSensitiveConfiguration`	`GuardedString`	`null`	🔒 Yes	✕ No
Custom Sensitive Configuration script for Groovy ConfigSlurper.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

Operation Script Files

Property	Type	Default	Encrypted[1]	Required[2]
`authenticateScriptFileName`	`String`	`null`		• Authenticate

Property	Type	Default	Encrypted(1)	Required(2)
The name of the file used to perform the AUTHENTICATE operation.				
`createScriptFileName`	`String`	`null`		• Create
The name of the file used to perform the CREATE operation.				
`customizerScriptFileName`	`String`	`null`		✕ No
The script used to customize some function of the connector. Read the documentation for more details.				
`deleteScriptFileName`	`String`	`null`		• Delete
The name of the file used to perform the DELETE operation.				
`resolveUsernameScriptFileName`	`String`	`null`		• Resolve Username
The name of the file used to perform the RESOLVE_USERNAME operation.				
`schemaScriptFileName`	`String`	`null`		• Schema
The name of the file used to perform the SCHEMA operation.				
`scriptOnResourceScriptFileName`	`String`	`null`		• Script on Resource
The name of the file used to perform the RUNSCRIPTONRESOURCE operation.				
`searchScriptFileName`	`String`	`null`		• Read • Search
The name of the file used to perform the SEARCH operation.				
`syncScriptFileName`	`String`	`null`		• Sync
The name of the file used to perform the SYNC operation.				

Property	Type	Default	Encrypted[1]	Required[2]
`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.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

HubSpot connector

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

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

For a complete example that includes the configuration required to synchronize users with this connector, refer to Synchronize data between IDM and HubSpot.

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

Install the HubSpot connector

> **Tip**
>
> To check for an Advanced Identity Cloud application for this connector, refer to:
>
> - Application management
> - App catalog

You can download any connector from Backstage, but some are included in the default deployment for Advanced Identity Cloud, IDM, or RCS. When using an included connector, you can skip installing it and move directly to configuration.

Connector included in default deployment

Connector	IDM	RCS
HubSpot	✗ No	✗ No

Download the connector .jar file from Backstage.

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

  ```
  mv ~/Downloads/hubspot-connector-1.5.20.30.jar /path/to/openidm/connectors/
  ```

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

Configure the HubSpot connector

Create a connector configuration using the IDM admin UI:

1. From the navigation bar, click **Configure** > **Connectors**.
2. On the **Connectors** page, click **New Connector**.
3. On the **New Connector** page, type a **Connector Name**.
4. From the **Connector Type** drop-down list, select **HubSpot Connector - 1.5.20.30**.
5. Complete the **Base Connector Details**.

   > **Tip**
   >
   > For a list of all configuration properties, refer to HubSpot Connector Configuration

6. Click **Save**.

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

Refer to this procedure to create a connector configuration over REST.

Alternatively, configure the connector with a configuration file. IDM provides a sample connector configuration file in the `/path/to/openidm/samples/example-configurations/provisioners` directory. Copy this sample file (`provisioner.openicf-hubspot.json`) to your project's `conf` directory.

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

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

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

Test the HubSpot connector

Test that the configuration is correct by running the following command:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system?_action=test"
[
  {
    "name": "hubspot",
    "enabled": true,
    "config": "config/provisioner.openicf/hubspot",
    "connectorRef": {
      "bundleVersion": "[1.5.0.0,1.6.0.0)",
      "bundleName": "org.forgerock.openicf.connectors.hubspot-connector",
      "connectorName": "org.forgerock.openicf.connectors.hubspot.HubspotConnector"
    },
    "displayName": "Hubspot Connector",
    "objectTypes": [
      "company",
      "contactProperties",
      "__ALL__",
      "companyProperties",
      "contact"
    ],
    "ok": true
  }
]
```

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

HubSpot remote connector

If you want to run this connector outside of PingOne Advanced Identity Cloud or IDM, you can configure the HubSpot connector as a remote connector. Java Connectors installed remotely on a Java Connector Server function identically to those bundled locally within PingOne Advanced Identity Cloud or installed locally on IDM.

You can download the HubSpot connector from here.

Refer to Remote connectors for configuring the HubSpot remote connector.

Configure connection pooling

The HubSpot connector supports HTTP pooling, which can substantially improve the performance of the connector. Learn more about the basic connection pooling configuration and different pooling mechanisms described in Connection pooling configuration.

Implementation specifics

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

Using the HubSpot connector With a proxy server

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

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

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

OpenICF Interfaces Implemented by the Hubspot Connector

The Hubspot Connector implements the following OpenICF interfaces. For additional details, see ICF interfaces:

Create

Creates an object and its `uid`.

Delete

Deletes an object, referenced by its `uid`.

Schema

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

Script on Connector

Enables an application to run a script in the context of the connector.

Any script that runs on the connector has the following characteristics:

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

Search

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

Test

Tests the connector configuration.

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

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

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

Update

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

Hubspot Connector Configuration

The Hubspot Connector has the following configurable properties:

Basic Configuration Properties

Property	Type	Default	Encrypted[1]	Required[2]
`clientId`	`String`	`null`		✓ Yes
Client ID of the OAuth application in Hubspot.				
`clientSecret`	`GuardedString`	`null`	🔒 Yes	✓ Yes
Client Secret for the preceding Client ID.				
`refreshToken`	`GuardedString`	`null`	🔒 Yes	✓ Yes
Refresh token for application in Hubspot.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

Advanced Connection Properties

Property	Type	Default	Encrypted[1]	Required[2]
`acceptSelfSignedCertificates`	`boolean`	`false`		✓ Yes
Specifies whether the HubSpot server should accept self-signed certificates. Defaults to false.				
`readSchema`	`Boolean`	`false`		✓ Yes
If false, the Hubspot connector provides a default schema for Hubspot contacts and companies.				
`disableHostNameVerifier`	`boolean`	`false`		✓ Yes
If hostname verification is disabled, the HubSpot server accepts connections from any host. Defaults to false.				
`maximumConnections`	`Integer`	`10`		✓ Yes
Maximum number of simultaneous connections to HubSpot.				
`permitsPerSecond`	`Integer`	`10`		✓ Yes
Number of Api calls to be made per second.				
`httpProxyHost`	`String`	`null`		✓ Yes
Specifies the Hostname if an HTTP proxy is used between the connector and HubSpot. Defaults to null.				
`httpProxyPort`	`Integer`	`null`		✓ Yes
Specifies the Port number if an HTTP proxy is used between the connector and HubSpot . Defaults to null.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

IBM RACF connector

IBM Resource Access Control Facility (RACF) is an access control system for IBM mainframes running z/OS. The RACF connector lets you manage and synchronize accounts between RACF and IDM managed user objects. A RACF administrator account is required for this connector to work.

Before you start

> **Important**
>
> The User ID and Password combination you use for connector setup must have access to the `/rseapi/api/v1/tso` RACF endpoint and be able to perform the following RACF commands:
>
> - `SEARCH`
> - `LISTUSER`
> - `ADDUSER`
> - `ALTUSER`
> - `DELUSER`
> - `LISTGRP`
> - `ADDGROUP`
> - `ALTGROUP`
> - `DELGROUP`

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

Host name

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

Port

The port RACF is configured to use.

User ID

The RACF administrator user ID.

Password

The password for the RACF administrator account.

Segments

A list of RACF user profile segments that are supported. Refer to RACF segments and attributes for a list of available segments.

Accept self-signed certificates

A boolean determining whether RACF is configured to allow self-signed certificates. This should usually be `false` in production environments, but may be `true` during development.

Client certificate alias

Alias name for the client certificate.

Client certificate password

Password for the client certificate.

Install a signed certificate

You can install a signed certificate to access the ZD&T Enterprise Edition web server. To generate your own pkcs12 keystore (`zdtkey.p12`) containing the certificate and add the encrypted password to the `server.env` file, do the following:

1. Check your installed web server's installation directory. For example, `/opt/ibm/zDT` is the default installation directory, but you can specify your own installation directory during the installation process.
2. Generate `zdtkey.p12` and place it in the `/path/to/zDT/zDTServer/resources/security` .

   ```
   openssl pkcs12 -export -out zdtkey.p12 -inkey cert.key -in cert.crt -password pass:$passcert
   ```

3. Modify the encrypted key store password:
   1. Get the value of `wlp.password.encryption.key` in the `/path/to/zDT/zDTServer/resources/security/bootstrap.properties` .
   2. Run the following command where you installed the web server:

      ```
      /path/to/zDT/Liberty/bin/securityUtility encode --encoding=aes --key=<value_of_wlp.password.encryption.
      key> <password>
      ```

      > **Note**
      >
      > To run `securityUtility` successfully, the Java path must be set. For more information, refer to Java requirements.

   3. Modify the `/path/to/zDT/Liberty/usr/servers/zDTServer/server.env` file with your encoded password value. For example:

      ```
      POSTGRES_SERVER=xxx
      POSTGRES_PORT=5432
      ...
      KEYSTORE_PASSWORD={aes}AG6i...JiS0p
      ```

Install the RACF connector

> **Tip**
>
> To check for an Advanced Identity Cloud application for this connector, refer to:
>
> - Application management
> - App catalog

You can download any connector from Backstage, but some are included in the default deployment for Advanced Identity Cloud, IDM, or RCS. When using an included connector, you can skip installing it and move directly to configuration.

Connector included in default deployment

Connector	IDM	RCS
IBM RACF	✕ No	✓ Yes

Download the connector .jar file from Backstage.

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

```
mv ~/Downloads/racf-connector-1.5.20.26.jar /path/to/openidm/connectors/
```

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

Configure the RACF connector

Create a connector configuration using the IDM admin UI:

1. From the navigation bar, click **Configure > Connectors**.
2. On the **Connectors** page, click **New Connector**.
3. On the **New Connector** page, type a **Connector Name**.
4. From the **Connector Type** drop-down list, select **RACF Connector - 1.5.20.26**.
5. Complete the **Base Connector Details**.

> **Tip**
>
> For a list of all configuration properties, refer to RACF Connector Configuration

6. Click **Save**.

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

Refer to this procedure to create a connector configuration over REST.

Test the RACF connector

You can test the configuration is correct by running the following command:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system/racf?_action=test"
{
  "name": "racf",
  "enabled": true,
  "config": "config/provisioner.openicf/racf",
  "connectorRef": {
    "bundleVersion": "[1.5.0.0,1.6.0.0)",
    "bundleName": "org.forgerock.openicf.connectors.racf-connector",
    "connectorName": "org.forgerock.openicf.connectors.racf.RacfConnector"
  },
  "displayName": "RACF Connector",
  "objectTypes": [
    "__ACCOUNT__",
    "__ALL__",
    "__GROUP__"
  ],
  "ok": true
}
```

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

RACF remote connector

If you want to run this connector outside of PingOne Advanced Identity Cloud or IDM, you can configure the RACF connector as a remote connector. Java Connectors installed remotely on a Java Connector Server function identically to those bundled locally within PingOne Advanced Identity Cloud or installed locally on IDM.

You can download the RACF connector from here.

Refer to Remote connectors for configuring the RACF remote connector.

Configure connection pooling

The RACF connector supports HTTP pooling, which can substantially improve the performance of the connector. Learn more about the basic connection pooling configuration and different pooling mechanisms described in Connection pooling configuration.

RACF segments and attributes

The following tables list available attributes by segment. Attributes listed in the `BASE` segment are available by default. To use any other attributes, include the segment name in the list of segments in the RACF connector configuration.

User accounts support create, update, query, and delete actions. Groups only support query actions.

Account attributes

The following attributes are available to the `__ACCOUNT__` resource object:

Attribute	Description
`userId`	The user's ID. Required.
`__NAME__`	The user's system name. Must match `userID`. Required.
`NAME`	The user's name.
`OWNER`	Owner of the user's profile.
`DFLTGRP`	User's default group.
`AUTHORITY`	User's authority in the default group.
`__PASSWORD__`	User's password.
`PHRASE`	Optional password phrase.
`REVOKE`	Expiration date for the user's system access.
`RESUME`	Date the user's system access is restored.
`WHEN`	Days of the week and hours of the day the user has access to the system.
`CLAUTH`	Classes in which the user can define profiles.
`MODEL`	Name of the data model profile used when creating new data profiles (either generic or discrete).
`GROUP`	The group the user belongs to.
`SECLABEL`	The user's default security label.
`GRPACC`	Whether other group members have access to any other group set the user protects.
`RESTRICTED`	Indicates that when checking global access, the account will not be used to allow access to a resource.
`AUDITOR`	Gives the user the system-wide auditor attribute.
`OPERATIONS`	Gives the user the system-wide operations attribute.
`SPECIAL`	Gives the user the system-wide special attribute.
`ADSP`	Indicates all permanent data sets this user creates should be discrete profiles in RACF.

Attribute	Description
CICS_OPCLASS	The classes the user is assigned in CICS. Determines which basic mapping support (BMS) messages are routed to the user. Represented as a number ranging from 01 to 24 .
CICS_OPIDENT	A 1-3 character identification of the user for use by BMS.
CICS_OPPRTY	The number (0 to 255) that represents the priority of the user.
CICS_RSLKEY	The resource security level (RSL) keys assigned to the user.
CICS_TIMEOUT	The time in hours and minutes (either HMM or HHMM format) that the operator is allowed to be idle before being signed out.
CICS_TSLKEY	The transaction security level (TLS) keys assigned to the user.
CICS_XRFSOFF	Indicates whether the user should be signed out when an XRF takeover occurs.

Attribute	Description
DCE_AUTOLOGIN	Single Sign On (SSO) processing. Either YES or NO .
DCE_DCENAME	The user's DCE principal name.
DCE_HOMECELL	The user's DCE home cell.
DCE_HOMEUUID	Defines the mapping between the user's RACF user ID and the corresponding DCE principal UUID.
DCE_UUID	The user's principal DCE UUID.

Attribute	Description
DFP_DATAAPPL	The user's DFP data application identifier.
DFP_DATACLAS	The user's default data class for attributes used during allocation of any new data sets.
DFP_MGMTCLAS	The user's default management class for attributes used in managing a data set after it is allocated.
DFP_STORCLAS	The user's default storage class for logical storage attributes.

Attribute	Description
KERB_ENCRYPT	The user's encryption key types. Available values include: DES , DES3 , DESD , AES128 , and AES256 .

Attribute	Description
`KERB_KERBNAME`	The user's local principal name. The value specified must be unique.
`KERB_MAXTKTLFE`	The maximum Kerberos ticket life specified in seconds. Note that `0` is not a valid value.

Attribute	Description
`LANGUAGE_PRIMARY`	The user's primary language.
`LANGUAGE_SECONDARY`	The user's secondary language.

Attribute	Description
`LNOTES_SNAME`	The user's short name for use with Lotus Notes in z/OS.

Attribute	Description
`NDS_UNAME`	The user's name for use with Novell Directory Services.

Attribute	Description
`NETVIEW_CONSNAME`	Master Console Station (MCS) console identifier. The console name value is an identifier 1-8 characters in length whose validity is checked by MVS processing.
`NETVIEW_CTL`	Specifies whether a security check is performed for this user. Either `GLOBAL`, `GENERAL`, or `SPECIFIC`.
`NETVIEW_DOMAINS`	The domain identifier for any domains where the user can start a cross-domain session.
`NETVIEW_IC`	The initial command or list of commands to be executed by NetView when the user logs in.
`NETVIEW_MSGRECVR`	Indicates whether the user can receive unsolicited messages.
`NETVIEW_NGMFADMN`	Indicates whether the user can use the NetView graphic monitor facility.
`NETVIEW_OPCLASS`	NetView scope classes the user has authority with. The class value is a number from `1` to `2040`.

Attribute	Description
`OMVS_ASSIZEMAX`	The user's z/OS maximum address space size.
`OMVS_CPUTIMEMAX`	The user's z/OS maximum CPU time allowed.

Attribute	Description
`OMVS_FILEPROCMAX`	The user's z/OS maximum number of files allowed per process.
`OMVS_HOME`	The user's z/OS home directory path.
`OMVS_MEMLIMIT`	The user's z/OS non-shared memory size limit.
`OMVS_MMAPAREAMAX`	The user's z/OS maximum memory map size.
`OMVS_PROCUSERMAX`	The user's maximum number of processes per UID in z/OS.
`OMVS_PROGRAM`	The user's z/OS path name, such as a default shell program.
`OMVS_SHMEMMAX`	The user's z/OS maximum shared memory size.
`OMVS_THREADSMAX`	The user's z/OS maximum number of threads per process.
`OMVS_UID`	The user's z/OS user ID.

Attribute	Description
`OPERPARM_ALTGRP`	Alternative console group used for recovery.
`OPERPARM_AUTH`	The user's command authority.
`OPERPARM_CMDSYS`	Name of the system to which the user is connected for command processing.
`OPERPARM_DOM`	Indicates whether the console can receive delete operator message (DOM) requests.
`OPERPARM_HC`	Indicates whether this console should receive all messages that are directed to hardcopy.
`OPERPARM_INTIDS`	Indicates whether or not a console should receive messages directed to the internal console.
`OPERPARM_KEY`	Indicates a data retrieval key used to search for user consoles using the `DISPLAY CONSOLES` command.
`OPERPARM_LEVEL`	Message level the user should receive. Available values include `R`, `I`, `CE`, `E`, `IN`, `NB`, or `ALL`. If you specify `ALL`, you cannot specify `R`, `I`, `CE`, `E`, or `IN`.
`OPERPARM_LOGCMDRESP`	Indicates whether command responses received by the user are logged.
`OPERPARM_MFORM`	Specifies the format messages are displayed in. Available values include `J`, `M`, `S`, `T`, and `X`.
`OPERPARM_MIGID`	Indicates whether the user should receive a migration console ID.
`OPERPARM_MONITOR`	List of events the user can monitor.
`OPERPARM_MSCOPE`	List of the systems this console can receive unsolicited messages from.

Attribute	Description
`OPERPARM_ROUTCODE`	Routing codes for messages this console receives.
`OPERPARM_STORAGE`	The amount of virtual storage (in megabytes) the console is allowed for message queuing.
`OPERPARM_UD`	Specifies whether this console should receive undelivered messages.
`OPERPARM_UNKNIDS`	Indicates whether a console should receive messages directed to unknown console IDs.

Attribute	Description
`OVM_UID`	The user's OpenExtensions for z/VM user ID.
`OVM_FSROOT`	The user's OpenExtensions for z/VM file system root directory path.
`OVM_HOME`	The user's OpenExtensions for z/VM home directory path.
`OVM_PROGRAM`	The user's OpenExtensions for z/VM program path, such as a default shell program.

Attribute	Description
`PROXY_LDAPHOST`	The URL of the LDAP server which the z/OS LDAP server contacts when acting as a proxy. An LDAP URL has a format such as `ldap://123.45.6:389` or `ldaps://123.45.6:636`.
`PROXY_BINDDN`	The distinguished name (DN) which the z/OS LDAP server uses when acting as a proxy. A DN is made using comma-separated attribute value pairs.

Attribute	Description
`TSO_ACCTNUM`	The user's default TSO account number.
`TSO_HOLDCLASS`	The user's default hold class.
`TSO_JOBCLASS`	The user's default job class.
`TSO_MAXSIZE`	The user's maximum region size.
`TSO_MSGCLASS`	The user's default message class.
`TSO_PROC`	The name of the user's default login procedure.
`TSO_SIZE`	The user's default region size.

Attribute	Description
`WORKATTR_WANAME`	User name on `SYSOUT`.
`WORKATTR_WABLDG`	Building on `SYSOUT`.
`WORKATTR_WADEPT`	Department on `SYSOUT`.
`WORKATTR_WAROOM`	Room on `SYSOUT`.
`WORKATTR_WAADDR1`	`SYSOUT` address line 1.
`WORKATTR_WAADDR2`	`SYSOUT` address line 2.
`WORKATTR_WAADDR3`	`SYSOUT` address line 3.
`WORKATTR_WAADDR4`	`SYSOUT` address line 4.
`WORKATTR_WAACCNT`	Account number.
`WORKATTR_WAEMAIL`	User email address.

Group attributes

The following attributes are available to the `__GROUP__` resource object:

Attribute	Description
`GID`	Group ID.
`NAME`	Group name.
`OWNER`	Group owner.
`MODEL`	Data set model profile to use when creating new data profiles, either generic or discrete.
`SUPGROUP`	Group’s superior group.
`TERMUACC`	RACF allows any user in this group to access a terminal based on the universal access authority for that terminal.
`UNIVERSAL`	Universal groups allow an effectively unlimited number of users to be connected.
`DATA`	Installation-defined data stored in the group profile.

Attribute	Description
`OVM_GID`	Group identifier.

Attribute	Description
`OMVS_GID`	Group identifier.

Attribute	Description
`DFP_DATACLAS`	Default data class.
`DFP_MGMTCLAS`	Default management class.
`DFP_STORCLAS`	Default storage class.
`DFP_DATAAPPL`	DFP data application identifier.

Attribute	Description
`TME_ROLES`	Group profile Tivoli security role.

Use the RACF connector

You can use the RACF connector to perform the following actions:

Users

Important

- When you create a new user, you must specify *at least* `__NAME__` and `userId`. Refer to the list of available attributes for more information.
- The length of `userId` must be 1-8 characters and can consist of any combination of:
  - Uppercase letters `A` - `Z`
  - Numbers `0` - `9`
  - `#` (`X'7B'`)
  - `$` (`X'5B'`)
  - `@` (`X'7C'`)
- `__NAME__` can be a maximum of 20 characters consisting of alphanumeric and non-alphanumeric characters.

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

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request POST \
--data '{
  "__NAME__": "BJENSEN",
  "userId": "BJENSEN"
}' \
"http://localhost:8080/openidm/system/racf/__ACCOUNT__?_action=create"
{
  "_id": "BJENSEN",
  "NAME": "UNKNOWN",
  "LAST-ACCESS": "UNKNOWN",
  "DFLTGRP": "SYS1",
  "WHEN": {
    "DAYS": "ANYDAY",
    "TIME": "ANYTIME"
  },
  "PASS-INTERVAL": "N/A",
  "PHRASEDATE": "N/A",
  "__NAME__": "BJENSEN",
  "__ENABLE__": true,
  "SECLABEL": "NONE SPECIFIED",
  "userId": "BJENSEN",
  "ATTRIBUTES": [
    "PROTECTED"
  ],
  "PASSDATE": "N/A",
  "SECLEVEL": "NONE SPECIFIED",
  "__GROUP__": [
    {
      "GROUP": "SYS1",
      "OWNER": "IBMUSER",
      "AUTH": "USE",
      "UACC": "NONE"
    }
  ],
  "OWNER": "IBMUSER"
}
```

The following example creates a user with additional attributes:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request POST \
--data '{
  "__NAME__": "Anto",
  "userId": "Anto",
  "__PASSWORD__": "Rvts1234",
  "__ENABLE__": true,
  "NAME": "Anto Monisha",
  "ATTRIBUTES": [
    "SPECIAL",
    "AUDITOR"
  ],
  "DFLTGRP": "IAM",
  "MODEL": "USER",
  "__GROUP__": [
    "SYS1"
  ],
  "OWNER": "Z1",
  "CLAUTH": [
    "TERMINAL",
    "TAPEVOL"
  ],
  "PROXY_LDAPHOST": "LDAP://12.34.567.89:389",
  "PROXY_BINDDN": "PATRICIA",
  "CICS_XRFSOFF": "NOFORCE",
  "CICS_TIMEOUT": "10",
  "CICS_OPCLASS": "2",
  "CICS_OPIDENT": "2",
  "CICS_TSLKEY": "99",
  "CICS_RSLKEY": "99",
  "CICS_OPPRTY": "5",
  "DCE_DCENAME": "TEST200",
  "DCE_HOMECELL": "/.../elvis.memphis.ibm.com",
  "DCE_HOMEUUID": "003456ab-ecb7-7de3-ebda-95531ed63dae",
  "DCE_UUID": "004386ea-ebb6-1ec3-bcae-10005ac90feb",
  "DCE_AUTOLOGIN": "No",
  "KERB_ENCRYPT": [
    "DES",
    "DES3",
    "DESD",
    "NOAES128",
    "AES128SHA2",
    "AES256SHA2"
  ],
  "KERB_KERBNAME": "KRBSEG002.SEC.COM",
  "KERB_MAXTKTLFE": "96400",
  "NETVIEW_CONSNAME": "CNSOLE06",
  "NETVIEW_CTL": "SPECIFIC",
  "NETVIEW_DOMAINS": "FR",
  "NETVIEW_IC": "NETVIEWCOMMAND",
  "NETVIEW_MSGRECVR": "YES",
  "NETVIEW_NGMFADMN": "YES",
  "NETVIEW_OPCLASS": "1",
  "DFP_DATAAPPL": "DFP4APPL",
  "DFP_DATACLAS": "DFP4DATA",
  "DFP_MGMTCLAS": "DFP4MGMT",
  "DFP_STORCLAS": "DFP4STOR",
  "TSO_ACCTNUM": "98655TT",
  "TSO_PROC": "TSOPROC3",
  "TSO_JOBCLASS": "Z",
  "TSO_HOLDCLASS": "X",
  "TSO_MSGCLASS": "Q",
  "TSO_SYS": "X",
  "TSO_MAXSIZE": "15000",
  "TSO_SIZE": "2500",
  "WHEN": {
    "DAYS": "WED",
    "TIME": "0800:1800"
  },
  "LNOTES_SNAME": "anto01",
  "NDS_UNAME": "anto01",
  "LANGUAGE_PRIMARY": "ENU",
  "LANGUAGE_SECONDARY": "DEU",
  "OVM_UID": 280,
  "OVM_FSROOT": "123",
  "OVM_HOME": "/u/pat",
  "OVM_PROGRAM": "/bin/sh"
}' \
"http://localhost:8080/openidm/system/racf/__ACCOUNT__?_action=create"
{
  "_id": "ANTO",
  "TSO_MSGCLASS": "Q",
  "NAME": "ANTO MONISHA",
  "CICS_RSLKEY": 99,
  "NETVIEW_OPCLASS": 1,
  "OWNER": "Z1",
  "DFLTGRP": "IAM",
  "SECLABEL": "NONE SPECIFIED",
  "TSO_SIZE": 2500,
  "SECLEVEL": "NONE SPECIFIED",
  "DFP_DATAAPPL": "DFP4APPL",
  "MODEL": "USER",
  "__ENABLE__": true,
  ...
}
```

You can modify an existing user with a PUT request, including all attributes of the account in the request. For a list of attributes, refer to RACF segments and attributes.

You can't modify any of the following attributes:

- `userId`
- `__NAME__`
- `DFLTGRP`

- `SECLEVEL`
- `SECLABEL`
- `LAST-ACCESS`
- `PASS-INTERVAL`
- `PHRASEDATE`
- `PASSDATE`

```
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__": "Rvts1234",
  "__ENABLE__": true,
  "NAME": "Monisha Vincer",
  "RESUME": null,
  "REVOKE": null,
  "__GROUP__": [
    "IAM",
    "EMPLOYEE"
  ],
  "ATTRIBUTES": [
    "SPECIAL",
    "AUDITOR"
  ],
  "MODEL": "RACF.ACCESS",
  "OWNER": "IBMUSER",
  "CLAUTH": [
    "TAPEVOL"
  ],
  "PROXY_LDAPHOST": "LDAP://12.34.567.89:389",
  "PROXY_BINDDN": "IBMUSER",
  "CICS_XRFSOFF": "NOFORCE",
  "CICS_TIMEOUT": "10",
  "CICS_OPCLASS": "2",
  "CICS_OPIDENT": "3",
  "CICS_TSLKEY": "99",
  "CICS_RSLKEY": "99",
  "CICS_OPPRTY": "5",
  "DCE_DCENAME": "TEST200",
  "DCE_HOMECELL": "/.../elvis.memphis.ibm.com",
  "DCE_HOMEUUID": "003456ab-ecb7-7de3-ebda-95531ed63dae",
  "DCE_UUID": "004386ea-ebb6-1ec3-bcae-10005ac90feb",
  "DCE_AUTOLOGIN": "No",
  "KERB_ENCRYPT": [
    "DES",
    "DES3",
    "DESD",
    "AES128",
    "AES128SHA2",
    "AES256SHA2"
  ],
  "KERB_KERBNAME": "KRBSEG002.SEC.COM",
  "KERB_MAXTKTLFE": "96400",
  "NETVIEW_CONSNAME": "CNSOLE06",
  "NETVIEW_CTL": "SPECIFIC",
  "NETVIEW_DOMAINS": "SK",
  "NETVIEW_IC": "NETVIEWCOMMAND",
  "NETVIEW_MSGRECVR": "YES",
  "NETVIEW_NGMFADMN": "YES",
  "NETVIEW_OPCLASS": "1",
  "DFP_DATAAPPL": "DFP4APPL",
  "DFP_DATACLAS": "DFP4DATA",
  "DFP_MGMTCLAS": "DFP4MGMT",
  "DFP_STORCLAS": "DFP4STOR",
  "TSO_ACCTNUM": "98655TT",
  "TSO_PROC": "TSOPROC3",
  "TSO_JOBCLASS": "Z",
  "TSO_HOLDCLASS": "X",
  "TSO_MSGCLASS": "Q",
  "TSO_SYS": "X",
  "TSO_MAXSIZE": "15000",
  "TSO_SIZE": "2500",
  "WHEN": {
    "DAYS": "WED",
    "TIME": "0800:1800"
  },
  "LNOTES_SNAME": "antovincer01",
  "NDS_UNAME": "antovincer01",
  "LANGUAGE_PRIMARY": "ENU",
  "LANGUAGE_SECONDARY": "DEU",
  "OVM_UID": 281,
  "OVM_FSROOT": "123",
  "OVM_HOME": "/u/pat1",
  "OVM_PROGRAM": "/bin/sh"
}' \
"http://localhost:8080/openidm/system/racf/__ACCOUNT__/ANTO"
{
  "_id": "ANTO",
  "TSO_MSGCLASS": "Q",
  "NAME": "MONISHA VINCER",
  "CICS_RSLKEY": 99,
  "NETVIEW_OPCLASS": 1,
  "OWNER": "IBMUSER",
  "DFLTGRP": "IAM",
  "SECLABEL": "NONE SPECIFIED",
  "TSO_SIZE": 2500,
  "SECLEVEL": "NONE SPECIFIED",
  "DFP_DATAAPPL": "DFP4APPL",
  "MODEL": "RACF.ACCESS",
  "__ENABLE__": true,
  "NETVIEW_CTL": "SPECIFIC",
  "__NAME__": "ANTO",
  "KERB_ENCRYPT": [
    "DES",
    "DES3",
    "DESD",
    "AES128",
    "AES256",
    "AES128SHA2",
    "AES256SHA2"
  ],
  "ATTRIBUTES": [
    "SPECIAL",
      "AUDITOR"
    ],
    "PROXY_BINDDN": "IBMUSER",
    "DCE_AUTOLOGIN": "NO",
    "NETVIEW_DOMAINS": "SK",
    "DFP_MGMTCLAS": "DFP4MGMT",
    "DCE_HOMECELL": "/.../elvis.memphis.ibm.com",
    "KERB_KERBNAME": "KRBSEG002.SEC.COM",
    "CLAUTH": [
      "TERMINAL",
      "TAPEVOL"
    ],
    "LANGUAGE_PRIMARY": "ENU",
    "NETVIEW_NGMFADMN": "YES",
    "CICS_OPCLASS": 2,
    "DCE_HOMEUUID": "003456ab-ecb7-7de3-ebda-95531ed63dae",
    "CICS_XRFSOFF": "NOFORCE",
    "TSO_MAXSIZE": 15000,
    "OVM_FSROOT": "123",
    "TSO_PROC": "TSOPROC3",
    "DFP_DATACLAS": "DFP4DATA",
    "userId": "ANTO",
    "NDS_UNAME": "antovincer01",
    "PHRASEDATE": "N/A",
    "TSO_ACCTNUM": "98655TT",
    "PASSDATE": "00.000",
    "TSO_SYS": "X",
    "DCE_UUID": "004386ea-ebb6-1ec3-bcae-10005ac90feb",
    "TSO_JOBCLASS": "Z",
    "OVM_UID": 281,
    "PROXY_LDAPHOST": "LDAP://12.34.567.89:389",
    "DFP_STORCLAS": "DFP4STOR",
    "CICS_TSLKEY": 99,
    "LAST-ACCESS": "22.181/23:34:59",
    "TSO_HOLDCLASS": "X",
    "NETVIEW_IC": "NETVIEWCOMMAND",
    "LANGUAGE_SECONDARY": "DEU",
    "NETVIEW_MSGRECVR": "YES",
    "WHEN": {
      "DAYS": "WED.",
      "TIME": "08:00 - 18:00"
    },
    "KERB_MAXTKTLFE": 96400,
    "CICS_TIMEOUT": "00:10 (HH:MM)",
    "NETVIEW_CONSNAME": "CNSOLE06",
    "OVM_HOME": "/u/pat1",
    "CICS_OPIDENT": "3",
    "__GROUP__": [
      "IAM",
      "EMPLOYEE"
    ],
    "DCE_DCENAME": "TEST200",
  "CICS_OPPRTY": 5,
  "PASS-INTERVAL": "180",
  "LNOTES_SNAME": "antovincer01",
  "OVM_PROGRAM": "/bin/sh"
}
```

The following example queries all RACF users:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/racf/__ACCOUNT__?_queryId=query-all-ids"
{
  "result": [
    {
      "_id": "ADCDY"
    },
    {
      "_id": "ADCDZ"
    },
    {
      "_id": "BJENSEN"
    },
    {
      "_id": "BPXOINIT"
    },
    {
      "_id": "CEA"
    },
    {
      "_id": "CFZSRV"
    },
    {
      "_id": "CICSUSER"
    },
    {
      "_id": "DANY101"
    },
    {
      "_id": "DANY102"
    },
    {
      "_id": "ZOSCAGL"
    },
    {
      "_id": "ZOSCSRV"
    },
    {
      "_id": "ZOSMFAD"
    },
    {
      "_id": "ZOSUGST"
    },
    {
      "_id": "ZWESIUSR"
    },
    {
      "_id": "ZWESVUSR"
    },
    ...
  ],
  "resultCount": 162,
  "pagedResultsCookie": null,
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

The following command queries a specific user by their ID:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/racf/__ACCOUNT__/ANTO"
{
  "_id": "ANTO",
  "TSO_MSGCLASS": "Q",
  "NAME": "MONISHA VINCER",
  "CICS_RSLKEY": 99,
  "NETVIEW_OPCLASS": 1,
  "OWNER": "IBMUSER",
  "OMVS_UID": 290,
  "DFLTGRP": "IAM",
  "SECLABEL": "NONE SPECIFIED",
  "OPERPARM_HC": "NO",
  "TSO_SIZE": 2500,
  "SECLEVEL": "NONE SPECIFIED",
  "DFP_DATAAPPL": "DFP4APPL",
  "OPERPARM_DOM": "NORMAL",
  "MODEL": "RACF.ACCESS",
  "OPERPARM_ROUTCODE": "ALL",
  "__ENABLE__": true,
  "OMVS_ASSIZEMAX": 10485769,
  "NETVIEW_CTL": "SPECIFIC",
  "__NAME__": "ANTO",
  ...
}
```

To reset the password for a RACF user account, use the connector to change the user’s password:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "if-Match:*" \
--request PATCH \
--data '{
  "__PASSWORD__": "Rvts123"
}' \
"http://localhost:8080/openidm/system/racf/__ACCOUNT__/ANTO"
{
  "_id": "ANTO",
  "TSO_MSGCLASS": "Q",
  "NAME": "ANTO MONISHA",
  "CICS_RSLKEY": 99,
  "NETVIEW_OPCLASS": 1,
  "OWNER": "Z1",
  "DFLTGRP": "IAM",
  "SECLABEL": "NONE SPECIFIED",
  "TSO_SIZE": 2500,
  "SECLEVEL": "NONE SPECIFIED",
  "DFP_DATAAPPL": "DFP4APPL",
  "MODEL": "USER",
  "__ENABLE__": true,
  ...
}
```

Note

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

The following example activates a user with the minimum required attribute:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "If-Match:*" \
--request PUT \
--data '{
  "__ENABLE__": true
}' \
"http://localhost:8080/openidm/system/racf/__ACCOUNT__/ANTO"
{
  "_id": "ANTO",
  "TSO_MSGCLASS": "Q",
  "NAME": "ANTO MONISHA",
  "CICS_RSLKEY": 99,
  "NETVIEW_OPCLASS": 1,
  "OWNER": "Z1",
  "DFLTGRP": "IAM",
  "SECLABEL": "NONE SPECIFIED",
  "TSO_SIZE": 2500,
  "SECLEVEL": "NONE SPECIFIED",
  "DFP_DATAAPPL": "DFP4APPL",
  "MODEL": "USER",
  "__ENABLE__": true,
  ...
}
```

The following example activates a user with the minimum required attribute:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "If-Match:*" \
--request PUT \
--data '{
  "__ENABLE__": false
}' \
"http://localhost:8080/openidm/system/racf/__ACCOUNT__/ANTO"
{
  "_id": "ANTO",
  "TSO_MSGCLASS": "Q",
  "NAME": "ANTO MONISHA",
  "CICS_RSLKEY": 99,
  "NETVIEW_OPCLASS": 1,
  "OWNER": "Z1",
  "DFLTGRP": "IAM",
  "SECLABEL": "NONE SPECIFIED",
  "TSO_SIZE": 2500,
  "SECLEVEL": "NONE SPECIFIED",
  "DFP_DATAAPPL": "DFP4APPL",
  "MODEL": "USER",
  "__ENABLE__": false,
  ...
}
```

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

The following example deletes a RACF account:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request DELETE \
"http://localhost:8080/openidm/system/racf/__ACCOUNT__/ANTO"
{
  "_id": "ANTO",
  "TSO_MSGCLASS": "Q",
  "NAME": "MONISHA VINCER",
  "CICS_RSLKEY": 99,
  "NETVIEW_OPCLASS": 1,
  "OWNER": "IBMUSER",
  "OMVS_UID": 290,
  "DFLTGRP": "IAM",
  "SECLABEL": "NONE SPECIFIED",
  "OPERPARM_HC": "NO",
  "TSO_SIZE": 2500,
  "SECLEVEL": "NONE SPECIFIED",
  "DFP_DATAAPPL": "DFP4APPL",
  "OPERPARM_DOM": "NORMAL",
  "MODEL": "RACF.ACCESS",
  "OPERPARM_ROUTCODE": "ALL",
  "__ENABLE__": true,
  ...
}
```

Groups

You can create a group using any of the following unique attributes:

- `GID` (must be the same as `__NAME__`)
- `OVM_GID`
- `OMVS_GID`

Important

- When you create a new group, you must specify *at least* `__NAME__` and `GID`. Refer to the list of available attributes for more information.
- The length of `GID` must be 1-8 characters and can consist of any combination of:
  - Uppercase letters `A` - `Z`
  - Numbers `0` - `9`
  - `#` (`X'7B'`)
  - `$` (`X'5B'`)
  - `@` (`X'7C'`)
- `__NAME__` can be a maximum of 20 characters consisting of alphanumeric and non-alphanumeric characters.

Note

When you create a new group, you must specify at least the `GID` and `__NAME__`.

The following example creates a group with all the creatable attributes:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request POST \
--data '{
  "GID": "SFDSFFH",
  "__NAME__": "SFDSFFH",
  "SUPGROUP": "SYS1",
  "TERMUACC": true,
  "UNIVERSAL": true,
  "DATA": "HELLOEVERYONE",
  "OVM_GID": 3245,
  "OMVS_GID": 4365,
  "TME_ROLES": "role002",
  "DFP_MGMTCLAS": "DFP3MGMT",
  "DFP_STORCLAS": "DFP3STOR",
  "DFP_DATAAPPL": "DFP3APPL",
  "DFP_DATACLAS": "DFP3DATA",
  "MODEL": "TEST",
  "OWNER": "IBMUSER"
}' \
"http://localhost:8080/openidm/system/racf/__GROUP__?_action=create"
{
  "_id": "SFDSFFH",
  "DFP_DATACLAS": "DFP3DATA",
  "Users": null,
  "OWNER": "IBMUSER",
  "DFP_DATAAPPL": "DFP3APPL",
  "GID": "SFDSFFH",
  "OVM_GID": 3245,
  "DATA": "HELLOEVERYONE",
  "__NAME__": "SFDSFFH",
  "TME_ROLES": "ROLE002",
  "CREATED": "24.005",
  "OMVS_GID": 4365,
  "SUPGROUP": "SYS1",
  "SUBGROUP": null,
  "UNIVERSAL": true,
  "TERMUACC": true,
  "DFP_STORCLAS": "DFP3STOR",
  "DFP_MGMTCLAS": "DFP3MGMT",
  "MODEL": "TEST"
}
```

The following example queries all RACF groups:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/racf/__GROUP__?_queryId=query-all-ids"
{
  "result": [
    {
      "_id": "AA01"
    },
    {
      "_id": "AA02"
    },
    {
      "_id": "ADCD"
    },
    {
      "_id": "BLZCFG"
    },
    {
      "_id": "BLZGRP"
    },
    {
      "_id": "BLZWRK"
    },
    {
      "_id": "CEAGP"
    },
    {
      "_id": "CFZADMGP"
    },
    {
      "_id": "CFZSRVGP"
    },
    {
      "_id": "CFZUSRGP"
    },
    {
      "_id": "CIMGP"
    },
    {
      "_id": "DB2"
    },
    {
      "_id": "DGFDGDH"
    },
    {
      "_id": "DSN710"
    },
    {
      "_id": "EMPLOYEE"
    },
    {
      "_id": "EXTERNAL"
    },
    ...
  ],
  "resultCount": 83,
  "pagedResultsCookie": null,
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

The following command queries a group by ID:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/racf/__GROUP__?_queryFilter=_id%20eq%20%22SFDSFFH%22"
{
  "result": [
    {
      "_id": "SFDSFFH",
      "DFP_DATACLAS": "DFP4DATA",
      "Users": null,
      "OWNER": "IBMUSER",
      "DFP_DATAAPPL": "DFP4APPL",
      "GID": "SFDSFFH",
      "OVM_GID": 3651,
      "DATA": "HELLY",
      "__NAME__": "SFDSFFH",
      "TME_ROLES": "ROLE004",
      "CREATED": "24.005",
      "OMVS_GID": 9011,
      "SUPGROUP": "SYS1",
      "SUBGROUP": null,
      "UNIVERSAL": true,
      "TERMUACC": true,
      "DFP_STORCLAS": "DFP4STOR",
      "DFP_MGMTCLAS": "DFP4MGMT",
      "MODEL": "TEST"
    }
  ],
  "resultCount": 1,
  "pagedResultsCookie": null,
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

You can modify an existing group with a PUT request, including all attributes of the group in the request. For a list of attributes, refer to RACF segments and attributes.

You can’t modify any of the following attributes:

- `GID`
- `__NAME__`
- `CREATED`
- `TERMUACC`
- `SUBGROUP`

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "If-Match:*" \
--request PUT \
--data '{
  "SUPGROUP": "SYS1",
  "TERMUACC": true,
  "DATA": "HELLY",
  "OVM_GID": 3651,
  "OMVS_GID": 9011,
  "TME_ROLES": "ROLE004",
  "DFP_MGMTCLAS": "DFP4MGMT",
  "DFP_STORCLAS": "DFP4STOR",
  "DFP_DATAAPPL": "DFP4APPL",
  "DFP_DATACLAS": "DFP4DATA",
  "OWNER": "IBMUSER",
  "MODEL": "TEST001"
}' \
"http://localhost:8080/openidm/system/racf/__GROUP__/SFDSFFH"
{
  "_id": "SFDSFFH",
  "DFP_DATACLAS": "DFP4DATA",
  "Users": null,
  "OWNER": "IBMUSER",
  "DFP_DATAAPPL": "DFP4APPL",
  "GID": "SFDSFFH",
  "OVM_GID": 3651,
  "DATA": "HELLY",
  "__NAME__": "SFDSFFH",
  "TME_ROLES": "ROLE004",
  "CREATED": "24.005",
  "OMVS_GID": 9011,
  "SUPGROUP": "SYS1",
  "SUBGROUP": null,
  "UNIVERSAL": true,
  "TERMUACC": true,
  "DFP_STORCLAS": "DFP4STOR",
  "DFP_MGMTCLAS": "DFP4MGMT",
  "MODEL": "TEST001"
}
```

You can use the RACF connector to delete a group from the RACF service.

The following example deletes a RACF group:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "If-Match: *" \
--request DELETE \
"http://localhost:8080/openidm/system/racf/__GROUP__/SFDSFFH"
{
  "_id": "SFDSFFH",
  "DFP_DATACLAS": "DFP4DATA",
  "Users": null,
  "OWNER": "IBMUSER",
  "DFP_DATAAPPL": "DFP4APPL",
  "GID": "SFDSFFH",
  "OVM_GID": 3651,
  "DATA": "HELLY",
  "__NAME__": "SFDSFFH",
  "TME_ROLES": "ROLE004",
  "CREATED": "24.005",
  "OMVS_GID": 9011,
  "SUPGROUP": "SYS1",
  "SUBGROUP": null,
  "UNIVERSAL": true,
  "TERMUACC": true,
  "DFP_STORCLAS": "DFP4STOR",
  "DFP_MGMTCLAS": "DFP4MGMT",
  "MODEL": "TEST"
}
```

OpenICF Interfaces Implemented by the RACF Connector

The RACF Connector implements the following OpenICF interfaces. For additional details, see ICF interfaces:

Create

Creates an object and its `uid`.

Delete

Deletes an object, referenced by its `uid`.

Schema

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

Script on Connector

Enables an application to run a script in the context of the connector.

Any script that runs on the connector has the following characteristics:

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

Search

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

Test

Tests the connector configuration.

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

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

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

Update

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

RACF Connector Configuration

The RACF Connector has the following configurable properties:

Configuration properties

Property	Type	Default	Encrypted[1]	Required[2]
`hostName`	`String`	`null`		✓ Yes
Host name or IP address of RACF.				
`port`	`Integer`	`null`		✓ Yes
TCP/IP port number used to communicate with the RACF.				
`userId`	`String`	`null`		✓ Yes
The user id used to login to RACF.				

Property	Type	Default	Encrypted[1]	Required[2]
`password`	`GuardedString`	`null`	🔒 Yes	✓ Yes
The password used to login to RACF.				
`segments`	`String`	`null`		✕ No
To retrieve data based on RACF segments.				
`acceptSelfSignedCertificates`	`boolean`	`false`		✓ Yes
Specifies whether to accept or not self-signed certificates.				
`clientCertAlias`	`String`	`null`		✕ No
Alias for the client certificate.				
`clientCertPassword`	`GuardedString`	`null`	🔒 Yes	✕ No
Password for the client certificate.				
`maximumConnections`	`Integer`	`10`		✕ No
Provides the maximum connections.				
`connectionTimeout`	`Integer`	`300`		✕ No
Provides the maximum connection timeout in seconds.				
`httpProxyHost`	`String`	`null`		✕ No
Provides the Proxy Host.				
`httpProxyPort`	`Integer`	`null`		✕ No
Provides the Proxy Port.				
`httpProxyUsername`	`String`	`null`		✕ No
Provides the Proxy Username.				
`httpProxyPassword`	`GuardedString`	`null`	🔒 Yes	✕ No
Provides the Proxy Password.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

Kerberos connector

The Kerberos connector is an implementation of the SSH connector, and is based on Java Secure Channel (JSch) and the Java implementation of the Expect library (Expect4j).

The Kerberos connector lets you manage Kerberos user principals from IDM. The connector bundles a number of Groovy scripts, to interact with a Kerberos admin server. You should not edit the bundled Groovy scripts. The scripts use the `kadmin` utility to communicate with the Kerberos server.

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

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

Kerberos connector schema

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

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

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

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

- `failedPasswordAttempts`

Install the Kerberos connector

> **Tip**
>
> To check for an Advanced Identity Cloud application for this connector, refer to:
>
> - Application management
> - App catalog

You can download any connector from Backstage, but some are included in the default deployment for Advanced Identity Cloud, IDM, or RCS. When using an included connector, you can skip installing it and move directly to configuration.

Connector included in default deployment

Connector	IDM	RCS
Kerberos	✓ Yes	✓ Yes

Download the connector .jar file from Backstage.

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

```
mv ~/Downloads/kerberos-connector-1.5.20.30.jar /path/to/openidm/connectors/
```

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

Configure the Kerberos connector

Create a connector configuration using the IDM admin UI:

1. From the navigation bar, click **Configure > Connectors**.
2. On the **Connectors** page, click **New Connector**.
3. On the **New Connector** page, type a **Connector Name**.
4. From the **Connector Type** drop-down list, select **Kerberos Connector - 1.5.20.30**.
5. Complete the **Base Connector Details**.

   > **Tip**
   >
   > For a list of all configuration properties, refer to Kerberos Connector Configuration

6. Click **Save**.

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

Refer to this procedure to create a connector configuration over REST.

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

1. Set the authentication properties, as described in Configure 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 connector configuration is as follows:

   ```
   "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 connector configuration is as follows:

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

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

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

This list describes the basic Kerberos connector configuration properties. For a list of all configuration properties, refer to Kerberos Connector Configuration:

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.

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

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

`password`

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

`prompt`

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

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

Consider customizing your Linux prompt with the `PS1` and `PS2` variables, to set a *safe* prompt. For information about customizing prompts, refer to 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, refer to 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, refer to connector-reference:ssh.adoc#ssh-authentication.

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 perform the ICF operations, relative to your installation directory. For the Kerberos connector, the scripts are bundled in the connector .jar file, so the sample connector configuration uses the path `jar:file:connectors/kerberos-connector-1.5.20.30.jar!/scripts/kerberos/`.

`classpath`

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

`ScriptFileName`

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

Kerberos remote connector

If you want to run this connector outside of PingOne Advanced Identity Cloud or IDM, you can configure the Kerberos connector as a remote connector. Java Connectors installed remotely on a Java Connector Server function identically to those bundled locally within PingOne Advanced Identity Cloud or installed locally on IDM.

You can download the Kerberos connector from here.

Refer to Remote connectors for configuring the Kerberos remote connector.

Configure connection pooling

The Kerberos connector uses ICF pooling to manage connections. Learn more about the different pooling mechanisms in Connectors by pooling mechanism.

OpenICF Interfaces Implemented by the Kerberos Connector

The Kerberos Connector implements the following OpenICF interfaces. For additional details, see ICF interfaces:

Authenticate

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

Create

Creates an object and its `uid`.

Delete

Deletes an object, referenced by its `uid`.

Resolve Username

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

Schema

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

Script on Connector

Enables an application to run a script in the context of the connector.

Any script that runs on the connector has the following characteristics:

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

Script on Resource

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

Search

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

Sync

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

Test

Tests the connector configuration.

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

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

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

Update

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

Kerberos Connector Configuration

The Kerberos Connector has the following configurable properties:

Basic Configuration Properties

Property	Type	Default	Encrypted[1]	Required[2]
`host`	`String`	`null`		✓ Yes
The hostname to connect to.				
`port`	`int`	`22`		✓ Yes
TCP port to use.				
`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.				

Property	Type	Default	Encrypted[1]	Required[2]
`prompt`	`String`	`root@localhost:# `		✓ Yes
A string representing the remote SSH session prompt.				
`sudoCommand`	`String`	`/usr/bin/sudo`		✓ Yes
A string representing the sudo command.				
`echoOff`	`boolean`	`true`		✓ Yes
Disable the input command echo.				
`terminalType`	`String`	`vt102`		✓ Yes
Defines the terminal type to use for the session.				
`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.				
`connectionTimeout`	`int`	`5000`		✓ Yes
Defines the connection timeout to the remote server in milliseconds.				
`expectTimeout`	`long`	`5000`		✓ Yes
Defines the timeout used by the expect() calls in the scripts in milliseconds.				
`throwOperationTimeoutException`	`boolean`	`true`		✓ Yes
Defines if an OperationTimeoutException should be thrown if any call to expect times out.				
`promptReadyTimeout`	`long`	`20`		✕ No
Defines the "prompt ready" timeout for the promptReady() command in milliseconds.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

Groovy Engine configuration

Property	Type	Default	Encrypted[1]	Required[2]
`scriptRoots`	`String[]`	`['!/scripts/kerberos/']`		✓ Yes
The root folder to load the scripts from. If the value is null or empty the classpath value is used.				
`classpath`	`String[]`	`[]`		✕ No
Classpath for use during compilation.				
`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.				
`minimumRecompilationInterval`	`int`	`100`		✕ No
Sets the minimum of time after a script can be recompiled.				
`recompileGroovySource`	`boolean`	`false`		✕ No
If set to true recompilation is enabled.				
`scriptBaseClass`	`String`	`null`		✕ No
Base class name for scripts (must derive from Script).				
`scriptExtensions`	`String[]`	`['groovy']`		✕ No
Gets the extensions used to find groovy files.				
`sourceEncoding`	`String`	`UTF-8`		✕ No
Encoding for source files.				
`targetDirectory`	`File`	`null`		✕ No
Directory into which to write classes.				
`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.				

Property	Type	Default	Encrypted[1]	Required[2]
`verbose`	`boolean`	`false`		✕ No
If true, the compiler should produce action information.				
`warningLevel`	`int`	`1`		✕ No
Warning Level of the compiler.				
`customConfiguration`	`String`	`kadmin { cmd = '/usr/sbin/kadmin.local'; user='<KADMIN USERNAME>'; default_realm='<REALM, e.g. EXAMPLE.COM>' }`		✕ No
Custom Configuration script for Groovy ConfigSlurper.				
`customSensitiveConfiguration`	`GuardedString`	`null`	🔒 Yes	✕ No
Custom Sensitive Configuration script for Groovy ConfigSlurper.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

Operation Script Files

Property	Type	Default	Encrypted[1]	Required[2]
`authenticateScriptFileName`	`String`	`null`		• Authenticate
The name of the file used to perform the AUTHENTICATE operation.				
`createScriptFileName`	`String`	`CreateKerberos.groovy`		• Create
The name of the file used to perform the CREATE operation.				
`customizerScriptFileName`	`String`	`null`		✕ No
The script used to customize some function of the connector. Read the documentation for more details.				

Property	Type	Default	Encrypted[1]	Required[2]
`deleteScriptFileName`	`String`	`DeleteKerberos.groovy`		- Delete
The name of the file used to perform the DELETE operation.				
`resolveUsernameScriptFileName`	`String`	`null`		- Resolve Username
The name of the file used to perform the RESOLVE_USERNAME operation.				
`schemaScriptFileName`	`String`	`SchemaKerberos.groovy`		- Schema
The name of the file used to perform the SCHEMA operation.				
`scriptOnResourceScriptFileName`	`String`	`ScriptOnResourceKerberos.groovy`		- Script on Resource
The name of the file used to perform the RUNSCRIPTONRESOURCE operation.				
`searchScriptFileName`	`String`	`SearchKerberos.groovy`		- Read - Search
The name of the file used to perform the SEARCH operation.				
`syncScriptFileName`	`String`	`null`		- Sync
The name of the file used to perform the SYNC operation.				
`testScriptFileName`	`String`	`TestKerberos.groovy`		- Test
The name of the file used to perform the TEST operation.				
`updateScriptFileName`	`String`	`UpdateKerberos.groovy`		- Update
The name of the file used to perform the UPDATE operation.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

LDAP connector

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

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

Install the LDAP connector

> **Tip**
>
> To check for an Advanced Identity Cloud application for this connector, refer to:
>
> - Application management
> - App catalog

You can download any connector from Backstage, but some are included in the default deployment for Advanced Identity Cloud, IDM, or RCS. When using an included connector, you can skip installing it and move directly to configuration.

Connector included in default deployment

Connector	IDM	RCS
LDAP	✓ Yes	✓ Yes

Download the connector .jar file from Backstage.

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

  ```
  mv ~/Downloads/ldap-connector-1.5.20.30.jar /path/to/openidm/connectors/
  ```

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

Configure the LDAP connector

Create a connector configuration using the IDM admin UI:

1. From the navigation bar, click **Configure > Connectors**.

2. On the **Connectors** page, click **New Connector**.

3. On the **New Connector** page, type a **Connector Name**.
4. From the **Connector Type** drop-down list, select **LDAP Connector - 1.5.20.30**.
5. Complete the **Base Connector Details**.

> **Tip**
>
> For a list of all configuration properties, refer to LDAP Connector Configuration

6. Click **Save**.

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

Refer to this procedure to create a connector configuration over REST.

Alternatively, configure the connector with a configuration file. IDM provides several sample LDAP connector configurations in the `path/to/openidm/samples/example-configurations/provisioners/` directory. Copy one of the sample connector configurations to your project's `conf` directory, and adjust it to match your LDAP environment:

- `provisioner.openicf-ldap.json` —a sample LDAP connector configuration for a generic LDAP server.
- `provisioner.openicf-dsldap.json` —a sample LDAP connector configuration for a PingDS (DS) server.
- `provisioner.openicf-adldap.json` —a sample LDAP connector configuration for an Active Directory server.

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

Sample LDAP connector configuration

This configuration shows the properties for an LDAP connector connecting to DS. For more information about the properties that affect synchronization, refer to Control what the LDAP connector synchronizes. For a list of all configuration properties, refer to LDAP Connector Configuration:

```
"configurationProperties" : {
    "host" : "localhost",
    "port" : 1389,
    "ssl" : false,
    "startTLS" : false,
    "privateKeyAlias" : null,
    "alternateKeyStore" : null,
    "alternateKeyStoreType" : null,
    "alternateKeyStorePassword" : null,
    "principal" : "uid=admin",
    "credentials" : "password",
    "baseContexts" : [
        "dc=example,dc=com"
    ],
    "baseContextsToSynchronize" : [
        "dc=example,dc=com"
    ],
    "accountSearchFilter" : null,
    "accountSynchronizationFilter" : null,
    "groupSearchFilter" : null,
    "groupSynchronizationFilter" : null,
    "removeLogEntryObjectClassFromFilter" : true,
    "modifiersNamesToFilterOut" : [ ],
    "changeLogBlockSize" : 100,
    "attributesToSynchronize" : [ ],
    "changeNumberAttribute" : "changeNumber",
    "filterWithOrInsteadOfAnd" : false,
    "objectClassesToSynchronize" : [
        "inetOrgPerson"
    ],
    "vlvSortAttribute" : "uid",
    "passwordAttribute" : "userPassword",
    "useBlocks" : false,
    "maintainPosixGroupMembership" : false,
    "failover" : [ ],
    "readSchema" : true,
    "accountObjectClasses" : [
        "top",
        "person",
        "organizationalPerson",
        "inetOrgPerson"
    ],
    "accountUserNameAttributes" : [
        "uid"
    ],
    "groupMemberAttribute" : "uniqueMember",
    "passwordHashAlgorithm" : null,
    "usePagedResultControl" : true,
    "blockSize" : 100,
    "uidAttribute" : "entryUUID",
    "maintainLdapGroupMembership" : false,
    "respectResourcePasswordPolicyChangeAfterReset" : false
},
```

`host`

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

`port`

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

`ssl`

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

For instructions on using the LDAP connector over SSL, refer to Configure the LDAP Connector to Use SSL and StartTLS.

`startTLS`

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

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

`principal`

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

`credentials`

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

`baseContexts`

One or more starting points in the LDAP tree that will be used when searching the tree. Searches are performed when discovering users from the LDAP server or when looking for the groups of which a user is a member. During reconciliation operations, IDM searches through the base contexts listed in this property for changes. For more information, refer to Control What the LDAP Connector Synchronizes.

`baseContextsToSynchronize`

One or more starting points in the LDAP tree that will be used to determine if a change should be synchronized. During liveSync operations, IDM searches through the base contexts listed in this property for changes. If no value is specified here, the values in listed in the `baseContexts` property are used. For more information, refer to Control What the LDAP Connector Synchronizes.

`accountSynchronizationFilter`

Used during synchronization actions to filter out LDAP accounts. For more information, refer to Control What the LDAP Connector Synchronizes.

`accountObjectClasses`

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

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

`accountSearchFilter`

Search filter that user accounts must match. For more information, refer to Control What the LDAP Connector Synchronizes.

`accountUserNameAttributes`

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

`attributesToSynchronize`

List of attributes used during object synchronization. IDM ignores change log updates that do not include any of the specified attributes. If empty, IDM considers all changes. For more information, refer to Control What the LDAP Connector Synchronizes.

`blockSize`

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

`changeLogBlockSize`

Block size used when fetching change log entries.

`changeNumberAttribute`

Change log attribute containing the last change number.

`failover`

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

`filterWithOrInsteadOfAnd`

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

`groupMemberAttribute`

LDAP attribute holding members for non-POSIX static groups.

`groupSearchFilter`

Search filter that group entries must match.

`maintainLdapGroupMembership`

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

Does not apply to Active Directory.

In the sample LDAP connector configuration, this property is set to `false`. This means that LDAP group membership is not modified when entries are renamed or deleted in IDM. To ensure that entries are removed from LDAP groups when the entries are deleted, set this property to `true` or enable referential integrity on the LDAP server.

Learn more about configuring referential integrity in DS, refer to Referential Integrity in the *Configuration Guide* for PingDS.

Learn more about LDAP groups in Configuring `ldapGroups` in the LDAP connector.

`maintainPosixGroupMembership`

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

`modifiersNamesToFilterOut`

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

`objectClassesToSynchronize`

IDM synchronizes only entries that have these object classes. For more information, refer to Control What the LDAP Connector Synchronizes.

`passwordAttribute`

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

`passwordHashAlgorithm`

Hash password values with the specified algorithm, if the LDAP server stores them in clear text. The only supported algorithm is `WIN-AD` (used when Active Directory is the target). A blank value indicates the system will not hash passwords. This will cause clear text passwords to be stored in LDAP, unless the LDAP server performs the hash.

`readSchema`

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

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

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

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

`removeLogEntryObjectClassFromFilter`

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

`respectResourcePasswordPolicyChangeAfterReset`

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

`uidAttribute`

Specifies the LDAP attribute that should be used as the immutable ID for the entry. You can use a DN (or any unique attribute) for the `id`. As a best practice, you _should use an attribute that is both unique and immutable, such as the `entryUUID`. For a DS resource, you must use the `entryUUID` as the `uidAttribute`, otherwise you might encounter problems with synchronizing delete operations.

`useBlocks`

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

`usePagedResultControl`

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

`useTimestampsForSync`

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

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

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

- MS Active Directory Global Catalog
- OpenLDAP
- Unknown

> **Note**
>
> An LDAP server type is marked Unknown if it is anything other than PingDirectory, IBM, Novell, UnboundID, RedHat/Fedora 389, CA LDAP, OpenDS, PingDS, Sun DSEE Directory, Microsoft Active Directory, Microsoft Active Directory Lightweight Directory Services (LDS), Microsoft Active Directory Global Catalog, or OpenLDAP.

`timestampSyncOffset`

An optional offset, specified in seconds, that is negatively applied to the timestamp for timestamp-based sync operations. This setting is useful when there are replication delays between LDAP instances. The default value is `0`, or no offset.

`vlvSortAttribute`

Attribute used as the sort key for virtual list view.

`sendCAUDTxId`

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

LDAP remote connector

If you want to run this connector outside of PingOne Advanced Identity Cloud or IDM, you can configure the LDAP connector as a remote connector. Java Connectors installed remotely on a Java Connector Server function identically to those bundled locally within PingOne Advanced Identity Cloud or installed locally on IDM.

You can download the LDAP connector from here.

Refer to Remote connectors for configuring the LDAP remote connector.

Configure connection pooling

The LDAP connector uses ICF pooling to manage connections. Learn more about the different pooling mechanisms in Connectors by pooling mechanism.

Configure the LDAP connector to use SSL and StartTLS

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

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

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

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

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

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

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

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

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

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

- Specify a different keystore for the connector.

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

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

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

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

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

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

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

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

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

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

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

The error returned is similar to the following:

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

Configure the LDAP connector for failover

You can configure the LDAP connector for failover in the connector configuration or in the provisioner configuration file (IDM only). This allows you to specify a primary server and alternative secondary servers.

When you configure failover:

- If Advanced Identity Cloud or IDM can't connect to the primary server, they will attempt to connect to one of the secondary servers in the order they're specified until a connection is successful.
- For new connections, if the primary server becomes available, Advanced Identity Cloud or IDM will reconnect to the primary server.
- For existing connections, as long as a connection is valid or not expired and is in the connection pool, the failover server will be used.

> **Note**
>
> This information applies only to the LDAP connector. If you're using DS as an external repository, learn more in Configure two DS repositories in an active/passive deployment.

Configure LDAP for failover using the admin UI

You can configure the LDAP connector for failover using the admin UI as follows:

1. Select the LDAP connector you want to update:
   - For the Advanced Identity Cloud admin UI: Go to **Native Consoles** > **Identity Management** > **Configure** > **Connectors** > **LDAP connector**.
   - For the IDM admin UI: Go to **Configure** > **Connectors** > **LDAP connector**.
2. From **Base Connector Details** on the **Details** tab, update the **Host Name or IP** and **Port** fields to point to the primary server.
3. Expand the **Additional Options** section on the **Details** tab.
4. Enter the full LDAP URLs of one or more secondary servers in the **Failover LDAP servers, by URL** fields.
5. Click **Save**.

Configure LDAP for failover using the provisioner configuration file (IDM only)

You can configure the LDAP connector for failover using the provisioner configuration file as follows:

1. Edit your provisioner configuration file. For example, `provisioner.openicf.ldap.json`, which is located in the `/path/to/idm/conf` directory.
2. Set the host and port properties in your provisioner configuration file to point to the primary server, for example:

```
"configurationProperties" : {
    "host" : "ds1.example.com",
    "port" : 1389,
    ...
}
```

3. Set the failover property in your provisioner configuration file to point to one or more secondary servers by specifying the full LDAP URLs, for example:

```
"configurationProperties" : {
    "host" : "ds1.example.com",
    "port" : 1389,
    "failover" : [
    "ldap://ds2.example.com:1389",
    "ldap://ds3.example.com:1389"
    ],
    ...
}
```

Control what the LDAP connector synchronizes

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

`accountSearchFilter`

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

`accountSynchronizationFilter`

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

`attributesToSynchronize`

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

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

`baseContexts`

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

`baseContextsToSynchronize`

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

`modifiersNamesToFilterOut`

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

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

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

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

`objectClassesToSynchronize`

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

Use the LDAP connector with Active Directory

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

`__ENABLE__`

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

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

`__ACCOUNT_EXPIRES__`

Sets the `accountExpires` attribute of an Active Directory object to reset an expired account, or to set a future expiration date.

To set an account that *never* expires, set `"__ACCOUNT_EXPIRES__": "0"` .

To set an expiration date, set `"__ACCOUNT_EXPIRES__": "date"` , where `date` is in ISO8601 format. For example:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request PUT \
--data '{
  "__ACCOUNT_EXPIRES__": "2020-12-31T00:00:00Z"
}' \
"http://localhost:8080/openidm/system/AD/account/e1418d64-096c-4cb0-b903-ebb66562d99d"
{
    "sn": "jensen",
    "__LOCK_OUT__": false,
    "__ENABLE__": true,
    "objectGUID": "e1418d64-096c-4cb0-b903-ebb66562d99d",
    "dn": "CN=bjensen,OU=create,DC=example,DC=com",
    "accountExpires": "2020-12-31T00:00:00Z"
}
```

__LOCK_OUT__

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

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

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

__PASSWORD_EXPIRED__

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

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

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

> **Note**
>
> Active Directory does not allow you to create an enabled account with an expired password. If you are using `__PASSWORD_EXPIRED__` to force a new user to change their password when they next log in, you can create the user account as disabled initially (`__ENABLE__=false`). You can then patch the new user account to enable it. You can use the same workaround for synchronization operations, creating new user accounts as disabled, then issuing an `openidm.patch` call in a `postCreate` script to enable the account.

__CURRENT_PASSWORD__

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

The sample connector configuration file (`openidm/samples/example-configurations/provisioners/provisioner.openicf-adldap.json`) includes these operational attributes. Note that the `passwordAttribute` property in this provisioner file is set to `unicodePwd` . This property specifies the attribute in Active Directory that holds the user password. When a user's password is changed, the new value is set in this attribute.

Manage Active Directory users with the LDAP connector

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

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

```
curl \
--header "Content-Type: application/json" \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
--data '{
  "dn": "CN=Brian Smith,CN=Users,DC=example,DC=com",
  "cn": "Brian Smith",
  "sAMAccountName": "bsmith",
  "userPrincipalName": "bsmith@example.com",
  "userAccountControl": "512",
  "givenName": "Brian",
  "mail": "bsmith@example.com",
  "__PASSWORD__": "Passw0rd"
}' \
"http://localhost:8080/openidm/system/ad/account?_action=create"
{
  "_id": "e1418d64-096c-4cb0-b903-ebb66562d99d",
  "mobile": null,
  "postalCode": null,
  "st": null,
  "employeeType": [],
  "objectGUID": "e1418d64-096c-4cb0-b903-ebb66562d99d",
  "cn": "Brian Smith",
  "department": null,
  "l": null,
  "description": null,
  "info": null,
  "manager": null,
  "sAMAccountName": "bsmith",
  "sn": null,
  "whenChanged": "20151217131254.0Z",
  "userPrincipalName": "bsmith@example.com",
  "userAccountControl": "512",
  "__ENABLE__": true,
  "displayName": null,
  "givenName": "Brian",
  "middleName": null,
  "facsimileTelephoneNumber": null,
  "lastLogon": "0",
  "countryCode": "0",
  "employeeID": null,
  "co": null,
  "physicalDeliveryOfficeName": null,
  "pwdLastSet": "2015-12-17T13:12:54Z",
  "streetAddress": null,
  "homePhone": null,
  "__PASSWORD_NOTREQD__": false,
  "telephoneNumber": null,
  "dn": "CN=Brian Smith,CN=Users,DC=example,DC=com",
  "title": null,
  "mail": "bsmith@example.com",
  "postOfficeBox": null,
  "__SMARTCARD_REQUIRED__": false,
  "uSNChanged": "86144",
  "__PASSWORD_EXPIRED__": false,
  "initials": null,
  "__LOCK_OUT__": false,
  "company": null,
  "employeeNumber": null,
  "accountExpires": "0",
  "c": null,
  "whenCreated": "20151217131254.0Z",
  "uSNCreated": "86142",
  "division": null,
  "groups": [],
  "__DONT_EXPIRE_PASSWORD__": false,
  "otherHomePhone": []
}
```

Important

- Previous versions of the LDAP connector appended `<GUID=` to the GUID for Active Directory objects. This behavior ensured compatibility with the legacy .NET connector.
  The LDAP connector no longer appends `<GUID=` to the object GUID. The new GUID format is compatible with objects created using the AD PowerShell connector; for example, `e1418d64-096c-4cb0-b903-ebb66562d99d` .
  In existing deployments, this could mean your links are incompatible with the new GUID format. To update links to the new format, run a reconciliation operation. To retain the legacy behavior, set `"useOldADGUIDFormat" : true` in your provisioner file.
- You cannot sort by `_id` when you return results from an Active Directory (or Active Directory LDS) server. The `_id` attribute used by default is the `objectGUID` , which is a binary attribute, and cannot be used for sorting.
- When you page and sort query results (using the `sortKeys` parameter), the `pagedResultsCookie` applies only to the first connection that makes the sorted, paginated query. Active Directory (and AD LDS) build a cached index for sorted searches, which is attached to the original connection.

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.

Manage Active Directory groups with the LDAP connector

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

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

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

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

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

The value of the `GROUP_SCOPE` attribute can be `global`, `domain`, or `universal`. If no group scope is set when the group is created, the scope is `global` by default. You can find more information about the different group scopes in 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. You can find more information about the different group types in the corresponding Microsoft documentation.

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

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

Support for nested Active Directory groups

By default, Active Directory does not return members of a nested group when querying the parent group. Some additional configuration is required to return members of a nested Active Directory group.

To include members of a nested group, create a new `nestedMembers` attribute in your provisioner file:

```
"nestedMembers" : {
    "type" : "array",
    "items" : {
        "type" : "string",
        "nativeType" : "string"
    },
    "nativeName" : "memberOf:1.2.840.113556.1.4.1941:",
    "nativeType" : "string",
    "flags" : [
        "NOT_CREATABLE",
        "NOT_UPDATEABLE",
        "NOT_RETURNED_BY_DEFAULT"
    ]
},
```

Note that the `nativeName` property includes `1.2.840.113556.1.4.1941`, which is the OID of `LDAP_MATCHING_RULE_IN_CHAIN` and `LDAP_MATCHING_RULE_TRANSITIVE_EVAL`. This is what tells Active Directory to include members of nested groups.

Querying this attribute will return results that include members of any nested groups of the queried group. For example:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
"http://localhost:8080/openidm/system/ad/account?
_queryFilter=nestedMembers+eq+"CN=ParentGroup,DC=example,DC=com"&_fields=cn"
```

Handle Active Directory dates

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

```
pwdLastSet: 130698687542272930
```

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

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

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

Multiple Active Directory domains

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

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

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

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

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

- Delete operations are not detected immediately.

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

- Not all group types are supported.

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

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

LDAP search filters

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 configuration indicates that the `accountObjectClasses` include the LDAP object classes `top`, `person`, `organizationalPerson`, and `inetOrgPerson`:

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

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

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

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

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

OpenICF Interfaces Implemented by the LDAP Connector

The LDAP Connector implements the following OpenICF interfaces. For additional details, see ICF interfaces:

Authenticate

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

Create

Creates an object and its `uid`.

Delete

Deletes an object, referenced by its `uid`.

Resolve Username

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

Schema

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

Script on Connector

Enables an application to run a script in the context of the connector.

Any script that runs on the connector has the following characteristics:

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

Search

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

Sync

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

Test

Tests the connector configuration.

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

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

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

Update

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

LDAP Connector Configuration

The LDAP Connector has the following configurable properties:

Configuration properties

Property	Type	Default	Encrypted[1]	Required[2]
`filterWithOrInsteadOfAnd`	`boolean`	`false`		- Sync
Normally the filter used to fetch change log entries is an and-based filter retrieving an interval of change entries. If this property is set, the filter will or together the required change numbers instead.				
`objectClassesToSynchronize`	`String[]`	`['inetOrPerson']`		- Sync
The object classes to synchronize. The change log is for all objects; this filters updates to just the listed object classes. You should not list the superclasses of an object class unless you intend to synchronize objects with any of the superclass values. For example, if only "inetOrgPerson" objects should be synchronized, but the superclasses of "inetOrgPerson" ("person", "organizationalperson" and "top") should be filtered out, then list only "inetOrgPerson" here. All objects in LDAP are subclassed from "top". For this reason, you should never list "top", otherwise no object would be filtered.				
`baseContextsToSynchronize`	`String[]`	`[]`		- Sync
One or more starting points in the LDAP tree that will be used to determine if a change should be synchronized. The base contexts attribute will be used to synchronize a change if this property is not set.				
`attributesToSynchronize`	`String[]`	`[]`		- Sync
The names of the attributes to synchronize. This ignores updates from the change log if they do not update any of the named attributes. For example, if only "department" is listed, then only changes that affect "department" will be processed. All other updates are ignored. If blank (the default), then all changes are processed.				
`timestampSyncOffset`	`int`	`0`		- Sync
An optional offset, specified in seconds to be negatively applied to the timestamp for Timestamp based Sync operations. Default value is 0, or no offset.				

Property	Type	Default	Encrypted[1]	Required[2]
`changeNumberAttribute`	`String`	`changeNumber`		• Sync
The name of the change number attribute in the change log entry.				
`modifiersNamesToFilterOut`	`String[]`	`[]`		• Sync
The list of names (DNs) to filter from the changes. Changes with the attribute "modifiersName" that match entries in this list will be filtered out. The standard value is the administrator name used by this adapter, to prevent loops. Entries should be of the format "cn=Directory Manager".				
`credentials`	`GuardedString`	`null`	🔒 Yes	✕ No
Password for the principal.				
`changeLogBlockSize`	`int`	`100`		• Sync
The number of change log entries to fetch per query.				
`useTimestampsForSync`	`boolean`	`false`		• Sync
If true, the connector will use the createTimestamp and modifyTimestamp system attributes to detect changes (Create/Update) on the directory instead of native change detection mechanism (cn=changelog on OpenDJ or Update Sequence Number -USN- on Active Directory for instance). Default value is false.				
`accountSynchronizationFilter`	`String`	`null`		• Sync
An optional LDAP filter for the objects to synchronize. Because the change log is for all objects, this filter updates only objects that match the specified filter. If you specify a filter, an object will be synchronized only if it matches the filter and includes a synchronized object class.				
`removeLogEntryObjectClassFromFilter`	`boolean`	`true`		• Sync
If this property is set (the default), the filter used to fetch change log entries does not contain the "changeLogEntry" object class, expecting that there are no entries of other object types in the change log.				
`alternateKeyStorePassword`	`GuardedString`	`null`	🔒 Yes	✕ No
Password to use for the alternate keystore.				

Property	Type	Default	Encrypted[1]	Required[2]
`groupSynchronizationFilter`	`String`	`null`		- Sync
An optional LDAP filter for the objects to synchronize. Because the change log is for all objects, this filter updates only objects that match the specified filter. If you specify a filter, an object will be synchronized only if it matches the filter and includes a synchronized object class.				
`groupMemberAttribute`	`String`	`uniqueMember`		✕ No
The name of the group attribute that will be updated with the distinguished name of the user when the user is added to the group.				
`accountSearchFilter`	`String`	`null`		✕ No
An optional LDAP filter to control which accounts are returned from the LDAP resource. If no filter is specified, only accounts that include all specified object classes are returned.				
`privateKeyAlias`	`String`	`null`		✕ No
Specifies the name of a private key alias from the keystore that should be used for SSL mutual authentication. If null, no private key is sent during SSL handshake so only server cert is used. This alias name is case sensitive.				
`ssl`	`boolean`	`false`		✕ No
Select the check box to connect to the LDAP server using SSL.				
`maintainPosixGroupMembership`	`boolean`	`false`		✕ No
When enabled and a user is renamed or deleted, update any POSIX groups to which the user belongs to reflect the new name. Otherwise, the LDAP resource must maintain referential integrity with respect to group membership.				
`checkAliveMinInterval`	`long`	`60`		✕ No
The minimum interval (seconds) at which the target directory is polled when a connection is reused from the pool. Defaults to 60 seconds.				
`groupSearchFilter`	`String`	`null`		✕ No
An optional LDAP filter to control which groups are returned from the LDAP resource. If no filter is specified, only groups that include all specified object classes are returned.				
`referralsHandling`	`String`	`follow`		✕ No
Defines how to handle LDAP referrals. Possible values can be follow, ignore or throw.				
`host`	`String`	`null`		✕ No

Property	Type	Default	Encrypted[1]	Required[2]
The name or IP address of the host where the LDAP server is running.				
`maintainLdapGroupMembership`	`boolean`	`false`		✕ No
When enabled and a user is renamed or deleted, update any LDAP groups to which the user belongs to reflect the new name. Otherwise, the LDAP resource must maintain referential integrity with respect to group membership.				
`resetSyncToken`	`String`	`never`		✕ No
Connector can reset the sync token if ever the value of the sync token is greater than the last change number in the directory changelog. Defaults to "never" (no reset). If set to "first" it will reset the sync token to the value of the firstChangeNumber changelog attribute. If set to "last" it will reset the sync token to the value of the lastChangeNumber changelog attribute.				
`vlvSortAttribute`	`String`	`uid`		✕ No
Specify the sort attribute to use for VLV indexes on the resource.				
`convertGTToISO8601`	`String[]`	`['whenCreated', 'whenChanged']`		✕ No
Converts the Greenwich Time to ISO8601 format.				
`baseContexts`	`String[]`	`[]`		✕ No
One or more starting points in the LDAP tree that will be used when searching the tree. Searches are performed when discovering users from the LDAP server or when looking for the groups of which a user is a member.				
`hostNameVerification`	`boolean`	`false`		✕ No
If true, the connector will verify the hostname in the certificate (subject + alternative subject) against the defined hostNameVerifierPattern.				
`blockSize`	`int`	`100`		✕ No
The maximum number of entries that can be in a block when retrieving entries in blocks.				
`groupObjectClasses`	`String[]`	`['top', 'groupOfUniqueNames']`		✕ 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.				
`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.				

Property	Type	Default	Encrypted[1]	Required[2]
`failover`	`String[]`	`[]`		✕ No
List all servers that should be used for failover in case the preferred server fails. If the preferred server fails, JNDI will connect to the next available server in the list. List all servers in the form of "ldap://ldap.example.com:389/", which follows the standard LDAP v3 URLs described in RFC 2255. Only the host and port parts of the URL are relevant in this setting.				
`port`	`int`	`389`		✕ No
TCP/IP port number used to communicate with the LDAP server.				
`convertADIntervalToISO8601`	`String[]`	`['pwdLastSet', 'accountExpires', 'lockoutTime', 'lastLogon']`		✕ No
Converts the AD Interval to ISO8601.				
`hostNameVerifierPattern`	`String`	`null`		✕ No
A simple pattern used to match the hostname from the certificate. It can contains * character (server1.example.com, *.example.com).				
`passwordAttribute`	`String`	`userPassword`		✕ No
The name of the LDAP attribute that holds the password. When changing a users password, the new password is set to this attribute.				
`useDNSSRVRecord`	`boolean`	`false`		✕ No
If true, the connector will do a DNS query to find SRV records associated with the value set for host property ("_ldap._tcp.example.com" for example). Defaults to false.				
`getGroupMemberId`	`boolean`	`false`		✕ No
Specifies whether to add an extra _memberId attribute to get the group members *UID*. CAUTION: Setting this property to true can incur a large performance cost on group handling.				
`lastCheckAlive`	`long`	`1751650574794`		✕ No
The last time the connector was checked to see if it was alive				
`ldapGroupsUseStaticGroups`	`boolean`	`false`		✕ No
When set to true, The ldapGroups attribute will search group membership through static groups only. If false, it will leverage the "memberOf" attribute of an object (defaults to true).				
`startTLS`	`boolean`	`false`		✕ No

Property	Type	Default	Encrypted[1]	Required[2]
Specifies whether to use the startTLS operation to initiate a TLS/SSL session.				
`allowTreeDelete`	`boolean`	`false`		✕ No
Connector can delete an entry (node) with leaf entry if this value is set to true (defaults to false). The LDAP control LDAP_SERVER_TREE_DELETE_OID (1.2.840.113556.1.4.805) is used.				
`respectResourcePasswordPolicyChangeAfterReset`	`boolean`	`false`		✕ No
When this resource is specified in a Login Module (i.e., this resource is a pass-through authentication target) and the resource's password policy is configured for change-after-reset, a user whose resource account password has been administratively reset will be required to change that password after successfully authenticating.				
`uidAttribute`	`String`	`entryUUID`		✕ No
The name of the LDAP attribute that is mapped to the OpenICF UID attribute.				
`principal`	`String`	`null`		✕ No
The distinguished name with which to authenticate to the LDAP server.				
`accountObjectClasses`	`String[]`	`['top', 'person', 'organizationalPerson', 'inetOrgPerson' ]`		✕ No
The default list of object classes that will be used when creating new user objects in the LDAP tree. This can be overridden by specifying the user object classes during the Create operation.				
`alternateKeyStoreType`	`String`	`null`		✕ No
Defines the type of the alternate key store. Valid values are JKS, JCEKS and PKCS12.				
`passwordHashAlgorithm`	`String`	`null`		✕ No
Indicates the algorithm that the Identity system should use to hash the password. The only supported value is WIN-AD (for when Active Directory is the target). A blank value indicates that the system will not hash passwords. This will cause clear text passwords to be stored in LDAP unless the LDAP server performs the hash (as Forgerocks OpenDJ does, for example).				
`alternateKeyStore`	`String`	`null`		✕ No
Defines the filename of an alternate keystore. If specified, the connector will not use the default keystore specified by the javax.net.ssl.keyStore property.				
`authType`	`String`	`simple`		✕ No

Property	Type	Default	Encrypted[1]	Required[2]
The authentication mechanism to use: Simple, EXTERNAL (mTLS) or SASL-GSSAPI (Kerberos). Defaults to "simple".				
`connectionTimeout`	`int`	`30000`		✕ No
The timeout (in ms) before the connection attempt is aborted.				
`customOctetStringAttributes`	`String[]`	`[]`		✕ No
A list of custom octet string attributes to be declared so that the LDAP connector manipulates these attributes as byte arrays.				
`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.				
`readSchema`	`boolean`	`true`		✕ No
If true, the connector will read the schema from the server. If false, the connector will provide a default schema based on the object classes in the configuration. This property must be true in order to use extended object classes.				
`usePagedResultControl`	`boolean`	`false`		✕ No
When enabled, the LDAP Paged Results control is preferred over the VLV control when retrieving entries. If disabled, paged queries will be ignored.				
`useOldADGUIDFormat`	`boolean`	`false`		✕ No
The connector used to transform the AD ObjectGUID in the form <GUID=xxxxxx>. It now used dashed notation (xxxx-xx-xx-xxxx-xxxxxx) by default. Set to true to keep the old format.				
`sendCAUDTxId`	`boolean`	`false`		✕ No
Connector can send the Common Audit Transaction Id (if present) to the target OpenDJ server when this value is set to true (defaults to false). The LDAP control TransactionIdControl (1.3.6.1.4.1.36733.2.1.5.1) is used.				
`gssapiLoginContext`	`String`	`null`		✕ No
Defines the name used in the JAAS configuration file to define the JAAS login configuration. If null, it defaults to "org.identityconnectors.ldap.LdapConnector".				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

Configuring `ldapGroups` in the LDAP connector

The LDAP Connector includes a special virtual attribute, `ldapGroups`, which simplifies managing LDAP group memberships. This topic explains the functionality of `ldapGroups`, provides configuration instructions, details important considerations, and best practices.

LDAP group membership

In standard LDAP directories, such as PingDS or Active Directory (AD), directory servers typically store group membership information on the **group** entry, not the **user** entry. A group object contains a list of its members, often as distinguished names (DNs) pointing to user entries (for example, using the `member` or `uniqueMember` attribute).

By default, a user entry doesn't list the groups to which it belongs, so a client application could need to search numerous group entries to determine a user's group memberships. To simplify this process, most directory servers provide a **computed** attribute on the user entry that dynamically lists the groups where the user is a member.

- In **PingDS**, this attribute is `isMemberOf`.
- In **AD**, this attribute is `memberOf`.

These attributes are valuable for reading membership information but are **read-only**. You cannot directly modify a user's `memberOf` or `isMemberOf` attribute to change group memberships; you must modify the actual group entry's member list.

Example: `memberOf` in AD

A query for an AD user might return:

```
dn: CN=user five,OU=test1,DC=example,DC=com
...
memberOf: CN=testgroup,OU=test1,DC=example,DC=com
...
```

Example: `isMemberOf` in PingDS (DS)

A query for a DS user, requesting the `isMemberOf` attribute, might return:

```
dn: uid=user.3,ou=People,dc=example,dc=com
...
isMemberOf: cn=Test Group,ou=groups,dc=example,dc=com
...
```

> **Note**
>
> Directory servers often do not return these calculated attributes by default because their computation incurs a performance cost. You typically need to request these attributes in LDAP searches.

`ldapGroups`: A connector-level solution

The `ldapGroups` attribute is a feature **specific to the LDAP Connector**. It functions as a writable proxy for managing group memberships directly using the user object within IDM.

When you map and use `ldapGroups`:

- **Reads (GET operations):** The connector determines the user's group memberships. By default, it does this by searching group entries on the target directory for the user's DN (for example, searching for `(member=<userDN>)` or `(uniqueMember=<userDN>)`).
- **Writes (Update operations):** When you add or remove group DNs from the `ldapGroups` attribute in IDM and save the user object, the connector translates these changes into the necessary LDAP modify operations against the **group** entries on the target directory (adding or removing the user's DN from the group's member list).

This mechanism lets you manage group assignments within IDM (for example, through role assignments) without directly interacting with group objects.

Configuration

To use `ldapGroups`, configure it in the LDAP connector's provisioner file (`provisioner.openicf-ldap.json` or similar) and include it in the user object mapping.

1. Add `ldapGroups` to the connector schema

Add the following definition to the `account` object type within the connector configuration's `objectTypes` section:

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

2. Update user mapping

The goal of this step is to produce a list of group DN values for `ldapGroups` during the synchronization process. There are a variety of methods which you can use, including:

- Property mappings
- An `onUpdate` script
- Role assignment processing

Ensure the synchronization mapping (`sync.json`) for the user object includes a mapping for `ldapGroups`. For example:

```
{
    "source": "ldapGroups",
    "target": "ldapGroups"
}
```

If you manage assignments using roles, configure the role assignment to target the `ldapGroups` attribute. You can also grant roles to users directly. Learn more in Roles in the PingIDM documentation.

Example: Assigning AD groups using roles

1. Configure the connector and mapping as described previously.
2. Create an internal role (for example, "AD Finance Group Users").
3. In the admin UI, navigate to the role's **Managed Assignments** (or equivalent section).
4. Create a new assignment targeting the appropriate AD connector or mapping.
5. Select the `ldapGroups` attribute.
6. Select the desired AD group (for example, `CN=Finance Users,OU=Groups,DC=example,DC=com`) from the provided list.
7. Save the assignment and the role.

Assigning this IDM role to a user automatically adds that user to the specified AD group. Learn more in Working with role assignments in the PingIDM documentation.

Performance and functional considerations

Although `ldapGroups` offers convenience, it's crucial to understand its implications.

Performance cost

- **Default read behavior:** By default, `ldapGroups` requires the connector to search group entries on the target directory. This makes read operations (such as GET User or reconciliation) consume more resources. The connector performs an additional search for each user to identify the **static** groups they belong to. This slows performance when querying for many users with `ldapGroups` included or, to a lesser extent, users who are members of many groups.
- **Update behavior:** Updates involving `ldapGroups` also add overhead. The connector must determine the user's current group memberships (often involving another search), calculate the difference between the current state and the desired state, and then execute LDAP modify operations on the relevant group entries. This can slow performance when updating groups with many members.

Functional limitations (default behavior)

- **Static groups only:** By default, `ldapGroups` discovers and manages only memberships in **static** groups (those that have explicit `member` or `uniqueMember` attributes). It does **not** recognize or manage memberships from:
- **Dynamic groups:** Groups where rules or LDAP URLs determine membership (common in DS).
- **Nested groups:** Groups that contain other groups as members.
- **Incomplete view:** Relying **only** on `ldapGroups` (using default settings) to ascertain a user's effective group memberships might give an incomplete picture if your environment uses dynamic or nested groups.

`ldapGroups` compared to `memberOf` and `isMemberOf`

Feature	`ldapGroups`	`memberOf` and `isMemberOf`

Purpose	Read/write proxy for group membership	Read-only view of group membership
Scope	Static groups only	All group types (static, dynamic, nested)
Writable?	Yes (using connector logic)	No (read-only virtual attribute)
Read cost	Higher (connector searches groups)	Lower (directory calculates; often faster)
Managed by	LDAP Connector	Target directory server

The `ldapGroupsUseStaticGroups` parameter

To address performance and functional limitations, the LDAP connector includes the `ldapGroupsUseStaticGroups` configuration parameter (located in the main connector configuration file, `provisioner.openicf-ldap.json`).

- `false` **(default):** `ldapGroups` operates as described previously, searching static group entries. This approach is safer for updates involving mixed group types but has performance drawbacks and limits visibility to static groups.
- `true` **:** `ldapGroups` uses the directory's `memberOf` (AD) or `isMemberOf` (DS) virtual attribute for **reading** group memberships.
- **Pros:** Improves read and query performance; includes dynamic and nested groups in the results that `ldapGroups` returns.
- **Cons: Presents high risk for updates.** If IDM tries to modify memberships derived from dynamic or nested groups (which `isMemberOf` or `memberOf` might return), the connector could attempt inappropriate LDAP operations (such as modifying a dynamic group as if it were static). This can lead to errors (for example, DS schema violation code 65) and failed updates. Use this setting only if the environment exclusively uses static groups and you have thoroughly tested it.

Base context filtering

- When `ldapGroupsUseStaticGroups` is `false` (default), the connector's search for static groups follows the `baseContexts` defined in the connector configuration. The connector considers only groups within these specified base contexts.
- When `ldapGroupsUseStaticGroups` is `true` , the `memberOf` or `isMemberOf` attribute returns **all** groups the user belongs to, regardless of the connector's `baseContexts` setting. This happens because the directory server typically does not filter the virtual attribute itself by base context.

AD considerations

- AD primarily uses static groups, which makes the default `ldapGroups` behavior suitable.
- AD's `memberOf` attribute does **not** natively display nested memberships unless the query includes a specific LDAP control (LDAP_SERVER_CHAINING_OID: `1.2.840.113556.1.4.1941`). The connector's use of `memberOf` (when `ldapGroupsUseStaticGroups` is true) might or might not retrieve nested memberships, depending on internal implementation specifics.

Recommendations and best practices

1. **Understand the trade-offs:** Recognize that `ldapGroups` offers convenience but affects performance and potentially adds complexity, particularly when different group types exist in the environment.

2. **Prefer `memberOf` or `isMemberOf` for reading:** To view a user's complete group memberships, you should map the directory's native `memberOf` or `isMemberOf` attribute as read-only (`"flags": ["NOT_CREATABLE", "NOT_UPDATEABLE"]`). This method is generally more efficient and comprehensive than reading `ldapGroups` .

3. **Use `ldapGroups` cautiously for writing:** If IDM must manage group memberships:

    - Keep the `ldapGroupsUseStaticGroups` setting as `false` (the default) unless you confirm the environment uses **only** static groups and you've performed comprehensive testing.

        > **Tip**
        >
        > If you can't guarantee that only static groups are used and want to update the static groups, use a transform script. Create logic which filters all of the non-static groups from the `isMemberOf` group DN list. This allows you make reasonable updates to static groups without the high performance cost of reading `ldapGroups`.

    - Consider adding the `"flags": ["NOT_RETURNED_BY_DEFAULT"]` to the `ldapGroups` definition within the provisioner file. This flag prevents the performance degradation that calculating `ldapGroups` causes during default read operations, such as reconciliation synchronization checks, while still allowing you to use it explicitly in updates or specific requests.

    - Add the `NOT_RETURNED_BY_DEFAULT` flag to optimize read performance:

        ```
        {
          "ldapGroups": {
            "type": "array",
            "nativeType": "string",
            "nativeName": "ldapGroups",
            "required": false,
            "items": {
              "type": "string",
              "nativeType": "string"
            },
            "flags": [
              "NOT_RETURNED_BY_DEFAULT"
            ]
          }
        }
        ```

4. **Index directory attributes:** Ensure the directory server indexes the `member` (AD) or `uniqueMember` (DS) attributes on group objects. Indexing is critical to lessen the performance impact when `ldapGroups` searches static groups (`ldapGroupsUseStaticGroups: false`). Consult the directory server documentation for indexing best practices.

5. **Alternative: Manage group objects:** The most technically direct LDAP approach involves managing memberships by modifying the group objects directly within IDM (assuming you synchronize group objects). However, this method often requires more complex configuration and scripting compared to using `ldapGroups` through role assignments.

By understanding how `ldapGroups` operates and its associated trade-offs, you can configure it effectively to meet group management requirements while minimizing performance impacts and avoiding potential issues.

Marketo connector

The Marketo connector lets you synchronize between IDM managed users and a Marketo leads database. You can synchronize any managed user to Marketo—those who have been added directly to the IDM repository, and those who have registered themselves through a Social Identity Provider.

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

To use the Marketo connector, you need:

- A Marketo account
- A client ID and client secret
- The REST API URL for your IDM service
- A custom list created in your Marketo leads database

To obtain these details from Marketo, refer to the Marketo documentation.

Install the Marketo connector

> **Tip**
>
> To check for an Advanced Identity Cloud application for this connector, refer to:
>
> - Application management
> - App catalog

You can download any connector from Backstage, but some are included in the default deployment for Advanced Identity Cloud, IDM, or RCS. When using an included connector, you can skip installing it and move directly to configuration.

Connector included in default deployment

Connector	IDM	RCS
Marketo	✓ Yes	✕ No

Download the connector .jar file from Backstage.

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

```
mv ~/Downloads/marketo-connector-1.5.20.29.jar /path/to/openidm/connectors/
```

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

Configure the Marketo connector

Create a connector configuration using the IDM admin UI:

1. From the navigation bar, click **Configure > Connectors**.
2. On the **Connectors** page, click **New Connector**.
3. On the **New Connector** page, type a **Connector Name**.
4. From the **Connector Type** drop-down list, select **Marketo Connector - 1.5.20.29**.
5. Complete the **Base Connector Details**.

   > **Tip**
   >
   > For a list of all configuration properties, refer to Marketo Connector Configuration

6. Click **Save**.

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

Refer to this procedure to create a connector configuration over REST.

Alternatively, configure the connector with a configuration file.

A sample connector configuration file is bundled with IDM, provided at `/path/to/openidm/samples/example-configurations/provisioners/provisioner.openicf-marketo.json`. Copy the sample connector configuration file to your project's `conf/` directory.

This sample connector configuration shows the mandatory properties:

```
{
    "displayName" : "MarketoConnector",
    "description" : "Connector used to sync users to Marketo leads",
    "author" : "ForgeRock",
    "enabled" : true,
    "connectorRef" : {
        "bundleName" : "org.forgerock.openicf.connectors.marketo-connector",
        "bundleVersion" : "[1.5.0.0,1.6.0.0)",
        "connectorName" : "org.forgerock.openicf.connectors.marketo.MarketoConnector"
    },
    ...
    "configurationProperties" : {
        "instance" : "<INSTANCE_FQDN>",
        "clientId" : "<CLIENT_ID>",
        "clientSecret" : "<CLIENT_SECRET>",
        "leadFields" : null,
        "partitionName" : null,
        "listName" : "<LEAD_LIST_NAME>",
        "scriptRoots" : [
            "jar:file:connectors/marketo-connector-1.5.20.29.jar!/scripts/marketo/"
        ],
        ...
    },
    ...
}
```

instance

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

clientId

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

clientSecret

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

listName

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

scriptRoots

The path to the Groovy scripts that perform the ICF operations, relative to your installation directory. For the Marketo connector, the scripts are bundled in the connector .jar file, so the sample connector configuration uses the path `jar:file:connectors/marketo-connector-1.5.20.29.jar!/scripts/marketo/`.

Tip

For a list of all configuration properties, refer to Marketo Connector Configuration.

Test the Marketo connector

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

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system?_action=test"
[
  {
    "name": "marketo",
    "enabled": true,
    "config": "config/provisioner.openicf/marketo",
    "objectTypes": [
      "__ALL__",
      "account"
    ],
    "connectorRef": {
      "bundleName": "org.forgerock.openicf.connectors.marketo-connector",
      "connectorName": "org.forgerock.openicf.connectors.marketo.MarketoConnector",
      "bundleVersion": "[1.5.0.0,1.6.0.0)"
    },
    "displayName": "Marketo Connector",
    "ok": true
  }
]
```

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

Marketo remote connector

If you want to run this connector outside of PingOne Advanced Identity Cloud or IDM, you can configure the Marketo connector as a remote connector. Java Connectors installed remotely on a Java Connector Server function identically to those bundled locally within PingOne Advanced Identity Cloud or installed locally on IDM.

You can download the Marketo connector from here.

Refer to Remote connectors for configuring the Marketo remote connector.

Configure connection pooling

The Marketo connector supports HTTP pooling, which can substantially improve the performance of the connector. Learn more about the basic connection pooling configuration and different pooling mechanisms described in Connection pooling configuration.

> **Note**
>
> The Marketo connector works with the default values that the `HTTPClient` library provides. It doesn't use SaaS configuration or specific pooling properties.

Reconcile users with a Marketo leads database

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

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

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

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

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

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

When a user registers with IDM, they can choose to accept this condition. As a regular user, they can also select (or deselect) the condition in the End User UI by logging into IDM at `http://localhost:8080/`, and selecting **Preferences**.

If a user deselects the marketing preference after their account has been reconciled to Marketo, the next reconciliation run will remove the account from the Marketo database.

For more information on how preferences work in a mapping, refer to User preferences.

Implementation specifics

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

OpenICF Interfaces Implemented by the Marketo Connector

The Marketo Connector implements the following OpenICF interfaces. For additional details, see ICF interfaces:

Authenticate

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

Create

Creates an object and its `uid`.

Delete

Deletes an object, referenced by its `uid`.

Resolve Username

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

Schema

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

Script on Connector

Enables an application to run a script in the context of the connector.

Any script that runs on the connector has the following characteristics:

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

Script on Resource

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

Search

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

Sync

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

Test

Tests the connector configuration.

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

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

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

Update

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

Marketo Connector Configuration

The Marketo Connector has the following configurable properties:

Configuration properties

Property	Type	Default	Encrypted[1]	Required[2]
`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.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

Groovy Engine configuration

Property	Type	Default	Encrypted[1]	Required[2]
`scriptRoots`	`String[]`	`['!/scripts/marketo/']`		✓ Yes
The root folder to load the scripts from. If the value is null or empty the classpath value is used.				
`classpath`	`String[]`	`[]`		✕ No
Classpath for use during compilation.				

Property	Type	Default	Encrypted[1]	Required[2]
`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.				
`minimumRecompilationInterval`	`int`	`100`		✕ No
Sets the minimum of time after a script can be recompiled.				
`recompileGroovySource`	`boolean`	`false`		✕ No
If set to true recompilation is enabled.				
`scriptBaseClass`	`String`	`null`		✕ No
Base class name for scripts (must derive from Script).				
`scriptExtensions`	`String[]`	`['groovy']`		✕ No
Gets the extensions used to find groovy files.				
`sourceEncoding`	`String`	`UTF-8`		✕ No
Encoding for source files.				
`targetDirectory`	`File`	`null`		✕ No
Directory into which to write classes.				
`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.				
`verbose`	`boolean`	`false`		✕ No
If true, the compiler should produce action information.				
`warningLevel`	`int`	`1`		✕ No
Warning Level of the compiler.				
`customConfiguration`	`String`	`null`		✕ No

Property	Type	Default	Encrypted[1]	Required[2]
Custom Configuration script for Groovy ConfigSlurper.				
`customSensitiveConfiguration`	`GuardedString`	`null`	🔒 Yes	✕ No
Custom Sensitive Configuration script for Groovy ConfigSlurper.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

Operation Script Files

Property	Type	Default	Encrypted[1]	Required[2]
`authenticateScriptFileName`	`String`	`null`		• Authenticate
The name of the file used to perform the AUTHENTICATE operation.				
`createScriptFileName`	`String`	`CreateMarketo.groovy`		• Create
The name of the file used to perform the CREATE operation.				
`customizerScriptFileName`	`String`	`null`		✕ No
The script used to customize some function of the connector. Read the documentation for more details.				
`deleteScriptFileName`	`String`	`DeleteMarketo.groovy`		• Delete
The name of the file used to perform the DELETE operation.				
`resolveUsernameScriptFileName`	`String`	`null`		• Resolve Username
The name of the file used to perform the RESOLVE_USERNAME operation.				
`schemaScriptFileName`	`String`	`SchemaMarketo.groovy`		• Schema
The name of the file used to perform the SCHEMA operation.				

Property	Type	Default	Encrypted[1]	Required[2]
`scriptOnResourceScriptFileName`	`String`	`null`		• Script on Resource
The name of the file used to perform the RUNSCRIPTONRESOURCE operation.				
`searchScriptFileName`	`String`	`SearchMarketo.groovy`		• Read • Search
The name of the file used to perform the SEARCH operation.				
`syncScriptFileName`	`String`	`null`		• Sync
The name of the file used to perform the SYNC operation.				
`testScriptFileName`	`String`	`TestMarketo.groovy`		• Test
The name of the file used to perform the TEST operation.				
`updateScriptFileName`	`String`	`UpdateMarketo.groovy`		• Update
The name of the file used to perform the UPDATE operation.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

Microsoft Graph API connector

The Microsoft (MS) Graph API connector uses the Microsoft Graph SDK for Java and the Authentication Providers for the Microsoft Graph Java SDK. Unlike the PowerShell connector for Azure, the MS Graph API connector is a Java connector and does not need .NET RCS to run. As a Java connector, the MS Graph API connector functions like any standard IDM connector.

The MS Graph API connector can read, search, and fetch data from Microsoft Azure when Azure is the authoritative data source. It can also provision to Azure when IDM or Advanced Identity Cloud is the authoritative data source.

The MS Graph API connector is bundled with IDM and Advanced Identity Cloud, and is also available from the Backstage. The connector bundles all its dependencies.

Important

Microsoft Graph API doesn’t support paging for all relationships/resources. Refer to the Microsoft Documentation.

Next steps:

1. Install and configure the connector.

2. Use the connector.

Install and configure the MS Graph API connector

Microsoft Azure requirements

Before you can use the connector, you must register an application with Azure. You need a Microsoft Azure subscription to complete this procedure:

1. Log in to the MS Azure portal as an administrative user.

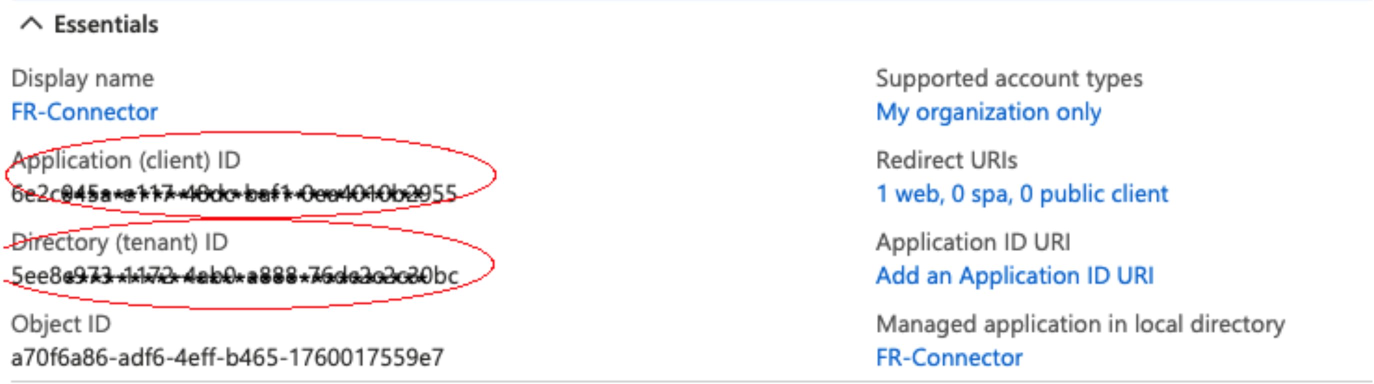
2. Under **Azure services** , select **App registrations**.

3. On the **Register an application** page, enter a name for the application; for example, **FR-Connector**.

4. Select the supported account types, and enter a **Redirect URI**.

   The redirect URI is the IDM URI that Azure should redirect to after successful authentication; for example, `https://idm.example.com:8443/` .

5. On the new registration page for your application, make a note of the **Application (client) ID** and the **Directory (tenant) ID**. You will need these to configure the connector:

   ^ Essentials

   Display name
   FR-Connector

   Application (client) ID
   6e2c[redacted]55

   Directory (tenant) ID
   5ee8[redacted]bc

   Object ID
   a70f6a86-adf6-4eff-b465-1760017559e7

   Supported account types
   My organization only

   Redirect URIs
   1 web, 0 spa, 0 public client

   Application ID URI
   Add an Application ID URI

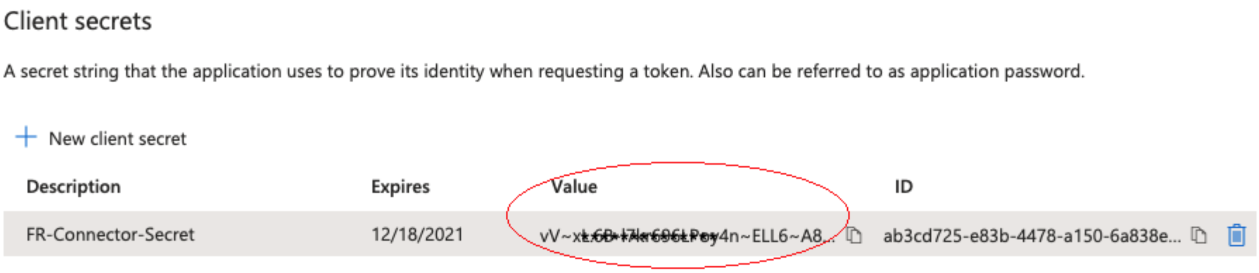
   Managed application in local directory
   FR-Connector

6. Generate a client secret:

   1. Select **Certificates & secrets** > **New client secret** .

   2. Enter a description, select an expiration date, and click **Add**.

   3. Copy the client secret **Value**:



Client secrets

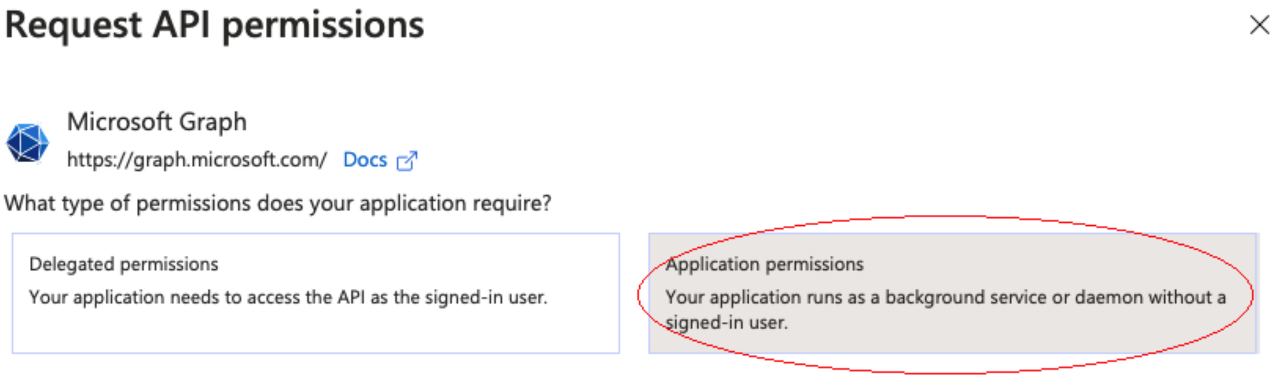
A secret string that the application uses to prove its identity when requesting a token. Also can be referred to as application password.

Description	Expires	Value	ID
FR-Connector-Secret	12/18/2021	vV~x[redacted]y4n~ELL6~A8...	ab3cd725-e83b-4478-a150-6a838e...

> **Important**
>
> You will not be able to retrieve the client secret in cleartext after you exit this screen.

7. Set the API permissions:

   1. Select **API permissions**, click **Microsoft Graph**, and then click **Application permissions**.



Request API permissions

Microsoft Graph

https://graph.microsoft.com/ Docs

What type of permissions does your application require?

Delegated permissions

Your application needs to access the API as the signed-in user.

Application permissions

Your application runs as a background service or daemon without a signed-in user.

   2. From the **User** item, select the following permissions:

      - `User.Export.All`
      - `User.ManageIdentities.All`
      - `User.Read.All`
      - `User.ReadWrite.All`

   3. From the **Group** item, select the following permissions:

      - `Group.Create`
      - `Group.Read.All`
      - `Group.ReadWrite.All`

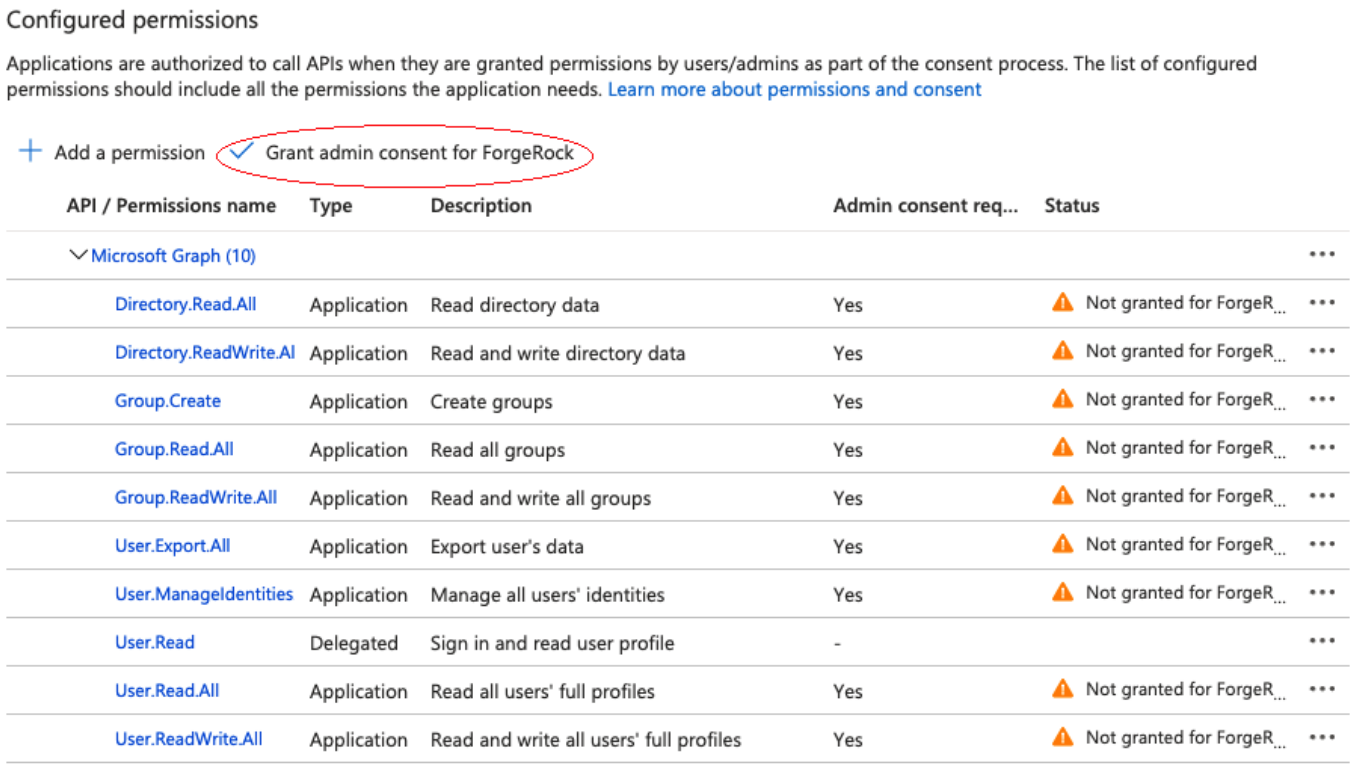
   4. From the **Directory** item, select the following permissions:

      - `Directory.Read.All`
      - `Directory.ReadWrite.All`

   5. Click **Add permissions** .

8. Grant admin consent for the API permissions:

On the **Configured permissions** page, **Grant admin consent for org-name**, then click **Yes**.



Configured permissions

Applications are authorized to call APIs when they are granted permissions by users/admins as part of the consent process. The list of configured permissions should include all the permissions the application needs. Learn more about permissions and consent

API / Permissions name	Type	Description	Admin consent req...	Status	
Microsoft Graph (10)					•••
Directory.Read.All	Application	Read directory data	Yes	⚠ Not granted for ForgeR...	•••
Directory.ReadWrite.Al	Application	Read and write directory data	Yes	⚠ Not granted for ForgeR...	•••
Group.Create	Application	Create groups	Yes	⚠ Not granted for ForgeR...	•••
Group.Read.All	Application	Read all groups	Yes	⚠ Not granted for ForgeR...	•••
Group.ReadWrite.All	Application	Read and write all groups	Yes	⚠ Not granted for ForgeR...	•••
User.Export.All	Application	Export user's data	Yes	⚠ Not granted for ForgeR...	•••
User.ManageIdentities	Application	Manage all users' identities	Yes	⚠ Not granted for ForgeR...	•••
User.Read	Delegated	Sign in and read user profile	-		•••
User.Read.All	Application	Read all users' full profiles	Yes	⚠ Not granted for ForgeR...	•••
User.ReadWrite.All	Application	Read and write all users' full profiles	Yes	⚠ Not granted for ForgeR...	•••

Install the MS Graph API connector

> **Tip**
>
> To check for an Advanced Identity Cloud application for this connector, refer to:
>
> - Application management
> - App catalog

You can download any connector from Backstage, but some are included in the default deployment for Advanced Identity Cloud, IDM, or RCS. When using an included connector, you can skip installing it and move directly to configuration.

Connector included in default deployment

Connector	IDM	RCS
Microsoft Graph API	✓ Yes	✕ No

Download the connector .jar file from Backstage.

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

```
mv ~/Downloads/msgraphapi-connector-1.5.20.30.jar /path/to/openidm/connectors/
```

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

Configure the MS Graph API connector

Create a connector configuration using the IDM admin UI:

1. From the navigation bar, click **Configure > Connectors**.
2. On the **Connectors** page, click **New Connector**.
3. On the **New Connector** page, type a **Connector Name**.
4. From the **Connector Type** drop-down list, select **MS Graph API Connector - 1.5.20.30**.
5. Complete the **Base Connector Details**.

> **Tip**
>
> For a list of all configuration properties, refer to MS Graph API Connector Configuration

6. Click **Save**.

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

Refer to this procedure to create a connector configuration over REST.

Alternatively, copy the sample connector configuration file from `/path/to/openidm/samples/example-configurations/provisioners/provisioner.openicf-azuread.json` to your project's `conf/` directory.

MS Graph API authentication configuration

You can configure authentication using one of the following methods:

Direct authentication configuration

Set the Azure `tenant`, `clientId` and `clientSecret` in the connector provisioner `configurationProperties`. For example:

```
"configurationProperties" : {
    "tenant" : "your tenant ID",
    "clientId" : "your client ID",
    "clientSecret" : "your client secret"
}
```

Environment-based authentication configuration [1]

If you don't specify the `tenant`, `clientId`, and `clientSecret` properties in the connector configuration, the connector uses the Azure Identity SDK's `DefaultAzureCredential` provider to search for credentials in the connector's system environment.

The provider searches for the following environment variables:

- `AZURE_TENANT_ID`
- `AZURE_CLIENT_ID`

- `AZURE_CLIENT_SECRET`

This feature supports other authentication mechanisms available through the Azure Identity SDK, such as `EnvironmentCredential`, `WorkloadIdentityCredential`, and `ManagedIdentityCredential`. Create the configuration for these mechanisms in the system's environment, not in the connector configuration.

MS Graph API remote connector

If you want to run this connector outside of PingOne Advanced Identity Cloud or IDM, you can configure the MS Graph API connector as a remote connector. Java Connectors installed remotely on a Java Connector Server function identically to those bundled locally within PingOne Advanced Identity Cloud or installed locally on IDM.

You can download the MS Graph API connector from here.

Refer to Remote connectors for configuring the MS Graph API remote connector.

Configure connection pooling

The MS Graph API connector uses ICF pooling to manage connections. Learn more about the different pooling mechanisms in Connectors by pooling mechanism.

Test the connector

One simple method for testing the connector configuration is using the `test` action on the `openidm/system` endpoint:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system?_action=test"
{
  "name": "azuread",
  "enabled": true,
  "config": "config/provisioner.openicf/azuread",
  "connectorRef": {
    "bundleVersion": "[1.5.0.0,1.6.0.0)",
    "bundleName": "org.forgerock.openicf.connectors.msgraphapi-connector",
    "connectorName": "org.forgerock.openicf.connectors.msgraphapi.MSGraphAPIConnector"
  },
  "displayName": "MSGraphAPI Connector",
  "objectTypes": [
    "servicePrincipal",
    "__GROUP__",
    "roleEligibilitySchedule",
    "roleEligibilityScheduleInstance",
    "__ALL__",
    "roleEligibilityScheduleRequest",
    "directoryRole",
    "team",
    "roleAssignmentSchedule",
    "roleDefinition",
    "servicePlan",
    "directoryRoleTemplate",
    "application",
    "roleAssignmentScheduleRequest",
    "roleAssignmentScheduleInstance",
    "subscribedSku",
    "__ACCOUNT__",
    "roleAssignment"
  ],
  "ok": true (1)
}
```

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

Synchronize accounts between IDM and Azure

To use the MS Graph API connector to synchronize accounts between IDM and Azure, set up a mapping between the two data stores.

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

OpenICF Interfaces Implemented by the MSGraphAPI Connector

The MSGraphAPI Connector implements the following OpenICF interfaces. For additional details, see ICF interfaces:

Authenticate

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

Create

Creates an object and its `uid`.

Delete

Deletes an object, referenced by its `uid`.

Schema

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

Script on Connector

Enables an application to run a script in the context of the connector.

Any script that runs on the connector has the following characteristics:

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

Search

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

Sync

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

Test

Tests the connector configuration.

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

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

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

Update

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

MSGraphAPI Connector Configuration

The MSGraphAPI Connector has the following configurable properties:

Basic Configuration Properties

Property	Type	Default	Encrypted(1)	Required(2)
`tenant`	`String`	`null`		✓ Yes
The Azure AD tenant name or id.				
`clientId`	`String`	`null`		✓ Yes
The clientID used by the connector during the OAuth flow.				
`clientSecret`	`GuardedString`	`null`	🔒 Yes	✕ No
The client secret used by the connector during the OAuth flow.				
`httpProxyHost`	`String`	`null`		✕ No
The Http proxy host.				
`httpProxyPort`	`Integer`	`null`		✕ No
The Http proxy port.				
`httpProxyUsername`	`String`	`null`		✕ No
The Http proxy user name.				
`httpProxyPassword`	`GuardedString`	`null`	🔒 Yes	✕ No
The Http proxy user password.				
`performHardDelete`	`boolean`	`false`		✕ No
If set to true, the Azure object will be deleted permanently on delete operation.				
`readRateLimit`	`String`	`null`		✕ No
Defines throttling for read operations either per seconds ("30/sec") or per minute ("100/min").				
`writeRateLimit`	`String`	`null`		✕ No
Defines throttling for write operations (create/update/delete) either per second ("30/sec") or per minute ("100/min").				
`licenseCacheExpiryTime`	`Long`	`null`		✕ No

Property	Type	Default	Encrypted[1]	Required[2]
Defines the expiry time for cached license information (in minutes).				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

1. MS Graph API connector version 1.5.20.30 and later supports environment-based authentication configuration.

Use the connector (MS Graph API)

After you configure the MS Graph API connector, you can use it to synchronize, read, and edit resources:

- Users & groups
- Service plans
- Licenses
- Contacts
- Role eligibility schedules
- Role assignment schedules
- Applications
- `servicePrincipal`
- Application permissions
- Authentication methods

> **Important**
>
> Microsoft Graph API doesn’t support paging for all relationships or resources. Refer to the Microsoft Documentation.

Users and groups (MS Graph API)

You can use the MS Graph API connector to list, create, update, and delete users and groups.

List user entries

This command retrieves a list of users in your Azure tenant. You can also use any system-enabled filter, such as those described in Construct Queries:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
"http://localhost:8080/openidm/system/azuread/user?_queryId=query-all-ids"
{
  "result": [
    {
      "_id": "c48be8cc-5846-4059-95e8-a7acbf6aec31"
    },
    {
      "_id": "c7fe57e2-3159-45e1-b67a-435232fd88d9"
    },
    {
      "_id": "9e714b5c-345a-430c-93f5-d8c6f9a2f225"
    },
    ...
  ],
  ...
}
```

Return a user entry

This command retrieves a specific user entry from your Azure tenant:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
"http://localhost:8080/openidm/system/azuread/user/c48be8cc-5846-4059-95e8-a7acbf6aec31"
{
  "_id": "c48be8cc-5846-4059-95e8-a7acbf6aec31",
  "surname": "Jensen",
  "displayName": "Babs Jensen",
  "memberOf": [
    "036f288c-6f71-41ae-9d09-6a68c8ba315b"
  ],
  "mail": "babs.jensen@example.onmicrosoft.com",
  "onPremisesExtensionAttributes": {
    ...
  },
  "usageLocation": "FR",
  "userType": "Member",
  "identities": [
    {
      "signInType": "userPrincipalName",
      "issuerAssignedId": "00991235@example.onmicrosoft.com",
      "issuer": "example.onmicrosoft.com"
    }
  ],
  "businessPhones": [],
  "createdDateTime": "2020-11-20T11:09:15Z",
  "accountEnabled": true,
  "userPrincipalName": "00991235@example.onmicrosoft.com",
  "proxyAddresses": [
    "smtp:00991235@example.onmicrosoft.com",
    "SMTP:babs.jensen@example.onmicrosoft.com"
  ],
  "imAddresses": [],
  "passwordPolicies": "None",
  "mailNickname": "00991235",
  "givenName": "Babs",
  "__NAME__": "00991235@example.onmicrosoft.com"
}
```

Create users or groups

This command creates a new user in your Azure tenant:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
--header "content-type: application/json" \
--data '{
  "surname": "Carter",
  "displayName": "Steve Carter",
  "givenName": "Steve",
  "userType": "Member",
  "accountEnabled": true,
  "mailNickname": "00654321",
  "userPrincipalName": "00654321@forgedemo.onmicrosoft.com",
  "__PASSWORD__": "MyPassw0rd"
}' \
"http://localhost:8080/openidm/system/azuread/user?_action=create"
{
  "_id": "9fa6c765-0872-45f6-8714-1dcd1ed94859",
  "surname": "Carter",
  "displayName": "Steve Carter",
  "memberOf": [],
  "onPremisesExtensionAttributes": {
    "extensionAttribute14": null,
    ...
  },
  "userType": "Member",
  "identities": [
    {
      "signInType": "userPrincipalName",
      "issuerAssignedId": "00654321@example.onmicrosoft.com",
      "issuer": "example.onmicrosoft.com"
    }
  ],
  "businessPhones": [],
  "createdDateTime": "2020-12-18T13:23:58Z",
  "accountEnabled": true,
  "userPrincipalName": "00654321@example.onmicrosoft.com",
  "proxyAddresses": [],
  "imAddresses": [],
  "mailNickname": "00654321",
  "givenName": "Steve",
  "__NAME__": "00654321@example.onmicrosoft.com"
}
```

Update entries

This command changes the password for the user created previously:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request PATCH \
--header "content-type: application/json" \
--data '[ {
  "operation": "replace",
  "field": "__PASSWORD__",
  "value": "MyNewPassw0rd"
} ]' \
"http://localhost:8080/openidm/system/azuread/user/9fa6c765-0872-45f6-8714-1dcd1ed94859"
{
  "_id": "9fa6c765-0872-45f6-8714-1dcd1ed94859",
  "surname": "Carter",
  "displayName": "Steve Carter",
  "memberOf": [],
  "onPremisesExtensionAttributes": {
    "extensionAttribute14": null,
    ...
  },
  "userType": "Member",
  "identities": [
    {
      "signInType": "userPrincipalName",
      "issuerAssignedId": "00654321@forgedemo.onmicrosoft.com",
      "issuer": "forgedemo.onmicrosoft.com"
    }
  ],
  "businessPhones": [],
  "createdDateTime": "2020-12-18T13:23:58Z",
  "accountEnabled": true,
  "userPrincipalName": "00654321@forgedemo.onmicrosoft.com",
  "proxyAddresses": [],
  "imAddresses": [],
  "mailNickname": "00654321",
  "givenName": "Steve",
  "__NAME__": "00654321@forgedemo.onmicrosoft.com"
}
```

Delete users and groups

This command deletes a user:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request DELETE \
"http://localhost:8080/openidm/system/azuread/user/9fa6c765-0872-45f6-8714-1dcd1ed94859"
{
  "_id": "9fa6c765-0872-45f6-8714-1dcd1ed94859",
  "surname": "Carter",
  "displayName": "Steve Carter",
  "memberOf": [],
  "onPremisesExtensionAttributes": {
    "extensionAttribute14": null,
    ...
  },
  "userType": "Member",
  "identities": [
    {
      "signInType": "userPrincipalName",
      "issuerAssignedId": "00654321@forgedemo.onmicrosoft.com",
      "issuer": "forgedemo.onmicrosoft.com"
    }
  ],
  "businessPhones": [],
  "createdDateTime": "2020-12-18T13:23:58Z",
  "accountEnabled": true,
  "userPrincipalName": "00654321@forgedemo.onmicrosoft.com",
  "proxyAddresses": [],
  "imAddresses": [],
  "mailNickname": "00654321",
  "givenName": "Steve",
  "__NAME__": "00654321@forgedemo.onmicrosoft.com"
}
```

Service plans (MS Graph API)

Use the MS Graph API connector to list the service plans in your Azure data source, get details on service plans, and manage service plans for specific users.

List available service plans in Azure

This command lists the values of the read-only `servicePlan` object:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
"http://localhost:8080/openidm/system/azuread/servicePlan?_queryFilter=true"
{
  "result": [
    {
      "_id": "80b2d799-d2ba-4d2a-8842-fb0d0f3a4b82:4a82b400-a79f-41a4-b4e2-e94f5787b113",
      "__NAME__": "EXCHANGEDESKLESS:EXCHANGE_S_DESKLESS",
      "appliesTo": "User",
      "subscribedSkuId": "9xdl4ChCwk2okl0_8zYaEIRZWEsbZYpEnlM7EPBpz38",
      "skuPartNumber": "EXCHANGEDESKLESS",
      "provisioningStatus": "Disabled",
      "servicePlanId": "4a82b400-a79f-41a4-b4e2-e94f5787b113"
    }
  ],
  "resultCount": 1,
  "pagedResultsCookie": null,
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

List service plans details

This command lists the details of a specific `servicePlan`:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
"http://localhost:8080/openidm/system/azuread/servicePlan/80b2d799-d2ba-4d2a-8842-fb0d0f3a4b82:4a82b400-
a79f-41a4-b4e2-e94f5787b113"
{
  "_id": "80b2d799-d2ba-4d2a-8842-fb0d0f3a4b82:4a82b400-a79f-41a4-b4e2-e94f5787b113",
  "__NAME__": "EXCHANGEDESKLESS:EXCHANGE_S_DESKLESS",
  "appliesTo": "User",
  "subscribedSkuId": "9xdl4ChCwk2okl0_8zYaEIRZWEsbZYpEnlM7EPBpz38",
  "skuPartNumber": "EXCHANGEDESKLESS",
  "provisioningStatus": "Disabled",
  "servicePlanId": "4a82b400-a79f-41a4-b4e2-e94f5787b113"
}
```

Add service plans to a user

Add service plans to a user by supplying the `__servicePlanIds__` array of strings with the format `skuId:servicePlanId`. This command adds two service plans to a specific user:

Request

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--header "If-None-Match: *" \
--request PUT \
--data '{
  "__servicePlanIds__": [
    "4b585984-651b-448a-9e53-3b10f069cf7f:a82fbf69-b4d7-49f4-83a6-915b2cf354f4",
    "4b585984-651b-448a-9e53-3b10f069cf7f:b76fb638-6ba6-402a-b9f9-83d28acb3d86"
  ]
}
"http://localhost:8080/openidm/system/azuread/user/5e85a7a1-2e57-4be2-b912-3e50bc26c856"
```

Response

```
{
  "_id": "5e85a7a1-2e57-4be2-b912-3e50bc26c856",
  "__NAME__": "mciowixxlrcnbob@{AZURE_DOMAIN}",
  "__servicePlanIds__": [
    "4b585984-651b-448a-9e53-3b10f069cf7f:a82fbf69-b4d7-49f4-83a6-915b2cf354f4",
    "4b585984-651b-448a-9e53-3b10f069cf7f:b76fb638-6ba6-402a-b9f9-83d28acb3d86"
  ],
  "country": "US",
  "givenName": "mciowixx",
  "userPrincipalName": "mciowixxlrcnbob@{AZURE_DOMAIN}",
  "licenses": [
    {
      "skuPartNumber": "DESKLESSPACK",
      "servicePlans": [
        {
          "servicePlanName": "VIVAENGAGE_CORE",
          "provisioningStatus": "Success",
          "appliesTo": "User",
          "servicePlanId": "a82fbf69-b4d7-49f4-83a6-915b2cf354f4"
        },
        {
          "servicePlanName": "VIVA_LEARNING_SEEDED",
          "provisioningStatus": "Success",
          "appliesTo": "User",
          "servicePlanId": "b76fb638-6ba6-402a-b9f9-83d28acb3d86"
        },
        {
          "servicePlanName": "Nucleus",
          "provisioningStatus": "Success",
          "appliesTo": "Company",
          "servicePlanId": "db4d623d-b514-490b-b7ef-8885eee514de"
        },
        {
          "servicePlanName": "MICROSOFTBOOKINGS",
          "provisioningStatus": "Disabled",
          "appliesTo": "User",
          "servicePlanId": "199a5c09-e0ca-4e37-8f7c-b05d533e1ea2"
        },
        {
          "servicePlanName": "RMS_S_BASIC",
          "provisioningStatus": "Success",
          "appliesTo": "Company",
          "servicePlanId": "31cf2cfc-6b0d-4adc-a336-88b724ed8122"
        },
        {
          "servicePlanName": "POWER_VIRTUAL_AGENTS_O365_F1",
          "provisioningStatus": "Disabled",
          "appliesTo": "User",
          "servicePlanId": "ba2fdb48-290b-4632-b46a-e4ecc58ac11a"
        },
        {
          "servicePlanName": "CDS_O365_F1",
          "provisioningStatus": "Disabled",
          "appliesTo": "User",
          "servicePlanId": "90db65a7-bf11-4904-a79f-ef657605145b"
        },
        {
          "servicePlanName": "PROJECT_O365_F3",
          "provisioningStatus": "Disabled",
          "appliesTo": "User",
          "servicePlanId": "7f6f28c2-34bb-4d4b-be36-48ca2e77e1ec"
        },
        {
          "servicePlanName": "DYN365_CDS_O365_F1",
          "provisioningStatus": "Disabled",
          "appliesTo": "User",
          "servicePlanId": "ca6e61ec-d4f4-41eb-8b88-d96e0e14323f"
        },
        {
          "servicePlanName": "PROJECTWORKMANAGEMENT",
          "provisioningStatus": "Disabled",
          "appliesTo": "User",
          "servicePlanId": "b737dad2-2f6c-4c65-90e3-ca563267e8b9"
        },
        {
          "servicePlanName": "KAIZALA_O365_P1",
          "provisioningStatus": "Disabled",
          "appliesTo": "User",
          "servicePlanId": "73b2a583-6a59-42e3-8e83-54db46bc3278"
        },
        {
          "servicePlanName": "MICROSOFT_SEARCH",
          "provisioningStatus": "Success",
          "appliesTo": "Company",
          "servicePlanId": "94065c59-bc8e-4e8b-89e5-5138d471eaff"
        },
        {
          "servicePlanName": "WHITEBOARD_FIRSTLINE1",
          "provisioningStatus": "Disabled",
          "appliesTo": "User",
          "servicePlanId": "36b29273-c6d0-477a-aca6-6fbe24f538e3"
        },
        {
          "servicePlanName": "OFFICEMOBILE_SUBSCRIPTION",
          "provisioningStatus": "Disabled",
          "appliesTo": "User",
          "servicePlanId": "c63d4d19-e8cb-460e-b37c-4d6c34603745"
        },
        {
          "servicePlanName": "BPOS_S_TODO_FIRSTLINE",
          "provisioningStatus": "Disabled",
          "appliesTo": "User",
          "servicePlanId": "80873e7a-cd2a-4e67-b061-1b5381a676a5"
        },
        {
          "servicePlanName": "FORMS_PLAN_K",
          "provisioningStatus": "Disabled",
          "appliesTo": "User",
          "servicePlanId": "f07046bd-2a3c-4b96-b0be-dea79d7cbfb8"
        },
        {
          "servicePlanName": "STREAM_O365_K",
          "provisioningStatus": "Disabled",
          "appliesTo": "User",
          "servicePlanId": "3ffba0d2-38e5-4d5e-8ec0-98f2b05c09d9"
        },
        {
          "servicePlanName": "FLOW_O365_S1",
          "provisioningStatus": "Disabled",
          "appliesTo": "User",
          "servicePlanId": "bd91b1a4-9f94-4ecf-b45b-3a65e5c8128a"
        },
        {
          "servicePlanName": "POWERAPPS_O365_S1",
          "provisioningStatus": "Disabled",
          "appliesTo": "User",
          "servicePlanId": "e0287f9f-e222-4f98-9a83-f379e249159a"
        },
        {
          "servicePlanName": "TEAMS1",
          "provisioningStatus": "Disabled",
          "appliesTo": "User",
          "servicePlanId": "57ff2da0-773e-42df-b2af-ffb7a2317929"
        },
        {
          "servicePlanName": "Deskless",
          "provisioningStatus": "Disabled",
          "appliesTo": "User",
          "servicePlanId": "8c7d2df8-86f0-4902-b2ed-a0458298f3b3"
        },
        {
          "servicePlanName": "MCOIMP",
          "provisioningStatus": "Disabled",
          "appliesTo": "User",
          "servicePlanId": "afc06cb0-b4f4-4473-8286-d644f70d8faf"
        },
        {
          "servicePlanName": "SHAREPOINTWAC",
          "provisioningStatus": "Disabled",
          "appliesTo": "User",
          "servicePlanId": "e95bec33-7c88-4a70-8e19-b10bd9d0c014"
        },
        {
          "servicePlanName": "SWAY",
          "provisioningStatus": "Disabled",
          "appliesTo": "User",
          "servicePlanId": "a23b959c-7ce8-4e57-9140-b90eb88a9e97"
        },
        {
          "servicePlanName": "INTUNE_O365",
          "provisioningStatus": "PendingActivation",
          "appliesTo": "Company",
          "servicePlanId": "882e1d05-acd1-4ccb-8708-6ee03664b117"
        },
        {
          "servicePlanName": "YAMMER_ENTERPRISE",
          "provisioningStatus": "Disabled",
          "appliesTo": "User",
          "servicePlanId": "7547a3fe-08ee-4ccb-b430-5077c5041653"
        },
        {
          "servicePlanName": "SHAREPOINTDESKLESS",
          "provisioningStatus": "Disabled",
          "appliesTo": "User",
          "servicePlanId": "902b47e5-dcb2-4fdc-858b-c63a90a2bdb9"
        },
        {
          "servicePlanName": "EXCHANGE_S_DESKLESS",
          "provisioningStatus": "Disabled",
          "appliesTo": "User",
          "servicePlanId": "4a82b400-a79f-41a4-b4e2-e94f5787b113"
```

```
        }
      ],
      "id": "9xdl4ChCwk2okl0_8zYaEIRZWEsbZYpEnlM7EPBpz38",
      "skuId": "4b585984-651b-448a-9e53-3b10f069cf7f"
    }
  ],
  "usageLocation": "US"
}
```

User licenses (MS Graph API)

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

List available licenses in Azure

This command lists the values of the read-only `subscribedSku` object. For more information about this object class, refer to the corresponding Microsoft documentation:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
"http://localhost:8080/openidm/system/azuread/subscribedSku?_queryFilter=true"
{
  "result": [
    {
      "_id": "5ee8xxxx-xxxx-xxxx-xxxx-76dc2c2c30bc_f245ecc8-xxxx-xxxx-xxxx-xxxx114de5f3",
      "prepaidUnits": {
        "warning": 0,
        "enabled": 1,
        "suspended": 0
      },
      "skuId": "f245ecc8-xxxx-xxxx-xxxx-xxxx114de5f3",
      "skuPartNumber": "O365_BUSINESS_PREMIUM",
      "capabilityStatus": "Enabled",
      "appliesTo": "User",
      "consumedUnits": 1,
      "__NAME__": "O365_BUSINESS_PREMIUM",
      "servicePlans": [
        {
          "servicePlanName": "RMS_S_BASIC",
          "provisioningStatus": "PendingProvisioning",
          "appliesTo": "Company",
          "servicePlanId": "31cxxxxxxxxxxxxxxxxxxxxxxxxxxx122"
        },
        {
          "servicePlanName": "POWER_VIRTUAL_AGENTS_O365_P2",
          "provisioningStatus": "PendingProvisioning",
          "appliesTo": "User",
          "servicePlanId": "041xxxxx-xxxx-xxxx-xxxx-xxxxxxxxxaee"
        },
        {
          "servicePlanName": "CDS_O365_P2",
          "provisioningStatus": "PendingProvisioning",
          "appliesTo": "User",
          "servicePlanId": "95bxxxxx-xxxx-xxxx-xxxx-xxxxxxxxx95a"
        },
        ...
      ]
    }
  ],
  ...
}
```

List a user’s licenses

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

This command lists a specific user’s licenses:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
"http://localhost:8080/openidm/system/azuread/user/c48be8cc-5846-4059-95e8-a7acbf6aec31?_fields=licenses"
{
  "_id": "c48be8cc-5846-4059-95e8-a7acbf6aec31",
  "licenses": [
    {
      "skuPartNumber": "O365_BUSINESS_PREMIUM",
      "servicePlans": [
        {
          "servicePlanName": "RMS_S_BASIC",
          "provisioningStatus": "PendingProvisioning",
          "appliesTo": "Company",
          "servicePlanId": "31cxxxxx-xxxx-xxxx-xxxx-xxxxxxxxx122"
        },
        {
          "servicePlanName": "POWER_VIRTUAL_AGENTS_O365_P2",
          "provisioningStatus": "PendingProvisioning",
          "appliesTo": "Company",
          "servicePlanId": "041xxxxx-xxxx-xxxx-xxxx-xxxxxxxxxaee"
        },
        {
          "servicePlanName": "CDS_O365_P2",
          "provisioningStatus": "PendingProvisioning",
          "appliesTo": "Company",
          "servicePlanId": "95bxxxxx-xxxx-xxxx-xxxx-xxxxxxxxx95a"
        },
        ...
      ],
      "id": "c8noxxxxsEqoxxxxLCwwxxxxRfKvxxxxth8nxxxx5fM",
      "skuId": "f24xxxxx-xxxx-xxxx-xxxx-xxxxxxxxx5f3"
    }
  ]
}
```

Add and remove a user’s licenses

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

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

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

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--header "If-None-Match: *" \
--request PUT \
--data '{
  "addLicenses": [
    {
      "skuId": "f24xxxxx-xxxx-xxxx-xxxx-xxxxxxxxx5f3"
    }
  ]
}' \
"http://localhost:8080/openidm/system/azuread/user/c48be8cc-5846-4059-95e8-a7acbf6aec31"
```

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

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--header "If-Match: *" \
--request PUT \
--data '{
  "removeLicenses": "f24xxxxx-xxxx-xxxx-xxxx-xxxxxxxxx5f3"
}' \
"http://localhost:8080/openidm/system/azuread/user/c48be8cc-5846-4059-95e8-a7acbf6aec31"
```

Contacts (MS Graph API)

A contact is a resource type in Microsoft Outlook used to organize and store information about an associated object. Contacts use the `/user` endpoint. For more information, refer to the Microsoft Graph documentation.

The MS Graph API connector offers limited support for contacts and includes the following write-only attributes:

```
{
  "type": "array",
  "items": {
    "type": "object",
    "nativeType": "object"
  },
  "nativeName": "__addContacts__",
  "nativeType": "object",
  "flags": [
    "NOT_READABLE",
    "NOT_RETURNED_BY_DEFAULT"
  ]
},
{
  "type": "array",
  "items": {
    "type": "string",
    "nativeType": "string"
  },
  "nativeName": "__removeContacts__",
  "nativeType": "string",
  "flags": [
    "NOT_READABLE",
    "NOT_RETURNED_BY_DEFAULT"
  ]
},
{
  "type": "array",
  "items": {
    "type": "object",
    "nativeType": "object"
  },
  "nativeName": "__updateContacts__",
  "nativeType": "object",
  "flags": [
    "NOT_READABLE",
    "NOT_RETURNED_BY_DEFAULT"
  ]
}
```

Add contacts to a user

Note

You must add contacts to an existing user.

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--header "If-Match: *" \
--request PUT \
--data '{
  "_id": "671fa173-ad81-41c3-89bf-af939426eee7",
  "__addContacts__": {
    "givenName": "Test-Contact",
    "businessAddress": {
      "city": "exampleCity",
      "state": "exampleState",
      "postalCode": "99999",
      "street": "example st",
      "countryOrRegion": "United States"
    }
  }
}' \
"http://localhost:8080/openidm/system/azuread/user/671fa173-ad81-41c3-89bf-af939426eee7"
```

Return a user entry with contacts

Request

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request GET \
'http://localhost:8080/openidm/system/azuread/user/671fa173-ad81-41c3-89bf-af939426eee7?
_fields="_id,contacts"'
```

Response

```
{
  "_id": "671fa173-ad81-41c3-89bf-af939426eee7",
  "contacts": [
    {
      "id": "{CONTACT-ID}",
      "givenName": "Test-Contact",
      "businessAddress": {
        "city": "exampleCity",
        "state": "exampleState",
        "postalCode": "99999",
        "street": "example st",
        "countryOrRegion": "United States"
      }
    }
  ]
}
```

Update a user’s contacts

Request

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--header "If-Match: *" \
--request PUT \
--data '{
  "_id": "671fa173-ad81-41c3-89bf-af939426eee7",
  "__updateContacts__": {
    "id": "{CONTACT-ID}",
    "givenName": "Test-Contact-Updated",
    "businessAddress": {
      "city": "exampleCity",
      "state": "exampleState",
      "postalCode": "99999",
      "street": "example st",
      "countryOrRegion": "United States"
    }
  }
}' \
"http://localhost:8080/openidm/system/azuread/user/671fa173-ad81-41c3-89bf-af939426eee7"
```

After updating the user’s contacts, a subsequent read on the user with the contacts field returns the updated contacts:

Request

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request GET \
'http://localhost:8080/openidm/system/azuread/user/671fa173-ad81-41c3-89bf-af939426eee7?
_fields="_id,contacts"'
```

Response

```
{
  "_id": "671fa173-ad81-41c3-89bf-af939426eee7",
  "contacts": [
    {
      "id": "{CONTACT-ID}",
      "givenName": "Test-Contact-Updated",
      "businessAddress": {
        "city": "exampleCity",
        "state": "exampleState",
        "postalCode": "99999",
        "street": "example st",
        "countryOrRegion": "United States"
      }
    }
  ]
}
```

Remove a user’s contacts

Request

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--header "If-Match: *" \
--request PUT \
--data '{
  "_id": "671fa173-ad81-41c3-89bf-af939426eee7",
  "__removeContacts__": [
    "{CONTACT-ID}"
  ]
}' \
"http://localhost:8080/openidm/system/azuread/user/671fa173-ad81-41c3-89bf-af939426eee7"
```

After removing the user’s contacts, a subsequent read on the user with the contacts field returns an empty contacts array:

Request

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request GET \
'http://localhost:8080/openidm/system/azuread/user/671fa173-ad81-41c3-89bf-af939426eee7?
_fields="_id,contacts"'
```

Response

```
{
  "_id": "671fa173-ad81-41c3-89bf-af939426eee7",
  "contacts": []
}
```

Role eligibility schedules (MS Graph API)

The MS Graph API connector lets you read and manage role eligibility schedules.

Create a role eligibility schedule request

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request POST \
--data '{
  "action": "adminAssign",
  "justification": "Justification is required",
  "roleDefinitionId": "fdd7a751-b60b-444a-984c-02652fe8fa1c",
  "directoryScopeId": "/",
  "principalId": "2588c7f0-776e-407e-a1dc-f3a77a28e4fe",
  "scheduleInfo": {
    "startDateTime": "2022-04-10T00:00:00Z",
    "expiration": {
      "type": "noExpiration"
    }
  }
}' \
"http://localhost:8080/openidm/system/azuread/roleEligibilityScheduleRequest"
{
  "_id": "0d8a7bbe-e4ab-4798-8539-728c410ac7b7",
  "isValidationOnly": false,
  "targetScheduleId": "0d8a7bbe-e4ab-4798-8539-728c410ac7b7",
  "createdDateTime": "2023-02-15T23:59:45.143Z",
  "__NAME__": "0d8a7bbe-e4ab-4798-8539-728c410ac7b7",
  "directoryScopeId": "/",
  "principalId": "2588c7f0-776e-407e-a1dc-f3a77a28e4fe",
  "roleDefinitionId": "fdd7a751-b60b-444a-984c-02652fe8fa1c",
  "action": "adminAssign",
  "ticketInfo": {},
  "completedDateTime": "2023-02-15T23:59:45.167Z",
  "justification": "Justification is required",
  "status": "Provisioned",
  "scheduleInfo": {
    "startDateTime": "2023-02-15T23:59:45.168101400Z",
    "expiration": {
      "type": "noExpiration"
    }
  },
  "createdBy": {
    "user": {
      "id": "f516bdc4-0171-42ba-823a-4cbdff160d0f"
    }
  }
}
```

Read a role eligibility schedule request

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
"http://localhost:8080/openidm/system/azuread/roleEligibilityScheduleRequest/0d8a7bbe-
e4ab-4798-8539-728c410ac7b7"
{
  "_id": "0d8a7bbe-e4ab-4798-8539-728c410ac7b7",
  "isValidationOnly": false,
  "targetScheduleId": "0d8a7bbe-e4ab-4798-8539-728c410ac7b7",
  "createdDateTime": "2023-02-15T23:59:45.143Z",
  "__NAME__": "0d8a7bbe-e4ab-4798-8539-728c410ac7b7",
  "directoryScopeId": "/",
  "principalId": "2588c7f0-776e-407e-a1dc-f3a77a28e4fe",
  "roleDefinitionId": "fdd7a751-b60b-444a-984c-02652fe8fa1c",
  "action": "adminAssign",
  "ticketInfo": {},
  "completedDateTime": "2023-02-15T23:59:45.167Z",
  "justification": "Justification is required",
  "status": "Provisioned",
  "scheduleInfo": {
    "startDateTime": "2023-02-15T23:59:45.168101400Z",
    "expiration": {
      "type": "noExpiration"
    }
  },
  "createdBy": {
    "user": {
      "id": "f516bdc4-0171-42ba-823a-4cbdff160d0f"
    }
  }
}
```

Get role eligibility schedules for a user

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
"http://localhost:8080/openidm/system/azuread/roleEligibilitySchedule?
_queryFilter=principalId%20eq%20'2588c7f0-776e-407e-a1dc-f3a77a28e4fe'"
{
  "result": [
    {
      "_id": "0d8a7bbe-e4ab-4798-8539-728c410ac7b7",
      "modifiedDateTime": "0001-01-01T08:00Z",
      "createdDateTime": "2023-02-15T23:59:45.450Z",
      "principalId": "2588c7f0-776e-407e-a1dc-f3a77a28e4fe",
      "scheduleInfo": {
        "startDateTime": "2023-02-15T23:59:45.450Z",
        "expiration": {
          "type": "noExpiration"
        }
      },
      "createdUsing": "0d8a7bbe-e4ab-4798-8539-728c410ac7b7",
      "status": "Provisioned",
      "directoryScopeId": "/",
      "__NAME__": "0d8a7bbe-e4ab-4798-8539-728c410ac7b7",
      "roleDefinitionId": "fdd7a751-b60b-444a-984c-02652fe8fa1c",
      "memberType": "Direct"
    }
  ],
  ...
}
```

Get role eligibility schedule instance

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
"http://localhost:8080/openidm/system/azuread/roleEligibilityScheduleInstance?
_queryFilter=principalId+eq+'2588c7f0-776e-407e-a1dc-f3a77a28e4fe'"
{
  "result": [
    {
      "_id": "UX6spHTVBkG5_Zv86oJthH0ZIKwfxAZIp1uoOmyPt1I-1-e",
      "roleDefinitionId": "a4ac7e51-d574-4106-b9fd-9bfcea826d84",
      "directoryScopeId": "/",
      "roleEligibilityScheduleId": "1248840c-f57d-4168-9e2c-1e0d0e9a46f4",
      "__NAME__": "UX6spHTVBkG5_Zv86oJthH0ZIKwfxAZIp1uoOmyPt1I-1-e",
      "principalId": "2588c7f0-776e-407e-a1dc-f3a77a28e4fe",
      "startDateTime": "2023-02-03T21:29:03.217Z",
      "memberType": "Direct"
    }
  ],
  ...
}
```

Role assignment schedules (MS Graph API)

The MS Graph API connector lets you read and manage role assignment schedules.

Create a role assignment schedule request

Request

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request POST \
--data '{
  "action": "adminAssign",
  "justification": "Justification is required",
  "roleDefinitionId": "fdd7a751-b60b-444a-984c-02652fe8fa1c",
  "directoryScopeId": "/",
  "principalId": "a4375665-cba5-4208-a4f2-12a0d2fc0e85",
  "scheduleInfo": {
    "startDateTime": "2022-04-10T00:00:00Z",
    "expiration": {
      "type": "noExpiration"
    }
  }
}' \
"http://localhost:8080/openidm/system/azuread/roleAssignmentScheduleRequest"
```

Response

```
{
  "_id": "4b49df1e-4b59-4a93-a7c7-ad3b13ad98b0",
  "scheduleInfo": {
    "startDateTime": "2023-02-16T22:21:04.079921Z",
    "expiration": {
      "type": "noExpiration"
    }
  },
  "isValidationOnly": false,
  "createdBy": {
    "user": {
      "id": "f516bdc4-0171-42ba-823a-4cbdff160d0f"
    }
  },
  "ticketInfo": {},
  "roleDefinitionId": "fdd7a751-b60b-444a-984c-02652fe8fa1c",
  "principalId": "f96413e8-1366-426e-ab24-4d9380f11e2e",
  "__NAME__": "4b49df1e-4b59-4a93-a7c7-ad3b13ad98b0",
  "completedDateTime": "2023-02-16T22:21:04.080Z",
  "targetScheduleId": "4b49df1e-4b59-4a93-a7c7-ad3b13ad98b0",
  "action": "adminAssign",
  "directoryScopeId": "/",
  "status": "Provisioned",
  "createdDateTime": "2023-02-16T22:21:04.070Z",
  "justification": "Justification is required"
}
```

Read a role assignment schedule request

Request

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
"http://localhost:8080/openidm/system/azuread/roleAssignmentScheduleRequest/4b49df1e-4b59-4a93-a7c7-
ad3b13ad98b0"
```

Response

```
{
  "_id": "4b49df1e-4b59-4a93-a7c7-ad3b13ad98b0",
  "scheduleInfo": {
    "startDateTime": "2023-02-16T22:21:04.079921Z",
    "expiration": {
      "type": "noExpiration"
    }
  },
  "isValidationOnly": false,
  "createdBy": {
    "user": {
      "id": "f516bdc4-0171-42ba-823a-4cbdff160d0f"
    }
  },
  "ticketInfo": {},
  "roleDefinitionId": "fdd7a751-b60b-444a-984c-02652fe8fa1c",
  "principalId": "f96413e8-1366-426e-ab24-4d9380f11e2e",
  "__NAME__": "4b49df1e-4b59-4a93-a7c7-ad3b13ad98b0",
  "completedDateTime": "2023-02-16T22:21:04.080Z",
  "targetScheduleId": "4b49df1e-4b59-4a93-a7c7-ad3b13ad98b0",
  "action": "adminAssign",
  "directoryScopeId": "/",
  "status": "Provisioned",
  "createdDateTime": "2023-02-16T22:21:04.070Z",
  "justification": "Justification is required"
}
```

Get role assignment schedules for a user

Request

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
"http://localhost:8080/openidm/system/azuread/roleAssignmentSchedule?
_queryFilter=principalId%20eq%20'f96413e8-1366-426e-ab24-4d9380f11e2e'"
```

Response

```
{
  "result": [
    {
      "_id": "4b49df1e-4b59-4a93-a7c7-ad3b13ad98b0",
      "__NAME__": "4b49df1e-4b59-4a93-a7c7-ad3b13ad98b0",
      "status": "Provisioned",
      "memberType": "Direct",
      "roleDefinitionId": "fdd7a751-b60b-444a-984c-02652fe8fa1c",
      "principalId": "f96413e8-1366-426e-ab24-4d9380f11e2e",
      "createdDateTime": "2023-02-16T22:21:07.727Z",
      "assignmentType": "Assigned",
      "directoryScopeId": "/",
      "createdUsing": "4b49df1e-4b59-4a93-a7c7-ad3b13ad98b0",
      "scheduleInfo": {
        "startDateTime": "2023-02-16T22:21:07.727Z",
        "expiration": {
          "type": "noExpiration"
        }
      }
    }
  ],
  ...
}
```

Get role assignment schedule instance

Request

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
"http://localhost:8080/openidm/system/azuread/roleAssignmentScheduleInstance?
_queryFilter=principalId+eq+'f96413e8-1366-426e-ab24-4d9380f11e2e'"
```

Response

```
{
  "result": [
    {
      "_id": "UafX_Qu2SkSYTAJlL-j6HOgTZPlmE25CqyRNk4DxHi4-1",
      "assignmentType": "Assigned",
      "memberType": "Direct",
      "principalId": "f96413e8-1366-426e-ab24-4d9380f11e2e",
      "startDateTime": "2023-02-16T22:21:07.727Z",
      "__NAME__": "UafX_Qu2SkSYTAJlL-j6HOgTZPlmE25CqyRNk4DxHi4-1",
      "roleAssignmentScheduleId": "4b49df1e-4b59-4a93-a7c7-ad3b13ad98b0",
      "directoryScopeId": "/",
      "roleDefinitionId": "fdd7a751-b60b-444a-984c-02652fe8fa1c",
      "roleAssignmentOriginId": "UafX_Qu2SkSYTAJlL-j6HOgTZPlmE25CqyRNk4DxHi4-1"
    }
  ],
  ...
}
```

Applications (MS Graph API)

The MS Graph API connector lets you read and manage applications.

Query all applications

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
"http://localhost:8080/openidm/system/azuread/application?_queryFilter=true"
```

Read an application

Request

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
"http://localhost:8080/openidm/system/azuread/application/e2dcfa77-5222-4715-a043-98baac00683d"
```

Response

```
{
  "_id": "e2dcfa77-5222-4715-a043-98baac00683d",
  "tags": [],
  "spa": {
    "redirectUris": []
  },
  "parentalControlSettings": {
    "legalAgeGroupRule": "Allow",
    "countriesBlockedForMinors": []
  },
  "api": {
    "requestedAccessTokenVersion": 2,
    "knownClientApplications": [],
    "oauth2PermissionScopes": [],
    "preAuthorizedApplications": []
  },
  "passwordCredentials": [],
  "info": {},
  "addIns": [],
  "keyCredentials": [],
  "publicClient": {
    "redirectUris": []
  },
  "verifiedPublisher": {},
  "identifierUris": [],
  "web": {
    "implicitGrantSettings": {
      "enableAccessTokenIssuance": false,
      "enableIdTokenIssuance": false
    },
    "redirectUris": []
  },
  "publisherDomain": "example.com",
  "createdDateTime": "2023-05-05T20:40:02Z",
  "displayName": "Test-Application",
  "appRoles": [],
  "isDeviceOnlyAuthSupported": false,
  "appId": "bc146d82-be72-4e16-814d-76e977ad198e",
  "signInAudience": "AzureADandPersonalMicrosoftAccount",
  "requiredResourceAccess": [
    {
      "resourceAppId": "00000002-0000-0000-c000-000000000000",
      "resourceAccess": [
        {
          "id": "311a71cc-e848-46a1-bdf8-97ff7156d8e6",
          "type": "Scope"
        }
      ]
    }
  ]
}
```

Create an application

Request

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request POST \
--data '{
  "displayName": "Test-Application",
  "requiredResourceAccess": [
    {
      "resourceAppId": "00000002-0000-0000-c000-000000000000",
      "resourceAccess": [
        {
          "id": "311a71cc-e848-46a1-bdf8-97ff7156d8e6",
          "type": "Scope"
        }
      ]
    }
  ]
}' \
"http://localhost:8080/openidm/system/azuread/application"
```

Response

```
{
  "_id": "e2dcfa77-5222-4715-a043-98baac00683d",
  "tags": [],
  "spa": {
    "redirectUris": []
  },
  "parentalControlSettings": {
    "legalAgeGroupRule": "Allow",
    "countriesBlockedForMinors": []
  },
  "api": {
    "requestedAccessTokenVersion": 2,
    "knownClientApplications": [],
    "oauth2PermissionScopes": [],
    "preAuthorizedApplications": []
  },
  "passwordCredentials": [],
  "info": {},
  "addIns": [],
  "keyCredentials": [],
  "publicClient": {
    "redirectUris": []
  },
  "verifiedPublisher": {},
  "identifierUris": [],
  "web": {
    "implicitGrantSettings": {
      "enableAccessTokenIssuance": false,
      "enableIdTokenIssuance": false
    },
    "redirectUris": []
  },
  "publisherDomain": "example.com",
  "createdDateTime": "2023-05-05T20:40:02Z",
  "displayName": "Test-Application",
  "appRoles": [],
  "isDeviceOnlyAuthSupported": false,
  "appId": "bc146d82-be72-4e16-814d-76e977ad198e",
  "signInAudience": "AzureADandPersonalMicrosoftAccount",
  "requiredResourceAccess": [
    {
      "resourceAppId": "00000002-0000-0000-c000-000000000000",
      "resourceAccess": [
        {
          "id": "311a71cc-e848-46a1-bdf8-97ff7156d8e6",
          "type": "Scope"
        }
      ]
    }
  ]
}
```

Add a password (client secret) to an application

Adding `passwordCredential` when creating applications is not supported. You must use the `addPassword` method to add passwords or secrets to an application.

Some actions require more than a UUID on return and have no object to follow up with a subsequent read. In this instance, you can use the `scriptOnConnector` action, which requires at least the `builtinAction` parameter. Adding client secrets using this method requires the parameter `builtinAction=addPassword`. You can learn more about the other required parameter `applicationId` and optional parameters in the Microsoft Graph documentation.

The above also requires a *dummy* system action. For example:

```
{
  "scriptId": "addPassword",
  "actions": [
    {
      "systemType": ".*MSGraphAPIConnector",
      "actionSource": "return;",
      "actionType": "Groovy"
    }
  ]
}
```

The `actionSource` is ignored for these `builtIn` requests, but still required to invoke the `scriptOnConnector` action.

Request

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request POST \
"http://localhost:8080/openidm/system/azuread/?
_action=script&scriptId=addPassword&displayName=TestSecretGenesis&applicationId=f619a0ac-0548-4e90-9314-84d967088d2b&
```

Response

```
{
  "actions": [
    {
      "result": {
        "secretText": "{GENERATED-CLIENT-SECRET}",
        "startDateTime": {
          "dateTime": {
            "date": {
              "month": 5,
              "year": 2023,
              "day": 5
            },
            "time": {
              "hour": 20,
              "nano": 771787000,
              "minute": 40,
              "second": 27
            }
          },
          "offset": {
            "totalSeconds": 0
          }
        },
        "displayName": "TestSecretGenesis",
        "hint": "LS8",
        "keyId": "8f48fb5e-a295-4969-b988-a723a02f2f28",
        "endDateTime": {
          "dateTime": {
            "date": {
              "month": 5,
              "year": 2025,
              "day": 5
            },
            "time": {
              "hour": 20,
              "nano": 771787000,
              "minute": 40,
              "second": 27
            }
          },
          "offset": {
            "totalSeconds": 0
          }
        }
      }
    }
  ]
}
```

Update an application

Request

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request PATCH \
--data '[
  {
    "operation": "replace",
    "field": "/displayName",
    "value": "Test-Application-Updated"
  }
]' \
"http://localhost:8080/openidm/system/azuread/application/4eff1242-bd95-463b-9c8c-f221ec489ba1"
```

Response

```
{
  "_id": "4eff1242-bd95-463b-9c8c-f221ec489ba1",
  "tags": [],
  "spa": {
    "redirectUris": []
  },
  "parentalControlSettings": {
    "legalAgeGroupRule": "Allow",
    "countriesBlockedForMinors": []
  },
  "api": {
    "requestedAccessTokenVersion": 2,
    "knownClientApplications": [],
    "oauth2PermissionScopes": [],
    "preAuthorizedApplications": []
  },
  "passwordCredentials": [],
  "info": {},
  "addIns": [],
  "keyCredentials": [],
  "publicClient": {
    "redirectUris": []
  },
  "verifiedPublisher": {},
  "identifierUris": [],
  "web": {
    "implicitGrantSettings": {
      "enableAccessTokenIssuance": false,
      "enableIdTokenIssuance": false
    },
    "redirectUris": []
  },
  "publisherDomain": "example.com",
  "createdDateTime": "2023-05-05T20:40:11Z",
  "displayName": "Test-Application-Updated",
  "appRoles": [],
  "appId": "68e06ad2-569f-407d-b117-6cc1d9f5d787",
  "signInAudience": "AzureADandPersonalMicrosoftAccount",
  "requiredResourceAccess": []
}
```

Delete an application

Request

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--header "If-Match: *" \
--request DELETE \
"http://localhost:8080/openidm/system/azuread/application/579d5781-6e39-4b94-b741-1748d1e14199"
```

Response

```
{
  "_id": "579d5781-6e39-4b94-b741-1748d1e14199",
  "tags": [],
  "spa": {
    "redirectUris": []
  },
  "parentalControlSettings": {
    "legalAgeGroupRule": "Allow",
    "countriesBlockedForMinors": []
  },
  "api": {
    "requestedAccessTokenVersion": 2,
    "knownClientApplications": [],
    "oauth2PermissionScopes": [],
    "preAuthorizedApplications": []
  },
  "passwordCredentials": [],
  "info": {},
  "addIns": [],
  "keyCredentials": [],
  "publicClient": {
    "redirectUris": []
  },
  "verifiedPublisher": {},
  "identifierUris": [],
  "web": {
    "implicitGrantSettings": {
      "enableAccessTokenIssuance": false,
      "enableIdTokenIssuance": false
    },
    "redirectUris": []
  },
  "publisherDomain": "example.com",
  "createdDateTime": "2023-05-05T20:40:18Z",
  "displayName": "Test-Application",
  "appRoles": [],
  "appId": "6e26b7a3-53ef-45ea-8492-fed30f1dd2ad",
  "signInAudience": "AzureADandPersonalMicrosoftAccount",
  "requiredResourceAccess": []
}
```

`servicePrincipal` (MS Graph API)

The `servicePrincipal` resource type represents an instance of an application in a directory. For more information, refer to the Microsoft Graph documentation.

Query all `servicePrincipal` objects

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
"http://localhost:8080/openidm/system/azuread/servicePrincipal?_queryFilter=true"
```

Read a `servicePrincipal`

Request

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
"http://localhost:8080/openidm/system/azuread/servicePrincipal/1c696b95-7f68-4018-b627-6c9601faa80b"
```

Response

```
{
  "_id": "1c696b95-7f68-4018-b627-6c9601faa80b",
  "addIns": [],
  "replyUrls": [],
  "keyCredentials": [],
  "oauth2PermissionScopes": [],
  "displayName": "Test-Application",
  "appRoleAssignments": [],
  "alternativeNames": [],
  "resourceSpecificApplicationPermissions": [],
  "appDisplayName": "Test-Application",
  "accountEnabled": true,
  "appOwnerOrganizationId": "9e91bf24-7a08-433e-b111-5542416b4f20",
  "passwordCredentials": [],
  "servicePrincipalNames": [
    "a293dbaf-ba5d-4692-8898-521a1da51bac"
  ],
  "appId": "a293dbaf-ba5d-4692-8898-521a1da51bac",
  "signInAudience": "AzureADandPersonalMicrosoftAccount",
  "notificationEmailAddresses": [],
  "servicePrincipalType": "Application",
  "tags": [],
  "appRoleAssignedTo": [],
  "info": {},
  "appRoles": [],
  "appRoleAssignmentRequired": false
}
```

Create a `servicePrincipal`

> **Note**
>
> A `servicePrincipal` requires an `appId`.

Request

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request POST \
--data '{
  "appId": "0b9179f4-f617-4ab8-9c33-18a870c76722"
}' \
"http://localhost:8080/openidm/system/azuread/servicePrincipal"
```

Response

```
{
  "_id": "7d164d58-6210-4c25-84db-d3dfce1171b4",
  "addIns": [],
  "replyUrls": [],
  "keyCredentials": [],
  "oauth2PermissionScopes": [],
  "displayName": "Test-Application",
  "appRoleAssignments": [],
  "alternativeNames": [],
  "resourceSpecificApplicationPermissions": [],
  "appDisplayName": "Test-Application",
  "accountEnabled": true,
  "appOwnerOrganizationId": "9e91bf24-7a08-433e-b111-5542416b4f20",
  "passwordCredentials": [],
  "servicePrincipalNames": [
    "0b9179f4-f617-4ab8-9c33-18a870c76722"
  ],
  "appId": "0b9179f4-f617-4ab8-9c33-18a870c76722",
  "signInAudience": "AzureADandPersonalMicrosoftAccount",
  "notificationEmailAddresses": [],
  "servicePrincipalType": "Application",
  "tags": [],
  "appRoleAssignedTo": [],
  "info": {},
  "appRoles": [],
  "appRoleAssignmentRequired": false
}
```

Add a password (client secret) to a `servicePrincipal`

Adding `passwordCredential` when creating a `servicePrincipal` is not supported. You must use the `addPassword` method to add passwords or secrets to a `servicePrincipal`.

Request

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request POST \
"http://localhost:8080/openidm/system/azuread/?
_action=script&scriptId=addPassword&displayName=TestSecretGenesis&servicePrincipalId=32e18e7a-cb23-4453-
b5f4-286bc1a629b8&builtinAction=addPassword"
```

Response

```
{
  "actions": [
    {
      "result": {
        "secretText": "{GENERATED-CLIENT-SECRET}",
        "startDateTime": {
          "dateTime": {
            "date": {
              "month": 5,
              "year": 2023,
              "day": 5
            },
            "time": {
              "hour": 20,
              "nano": 91094000,
              "minute": 41,
              "second": 8
            }
          },
          "offset": {
            "totalSeconds": 0
          }
        },
        "displayName": "TestSecretGenesis",
        "hint": "rJn",
        "keyId": "862c0883-45ac-4e13-8adc-ce9bf3036570",
        "endDateTime": {
          "dateTime": {
            "date": {
              "month": 5,
              "year": 2025,
              "day": 5
            },
            "time": {
              "hour": 20,
              "nano": 91094000,
              "minute": 41,
              "second": 8
            }
          },
          "offset": {
            "totalSeconds": 0
          }
        }
      }
    }
  ]
}
```

Update a `servicePrincipal`

Request

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request PATCH \
--data '[
  {
    "operation": "replace",
    "field": "/appRoleAssignmentRequired",
    "value": true
  }
]' \
"http://localhost:8080/openidm/system/azuread/servicePrincipal/7d164d58-6210-4c25-84db-d3dfce1171b4"
```

Response

```
 {
   "_id": "7d164d58-6210-4c25-84db-d3dfce1171b4",
   "addIns": [],
   "replyUrls": [],
   "keyCredentials": [],
   "oauth2PermissionScopes": [],
   "displayName": "Test-Application",
   "appRoleAssignments": [],
   "alternativeNames": [],
   "resourceSpecificApplicationPermissions": [],
   "appDisplayName": "Test-Application",
   "accountEnabled": true,
   "appOwnerOrganizationId": "9e91bf24-7a08-433e-b111-5542416b4f20",
   "passwordCredentials": [],
   "servicePrincipalNames": [
     "0b9179f4-f617-4ab8-9c33-18a870c76722"
   ],
   "appId": "0b9179f4-f617-4ab8-9c33-18a870c76722",
   "signInAudience": "AzureADandPersonalMicrosoftAccount",
   "notificationEmailAddresses": [],
   "servicePrincipalType": "Application",
   "tags": [],
   "appRoleAssignedTo": [],
   "info": {},
   "appRoles": [],
   "appRoleAssignmentRequired": true
 }
```

Delete a `servicePrincipal`

Request

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--header "If-Match: *" \
--request DELETE \
"http://localhost:8080/openidm/system/azuread/servicePrincipal/1df34a52-3491-4b3a-8ec7-51d77ab50860"
```

Response

```
{
  "_id": "1df34a52-3491-4b3a-8ec7-51d77ab50860",
  "addIns": [],
  "replyUrls": [],
  "keyCredentials": [],
  "oauth2PermissionScopes": [],
  "displayName": "Test-Application",
  "appRoleAssignments": [],
  "alternativeNames": [],
  "resourceSpecificApplicationPermissions": [],
  "appDisplayName": "Test-Application",
  "accountEnabled": true,
  "appOwnerOrganizationId": "9e91bf24-7a08-433e-b111-5542416b4f20",
  "passwordCredentials": [],
  "servicePrincipalNames": [
    "a2179b48-33f0-4933-8c59-39639469bb13"
  ],
  "appId": "a2179b48-33f0-4933-8c59-39639469bb13",
  "signInAudience": "AzureADandPersonalMicrosoftAccount",
  "notificationEmailAddresses": [],
  "servicePrincipalType": "Application",
  "tags": [],
  "appRoleAssignedTo": [],
  "info": {},
  "appRoles": [],
  "appRoleAssignmentRequired": false
}
```

Application permissions (MS Graph API)

Application permissions are also known as *app roles* or *app role assignments*. You can grant application permissions directly by adding an app role assignment to an object, such as user, group, or `servicePrincipal`. For more information about app role assignments, refer to the Microsoft Graph documentation.

Note

The following table displays what the different id’s involved in app role assignment represent:

`principalId`	The id of a user, group, or client `servicePrincipal` . Depends on the type of object receiving the app role assignment.
`resourceId`	The object id of the `servicePrincipal` containing the `appRole` .
`appRoleId`	The id of the `appRole` .

Special schema definitions for app role assignments

The following schema definitions are special attributes in the connector, not real, readable properties of a `servicePrincipal` or other directory objects. They allow the connector to add and remove the respective app role assignments that appear in their related relationship fields.

For example, `__addAppRoleAssignments__` stores a list of object data to populate the *actual* attribute `appRoleAssignments`.

__addAppRoleAssignments__

```
{
  "type": "array",
  "items": {
    "type": "object",
    "nativeType": "object"
  },
  "nativeName": "__addAppRoleAssignments__",
  "nativeType": "object"
}
```

__removeAppRoleAssignments__

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

__addAppRoleAssignedTo__

```
{
  "type": "array",
  "items": {
    "type": "object",
    "nativeType": "object"
  },
  "nativeName": "__addAppRoleAssignedTo__",
  "nativeType": "object"
}
```

__removeAppRoleAssignedTo__

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

Add an app role assignment to a `servicePrincipal`

Note

This process is identical for users and groups.

Request

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--header "If-Match: *" \
--request PUT \
--data '{
  "__addAppRoleAssignments__": {
    "principalId": "05b49121-0bf5-479e-8a4e-140212648879",
    "resourceId": "b3e4e58e-16fa-4b3d-a7b5-f134b7387e62",
    "appRoleId": "df021288-bdef-4463-88db-98f22de89214"
  }
}' \
"http://localhost:8080/openidm/system/azuread/servicePrincipal/05b49121-0bf5-479e-8a4e-140212648879"
```

Response

```
{
  "_id": "05b49121-0bf5-479e-8a4e-140212648879",
  "addIns": [],
  "replyUrls": [],
  "keyCredentials": [],
  "oauth2PermissionScopes": [],
  "displayName": "Test-Application",
  "appRoleAssignments": [
    {
      "resourceDisplayName": "Microsoft Graph",
      "resourceId": "b3e4e58e-16fa-4b3d-a7b5-f134b7387e62",
      "principalDisplayName": "Test-Application",
      "appRoleId": "df021288-bdef-4463-88db-98f22de89214",
      "createdDateTime": "2023-05-05T20:41:15.373168300Z",
      "principalId": "05b49121-0bf5-479e-8a4e-140212648879",
      "id": "IZG0BfULnkeKThQCEmSIeS7n5ay2n99BiFNwyj97w8Y",
      "principalType": "ServicePrincipal"
    }
  ],
  "alternativeNames": [],
  "resourceSpecificApplicationPermissions": [],
  "appDisplayName": "Test-Application",
  "accountEnabled": true,
  "appOwnerOrganizationId": "9e91bf24-7a08-433e-b111-5542416b4f20",
  "passwordCredentials": [],
  "servicePrincipalNames": [
    "93dd36a4-61ca-4a1d-89cf-eac96587de35"
  ],
  "appId": "93dd36a4-61ca-4a1d-89cf-eac96587de35",
  "signInAudience": "AzureADandPersonalMicrosoftAccount",
  "notificationEmailAddresses": [],
  "servicePrincipalType": "Application",
  "tags": [],
  "appRoleAssignedTo": [],
  "info": {},
  "appRoles": [],
  "appRoleAssignmentRequired": false
}
```

Remove an app role assignment from a `servicePrincipal`

Note

This process is identical for users and groups.

Request

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--header "If-Match: *" \
--request PUT \
--data '{
  "__removeAppRoleAssignments__": "IZG0BfULnkeKThQCEmSIeS7n5ay2n99BiFNwyj97w8Y"
}' \
"http://localhost:8080/openidm/system/azuread/servicePrincipal/05b49121-0bf5-479e-8a4e-140212648879"
```

Response

```
{
  "_id": "05b49121-0bf5-479e-8a4e-140212648879",
  "addIns": [],
  "replyUrls": [],
  "keyCredentials": [],
  "oauth2PermissionScopes": [],
  "displayName": "Test-Application",
  "appRoleAssignments": [],
  "alternativeNames": [],
  "resourceSpecificApplicationPermissions": [],
  "appDisplayName": "Test-Application",
  "accountEnabled": true,
  "appOwnerOrganizationId": "9e91bf24-7a08-433e-b111-5542416b4f20",
  "passwordCredentials": [],
  "servicePrincipalNames": [
    "93dd36a4-61ca-4a1d-89cf-eac96587de35"
  ],
  "appId": "93dd36a4-61ca-4a1d-89cf-eac96587de35",
  "signInAudience": "AzureADandPersonalMicrosoftAccount",
  "notificationEmailAddresses": [],
  "servicePrincipalType": "Application",
  "tags": [],
  "appRoleAssignedTo": [],
  "info": {},
  "appRoles": [],
  "appRoleAssignmentRequired": false
}
```

Add an app role to a principal (user/group/servicePrincipal) via a `servicePrincipal`

Request

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--header "If-Match: *" \
--request PUT \
--data '{
  "__addAppRoleAssignedTo__": {
    "principalId": "87f5b3f8-6a8c-4e50-8fd6-0467d5e97e0c",
    "resourceId": "bf960539-a1d8-4eab-a46e-e9ce0b3f15c8",
    "appRoleId": "00000000-0000-0000-0000-000000000000"
  }
}' \
"http://localhost:8080/openidm/system/azuread/servicePrincipal/bf960539-a1d8-4eab-a46e-e9ce0b3f15c8"
```

Response

```
{
  "_id": "bf960539-a1d8-4eab-a46e-e9ce0b3f15c8",
  "addIns": [],
  "replyUrls": [],
  "keyCredentials": [],
  "oauth2PermissionScopes": [],
  "displayName": "Test-Application",
  "appRoleAssignments": [],
  "alternativeNames": [],
  "resourceSpecificApplicationPermissions": [],
  "appDisplayName": "Test-Application",
  "accountEnabled": true,
  "appOwnerOrganizationId": "9e91bf24-7a08-433e-b111-5542416b4f20",
  "passwordCredentials": [],
  "servicePrincipalNames": [
    "62212657-8f49-40b3-874b-9d1c25cb4388"
  ],
  "appId": "62212657-8f49-40b3-874b-9d1c25cb4388",
  "signInAudience": "AzureADandPersonalMicrosoftAccount",
  "notificationEmailAddresses": [],
  "servicePrincipalType": "Application",
  "tags": [],
  "appRoleAssignedTo": [
    {
      "resourceDisplayName": "Test-Application",
      "resourceId": "bf960539-a1d8-4eab-a46e-e9ce0b3f15c8",
      "principalDisplayName": "qcmozfwwygkebie",
      "appRoleId": "00000000-0000-0000-0000-000000000000",
      "createdDateTime": "2023-05-05T20:41:25.405071800Z",
      "principalId": "87f5b3f8-6a8c-4e50-8fd6-0467d5e97e0c",
      "id": "-LP1h4xqUE6P1gRn1el-DCzqXtqJH6NBt0Fr0lT0g2g",
      "principalType": "User"
    }
  ],
  "info": {},
  "appRoles": [],
  "appRoleAssignmentRequired": false
}
```

Remove an app role from a principal (user/group/servicePrincipal) via a `servicePrincipal`

Request

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--header "If-Match: *" \
--request PUT \
--data '{
  "__removeAppRoleAssignedTo__": "-LP1h4xqUE6P1gRn1el-DCzqXtqJH6NBt0Fr0lT0g2g"
}' \
"http://localhost:8080/openidm/system/azuread/servicePrincipal/bf960539-a1d8-4eab-a46e-e9ce0b3f15c8"
```

Response

```
{
  "_id": "bf960539-a1d8-4eab-a46e-e9ce0b3f15c8",
  "addIns": [],
  "replyUrls": [],
  "keyCredentials": [],
  "oauth2PermissionScopes": [],
  "displayName": "Test-Application",
  "appRoleAssignments": [],
  "alternativeNames": [],
  "resourceSpecificApplicationPermissions": [],
  "appDisplayName": "Test-Application",
  "accountEnabled": true,
  "appOwnerOrganizationId": "9e91bf24-7a08-433e-b111-5542416b4f20",
  "passwordCredentials": [],
  "servicePrincipalNames": [
    "62212657-8f49-40b3-874b-9d1c25cb4388"
  ],
  "appId": "62212657-8f49-40b3-874b-9d1c25cb4388",
  "signInAudience": "AzureADandPersonalMicrosoftAccount",
  "notificationEmailAddresses": [],
  "servicePrincipalType": "Application",
  "tags": [],
  "appRoleAssignedTo": [],
  "info": {},
  "appRoles": [],
  "appRoleAssignmentRequired": false
}
```

Authentication methods (MS Graph API)

The MS Graph API connector lets you read and manage the following multi-factor authentication (MFA) methods from the user resource:

MFA method	Supported operations
Email	Create, Update, Delete
Phone	Create, Update, Delete
FIDO2	Delete
Microsoft Authenticator	Delete
Software OATH	Delete

List MFA methods

List the authentication methods with the `authenticationMethods` user relationship:

Request

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
"http://localhost:8080/openidm/system/azuread/user/88080bec-30bd-4026-b5af-5d4607dc7ccc/?
_fields=authenticationMethods"
```

Response

```
{
  "_id": "88080bec-30bd-4026-b5af-5d4607dc7ccc",
  ...
  "authenticationMethods": [
    {
      "@odata.type": "#microsoft.graph.passwordAuthenticationMethod",
      "createdDateTime": "2024-12-11T01:07:44Z",
      "id": "28c10230-6103-485e-b985-444c60001490"
    }
  ]
}
```

Create or update email MFA method

Note

The create and update requests are identical.

Request

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--header "If-None-Match: *" \
--request PUT \
--data '{
  "__emailAuthenticationMethod__": "add_email@example.com"
}' \
"http://localhost:8080/openidm/system/azuread/user/a9299a21-c384-4882-b363-8d7427b36fc5"
```

Response

```
{
  "_id": "a9299a21-c384-4882-b363-8d7427b36fc5",
  ...
  "authenticationMethods": [
    {
      "@odata.type": "#microsoft.graph.passwordAuthenticationMethod",
      "createdDateTime": "2024-12-11T01:07:47Z",
      "id": "28c10230-6103-485e-b985-444c60001490"
    },
    {
      "emailAddress": "add_email@example.com",
      "@odata.type": "#microsoft.graph.emailAuthenticationMethod",
      "id": "3ddfcfc8-9383-446f-83cc-3ab9be4be18f"
    }
  ]
}
```

Remove email MFA method

Request

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--header "If-None-Match: *" \
--request PUT \
--data '{
  "__emailAuthenticationMethod__": null
}' \
"http://localhost:8080/openidm/system/azuread/user/bfe4b140-3b76-4633-855c-26e21a7517c9"
```

Response

```
{
  "_id": "bfe4b140-3b76-4633-855c-26e21a7517c9",
  ...
  "authenticationMethods": [
    {
      "@odata.type": "#microsoft.graph.passwordAuthenticationMethod",
      "createdDateTime": "2024-12-11T01:07:56Z",
      "id": "28c10230-6103-485e-b985-444c60001490"
    }
  ]
}
```

Manage phone MFA method

To manage phone MFA methods, provide an authoritative list of numbers using a PUT request on the user object. Phone numbers use the format `{phoneNumber}:{phoneType}` under the special user attribute `__phoneAuthenticationMethods__`. The connector performs a diff between the request and the user's current list of numbers, and does the following:

- Adds new numbers.
- Removes existing numbers not in the request.
- Replaces non-matching existing numbers for phone types.

Request

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--header "If-None-Match: *" \
--request PUT \
--data '{
  "__phoneAuthenticationMethods__": [
    "+1 7322714703:mobile",
    "+1 7322714705:alternate_mobile",
    "+1 7322714709:office"
  ]
}' \
"http://localhost:8080/openidm/system/azuread/user/00490cf6-4fdd-4b7b-86dd-907e3d613fd0"
```

Response

```
{
  "_id": "00490cf6-4fdd-4b7b-86dd-907e3d613fd0",
  ...
  "__phoneAuthenticationMethods__": [
    "+1 7322714709:OFFICE",
    "+1 7322714705:ALTERNATE_MOBILE",
    "+1 7322714703:MOBILE"
  ]
}
```

Remove Microsoft Authenticator, FIDO2, and software OATH MFA methods

Microsoft's API only supports the removal of Microsoft Authenticator, FIDO2, and software OATH MFA methods. The connector implements the removal of these MFA methods using the following virtual attributes:

- `__removeMicrosoftAuthenticatorMethods__`
- `__removeFido2Methods__`
- `__removeSoftwareOathMethods__`

The following example removes a Microsoft Authenticator MFA method:

Request

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--header "If-None-Match: *" \
--request PUT \
--data '{
  "__removeMicrosoftAuthenticatorMethods__": [
    "00490cf6-4fdd-4b7b-86dd-907e3d613fd0"
  ]
}' \
"http://localhost:8080/openidm/system/azuread/user/bfe4b140-3b76-4633-855c-26e21a7517c9"
```

Response

```
{
  "_id": "bfe4b140-3b76-4633-855c-26e21a7517c9",
  ...
  "authenticationMethods": [
    {
      "@odata.type": "#microsoft.graph.passwordAuthenticationMethod",
      "createdDateTime": "2024-12-11T01:07:56Z",
      "id": "28c10230-6103-485e-b985-444c60001490"
    }
  ]
}
```

MongoDB connector

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

Note

The MongoDB connector uses the following Java MongoDB driver version:

Connector version	Driver version
1.5.20.21 and earlier	v4.5.1
1.5.20.22 - 1.5.20.25	v4.11.2
1.5.20.26 and later	v4.11.4

For MongoDB version compatibility information, refer to Compatibility in the MongoDB Documentation.

Before you start

In a production environment, enable access control on your MongoDB database. If your connector will manage MongoDB users and roles, you must create an administrative user in the `admin` database. If your connector will manage collections in a database, this administrative user must create a specific user and role for the connector for the target database.

Learn more about enabling access control in MongoDB in the MongoDB documentation.

The commands in this chapter assume an administrative user named `myUserAdmin` with password `Passw0rd` who has the `readWrite` role on the `test` database.

Install the MongoDB connector

> **Tip**
>
> To check for an Advanced Identity Cloud application for this connector, refer to:
>
> - Application management
> - App catalog

You can download any connector from Backstage, but some are included in the default deployment for Advanced Identity Cloud, IDM, or RCS. When using an included connector, you can skip installing it and move directly to configuration.

Connector included in default deployment

Connector	IDM	RCS
MongoDB	✓ Yes	✓ Yes

Download the connector .jar file from Backstage.

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

```
mv ~/Downloads/mongodb-connector-1.5.20.28.jar /path/to/openidm/connectors/
```

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

Configure the MongoDB connector

Create a connector configuration using the IDM admin UI:

1. From the navigation bar, click **Configure > Connectors**.
2. On the **Connectors** page, click **New Connector**.
3. On the **New Connector** page, type a **Connector Name**.
4. From the **Connector Type** drop-down list, select **MongoDB Connector - 1.5.20.28**.
5. Complete the **Base Connector Details**.

   > **Tip**
   >
   > For a list of all configuration properties, refer to MongoDB Connector Configuration

6. Click **Save**.

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

Refer to this procedure to create a connector configuration over REST.

Alternatively, configure the connector with a configuration file. A sample connector configuration file (`provisioner.openicf-mongodb.json`) is provided in the `/path/to/openidm/samples/example-configurations/provisioners` directory in IDM. Copy the sample connector configuration to your project's `conf/` directory, and adjust the `configurationProperties` to match your MongoDB instance:

```
"configurationProperties" : {
    "connectionURI" : "mongodb://localhost:27017",
    "host" : "localhost",
    "port" : "27017",
    "user" : "myUserAdmin",
    "password" : "Passw0rd",
    "userDatabase" : "admin",
    "database" : "test",
    ...
}
```

Set `"enabled" : true` to enable the connector.

MongoDB connector bundled scripts

The connector bundles two sets of sample Groovy scripts:

MongoDB management example scripts

- `CreateMongoDB.groovy`
- `DeleteMongoDB.groovy`
- `SchemaMongoDB.groovy`
- `SearchMongoDB.groovy`
- `TestMongoDB.groovy`
- `UpdateMongoDB.groovy`

MongoDB user and role system management example scripts

- `usersRoles/Create.groovy`
- `usersRoles/Delete.groovy`
- `usersRoles/Schema.groovy`
- `usersRoles/Search.groovy`
- `usersRoles/Test.groovy`
- `usersRoles/Update.groovy`

You can customize these scripts to suit your deployment by extracting them from the connector JAR and updating the connector configuration to point to the new file path.

Test the MongoDB connector

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

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system?_action=test"
[
  {
    "name": "mongodb",
    "enabled": true,
    "config": "config/provisioner.openicf/mongodb",
    "connectorRef": {
      "bundleVersion": "[1.5.0.0,1.6.0.0)",
      "bundleName": "org.forgerock.openicf.connectors.mongodb-connector",
      "connectorName": "org.forgerock.openicf.connectors.mongodb.MongoDBConnector"
    },
    "displayName": "MongoDB Connector",
    "objectTypes": [
      "__ALL__",
      "account",
      "role"
    ],
    "ok": true
  }
]
```

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

MongoDB remote connector

If you want to run this connector outside of PingOne Advanced Identity Cloud or IDM, you can configure the MongoDB connector as a remote connector. Java Connectors installed remotely on a Java Connector Server function identically to those bundled locally within PingOne Advanced Identity Cloud or installed locally on IDM.

You can download the MongoDB connector from here.

Refer to Remote connectors for configuring the MongoDB remote connector.

Configure connection pooling

The MongoDB connector supports HTTP pooling, which can substantially improve the performance of the connector. Learn more about the basic connection pooling configuration and different pooling mechanisms described in Connection pooling configuration.

OpenICF Interfaces Implemented by the MongoDB Connector

The MongoDB Connector implements the following OpenICF interfaces. For additional details, see ICF interfaces:

Authenticate

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

Create

Creates an object and its `uid`.

Delete

Deletes an object, referenced by its `uid`.

Resolve Username

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

Schema

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

Script on Connector

Enables an application to run a script in the context of the connector.

Any script that runs on the connector has the following characteristics:

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

Script on Resource

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

Search

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

Sync

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

Test

Tests the connector configuration.

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

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

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

Update

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

MongoDB Connector Configuration

The MongoDB Connector has the following configurable properties:

Basic Configuration Properties

Property	Type	Default	Encrypted(1)	Required(2)
`connectionURI`	`String`	`null`		✕ No
The MongoDB client connection URI, for example "mongodb://localhost:27017". Overrides other connection parameters.				
`host`	`String`	`localhost`		✕ No
The MongoDB server host name.				
`port`	`int`	`27017`		✕ No
The MongoDB server port number.				
`user`	`String`	`null`		✕ No
The MongoDB username.				
`password`	`GuardedString`	`null`	🔒 Yes	✕ No
The password used to connect to MongoDB.				
`userDatabase`	`String`	`null`		✕ No
The name of the database in which the MongoDB user is defined.				
`clusterAddresses`	`String[]`	`null`		✕ No
A list of additional mongodbDB servers when connecting to a MongoDB cluster (["host1:27017","host2:27017",...]).				
`dateAttributes`	`String[]`	`[]`		✕ No
Defines the list of attributes to convert to MongoDB BSON Date type on create/update.				
`database`	`String`	`null`		✕ No
The database to use.				
`arrayAttributes`	`String[]`	`[]`		✕ No

Property	Type	Default	Encrypted[1]	Required[2]
Defines the list of attributes that should be considered as BSON Arrays.				
`includeNullValue`	`boolean`	`false`		✕ No
If set to true, retains null values in the target MongoDB document.				
`includeEmptyList`	`boolean`	`false`		✕ No
If set to true, retains null values in the target MongoDB document.				
`dateFormat`	`String`	`yyyy-MM-dd’T’HH:mm:ss’Z'`		✕ No
Defines the date format to use for MongoDB Date attributes (defaults to ISO 8601 "yyyy-MM-ddTHH:mm:ssZ").				
`timeZone`	`String`	`UTC`		✕ No
Defines the timezone to use for MongoDB Date attributes.				
`ICFName`	`String`	`name`		✕ No
Defines the name to use in the target MongoDB document for the ICF *NAME* attribute.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

Connection Configuration Properties

Property	Type	Default	Encrypted[1]	Required[2]
`sslEnabled`	`boolean`	`true`		✕ No
Use secure socket layer to connect to MongoDB.				
`sslHostNameValidation`	`boolean`	`true`		✕ No
Defines if host name should be validated when SSL is enabled.				
`maxConnectionIdleTime`	`int`	`0`		✕ No
The maximum idle time for a pooled connection in ms (0 means no limit).				
`maxConnectionLifeTime`	`int`	`0`		✕ No
The maximum life time for a pooled connection in ms (0 means no limit).				

Property	Type	Default	Encrypted[1]	Required[2]
`minConnectionsPerHost`	`int`	`0`		✗ No
The minimum number of connections per host (must be >= 0).				
`maxConnectionsPerHost`	`int`	`5`		✗ No
The maximum number of connections per host (must be > 0).				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

Groovy Engine configuration

Property	Type	Default	Encrypted[1]	Required[2]
`scriptRoots`	`String[]`	`['!/scripts/mongodb/']`		✓ Yes
The root folder to load the scripts from. If the value is null or empty the classpath value is used.				
`classpath`	`String[]`	`[]`		✗ No
Classpath for use during compilation.				
`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.				
`minimumRecompilationInterval`	`int`	`100`		✗ No
Sets the minimum of time after a script can be recompiled.				
`recompileGroovySource`	`boolean`	`false`		✗ No
If set to true recompilation is enabled.				
`scriptBaseClass`	`String`	`null`		✗ No
Base class name for scripts (must derive from Script).				
`scriptExtensions`	`String[]`	`['groovy']`		✗ No

Property	Type	Default	Encrypted[1]	Required[2]
Gets the extensions used to find groovy files.				
`sourceEncoding`	`String`	`UTF-8`		✕ No
Encoding for source files.				
`targetDirectory`	`File`	`null`		✕ No
Directory into which to write classes.				
`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.				
`verbose`	`boolean`	`false`		✕ No
If true, the compiler should produce action information.				
`warningLevel`	`int`	`1`		✕ No
Warning Level of the compiler.				
`customConfiguration`	`String`	`null`		✕ No
Custom Configuration script for Groovy ConfigSlurper.				
`customSensitiveConfiguration`	`GuardedString`	`null`	🔒 Yes	✕ No
Custom Sensitive Configuration script for Groovy ConfigSlurper.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

Operation Script Files

Property	Type	Default	Encrypted[1]	Required[2]
`authenticateScriptFileName`	`String`	`null`		• Authenticate
The name of the file used to perform the AUTHENTICATE operation.				
`createScriptFileName`	`String`	`usersRoles/Create.groovy`		• Create

Property	Type	Default	Encrypted[(1)]	Required[(2)]
The name of the file used to perform the CREATE operation.				
`customizerScriptFileName`	`String`	`null`		✗ No
The script used to customize some function of the connector. Read the documentation for more details.				
`deleteScriptFileName`	`String`	`usersRoles/Delete.groovy`		• Delete
The name of the file used to perform the DELETE operation.				
`resolveUsernameScriptFileName`	`String`	`null`		• Resolve Username
The name of the file used to perform the RESOLVE_USERNAME operation.				
`schemaScriptFileName`	`String`	`usersRoles/Schema.groovy`		• Schema
The name of the file used to perform the SCHEMA operation.				
`scriptOnResourceScriptFileName`	`String`	`null`		• Script on Resource
The name of the file used to perform the RUNSCRIPTONRESOURCE operation.				
`searchScriptFileName`	`String`	`usersRoles/Search.groovy`		• Read • Search
The name of the file used to perform the SEARCH operation.				
`syncScriptFileName`	`String`	`null`		• Sync
The name of the file used to perform the SYNC operation.				
`testScriptFileName`	`String`	`usersRoles/Test.groovy`		• Test
The name of the file used to perform the TEST operation.				

Property	Type	Default	Encrypted[1]	Required[2]
`updateScriptFileName`	`String`	`usersRoles/Update.groovy`		- Update
The name of the file used to perform the UPDATE operation.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

Multiple CSV connector

Important

You can only use Multiple CSV connector version 1.5.20.29 and later with:

- Connector framework 1.5.20.24 or later
- RCS 1.5.20.24 or later

Learn more in Changed functionality.

The Multiple CSV connector allows for resources such as users and groups to be retrieved from one or more CSV files as defined by the connector configuration. When used continuously in production, CSV files serve as a change log, often containing only records that have changed.

Warning

- This connector does not verify CSV data. You must ensure that your CSV file is complete and properly formed before using the connector.
- Do not remove or replace CSV files that are the source or target of an active scheduled reconciliation.
- Modifying the CSV header by adding, removing, or modifying the columns requires reconfiguring the connector for the changes to take effect.

Install the Multiple CSV connector

Tip

To check for an Advanced Identity Cloud application for this connector, refer to:

- Application management
- App catalog

You can download any connector from Backstage, but some are included in the default deployment for Advanced Identity Cloud, IDM, or RCS. When using an included connector, you can skip installing it and move directly to configuration.

Connector included in default deployment

Connector	IDM	RCS
Multiple CSV	✕ No	✕ No

Download the connector .jar file from Backstage.

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

  ```
  mv ~/Downloads/multicsv-connector-1.5.20.28.jar /path/to/openidm/connectors/
  ```

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

Configure the Multiple CSV connector

Create a connector configuration using the IDM admin UI:

1. From the navigation bar, click **Configure > Connectors**.
2. On the **Connectors** page, click **New Connector**.
3. On the **New Connector** page, type a **Connector Name**.
4. From the **Connector Type** drop-down list, select **Multiple CSV Connector - 1.5.20.28**.
5. Complete the **Base Connector Details**.

   > **Tip**
   >
   > For a list of all configuration properties, refer to Multiple CSV Connector Configuration

6. Click **Save**.

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

Refer to this procedure to create a connector configuration over REST.

```
{
  "configurationProperties": {
    "minBytesToEstimate": 100000000,
    "multiValuedFieldDelimiter": ";",
    "newLineString": "\n",
    "quotationMode": "ALL",
    "csvFiles": [
      {
        "objectType": "__ACCOUNT__",
        "path": "&{idm.instance.dir}/data/file_name.csv",
        "uid": "_CHANGEME_",
        "name": "_CHANGEME_",
        "password": "_CHANGEME_",
        "multiValuedAttr": [
          "_CHANGEME_"
        ]
      },
      {
        "objectType": "__GROUP__",
        "path": "&{idm.instance.dir}/data/file_name.csv",
        "name": "_CHANGEME_",
        "uid": "_CHANGEME_",
        "multiValuedAttr": [
          "_CHANGEME_"
        ]
      },
      {
        "objectType": "__BALANCE__",
        "path": "&{idm.instance.dir}/data/file_name.csv",
        "name": "_CHANGEME_",
        "uid": "_CHANGEME_"
      }
    ],
    "quoteCharacter": "\"",
    "escapeCharacter": "\\",
    "fieldDelimiter": ",",
    "ignoreSurroundingSpaces": false
  }
}
```

minBytesToEstimate

The total number of rows in the CSV file will begin to be estimated when the file size in bytes exceeds the value specified in this field. The default value is 100MB.

multiValuedFieldDelimiter

String value that delimits each field inside a multivalued field.

newLineString

Indicates the character that represents a new line in the CSV file.

`quotationMode`

Defines quoting behavior.

- Available values:
  - `ALL` : Quotes all fields.
  - `ALL_NON_NULL` : Quotes all non-null fields.
  - `MINIMAL` : Quotes fields that contain special characters such as a field delimiter, quote character, or any of the characters in the line separator string.
  - `NON_NUMERIC` : Quotes all non-numeric fields.
  - `NONE` : Never quotes fields.

`csvFiles`

This field is required. It’s an array of objects, where each object represents the CSV file associated with a corresponding object type.

`objectType`

The name of the object type.

`path`

The path where the CSV file is located.

`name`

The field that represents the name of the record.

`password`

The field to be handled as a password. It will be hidden and encrypted.

`uid`

The field that represents the ID of the record

`multiValuedAttr`

Refers to an array that can contain multiple values for the same row.

`quoteCharacter`

Defines the characters used to enclose values.

`escapeCharacter`

Defines the characters used to escape special characters.

`fieldDelimiter`

String value that delimits each field. Defaults to `","`.

`ignoreSurroundingSpaces`

Ignores whitespace around field separators. Can alter CSV structure. Defaults to `false`.

> **Note**
>
> The character used for `quoteCharacter`, `escapeCharacter`, and `fieldDelimiter` must be unique.

Multiple CSV remote connector

If you want to run this connector outside of PingOne Advanced Identity Cloud or IDM, you can configure the Multiple CSV connector as a remote connector. Java Connectors installed remotely on a Java Connector Server function identically to those bundled locally within PingOne Advanced Identity Cloud or installed locally on IDM.

You can download the Multiple CSV connector from here.

Refer to Remote connectors for configuring the Multiple CSV remote connector.

Configure connection pooling

The Multiple CSV connector uses a non-poolable mechanism to manage connections. Learn more about the different pooling mechanisms in Connectors by pooling mechanism.

Use the Multiple CSV connector

> **Important**
>
> In the following example, we will use account, but it can be replaced by any object type.
> If you change the structure of the CSV file resource by adding or removing columns, you must update the corresponding object `type` in the connector configuration accordingly.

You can use the Multiple CSV connector to perform the following actions.

List all Multiple CSV accounts

This example queries all Multiple CSV accounts:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--request GET \
"http://localhost:8080/openidm/system/Multiple CSV/__ACCOUNT__?_queryFilter=true"
{
  "result": [
    {
      "_id": "9012",
      "Department": "Sales",
      "First_name": "Rachel",
      "Recovery_code": "rb9012",
      "Phones": [
        "1234",
        "5678"
      ],
      "__NAME__": "rachel@example.com",
      "Last_name": "Booker",
      "Location": "Manchester"
    },
    ...
    {
      "_id": "2070",
      "Department": "Depot",
      "First_name": "Laura",
      "Recovery_code": "lg2070",
      "Phones": [
        "1234",
        "5678"
      ],
      "__NAME__": "laura@example.com",
      "Last_name": "Grey",
      "Location": "London"
    }
  ],
  "resultCount": 10,
  "pagedResultsCookie": null,
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

Note

The value of the `totalPagedResults` field is estimated for files larger than 100000000 (100MB).

List Multiple CSV field account by ID

This example queries Multiple CSV file account by ID:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--request GET \
"http://localhost:8080/openidm/system/Multiple CSV/__ACCOUNT__/9012"
{
  "_id": "9012",
  "Department": "Sales",
  "First_name": "Rachel",
  "Recovery_code": "rb9012",
  "Phones": [
      "1234",
      "5678"
  ],
  "__NAME__": "rachel@example.com",
  "Last_name": "Booker",
  "Location": "Manchester"
}
```

Create a Multiple CSV file full account

This example creates a Multiple CSV file account with the minimum required attributes.

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request POST \
--data '{
{
  "__UID__": "7854",
  "__NAME__": "Jane Doe",
  "__PASSWORD__": "12345678",
  "email": "janedoe@example.com"
  "First_name": "Jane",
  "Last_name" : "Doe",
  "Department": "Sales",
  "Location": "London",
  "Phones": [1234,5678]
}' \
"http://localhost:8080/openidm/system/connectorName/__ACCOUNT__?_action=create"
{
  "_id": "859d70a5-573e-4428-8294-1a13a38e0fec",
  "Department": "Sales",
  "First_name": "Jane",
  "Recovery_code": "cj4082",
  "Phones": [
    "123",
    "45678"
  ],
  "__NAME__": "janedoe@example.com",
  "Last_name": "Doe",
  "Location": "London"
}
```

Note

A record can be created with a specific __UID__, or a random ID is assigned if not specified. When you create a new account you must specify at least the `__NAME__` attribute.

Update a Multiple CSV account

The following command updates a specific Multiple CSV account by its ID:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "If-Match: *" \
--request PUT \
--data '{
  "Department": "Engineering",
  "First_name": "Jhon",
  "Phones": [876,54321],
  "__NAME__": "jhondoe@example.com",
  "Last_name": "Doe",
  "Location": "Manchester"
}' \
'http://localhost:8080/openidm/system/Multiple CSV/__ACCOUNT__/859d70a5-573e-4428-8294-1a13a38e0fec'
{
  "_id": "859d70a5-573e-4428-8294-1a13a38e0fec",
  "Department": "Engineering",
  "First_name": "Jhon",
  "Recovery_code": "cj4082",
  "Phones": [
    "876",
    "54321"
  ],
  "__NAME__": "jhondoe@example.com",
  "Last_name": "Doe",
  "Location": "Manchester"
}
```

Delete a Multiple CSV account

The following example deletes a Multiple CSV account:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request DELETE \
'http://localhost:8080/openidm/system/Multiple CSV/__ACCOUNT__/859d70a5-573e-4428-8294-1a13a38e0fec'
{
  "_id": "859d70a5-573e-4428-8294-1a13a38e0fec",
  "Department": "Engineering",
  "First_name": "Jhon",
  "Recovery_code": "cj4082",
  "Phones": [
    "876",
    "54321"
  ],
  "__NAME__": "jhondoe@example.com",
  "Last_name": "Doe",
  "Location": "Manchester"
}
```

> **Note**
>
> The response returns the account object before deletion.

OpenICF Interfaces Implemented by the Multi CSV Connector

The Multi CSV Connector implements the following OpenICF interfaces. For additional details, see ICF 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.

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.

Multi CSV Connector Configuration

The Multi CSV Connector has the following configurable properties:

Configuration properties

Property	Type	Default	Encrypted[1]	Required[2]
`csvFiles`	`Map[]`	`['{path=null, uid=null, password=null, multiValuedAttr=[], name=null, objectType=null}']`		✓ Yes
Description is not available				
`quoteCharacter`	`String`	`"`		✕ No
Description is not available				
`fieldDelimiter`	`String`	`,`		✕ No
Description is not available				
`escapeCharacter`	`String`	`\`		✕ No
Description is not available				
`newLineString`	`String`	`\n`		✕ No
Description is not available				
`spaceReplacementString`	`String`	`_`		✕ No
Description is not available				
`minBytesToEstimate`	`double`	`1.0E8`		✕ No

Property	Type	Default	Encrypted[1]	Required[2]
Description is not available				
`multiValuedFieldDelimiter`	`String`	`null`		✕ No
Description is not available				
`quotationMode`	`String`	`ALL`		✕ No
Description is not available				
`ignoreSurroundingSpaces`	`boolean`	`false`		✕ No
Description is not available				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

Oracle EBS connector

The Oracle E-Business Suite (EBS) connector enables you to manage EBS accounts and synchronize accounts between EBS and the IDM managed user repository.

Before you start

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

Username

Your EBS administrator account username.

Password

Your EBS administrator account password.

JDBC Connection URL

The URL to establish the connection between the connector and the EBS application.

For more information, refer to the Oracle E-Business Suite documentation.

Install the EBS connector

> **Tip**
>
> To check for an Advanced Identity Cloud application for this connector, refer to:
>
> - Application management
> - App catalog

You can download any connector from Backstage, but some are included in the default deployment for Advanced Identity Cloud, IDM, or RCS. When using an included connector, you can skip installing it and move directly to configuration.

Connector included in default deployment

Connector	IDM	RCS
Oracle EBS	✗ No	✓ Yes

Download the connector .jar file from Backstage.

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

```
mv ~/Downloads/ebs-connector-1.5.20.26.jar /path/to/openidm/connectors/
```

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

Download the Oracle JDBC driver (`ojdbc8.jar`).

- If you are running the connector locally, place the library in the `/path/to/openidm/lib/` directory:

```
mv ~/Downloads/ojdbc8.jar /path/to/openidm/lib/
```

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

Configure the EBS connector

Create a connector configuration using the IDM admin UI:

1. From the navigation bar, click **Configure > Connectors**.

2. On the **Connectors** page, click **New Connector**.

3. On the **New Connector** page, type a **Connector Name**.

4. From the **Connector Type** drop-down list, select **EBS Connector - 1.5.20.26**.

5. Complete the **Base Connector Details**.

> **Tip**
>
> For a list of all configuration properties, refer to EBS Connector Configuration

6. Click **Save**.

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

Refer to this procedure to create a connector configuration over REST.

Test the EBS connector

Test that the configuration is correct by running the following command:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"https://localhost:8443/openidm/system/EBS?_action=test"
{
  "name" : "EBS",
  "enabled" : true,
  "config" : "config/provisioner.openicf/EBS",
  "connectorRef" : {
    "bundleVersion" : [1.5.0.0,1.6.0.0),
    "bundleName" : "org.forgerock.openicf.connectors.ebs-connector",
    "connectorName" : "org.forgerock.openicf.connectors.oracleebs.OracleEbsConnector"
  },
  "displayName" : "Oracle EBS Connector",
  "objectTypes" : [
    "__ACCOUNT__",
    "__ALL__"
  ],
  "ok" : true
}
```

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

EBS remote connector

If you want to run this connector outside of PingOne Advanced Identity Cloud or IDM, you can configure the EBS connector as a remote connector. Java Connectors installed remotely on a Java Connector Server function identically to those bundled locally within PingOne Advanced Identity Cloud or installed locally on IDM.

You can download the EBS connector from here.

Refer to Remote connectors for configuring the EBS remote connector.

Configure connection pooling

The EBS connector embeds the Apache Tomcat 9 JDBC Connection Pool. Learn more about the different pooling mechanisms in Connectors by pooling mechanism.

Supported resource types

The EBS connector supports the following resource types:

EBS connector supported resource types

ICF Native Type	EBS Resource Type	Naming Attribute
__ACCOUNT__	User	__NAME__
RESPONSIBILITY	Responsibilities	__NAME__

Supported search filters

The EBS connector supports Search operations with the following filter operators and attributes:

Supported Operators and Filter Attributes With EBS Searches

Object Type	Operators	Attributes
__ACCOUNT__	And, Contains, StartsWith, Equals, EndsWith, Or	• USER_ID • USER_NAME • LAST_UPDATE_DATE • LAST_UPDATE_BY • CREATION_DATE • CREATED_BY • LAST_UPDATE_LOGIN • ENCRYPTED_USER_PASSWORD • SESSION_NUMBER • START_DATE • END_DATE • DESCRIPTION • LAST_LOGON_DATE • PASSWORD_DATE • PASSWORD_ACCESSES_LEFT • PASSWORD_LIFESPAN_ACCESSES • PASSWORD_LIFESPAN_DAYS • EMAIL_ADDRESS • FAX • RESPONSIBILITY

Object Type	Operators	Attributes
RESPONSIBILITY	Equals, And, Or	• RESPONSIBILITY_ID • APPLICATION_ID • LAST_UPDATE_DATE • LAST_UPDATED_BY • CREATION_DATE • CREATED_BY • LAST_UPDATE_LOGIN • DATA_GROUP_APPLICATION_ID • DATA_GROUP_ID • MENU_ID • TERM_SECURITY_ENABLED_FLAG • START_DATE • END_DATE • GROUP_APPLICATION_ID • REQUEST_GROUP_ID • VERSION • WEB_HOST_NAME • WEB_AGENT_NAME • RESPONSIBILITY_KEY • RESPONSIBILITY_NAME • SECURITY_GROUP_ID • ZD_EDITION_NAME • ZD_SYNC

Attributes

The following attributes are supported by the connector:

Attributes	Description
`USER_ID`	The user’s User ID
`USER_NAME`	The user’s username
`ENCRYPTED_USER_PASSWORD`	The user’s encrypted password
`SESSION_NUMBER`	Number of sessions
`START_DATE`	Start date for the created user
`END_DATE`	End date for the created user
`DESCRIPTION`	The user’s description
`LAST_LOGON_DATE`	Last logged on date

Attributes	Description
`PASSWORD_DATE`	The date the current password was set
`PASSWORD_ACCESSES_LEFT`	The number of accesses left for the password
`PASSWORD_LIFESPAN_ACCESSES`	The number of accesses allowed for the password
`PASSWORD_LIFESPAN_DAYS`	The number of days allowed for the password
`EMAIL_ADDRESS`	The user’s email address
`FAX`	The user’s fax number

Use the EBS connector

The EBS connector can perform the following actions:

Users

The following example creates a user with all the creatable attributes:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request POST \
--data
'{
  "__NAME__": "BJENSEN",
  "__PASSWORD__": "Test@123",
  "EMAIL_ADDRESS": "bjensen@forgerock.com",
  "SESSION_NUMBER": "2",
  "START_DATE": "03-Nov-22",
  "END_DATE": "08-Nov-22",
  "LAST_LOGON_DATE": "08-DEC-2021",
  "PASSWORD_DATE": "08-JUN-2021",
  "PASSWORD_ACCESSES_LEFT": "1",
  "PASSWORD_LIFESPAN_ACCESSES": "1",
  "PASSWORD_LIFESPAN_DAYS": "1"
}' \
"https://localhost:8443/openidm/system/EBS/__ACCOUNT__?_action=create"
{
  "_id": "1015488",
  "USER_ID": "1015488",
  "START_DATE": "03-Nov-22",
  "LAST_UPDATE_LOGIN": 1015131,
  "USER_NAME": "BJENSEN",
  "__ENABLE__": false,
  "EMAIL_ADDRESS": "bjensen@forgerock.com",
  "SESSION_NUMBER": 2,
  "LAST_LOGON_DATE": "08-Dec-21",
  "PASSWORD_ACCESSES_LEFT": 1,
  "PASSWORD_LIFESPAN_ACCESSES": 1,
  "END_DATE": "08-Nov-22",
  "PASSWORD_LIFESPAN_DAYS": 1,
  "PASSWORD_DATE": "08-Jun-21",
  "__NAME__": "BJENSEN",
  "LAST_UPDATE_DATE": "02-Dec-22"
}
```

Important

When you create a new user, you must specify at least the `__NAME__` attribute. `__NAME__` has a maximum length of 100 characters, should be in UPPER CASE, and must be unique.

The following queries a specific user by their ID:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"https://localhost:8443/openidm/system/EBS/__ACCOUNT__/1015488"
{
  "_id": "1015488",
  "USER_ID": "1015488",
  "START_DATE": "03-Nov-22",
  "LAST_UPDATE_LOGIN": 1015131,
  "USER_NAME": "BJENSEN",
  "__ENABLE__": false,
  "EMAIL_ADDRESS": "bjensen@forgerock.com",
  "SESSION_NUMBER": 7,
  "LAST_LOGON_DATE": "08-Dec-21",
  "PASSWORD_ACCESSES_LEFT": 1,
  "PASSWORD_LIFESPAN_ACCESSES": 1,
  "END_DATE": "08-Nov-22",
  "PASSWORD_LIFESPAN_DAYS": 1,
  "PASSWORD_DATE": "08-Jun-21",
  "__NAME__": "BJENSEN",
  "DESCRIPTION": "ebsuser",
  "LAST_UPDATE_DATE": "02-Dec-22"
}
```

The following example queries all users:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
"https://localhost:8443/openidm/system/EBS/__ACCOUNT__?_queryId=query-all-ids"
{
  "result":[
    {"_id":"1000001"},
    {"_id":"3"},
    {"_id":"2"},
    {"_id":"0"},
    {"_id":"1001"},
    {"_id":"1555"},
    {"_id":"1003"},
    {"_id":"1004"},
    {"_id":"1005"},
    {"_id":"1007"}
    ],
  "resultCount":10,
  "pagedResultsCookie":null,
  "totalPagedResultsPolicy":"NONE",
  "totalPagedResults":-1,
  "remainingPagedResults":-1
}
```

The EBS Connector can modify the following attributes of a user entry:

- `__PASSWORD__`
- `__ENABLE__`
- `EMAIL_ADDRESS`
- `START_DATE`
- `END_DATE`

The following example updates a user:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "If-Match: *" \
--header "Content-Type: application/json" \
--request  PUT \
--data '{
  "__NAME__": "BJENSEN",
  "__PASSWORD__": "Test@123",
  "EMAIL_ADDRESS": "bjensen@forgerock.com",
  "SESSION_NUMBER": "7",
  "START_DATE": "03-Nov-22",
  "END_DATE": "08-Nov-22",
  "LAST_LOGON_DATE": "08-DEC-2021",
  "PASSWORD_DATE": "08-JUN-2021",
  "PASSWORD_ACCESSES_LEFT": "1",
  "PASSWORD_LIFESPAN_ACCESSES": "1",
  "PASSWORD_LIFESPAN_DAYS": "1",
  "DESCRIPTION": "ebsuser"
}' \
"https://localhost:8443/openidm/system/EBS/__ACCOUNT__/1015488"
{
  "_id" : "1015488",
  "USER_ID" : "1015488",
  "START_DATE" : "03-Nov-22",
  "LAST_UPDATE_LOGIN" : 1015131,
  "USER_NAME" : "BJENSEN",
  "__ENABLE__" : false,
  "EMAIL_ADDRESS" : "bjensen@forgerock.com",
  "SESSION_NUMBER" : 7,
  "LAST_LOGON_DATE" : "08-Dec-21",
  "PASSWORD_ACCESSES_LEFT" : 1,
  "PASSWORD_LIFESPAN_ACCESSES" : 1,
  "END_DATE" : "08-Nov-22",
  "PASSWORD_LIFESPAN_DAYS" : 1,
  "PASSWORD_DATE" : "08-Jun-21",
  "__NAME__" : "BJENSEN",
  "DESCRIPTION" : "ebsuser",
  "LAST_UPDATE_DATE" : "02-Dec-22"
}
```

To reset the password for a user account, update the user's `"__PASSWORD__"` attribute:

```
curl \
--header "Content-Type: application/json" \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "if-Match:*" \
--request PUT \
--data '{
  "__NAME__": "BJENSEN",
  "__PASSWORD__": "RRvts125!"
}' \
"https://localhost:8443/openidm/system/EBS/__ACCOUNT__/1015488"
{
  "_id" : "1015488",
  "USER_ID" : "1015488",
  "START_DATE" : "03-Nov-22",
  "LAST_UPDATE_LOGIN" : 1015131,
  "USER_NAME" : "BJENSEN",
  "__ENABLE__" : false,
  "EMAIL_ADDRESS" : "bjensen@forgerock.com",
  "SESSION_NUMBER" : 7,
  "LAST_LOGON_DATE" : "08-Dec-21",
  "PASSWORD_ACCESSES_LEFT" : 1,
  "PASSWORD_LIFESPAN_ACCESSES" : 1,
  "END_DATE" : "08-Nov-22",
  "PASSWORD_LIFESPAN_DAYS" : 1,
  "PASSWORD_DATE" : "08-Jun-21",
  "__NAME__": "BJENSEN",
  "DESCRIPTION" : "ebsuser",
  "LAST_UPDATE_DATE" : "02-Dec-22"
}
```

The following example activates a user:

```
curl \
--header "Content-Type: application/json" \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "if-Match:*" \
--request PUT \
--data '{
  "__NAME__":"BJENSEN",
  "__PASSWORD__":"Rvts12345",
  "__ENABLE__": true
}' \
"https://localhost:8443/openidm/system/EBS/__ACCOUNT__/1015488"
{
  "_id" : "1015488",
  "USER_ID" : "1015488",
  "START_DATE" : "02-Dec-22",
  "LAST_UPDATE_LOGIN" : -1,
  "USER_NAME" : "BJENSEN",
  "__ENABLE__" : true,
  "EMAIL_ADDRESS" : "bjensen@forgerock.com",
  "SESSION_NUMBER" : 7,
  "LAST_LOGON_DATE" : "08-Dec-21",
  "PASSWORD_ACCESSES_LEFT" : 1,
  "PASSWORD_LIFESPAN_ACCESSES" : 1,
  "PASSWORD_LIFESPAN_DAYS" : 1,
  "PASSWORD_DATE" : "08-Jun-21",
  "__NAME__" : "BJENSEN",
  "DESCRIPTION" : "ebsuser",
  "LAST_UPDATE_DATE" : "02-Dec-22"
}
```

The following example deactivates a user:

```
curl \
--header "Content-Type: application/json" \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "if-Match:*" \
--request PUT \
--data '{
  "__NAME__":"BJENSEN",
  "__PASSWORD__":"Rvts12345",
  "__ENABLE__": false
}' \
"https://localhost:8443/openidm/system/EBS/__ACCOUNT__/1015488"
{
  "_id" : "1015488",
  "USER_ID" : "1015488",
  "START_DATE" : "03-Nov-22",
  "LAST_UPDATE_LOGIN" : -1,
  "USER_NAME" : "BJENSEN",
  "__ENABLE__" : false,
  "EMAIL_ADDRESS" : "bjensen@forgerock.com",
  "SESSION_NUMBER" : 7,
  "LAST_LOGON_DATE" : "08-Dec-21",
  "PASSWORD_ACCESSES_LEFT" : 1,
  "PASSWORD_LIFESPAN_ACCESSES" : 1,
  "END_DATE" : "02-Dec-22",
  "PASSWORD_LIFESPAN_DAYS" : 1,
  "PASSWORD_DATE" : "08-Jun-21",
  "__NAME__" : "BJENSEN",
  "DESCRIPTION" : "ebsuser",
  "LAST_UPDATE_DATE" : "02-Dec-22"
}
```

Responsibilities

Note

After a user has been assigned a responsibility, the `START_DATE` and `END_DATE` can’t be changed. To edit either of these values, remove the responsibility and then add it again.

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "if-Match:*" \
--request PUT \
--data '{
  "__NAME__": "STUBBS",
  "__PASSWORD__": "Test@231",
  "EMAIL_ADDRESS": "STUBBS.SON@example.com",
  "SESSION_NUMBER": "3",
  "FAX": "56",
  "START_DATE": "15-Jan-24",
  "END_DATE": "23-Jan-24",
  "LAST_LOGON_DATE": "23-DEC-2021",
  "PASSWORD_DATE": "12-JUN-2022",
  "PASSWORD_ACCESSES_LEFT": "2",
  "PASSWORD_LIFESPAN_ACCESSES": "3",
  "DESCRIPTION": "EBSBARUSER",
  "PASSWORD_LIFESPAN_DAYS": "2",
  "RESPONSIBILITY": [
    {
      "RESP_APP": "IBE",
      "RESP_KEY": "IBE_CUSTOMER",
      "RESPONSIBILITY_SECURITY_GROUP": "STANDARD",
      "RESPONSIBILITY_DESCRIPTION": "updated",
      "RESPONSIBILITY_START_DATE": "29-SEP-2021",
      "RESPONSIBILITY_END_DATE": "29-SEP-2024"
    },
    {
      "RESP_APP": "CS",
      "RESP_KEY": "CS_KB_SYSTEM_ADMIN",
      "RESPONSIBILITY_SECURITY_GROUP": "STANDARD",
      "RESPONSIBILITY_DESCRIPTION": "updated",
      "RESPONSIBILITY_START_DATE": "29-SEP-2021",
      "RESPONSIBILITY_END_DATE": "29-SEP-2024"
    },
    {
      "RESP_APP": "PA",
      "RESP_KEY": "PROJECT_BILLING_SUPER_USER",
      "RESPONSIBILITY_SECURITY_GROUP": "STANDARD",
      "RESPONSIBILITY_DESCRIPTION": "responded",
      "RESPONSIBILITY_START_DATE": "15-JUL-2021",
      "RESPONSIBILITY_END_DATE": "13-SEP-2024"
    }
  ]
}' \
"https://localhost:8443/openidm/system/EBS/__ACCOUNT__/1017315"
{
  "_id": "1017315",
  "PASSWORD_DATE": "12-Jun-22",
  "LAST_UPDATE_LOGIN": 1015131,
  "__ENABLE__": false,
  "PASSWORD_ACCESSES_LEFT": 2,
  "__NAME__": "STUBBS",
  "END_DATE": "23-Jan-24",
  "DESCRIPTION": "EBSBARUSER",
  "EMAIL_ADDRESS": "STUBBS.SON@example.com",
  "LAST_LOGON_DATE": "23-Dec-21",
  "RESPONSIBILITY": [
    {
      "RESP_KEY": "PROJECT_BILLING_SUPER_USER",
      "RESPONSIBILITY_SECURITY_GROUP": "STANDARD",
      "RESP_APP": "PA",
      "RESPONSIBILITY_DESCRIPTION": "responded",
      "RESPONSIBILITY_START_DATE": "15-Jul-21",
      "RESPONSIBILITY_END_DATE": "13-Sep-24"
    },
    {
      "RESP_KEY": "CS_KB_SYSTEM_ADMIN",
      "RESPONSIBILITY_SECURITY_GROUP": "STANDARD",
      "RESP_APP": "CS",
      "RESPONSIBILITY_DESCRIPTION": "updated",
      "RESPONSIBILITY_START_DATE": "29-Sep-21",
      "RESPONSIBILITY_END_DATE": "29-Sep-24"
    },
    {
      "RESP_KEY": "IBE_CUSTOMER",
      "RESPONSIBILITY_SECURITY_GROUP": "STANDARD",
      "RESP_APP": "IBE",
      "RESPONSIBILITY_DESCRIPTION": "updated",
      "RESPONSIBILITY_START_DATE": "29-Sep-21",
      "RESPONSIBILITY_END_DATE": "29-Sep-24"
    }
  ],
  "PASSWORD_LIFESPAN_DAYS": 2,
  "START_DATE": "15-Jan-24",
  "LAST_UPDATE_DATE": "17-May-24",
  "PASSWORD_LIFESPAN_ACCESSES": 3,
  "USER_ID": "1017315",
  "SESSION_NUMBER": 3,
  "FAX": "56"
}
```

Note

When you remove a responsibility the `END_DATE` is updated to the current date.

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "if-Match:*" \
--request PUT \
--data '{
  "__NAME__": "STUBBS",
  "__PASSWORD__": "Test@231",
  "EMAIL_ADDRESS": "STUBBS.SON@example.com",
  "SESSION_NUMBER": "3",
  "FAX": "56",
  "START_DATE": "15-Jan-24",
  "END_DATE": "23-Jan-24",
  "LAST_LOGON_DATE": "23-DEC-2021",
  "PASSWORD_DATE": "12-JUN-2022",
  "PASSWORD_ACCESSES_LEFT": "2",
  "PASSWORD_LIFESPAN_ACCESSES": "3",
  "DESCRIPTION": "EBSBARUSER",
  "PASSWORD_LIFESPAN_DAYS": "2",
  "RESPONSIBILITY": [
    {
      "RESP_APP": "IBE",
      "RESP_KEY": "IBE_CUSTOMER",
      "RESPONSIBILITY_SECURITY_GROUP": "STANDARD",
      "RESPONSIBILITY_DESCRIPTION": "updated",
      "RESPONSIBILITY_START_DATE": "29-SEP-2021",
      "RESPONSIBILITY_END_DATE": "29-SEP-2024"
    },
    {
      "RESP_APP": "CS",
      "RESP_KEY": "CS_KB_SYSTEM_ADMIN",
      "RESPONSIBILITY_SECURITY_GROUP": "STANDARD",
      "RESPONSIBILITY_DESCRIPTION": "updated",
      "RESPONSIBILITY_START_DATE": "29-SEP-2021",
      "RESPONSIBILITY_END_DATE": "29-SEP-2024"
    }
  ]
}' \
"https://localhost:8443/openidm/system/EBS/__ACCOUNT__/1017315"
{
  "_id": "1017315",
  "PASSWORD_DATE": "12-Jun-22",
  "LAST_UPDATE_LOGIN": 1015131,
  "__ENABLE__": false,
  "PASSWORD_ACCESSES_LEFT": 2,
  "__NAME__": "STUBBS",
  "END_DATE": "23-Jan-24",
  "DESCRIPTION": "EBSBARUSER",
  "EMAIL_ADDRESS": "STUBBS.SON@example.com",
  "LAST_LOGON_DATE": "23-Dec-21",
  "RESPONSIBILITY": [
    {
      "RESP_KEY": "PROJECT_BILLING_SUPER_USER",
      "RESPONSIBILITY_SECURITY_GROUP": "STANDARD",
      "RESP_APP": "PA",
      "RESPONSIBILITY_DESCRIPTION": "",
      "RESPONSIBILITY_START_DATE": "15-Jul-21",
      "RESPONSIBILITY_END_DATE": "17-May-24"
    },
    {
      "RESP_KEY": "CS_KB_SYSTEM_ADMIN",
      "RESPONSIBILITY_SECURITY_GROUP": "STANDARD",
      "RESP_APP": "CS",
      "RESPONSIBILITY_DESCRIPTION": "updated",
      "RESPONSIBILITY_START_DATE": "29-Sep-21",
      "RESPONSIBILITY_END_DATE": "29-Sep-24"
    },
    {
      "RESP_KEY": "IBE_CUSTOMER",
      "RESPONSIBILITY_SECURITY_GROUP": "STANDARD",
      "RESP_APP": "IBE",
      "RESPONSIBILITY_DESCRIPTION": "updated",
      "RESPONSIBILITY_START_DATE": "29-Sep-21",
      "RESPONSIBILITY_END_DATE": "29-Sep-24"
    }
  ],
  "PASSWORD_LIFESPAN_DAYS": 2,
  "START_DATE": "15-Jan-24",
  "LAST_UPDATE_DATE": "17-May-24",
  "PASSWORD_LIFESPAN_ACCESSES": 3,
  "USER_ID": "1017315",
  "SESSION_NUMBER": 3,
  "FAX": "56"
}
```

The following example queries all user responsibilities by their IDs:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"https://localhost:8443/openidm/system/EBS/RESPONSIBILITY?_queryId=query-all-ids"
{
  "result": [
    {
      "_id": "21766"
    },
    {
      "_id": "21765"
    },
    {
      "_id": "21764"
    },
    ...
  ],
  "resultCount": 5104,
  "pagedResultsCookie": null,
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

The following example queries a single user responsibility (`50832`):

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"https://localhost:8443/openidm/system/EBS/RESPONSIBILITY?_queryFilter=_id%20eq%20%2250832%22"
{
  "result": [
    {
      "_id": "50832",
      "LAST_UPDATE_LOGIN": 0,
      "ZD_SYNC": "SYNCED",
      "RESPONSIBILITY_KEY": "CHEF CULINAIR",
      "ZD_EDITION_NAME": "SET1",
      "DATA_GROUP_ID": "0",
      "DATA_GROUP_APPLICATION_ID": "20021",
      "MENU_ID": "1000542",
      "WEB_HOST_NAME": "`",
      "START_DATE": "10-Feb-98",
      "CREATION_DATE": "10-Feb-98",
      "LAST_UPDATE_DATE": "10-Feb-98",
      "VERSION": "4",
      "APPLICATION_ID": "20021",
      "RESPONSIBILITY_ID": "50832",
      "LAST_UPDATED_BY": 2,
      "__NAME__": "CHEF CULINAIR",
      "CREATED_BY": 0,
      "RESPONSIBILITY_NAME": "Chef"
    }
  ],
  "resultCount": 1,
  "pagedResultsCookie": null,
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

OpenICF Interfaces Implemented by the Oracle EBS Connector

The Oracle EBS Connector implements the following OpenICF interfaces. For additional details, see ICF interfaces:

Create

Creates an object and its `uid`.

Schema

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

Script on Connector

Enables an application to run a script in the context of the connector.

Any script that runs on the connector has the following characteristics:

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

Search

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

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.

Oracle EBS Connector Configuration

The Oracle EBS Connector has the following configurable properties:

Configuration properties

Property	Type	Default	Encrypted[1]	Required[2]
`connectionProperties`	`String`	`null`		✕ No
The connection properties that will be sent to our JDBC driver when establishing new connections. Format of the string must be [propertyName=property;]* NOTE - The "user" and "password" properties will be passed explicitly, so they do not need to be included here. The default value is null.				
`propagateInterruptState`	`boolean`	`false`		✕ No

Property	Type	Default	Encrypted[1]	Required[2]
Set this to true to propagate the interrupt state for a thread that has been interrupted (not clearing the interrupt state). Default value is false for backwards compatibility.				
`validationQuery`	`String`	`select 1 from dual`		✗ No
The SQL query that will be used to validate connections from this pool before returning them to the caller. If specified, this query does not have to return any data, it just cant throw a SQLException. The default value is null. Example values are SELECT 1(mysql), select 1 from dual(oracle), SELECT 1(MS Sql Server).				
`rollbackOnReturn`	`boolean`	`false`		✗ No
If autoCommit==false then the pool can terminate the transaction by calling rollback on the connection as it is returned to the pool Default value is false.				
`useDisposableConnectionFacade`	`boolean`	`true`		✗ No
Set this to true if you wish to put a facade on your connection so that it cannot be reused after it has been closed. This prevents a thread holding on to a reference of a connection it has already called closed on, to execute queries on it.				
`defaultCatalog`	`String`	`null`		✗ No
The default catalog of connections created by this pool.				
`validationInterval`	`long`	`3000`		✗ No
To avoid excess validation, run validation at most at this frequency (in milliseconds). If a connection is due for validation, but was validated within this interval, it will not be validated again. The default value is 3000 (3 seconds).				
`ignoreExceptionOnPreLoad`	`boolean`	`false`		✗ No
Flag whether ignore error of connection creation while initializing the pool. Set to true if you want to ignore error of connection creation while initializing the pool. Set to false if you want to fail the initialization of the pool by throwing exception.				
`jmxEnabled`	`boolean`	`true`		✗ No
Register the pool with JMX or not. The default value is true.				
`commitOnReturn`	`boolean`	`false`		✗ No
If autoCommit==false then the pool can complete the transaction by calling commit on the connection as it is returned to the pool If rollbackOnReturn==true then this attribute is ignored. Default value is false.				
`logAbandoned`	`boolean`	`false`		✗ No
Flag to log stack traces for application code which abandoned a Connection. Logging of abandoned Connections adds overhead for every Connection borrow because a stack trace has to be generated. The default value is false.				

Property	Type	Default	Encrypted[1]	Required[2]
`alternateUsernameAllowed`	`boolean`	`false`		✕ No
By default, the jdbc-pool will ignore the DataSource.getConnection(username,password) call, and simply return a previously pooled connection under the globally configured properties username and password, for performance reasons. The pool can however be configured to allow use of different credentials each time a connection is requested. To enable the functionality described in the DataSource.getConnection(username,password) call, simply set the property alternateUsernameAllowed to true. Should you request a connection with the credentials user1/password1 and the connection was previously connected using different user2/password2, the connection will be closed, and reopened with the requested credentials. This way, the pool size is still managed on a global level, and not on a per schema level.				
`validatorClassName`	`String`	`null`		✕ No
The name of a class which implements the org.apache.tomcat.jdbc.pool.Validator interface and provides a no-arg constructor (may be implicit). If specified, the class will be used to create a Validator instance which is then used instead of any validation query to validate connections. The default value is null. An example value is com.mycompany.project.SimpleValidator.				
`maxIdle`	`int`	`100`		✕ No
The maximum number of connections that should be kept in the pool at all times. Idle connections are checked periodically (if enabled) and connections that have been idle for longer than minEvictableIdleTimeMillis are released. The default value is derived from maxActive:100. (Also see testWhileIdle).				
`testWhileIdle`	`boolean`	`false`		✕ No
The indication of whether objects will be validated by the idle object evictor (if any). If an object fails to validate, it will be dropped from the pool. NOTE - for a true value to have any effect, the validationQuery parameter must be set to a non-null string. The default value is false and this property has to be set in order for the pool cleaner/test thread is to run (also see timeBetweenEvictionRunsMillis).				
`removeAbandoned`	`boolean`	`false`		✕ No
Flag to remove abandoned connections if they exceed the removeAbandonedTimeout. If set to true a connection is considered abandoned and eligible for removal if it has been in use longer than the removeAbandonedTimeout Setting this to true can recover db connections from applications that fail to close a connection. See also logAbandoned The default value is false.				
`suspectTimeout`	`int`	`0`		✕ No
Timeout value in seconds. Similar to to the removeAbandonedTimeout value but instead of treating the connection as abandoned, and potentially closing the connection, this simply logs the warning if logAbandoned is set to true. If this value is equal or less than 0, no suspect checking will be performed. Suspect checking only takes place if the timeout value is larger than 0 and the connection was not abandoned or if abandon check is disabled. If a connection is suspect a WARN message gets logged and a JMX notification gets sent once.				
`useEquals`	`boolean`	`true`		✕ No

Property	Type	Default	Encrypted[1]	Required[2]
Set to true if you wish the ProxyConnection class to use String.equals and set to false when you wish to use == when comparing method names. This property does not apply to added interceptors as those are configured individually. The default value is true.				
`removeAbandonedTimeout`	`int`	`60`		✕ No
Timeout in seconds before an abandoned(in use) connection can be removed. The default value is 60 (60 seconds). The value should be set to the longest running query your applications might have.				
`defaultAutoCommit`	`Boolean`	`null`		✕ No
The default auto-commit state of connections created by this pool. If not set, default is JDBC driver default (If not set then the setAutoCommit method will not be called).				
`testOnConnect`	`boolean`	`false`		✕ No
Returns true if we should run the validation query when connecting to the database for the first time on a connection. Normally this is always set to false, unless one wants to use the validationQuery as an init query.				
`abandonWhenPercentageFull`	`int`	`0`		✕ No
Connections that have been abandoned (timed out) wont get closed and reported up unless the number of connections in use are above the percentage defined by abandonWhenPercentageFull. The value should be between 0-100. The default value is 0, which implies that connections are eligible for closure as soon as removeAbandonedTimeout has been reached.				
`jdbcInterceptors`	`String`	`null`		✕ No
A semicolon separated list of classnames extending org.apache.tomcat.jdbc.pool.JdbcInterceptor class. See Configuring JDBC interceptors below for more detailed description of syntaz and examples. These interceptors will be inserted as an interceptor into the chain of operations on a java.test_sample.Connection object. The default value is null.				
`minIdle`	`int`	`10`		✕ No
The minimum number of established connections that should be kept in the pool at all times. The connection pool can shrink below this number if validation queries fail. The default value is derived from initialSize:10. (Also see testWhileIdle).				
`defaultReadOnly`	`Boolean`	`null`		✕ No
The default read-only state of connections created by this pool. If not set then the setReadOnly method will not be called. (Some drivers dont support read only mode, ex: Informix).				
`initialSize`	`int`	`10`		✕ No
The initial number of connections that are created when the pool is started. Default value is 10.				
`maxWait`	`int`	`30000`		✕ No

Property	Type	Default	Encrypted[1]	Required[2]
The maximum number of milliseconds that the pool will wait (when there are no available connections) for a connection to be returned before throwing an exception. Default value is 30000 (30 seconds).				
`defaultTransactionIsolation`	`int`	`-1`		✕ No
The default TransactionIsolation state of connections created by this pool. One of the following: NONE, READ_COMMITTED, READ_UNCOMMITTED, REPEATABLE_READ, SERIALIZABLE If not set, the method will not be called and it defaults to the JDBC driver.				
`numTestsPerEvictionRun`	`int`	`0`		✕ No
Property not used in tomcat-jdbc-pool.				
`url`	`String`	`null`		✕ No
The URL used to connect to the database.				
`testOnBorrow`	`boolean`	`true`		✕ No
The indication of whether objects will be validated before being borrowed from the pool. If the object fails to validate, it will be dropped from the pool, and we will attempt to borrow another. NOTE - for a true value to have any effect, the validationQuery parameter must be set to a non-null string. In order to have a more efficient validation, see validationInterval. Default value is false.				
`fairQueue`	`boolean`	`true`		✕ No
Set to true if you wish that calls to getConnection should be treated fairly in a true FIFO fashion. This uses the org.apache.tomcat.jdbc.pool.FairBlockingQueue implementation for the list of the idle connections. The default value is true. This flag is required when you want to use asynchronous connection retrieval. Setting this flag ensures that threads receive connections in the order they arrive. During performance tests, there is a very large difference in how locks and lock waiting is implemented. When fairQueue=true there is a decision making process based on what operating system the system is running. If the system is running on Linux (property os.name=Linux. To disable this Linux specific behavior and still use the fair queue, simply add the property org.apache.tomcat.jdbc.pool.FairBlockingQueue.ignoreOS=true to your system properties before the connection pool classes are loaded.				
`logValidationErrors`	`boolean`	`false`		✕ No
Set this to true to log errors during the validation phase to the log file. If set to true, errors will be logged as SEVERE. Default value is false for backwards compatibility.				
`accessToUnderlyingConnectionAllowed`	`boolean`	`true`		✕ No
Property not used. Access can be achieved by calling unwrap on the pooled connection. see javax.test_sample.DataSource interface, or call getConnection through reflection or cast the object as javax.test_sample.PooledConnection.				
`maxAge`	`long`	`0`		✕ No

Property	Type	Default	Encrypted[1]	Required[2]
Time in milliseconds to keep this connection. When a connection is returned to the pool, the pool will check to see if the now - time-when-connected > maxAge has been reached, and if so, it closes the connection rather than returning it to the pool. The default value is 0, which implies that connections will be left open and no age check will be done upon returning the connection to the pool.				
`minEvictableIdleTimeMillis`	`int`	`60000`		✕ No
The minimum amount of time an object may sit idle in the pool before it is eligible for eviction. The default value is 60000 (60 seconds).				
`timeBetweenEvictionRunsMillis`	`int`	`5000`		✕ No
The number of milliseconds to sleep between runs of the idle connection validation/cleaner thread. This value should not be set under 1 second. It dictates how often we check for idle, abandoned connections, and how often we validate idle connections. The default value is 5000 (5 seconds).				
`testOnReturn`	`boolean`	`false`		✕ No
The indication of whether objects will be validated before being returned to the pool. NOTE - for a true value to have any effect, the validationQuery parameter must be set to a non-null string. The default value is false.				
`driverClassName`	`String`	`null`		✕ No
The fully qualified Java class name of the JDBC driver to be used. The driver has to be accessible from the same classloader as tomcat-jdbc.jar.				
`name`	`String`	`Tomcat Connection Pool[4-351632824]`		✕ No
Returns the name of the connection pool. By default a JVM unique random name is assigned.				
`useStatementFacade`	`boolean`	`true`		✕ No
Returns true if this connection pool is configured to wrap statements in order to enable equals() and hashCode() methods to be called on the closed statements if any statement proxy is set.				
`useLock`	`boolean`	`false`		✕ No
Return true if a lock should be used when operations are performed on the connection object. Should be set to false unless you plan to have a background thread of your own doing idle and abandon checking such as JMX clients. If the pool sweeper is enabled, then the lock will automatically be used regardless of this setting.				
`initSQL`	`String`	`null`		✕ No
A custom query to be run when a connection is first created. The default value is null.				

Property	Type	Default	Encrypted[1]	Required[2]
`maxActive`	`int`	`100`		✕ No
The maximum number of active connections that can be allocated from this pool at the same time. The default value is 100.				
`username`	`String`	`null`		✕ No
The connection username to be passed to our JDBC driver to establish a connection. Note that method DataSource.getConnection(username,password) by default will not use credentials passed into the method, but will use the ones configured here. See alternateUsernameAllowed property for more details.				
`validationQueryTimeout`	`int`	`-1`		✕ No
The timeout in seconds before a connection validation queries fail. This works by calling java.test_sample.Statement.setQueryTimeout(seconds) on the statement that executes the validationQuery. The pool itself doesnt timeout the query, it is still up to the JDBC driver to enforce query timeouts. A value less than or equal to zero will disable this feature. The default value is -1.				
`password`	`String`	`null`	🔒 Yes	✓ Yes
Oracle EBS login password to authenticate the user.				
`pageSize`	`int`	`50`		✕ No
Page size of search.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

PeopleSoft connector

The PeopleSoft connector lets you manage and synchronize accounts between Oracle PeopleSoft and IDM managed user objects. A PeopleSoft administrator account is required for this connector to work.

Before you start

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

Host

The host address of the PeopleSoft instance.

Port

The port for the PeopleSoft instance.

User ID

The username to log into the PeopleSoft instance.

Password

The password to log into the PeopleSoft instance.

Domain Connect Password

The domain connection password for the PeopleSoft WebLogic application server.

Install the PeopleSoft connector

> **Tip**
>
> To check for an Advanced Identity Cloud application for this connector, refer to:
>
> - Application management
> - App catalog

You can download any connector from Backstage, but some are included in the default deployment for Advanced Identity Cloud, IDM, or RCS. When using an included connector, you can skip installing it and move directly to configuration.

Connector included in default deployment

Connector	IDM	RCS
PeopleSoft	✕ No	✓ Yes

Download the connector .jar file from Backstage.

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

```
mv ~/Downloads/peoplesoft-connector-1.5.20.26.jar /path/to/openidm/connectors/
```

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

Download the connector dependencies.

- `psjoa.jar` is a file unique to each installation of PeopleSoft. It is compiled and provided by your PeopleSoft administrator. If it is not provided to you, refer to Generate `psjoa.jar`

- `psft.jar` is generated by the following commands:

```
set CLASSPATH=%JAVA_HOME%\lib\tools.jar;%CLASSPATH%
jar cvf psft.jar .\PeopleSoft\Generated\CompIntfc*.class
```

Generate `psjoa.jar`

> **Note**
>
> This procedure is only required if your PeopleSoft Administrator did not provide `psjoa.jar`.

1. Start **PeopleSoft Application Designer**, and open any **Component Interface** definition.
2. Select **Build** > **PeopleSoft APIs**.

   The **Build PeopleSoft API Bindings** window displays.
3. Under the **Java Classes** group box, select **Build**, and specify a target directory.
4. To build the selected bindings, click **OK**.

   The app builds the selected bindings in the specified directory. If the operation is successful, a **Done** message appears in the **PeopleSoft Application Designer Build** window.
5. Compile the generated APIs:

   Windows

   ```
   cd %PS_HOME%\class\PeopleSoft\Generated\CompIntfc
   javac -classpath %PS_HOME%\class\psjoa.jar *.java
   cd c:\pt8\class\PeopleSoft\ Generated\ PeopleSoft
   javac -classpath %PS_HOME%\class\psjoa.jar *.java
   ```

   Linux

   ```
   cd $PS_HOME/class/PeopleSoft/Generated/CompIntfc
   javac classpath $PS_HOME/class/psjoa.jar *.java
   cd $PS_HOME/class/PeopleSoft/Generated/PeopleSoft
   javac classpath $PS_HOME/class/psjoa.jar *.java
   ```
6. Copy `psjoa.jar` and generated jar into `/path/to/openicf/lib`.

Configure the PeopleSoft connector

Create a connector configuration using the IDM admin UI:

1. From the navigation bar, click **Configure** > **Connectors**.
2. On the **Connectors** page, click **New Connector**.
3. On the **New Connector** page, type a **Connector Name**.

4. From the **Connector Type** drop-down list, select **PeopleSoft Connector - 1.5.20.26**.
5. Complete the **Base Connector Details**.

> **Tip**
>
> For a list of all configuration properties, refer to PeopleSoft Connector Configuration

6. Click **Save**.

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

Refer to this procedure to create a connector configuration over REST.

Test the PeopleSoft connector

Test that the configuration is correct by running the following command:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system/peoplesoft?_action=test"
{
  "name": "peoplesoft",
  "enabled": true,
  "config": "config/provisioner.openicf/peoplesoft",
  "connectorRef": {
    "bundleVersion": "[1.5.0.0,1.6.0.0)",
    "bundleName": "org.forgerock.openicf.connectors.peoplesoft-connector",
    "connectorName": "org.forgerock.openicf.connectors.peoplesoft.PeopleSoftConnector"
  },
  "displayName": "PeopleSoft Connector",
  "objectTypes": [
    "__ACCOUNT__",
    "__ALL__"
  ],
  "ok": true
}
```

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

PeopleSoft remote connector

If you want to run this connector outside of PingOne Advanced Identity Cloud or IDM, you can configure the PeopleSoft connector as a remote connector. Java Connectors installed remotely on a Java Connector Server function identically to those bundled locally within PingOne Advanced Identity Cloud or installed locally on IDM.

You can download the PeopleSoft connector from here.

Refer to Remote connectors for configuring the PeopleSoft remote connector.

[[config-connection-pooling-peoplesoft]

Configure connection pooling

The PeopleSoft connector uses ICF pooling to manage connections. Learn more about the different pooling mechanisms in Connectors by pooling mechanism.

Use the PeopleSoft connector

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

<table>
<tr><th>Attribute</th><th>Description</th></tr>
<tr><td><code>__NAME__</code></td><td>The name of the user. Required.</td></tr>
<tr><td><code>UserID</code></td><td>ID of the user. Required.</td></tr>
<tr><td><code>IDTypes</code></td><td>The type of ID and ID value for the user. Required. This is an object, containing <code>IDType</code> and <code>AttributeValue</code> as sub-attributes. For example,
<pre>
"IDTypes": [{
  "IDType": "EMP",
  "AttributeValue": "0001"
}]
</pre>
<i>Supported ID types</i>
<table>
<tr><th>ID Type</th><th>Name</th></tr>
<tr><td><code>BID</code></td><td>Bidder</td></tr>
<tr><td><code>CNT</code></td><td>Customer Contact</td></tr>
<tr><td><code>CST</code></td><td>Customer</td></tr>
<tr><td><code>EJA</code></td><td>External Job Applicant</td></tr>
<tr><td><code>EMP</code></td><td>Employee</td></tr>
<tr><td><code>NON</code></td><td>None</td></tr>
<tr><td><code>ORG</code></td><td>Organization ID</td></tr>
<tr><td><code>PER</code></td><td>Person (CRM)</td></tr>
<tr><td><code>VND</code></td><td>Vendor</td></tr>
<tr><td><code>PTN</code></td><td>Partner</td></tr>
</table>
</td></tr>
</table>

Attribute	Description
`UserIDAlias`	Alias ID of the user. This should be a fully qualified email address.
`UserDescription`	A description of the user.
`PrimaryPermissionList`	Primary permission list for the user. Displays which permissions the user is granted in the primary permission list.
`RowSecurityPermissionList`	Row security permission list for the user. Displays which permissions the user is granted in the row security permission list.
`ProcessProfilePermissionList`	Process profile permission list for the user. Displays which permissions the user is granted in the process profile permission list.
`NavigatorHomePermissionList`	Navigator home permission list for the user. Displays which permissions the user is granted in the navigator home permission list.
`SymbolicID`	The symbolic ID of the user.

<table>
<tr>
<td><code>LanguageCode</code></td>
<td>The user’s language preference.

<i>Supported languages</i>
<table>
<tr><th>Language</th><th>Code</th></tr>
<tr><td>Arabic</td><td><code>ARA</code></td></tr>
<tr><td>Afrikaans</td><td><code>AFR</code></td></tr>
<tr><td>Bulgarian</td><td><code>BUL</code></td></tr>
<tr><td>Simplified Chinese</td><td><code>ZHS</code></td></tr>
<tr><td>Traditional Chinese</td><td><code>ZHT</code></td></tr>
<tr><td>Croatian</td><td><code>CRO</code></td></tr>
<tr><td>Czech</td><td><code>CZE</code></td></tr>
<tr><td>Danish</td><td><code>DAN</code></td></tr>
<tr><td>Dutch</td><td><code>DUT</code></td></tr>
<tr><td>English</td><td><code>ENG</code></td></tr>
<tr><td>UK English</td><td><code>UKE</code></td></tr>
<tr><td>French</td><td><code>FRA</code></td></tr>
<tr><td>Canadian French</td><td><code>CFR</code></td></tr>
<tr><td>German</td><td><code>GER</code></td></tr>
<tr><td>Greek</td><td><code>GRK</code></td></tr>
<tr><td>Finnish</td><td><code>FIN</code></td></tr>
<tr><td>Hebrew</td><td><code>HEB</code></td></tr>
<tr><td>Hungarian</td><td><code>HUN</code></td></tr>
<tr><td>Italian</td><td><code>ITA</code></td></tr>
<tr><td>Japanese</td><td><code>JPN</code></td></tr>
<tr><td>Korean</td><td><code>KOR</code></td></tr>
<tr><td>Bahasa Malay</td><td><code>MAY</code></td></tr>
</table>
</td>
</tr>
</table>

<table>
<tr><th>Attribute</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Language</th><th>Code</th></tr>
<tr><td>Norwegian</td><td>`NOR`</td></tr>
<tr><td>Polish</td><td>`POL`</td></tr>
<tr><td>Portuguese</td><td>`POR`</td></tr>
<tr><td>Romanian</td><td>`ROM`</td></tr>
<tr><td>Russian</td><td>`RUS`</td></tr>
<tr><td>Serbian</td><td>`SER`</td></tr>
<tr><td>Slovak</td><td>`SLK`</td></tr>
<tr><td>Slovenian</td><td>`SLV`</td></tr>
<tr><td>Spanish</td><td>`ESP`</td></tr>
<tr><td>Swedish</td><td>`SVE`</td></tr>
<tr><td>Thai</td><td>`THA`</td></tr>
<tr><td>Turkish</td><td>`TUR`</td></tr>
<tr><td>Vietnamese</td><td>`VIE`</td></tr>
</table>

Note
The list of supported languages can vary depending on your Oracle PeopleSoft version.
</td></tr>
<tr><td>`MultiLanguageEnabled`</td><td>Enable support for multiple languages for the user.</td></tr>
<tr><td>`AccountLocked`</td><td>Whether the user account is locked.</td></tr>
<tr><td>`CurrencyCode`</td><td>Three letter code for the user’s preferred currency.</td></tr>
<tr><td>`FailedLogins`</td><td>The number of failed logins for the user.</td></tr>
<tr><td>`ExpertEntry`</td><td>Whether the user is marked as an expert.</td></tr>
<tr><td>`Opertype`</td><td>The type of operation.</td></tr>
<tr><td>`AllowSwitchUser`</td><td>Determines whether the user has access to user switching.</td></tr>
<tr><td>`WorklistEntriesCount`</td><td>Number of worklist entries associated with the user.</td></tr>
</table>

<table>
<tr><th>Attribute</th><th>Description</th></tr>
<tr><td><code>WorklistUser</code></td><td>Whether there is a worklist associated with the user. Must be either <code>Y</code> (Yes) or <code>N</code> (No).</td></tr>
<tr><td><code>EmailUser</code></td><td>Email preference of the user. Must be either <code>Y</code> (Yes) or <code>N</code> (No).</td></tr>
<tr><td><code>AlternateUserID</code></td><td>Fallback user to route to if the user is unavailable. This must be filled out if you specify <code>EffectiveDateFrom</code> or <code>EffectiveDateTo</code>.</td></tr>
<tr><td><code>EffectiveDateFrom</code></td><td>Effective start date that a user will be unavailable. Must be in <code>MM/DD/YYYY</code> format.</td></tr>
<tr><td><code>EffectiveDateTo</code></td><td>Effective end date, marking when a user will become available again. Must be in <code>MM/DD/YYYY</code> format.</td></tr>
<tr><td><code>EmailAddresses</code></td><td>List of email addresses associated with the user. This is an object, with <code>EmailType</code>, <code>EmailAddress</code>, and <code>PrimaryEmail</code> as sub-attributes. For example:
<pre><code>"EmailAddresses": [{
  "EmailType":"BUS",
  "EmailAddress":"test@gmail.com",
  "PrimaryEmail":"Y"
}]</code></pre>
<i>Supported email types</i>
<table>
<tr><th>Email Code</th><th>Email Type</th></tr>
<tr><td><code>BB</code></td><td>Blackberry</td></tr>
<tr><td><code>HOME</code></td><td>Home</td></tr>
<tr><td><code>WORK</code></td><td>Work</td></tr>
<tr><td><code>BUS</code></td><td>Business</td></tr>
<tr><td><code>OTH</code></td><td>Other</td></tr>
<tr><td><code>EMPTY</code></td><td>Empty field</td></tr>
</table>
</td></tr>
<tr><td><code>Roles</code></td><td>List of roles the user has. Users inherit permissions based on the roles the user has. This is an object, with <code>RoleName</code> and <code>Dynamic</code> as sub-attributes. For example:
<pre><code>"Roles": [{
  "RoleName": "PeopleSoft User"
}]</code></pre>
</td></tr>
<tr><td><code>__PASSWORD__</code></td><td>The password for the user.</td></tr>
</table>

Attribute	Description
`ConfirmPassword`	Used to confirm the password of the user. This needs to match the user’s password.
`Encrypted`	Status showing whether or not the user profile is encrypted.

Operations on PeopleSoft accounts

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

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

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request POST \
--data '{
  "__NAME__": "Barbara Jensen",
  "UserID": "BJENSEN",
  "IDTypes": [{
    "IDType": "EMP",
    "AttributeValue": "0001"
  }]
}' \
"http://localhost:8080/openidm/system/peoplesoft/__ACCOUNT__?_action=create"
{
  "_id": "BJENSEN",
  "ExpertEntry": 0,
  "LanguageCode": "ENG",
  "EmailUser": "Y",
  "__ENABLE__": 0,
  "__NAME__": "Barbara Jensen",
  "IDTypes": [
    {
      "IDType": "EMP",
      "AttributeValue": "0001"
    }
  ],
  "Encrypted": 1,
  "UserID": "BJENSEN",
  "Opertype": 0,
  "MultiLanguageEnabled": 0,
  "WorklistUser": "Y",
  "WorklistEntriesCount": 0,
  "AllowSwitchUser": 0,
  "FailedLogins": 0
}
```

> **Note**
>
> When you create a new user, you must specify *at least* `__NAME__`, `UserID`, and `IDTypes`. refer to the list of available attributes for more information.

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

- `UserIDAlias`
- `UserDescription`
- `PrimaryPermissionList`
- `RowSecurityPermissionList`
- `ProcessProfilePermissionList`
- `NavigatorHomePermissionList`
- `SymbolicID`
- `LanguageCode`
- `MultiLanguageEnabled`
- `AccountLocked`
- `CurrencyCode`
- `FailedLogins`
- `ExpertEntry`
- `Opertype`
- `AllowSwitchUser`
- `WorklistUser`
- `EmailUser`
- `AlternateUserID`
- `EffectiveDateFrom`
- `EffectiveDateTo`
- `EmailAddresses`
- `Roles`
- `IDTypes`
- `Password`
- `ConfirmPassword`
- `Encrypted`

For example, to update the `EmailAddresses` for a user:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "If-Match:*" \
--request PUT \
--data '{
  "__NAME__": "Barbara Jensen",
  "UserID": "BJENSEN",
  "IDTypes": [{
    "IDType": "EMP",
    "AttributeValue": "0001"
  }],
  "EmailAddresses": [{
    "EmailType":"BUS",
    "EmailAddress":"test@example.com",
    "PrimaryEmail":"Y"
  }]
}' \
"http://localhost:8080/openidm/system/peoplesoft/__ACCOUNT__/BJENSEN"
{
  "_id": "BJENSEN",
  "ExpertEntry": 0,
  "LanguageCode": "ENG",
  "EmailUser": "Y",
  "__ENABLE__": 0,
  "__NAME__": "Barbara Jensen",
  "IDTypes": [
    {
      "IDType": "EMP",
      "AttributeValue": "0001"
    }
  ],
  "Encrypted": 1,
  "EmailAddresses": [
    {
      "EmailType": "BUS",
      "EmailAddress": "test@example.com",
      "PrimaryEmail": "Y"
    }
  ],
  "UserID": "BJENSEN",
  "Opertype": 0,
  "MultiLanguageEnabled": 0,
  "WorklistUser": "Y",
  "WorklistEntriesCount": 0,
  "AllowSwitchUser": 0,
  "FailedLogins": 0
}
```

The following example queries all PeopleSoft users:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/peoplesoft/__ACCOUNT__?_queryId=query-all-ids"
{
  "result": [
    {
      "_id": "AZIGLAR"
    },
    {
      "_id": "BCHALMERS"
    },
    {
      "_id": "BDAVIS"
    },
    {
      "_id": "BFRANCISCO"
    },
    {
      "_id": "BGONZALES"
    },
    {
      "_id": "BJENSEN"
    },
    {
      "_id": "BLOCHERTY"
    },
    [ ... ]
    {
      "_id": "SUNDERWOOD"
    },
    {
      "_id": "SVANDERSTEEN"
    },
    {
      "_id": "SWALTERS"
    },
    {
      "_id": "TCORY"
    },
    {
      "_id": "TELLIS"
    }
  ],
  "resultCount": 300,
  "pagedResultsCookie": null,
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

The following command queries a specific user by their ID:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/peoplesoft/__ACCOUNT__/BJENSEN"
{
  "_id": "BJENSEN",
  "ExpertEntry": 0,
  "LanguageCode": "ENG",
  "EmailUser": "Y",
  "__ENABLE__": 0,
  "__NAME__": "Barbara Jensen",
  "IDTypes": [
    {
      "IDType": "EMP",
      "AttributeValue": "0001"
    }
  ],
  "Encrypted": 1,
  "EmailAddresses": [
    {
      "EmailType": "BUS",
      "EmailAddress": "test@example.com",
      "PrimaryEmail": "Y"
    }
  ],
  "UserID": "BJENSEN",
  "Opertype": 0,
  "MultiLanguageEnabled": 0,
  "WorklistUser": "Y",
  "WorklistEntriesCount": 0,
  "AllowSwitchUser": 0,
  "FailedLogins": 0
}
```

```
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__": "Passw0rd",
  "__CURRENT_PASSWORD__": "Passw0rd"
}' \
"http://localhost:8080/openidm/system/peoplesoft/__ACCOUNT__/BJENSEN"
{
  "_id": "BJENSEN",
  "ExpertEntry": 0,
  "LanguageCode": "ENG",
  "EmailUser": "Y",
  "__ENABLE__": 0,
  "__NAME__": "Barbara Jensen",
  "IDTypes": [
    {
      "IDType": "EMP",
      "AttributeValue": "0001"
    }
  ],
  "Encrypted": 1,
  "EmailAddresses": [
    {
      "EmailType": "BUS",
      "EmailAddress": "test@example.com",
      "PrimaryEmail": "Y"
    }
  ],
  "UserID": "BJENSEN",
  "Opertype": 0,
  "MultiLanguageEnabled": 0,
  "WorklistUser": "Y",
  "WorklistEntriesCount": 0,
  "AllowSwitchUser": 0,
  "FailedLogins": 0
}
```

Note

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

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "If-Match:*" \
--request PUT \
--data '{
  "__NAME__": "Barbara Jensen",
  "__ENABLE__": 1
}' \
"http://localhost:8080/openidm/system/peoplesoft/__ACCOUNT__/BJENSEN"
{
  "_id": "BJENSEN",
  "ExpertEntry": 0,
  "LanguageCode": "ENG",
  "EmailUser": "N",
  "__ENABLE__": 1,
  "__NAME__": "Barbara Jensen",
  "IDTypes": [
    {
      "IDType": "EMP",
      "AttributeValue": "0001"
    }
  ],
  "Encrypted": 1,
  "EmailAddresses": [
    {
      "EmailType": "BUS",
      "EmailAddress": "test@example.com",
      "PrimaryEmail": "Y"
    }
  ],
  "UserID": "BJENSEN",
  "Opertype": 0,
  "MultiLanguageEnabled": 0,
  "WorklistUser": "N",
  "WorklistEntriesCount": 0,
  "AllowSwitchUser": 0,
  "FailedLogins": 0
}
```

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "If-Match:*" \
--request PUT \
--data '{
  "__NAME__": "Barbara Jensen",
  "__ENABLE__": 0
}' \
"http://localhost:8080/openidm/system/peoplesoft/__ACCOUNT__/BJENSEN"
{
  "_id": "BJENSEN",
  "ExpertEntry": 0,
  "LanguageCode": "ENG",
  "EmailUser": "N",
  "__ENABLE__": 0,
  "__NAME__": "Barbara Jensen",
  "IDTypes": [
    {
      "IDType": "EMP",
      "AttributeValue": "0001"
    }
  ],
  "Encrypted": 1,
  "EmailAddresses": [
    {
      "EmailType": "BUS",
      "EmailAddress": "test@example.com",
      "PrimaryEmail": "Y"
    }
  ],
  "UserID": "BJENSEN",
  "Opertype": 0,
  "MultiLanguageEnabled": 0,
  "WorklistUser": "N",
  "WorklistEntriesCount": 0,
  "AllowSwitchUser": 0,
  "FailedLogins": 0
}
```

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

The following example deletes an PeopleSoft account:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request DELETE \
"http://localhost:8080/openidm/system/peoplesoft/__ACCOUNT__/BJENSEN"
{
  "_id": "BJENSEN",
  "ExpertEntry": 0,
  "LanguageCode": "ENG",
  "EmailUser": "N",
  "__ENABLE__": 0,
  "__NAME__": "Barbara Jensen",
  "IDTypes": [
    {
      "IDType": "EMP",
      "AttributeValue": "0001"
    }
  ],
  "Encrypted": 1,
  "EmailAddresses": [
    {
      "EmailType": "BUS",
      "EmailAddress": "test@example.com",
      "PrimaryEmail": "Y"
    }
  ],
  "UserID": "BJENSEN",
  "Opertype": 0,
  "MultiLanguageEnabled": 0,
  "WorklistUser": "N",
  "WorklistEntriesCount": 0,
  "AllowSwitchUser": 0,
  "FailedLogins": 0
}
```

Operations on other objects

The following operations are supported for other objects, including Employee, Permission, External Job Applicant, and Role:

The following example queries all employees' details:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/peoplesoft/__EMPLOYEE__?_queryId=query-all-ids"
{
  "result": [
    {"_id":"0001"},
    {"_id":"21"},
    {"_id":"22"},
    {"_id":"25"},
    {"_id":"AA0001"}
  ],
  "resultCount":5,
  "pagedResultsCookie":null,
  "totalPagedResultsPolicy":"NONE",
  "totalPagedResults":-1,
  "remainingPagedResults":-1
}
```

The following example queries a single employee’s details:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/peoplesoft/__EMPLOYEE__/BJENSEN?_prettyprint=true"
{
  "_id" : "BJENSEN",
  "__NAME__" : "BJENSEN",
  "LAST_NAME" : "Jensen",
  "PROP_DERIVED_EMP" : "N",
  "COLL_NAME_TYPE_VW" : [ {
    "KEYPROP_NAME_TYPE" : "PRI",
    "FIRST_NAME" : "Barbara",
    "LAST_NAME" : "Jensen"
  }, {
    "KEYPROP_NAME_TYPE" : "PRF",
    "FIRST_NAME" : "Barbara",
    "LAST_NAME" : "Jensen"
  } ],
  "PROP_NAME" : "Barbara Jensen",
  "__UID__" : "BJENSEN",
  "COLL_ADDRESS_TYPE_VW" : [ {
    "KEYPROP_ADDRESS_TYPE" : "",
    "KEYPROP_EFFDT" : "11/14/2022",
    "PROP_EFF_STATUS" : "A",
    "PROP_COUNTRY" : "",
    "PROP_ADDRESS1" : "",
    "PROP_ADDRESS2" : "",
    "PROP_ADDRESS3" : "",
    "PROP_ADDRESS4" : "",
    "PROP_CITY" : "",
    "PROP_NUM1" : "",
    "PROP_NUM2" : "",
    "PROP_HOUSE_TYPE" : "",
    "PROP_ADDR_FIELD1" : "",
    "PROP_ADDR_FIELD2" : "",
    "PROP_ADDR_FIELD3" : "",
    "PROP_COUNTY" : "",
    "PROP_STATE" : "",
    "PROP_POSTAL" : "",
    "PROP_GEO_CODE" : "",
    "PROP_IN_CITY_LIMIT" : "",
    "PROP_ADDRESS1_AC" : "",
    "PROP_ADDRESS2_AC" : "",
    "PROP_ADDRESS3_AC" : "",
    "PROP_CITY_AC" : "",
    "PROP_REG_REGION" : ""
  } ],
  "COLL_PERSONAL_PHONE" : [ {
    "KEYPROP_PHONE_TYPE" : "",
    "PROP_COUNTRY_CODE" : "",
    "PROP_PHONE" : "",
    "PROP_EXTENSION" : "",
    "PROP_PREF_PHONE_FLAG" : "N"
  } ],
  "COLL_EMAIL_ADDRESSES" : [ {
    "KEYPROP_E_ADDR_TYPE" : "",
    "PROP_EMAIL_ADDR" : "",
    "PROP_PREF_EMAIL_FLAG" : "N"
  } ]
}
```

The following example queries all employee permissions:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/peoplesoft/__PERMISSION__?_queryId=query-all-ids"
{
  "result": [
    {"_id":"11"},
    {"_id":"CI_PERSONAL_DATA"},
    {"_id":"CRM8000"},
    {"_id":"CRRW1000"},
    {"_id":"EOCB_CLIENT_USER"}
  ],
  "resultCount":5,
  "pagedResultsCookie":null,
  "totalPagedResultsPolicy":"NONE",
  "totalPagedResults":-1,
  "remainingPagedResults":-1
}
```

The following example queries a single permission's details:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/peoplesoft/__PERMISSION__/HCCPCSALL?_prettyprint=true"
{
  "_id" : "HCCPCSALL",
  "__UID__" : "HCCPCSALL",
  "__NAME__" : "Campus - Hidden Objects",
  "KEYPROP_CLASSID" : "HCCPCSALL"
}
```

The following example queries all external job applicants:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/peoplesoft/__EXTERNAL_JOB_APPLICANT__?_queryId=query-all-ids"
{
  "result": [
    {"_id":"500000"},
    {"_id":"500001"},
    {"_id":"500002"},
    {"_id":"500003"},
    {"_id":"500004"}
  ],
  "resultCount":5,
  "pagedResultsCookie":null,
  "totalPagedResultsPolicy":"NONE",
  "totalPagedResults":-1,
  "remainingPagedResults":-1
}
```

The following example queries a single external job applicant's details:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/peoplesoft/__EXTERNAL_JOB_APPLICANT__/500258?_prettyprint=true"
{
  "_id" : "500258",
  "__NAME__" : "500258",
  "__UID__" : "500258"
}
```

The following example queries all employee roles:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/peoplesoft/__ROLE__?_queryId=query-all-ids"
{
  "result": [
    {"_id":"ACM Administrator"},
    {"_id":"ADS Designer"},
    {"_id":"AG Composer Administrator"},
    {"_id":"AG Composer User"},
    {"_id":"AM Administrator"}
  ],
  "resultCount":5,
  "pagedResultsCookie":null,
  "totalPagedResultsPolicy":"NONE",
  "totalPagedResults":-1,
  "remainingPagedResults":-1
}
```

The following example queries a single role's details:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/peoplesoft/__ROLE__/HR%20Matrix%20Manager?_prettyprint=true"
{
  "_id" : "HR Matrix Manager",
  "PSROLEGRANTORVW" : [ {
    "GRANTROLENAME" : "",
    "ROLENAME" : "HR Matrix Manager"
  } ],
  "PC_FUNCTION_NAME" : "HR Matrix Manager",
  "__UID__" : "HR Matrix Manager",
  "DESCRLONG" : "HR Matrix Manager",
  "ALLOWNOTIFY" : "HR Matrix Manager",
  "ROLE_PCODE_RULE_ON" : "HR Matrix Manager",
  "__NAME__" : "HR Matrix Manager",
  "PSROLECANGRANT" : [ {
    "GRANTROLENAME" : "",
    "ROLENAME" : "HR Matrix Manager"
  } ],
  "DESCR" : "HR Matrix Manager",
  "QRYNAME" : "HR Matrix Manager",
  "ROLE_QUERY_RULE_ON" : "HR Matrix Manager",
  "RECNAME" : "HR Matrix Manager",
  "FIELDNAME" : "HR Matrix Manager",
  "PSROLEMEMBER" : [ {
    "ROLEUSER" : "",
    "ROLENAME" : "HR Matrix Manager"
  } ],
  "PSROLEDYNMEMBER" : [ {
    "ROLEUSER" : "",
    "ROLENAME" : "HR Matrix Manager"
  } ],
  "ALLOWLOOKUP" : "HR Matrix Manager",
  "PSROLECLASS" : [ {
    "CLASSID" : "HCCPHR9435"
  } ],
  "LDAP_RULE_ON" : "HR Matrix Manager"
}
```

OpenICF Interfaces Implemented by the PeopleSoft Connector

The PeopleSoft Connector implements the following OpenICF interfaces. For additional details, see ICF interfaces:

Create

Creates an object and its `uid`.

Delete

Deletes an object, referenced by its `uid`.

Schema

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

Script on Connector

Enables an application to run a script in the context of the connector.

Any script that runs on the connector has the following characteristics:

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

Search

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

Test

Tests the connector configuration.

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

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

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

Update

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

PeopleSoft Connector Configuration

The PeopleSoft Connector has the following configurable properties:

Configuration properties

Property	Type	Default	Encrypted[1]	Required[2]
`host`	`String`	`null`		✓ Yes
Host name or IP address to connect to PeopleSoft server.				

Property	Type	Default	Encrypted[1]	Required[2]
`port`	`int`	`0`		✓ Yes
Port to connect to PeopleSoft server.				
`userId`	`String`	`null`		✓ Yes
The userid used to login to PeopleSoft server.				
`password`	`GuardedString`	`null`	🔒 Yes	✓ Yes
The password used to login to PeopleSoft server.				
`domainConnectPassword`	`GuardedString`	`null`	🔒 Yes	✓ Yes
The password for PeopleSoft app server domain.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

PingOne connector

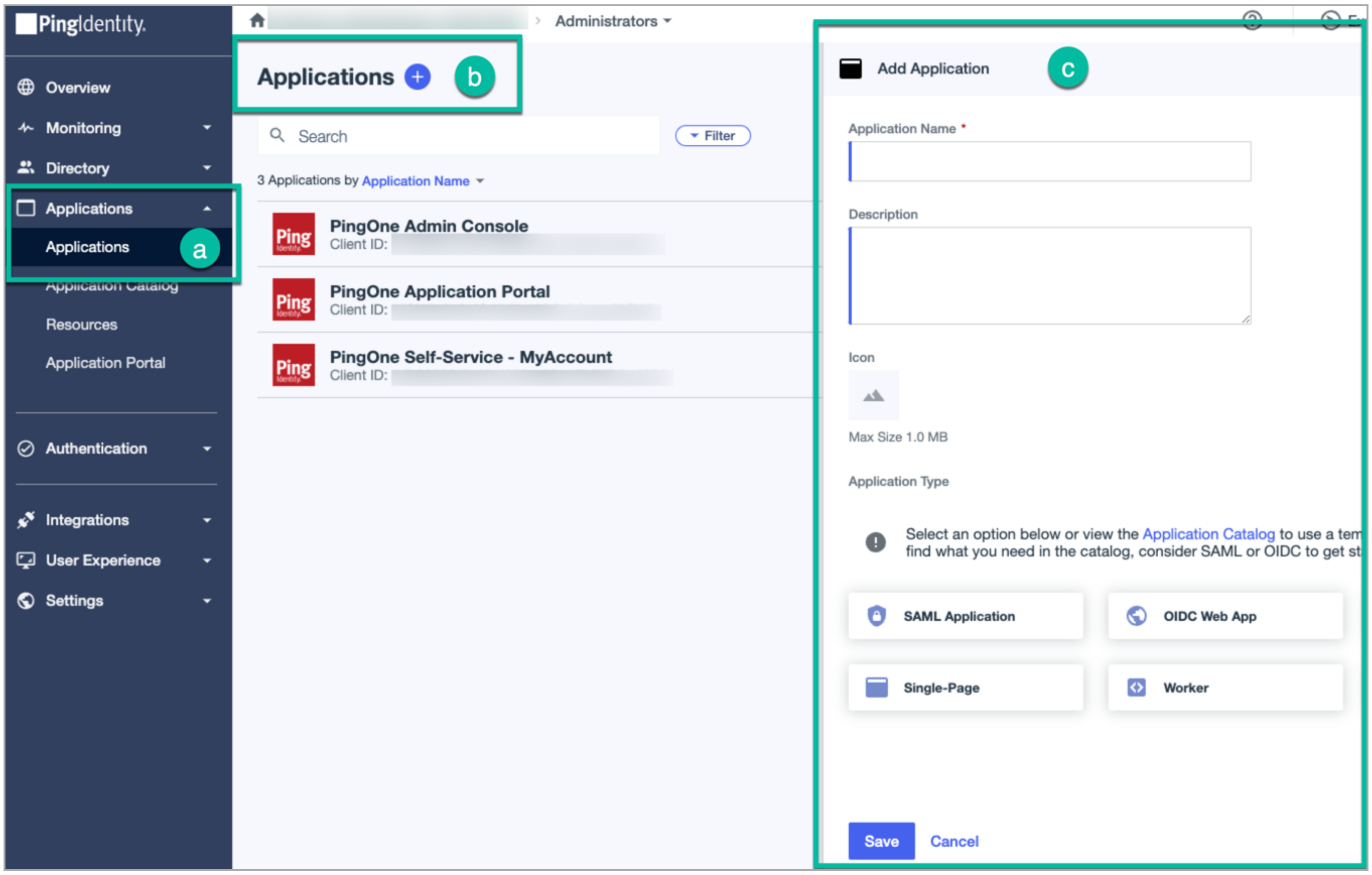
> **Tip**
>
> This is a SaaS common connector.

The PingOne connector lets you manage and synchronize data between PingOne and IDM or Advanced Identity Cloud. A PingOne administrator account is required for this connector to work.

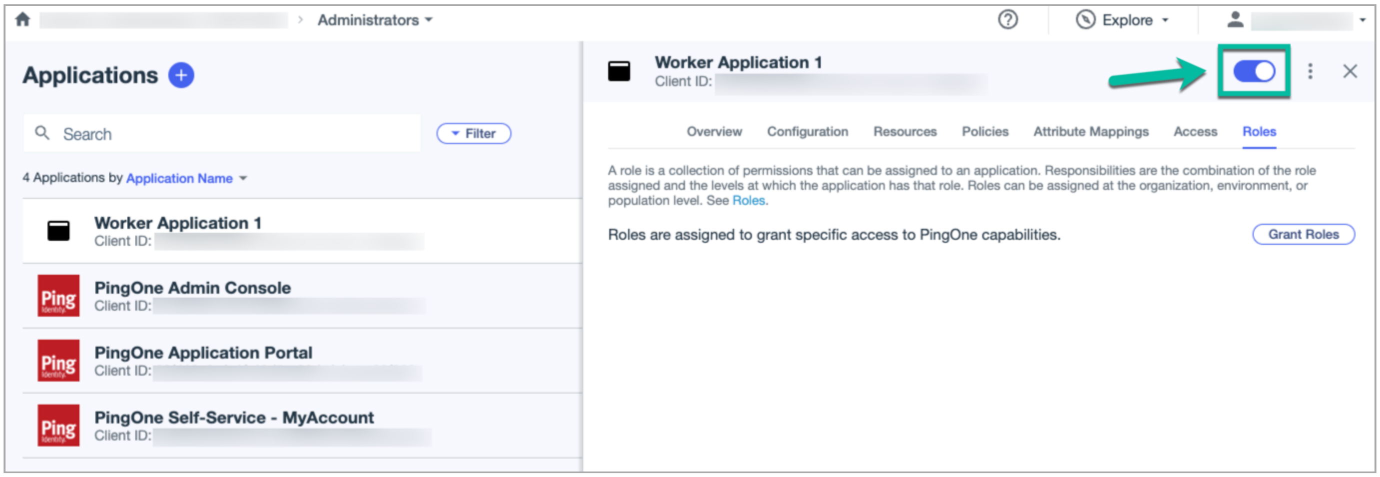
PingOne requirements

Before you can configure the connector, you must register an application in PingOne. You need a PingOne environment to complete this procedure:

1. In your PingOne environment, create a new application:
   1. From the menu, expand the **Applications** node, and click **Applications**.
   2. On the **Applications** page, click the add button.
   3. In the **Add Application** window, enter the necessary details, select the **Worker** application type, and click **Save**.



2. In the **Application Name** window, enable the application.

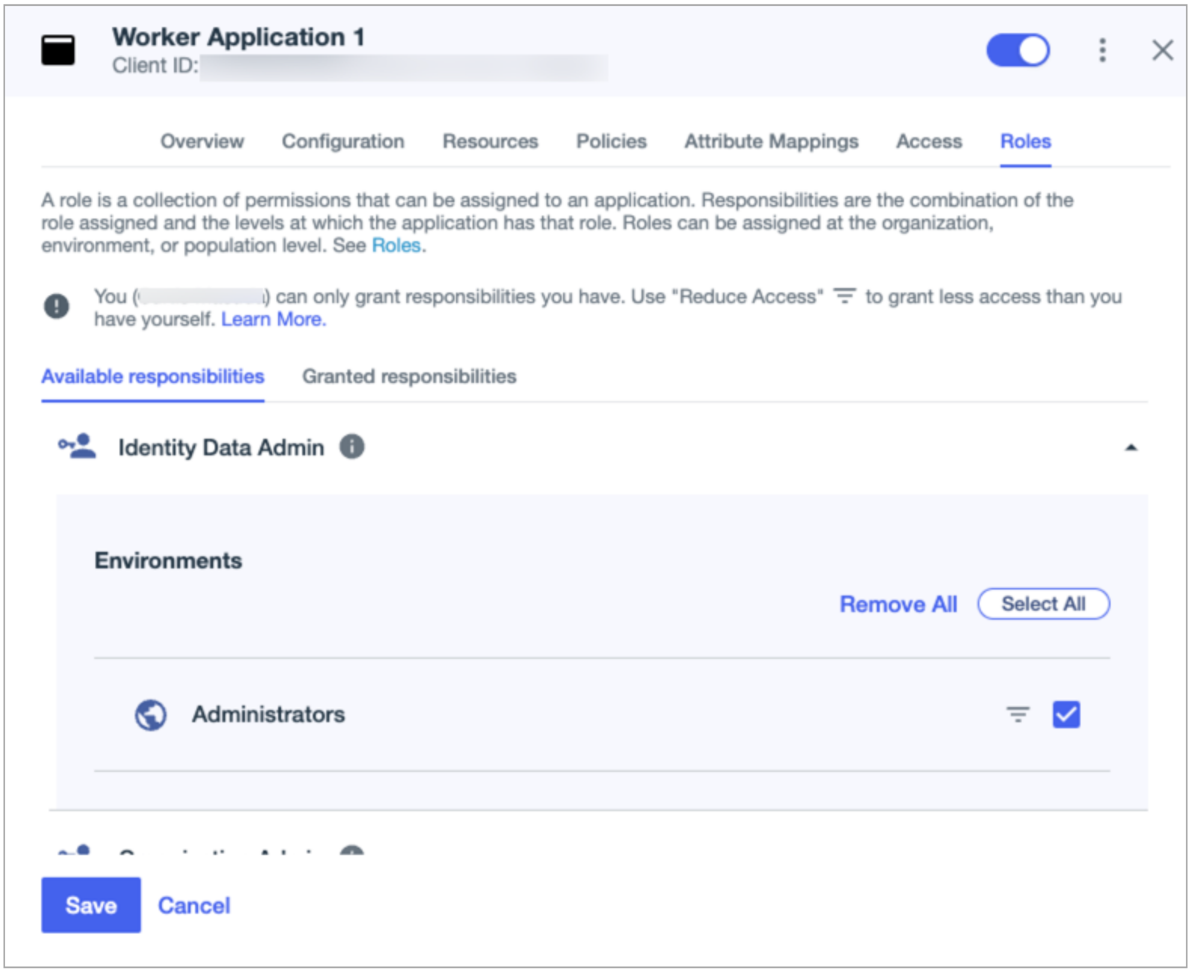


A role is a collection of permissions that can be assigned to an application. Responsibilities are the combination of the role assigned and the levels at which the application has that role. Roles can be assigned at the organization, environment, or population level. See Roles.

Roles are assigned to grant specific access to PingOne capabilities.

3. On the **Roles** tab, click **Grant Roles**.

4. On the **Available Responsibilities** tab, expand the **Identity Data Admin** node, select the applicable environment, and click **Save**.



Worker Application 1
Client ID: [redacted]

A role is a collection of permissions that can be assigned to an application. Responsibilities are the combination of the role assigned and the levels at which the application has that role. Roles can be assigned at the organization, environment, or population level. See Roles.

You ([redacted]) can only grant responsibilities you have. Use "Reduce Access" to grant less access than you have yourself. Learn More.

5. Click the **Configuration** tab, and make note of the following:

   - **URLs > Token Endpoint**
   - **General > Client ID**
   - **General > Client Secret**
   - **General > Environment ID**

   Use these values when you Configure the PingOne connector.

Install the PingOne connector

> **Tip**
>
> To check for an Advanced Identity Cloud application for this connector, refer to:
>
> - Application management
> - App catalog

You can download any connector from Backstage, but some are included in the default deployment for Advanced Identity Cloud, IDM, or RCS. When using an included connector, you can skip installing it and move directly to configuration.

Connector included in default deployment

Connector	IDM	RCS
PingOne	✕ No	✕ No

Download the connector .jar file from Backstage.

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

```
mv ~/Downloads/pingone-connector-1.5.20.30.jar /path/to/openidm/connectors/
```

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

Configure the PingOne connector

Create a connector configuration using the IDM admin UI:

1. From the navigation bar, click **Configure > Connectors**.
2. On the **Connectors** page, click **New Connector**.
3. On the **New Connector** page, type a **Connector Name**.
4. From the **Connector Type** drop-down list, select **PingOne Connector - 1.5.20.30**.
5. Complete the **Base Connector Details**.

> **Tip**
>
> For a list of all configuration properties, refer to PingOne Connector Configuration

6. Click **Save**.

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

Refer to this procedure to create a connector configuration over REST.

Example PingOne configuration

This excerpt shows a sample PingOne connector configuration:

```
{
    "connectorRef" : {
        "displayName" : "org.forgerock.openicf.connectors.pingone.PingOneConnector",
        "bundleVersion" : "[1.5.0.0,1.6.0.0)",
        "systemType" : "provisioner.openicf",
        "bundleName" : "org.forgerock.openicf.connectors.pingone-connector",
        "connectorName" : "org.forgerock.openicf.connectors.pingone.PingOneConnector",
        "connectorHostRef" : ""
    },
    "poolConfigOption" : {
        "maxObjects" : 10,
        "maxIdle" : 10,
        "maxWait" : 150000,
        "minEvictableIdleTimeMillis" : 120000,
        "minIdle" : 1
    },
    "resultsHandlerConfig" : {
        "enableNormalizingResultsHandler" : false,
        "enableFilteredResultsHandler" : false,
        "enableCaseInsensitiveFilter" : false,
        "enableAttributesToGetSearchResultsHandler" : true
    },
    "operationTimeout" : {
        "CREATE" : -1,
        "UPDATE" : -1,
        "DELETE" : -1,
        "TEST" : -1,
        "SCRIPT_ON_CONNECTOR" : -1,
        "SCRIPT_ON_RESOURCE" : -1,
        "GET" : -1,
        "RESOLVEUSERNAME" : -1,
        "AUTHENTICATE" : -1,
        "SEARCH" : -1,
        "VALIDATE" : -1,
        "SYNC" : -1,
        "SCHEMA" : -1
    },
    "configurationProperties" : {
        "environmentId" : null,
        "serviceUri" : null,
        "login" : null,
        "password" : null,
        "authenticationMethod" : "OAUTH",
        "tokenEndpoint" : null,
        "clientId" : null,
        "clientSecret" : null,
        "authToken" : null,
        "acceptSelfSignedCertificates" : false,
        "disableHostNameVerifier" : false,
        "disableHttpCompression" : false,
        "clientCertAlias" : null,
        "clientCertPassword" : null,
        "maximumConnections" : 10,
        "httpProxyHost" : null,
        "httpProxyPort" : null,
        "httpProxyUsername" : null,
        "httpProxyPassword" : null,
        "connectionTimeout" : 30,
        "refreshToken" : null,
        "grantType" : null,
        "scope" : null,
        "authorizationTokenPrefix" : "Bearer",
        "useBasicAuthForOauthTokenNeg" : true
    },
    "enabled" : true,
    "objectTypes" : {
        "__GROUP__" : {
            "$schema" : "http://json-schema.org/draft-03/schema",
            "id" : "__GROUP__",
            "type" : "object",
            "nativeType" : "__GROUP__",
            ...
            }
        },
        "__POPULATION__" : {
            "$schema" : "http://json-schema.org/draft-03/schema",
            "id" : "__POPULATION__",
            "type" : "object",
            "nativeType" : "__POPULATION__",
            ...
            }
        },
        "__ACCOUNT__" : {
            "$schema" : "http://json-schema.org/draft-03/schema",
            "id" : "__ACCOUNT__",
            "type" : "object",
            "nativeType" : "__ACCOUNT__",
            ...
            }
        },
        "__AVAILABLE_ROLE__" : {
            "$schema" : "http://json-schema.org/draft-03/schema",
            "id" : "__AVAILABLE_ROLE__",
            "type" : "object",
            "nativeType" : "__AVAILABLE_ROLE__",
            ...
            }
        },
        "__ROLE__" : {
            "$schema" : "http://json-schema.org/draft-03/schema",
            "id" : "__ROLE__",
            "type" : "object",
            "nativeType" : "__ROLE__",
            ...
            }
        }
    }
}
```

Test the PingOne connector

Test that the configuration is correct by running the following command:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system/pingone?_action=test"
{
  "name": "pingone",
  "enabled": true,
  "config": "config/provisioner.openicf/pingone",
  "connectorRef": {
    "bundleVersion": "[1.5.0.0,1.6.0.0)",
    "bundleName": "org.forgerock.openicf.connectors.pingone-connector",
    "connectorName": "org.forgerock.openicf.connectors.pingone.PingOneConnector"
  },
  "displayName": "PingOne Connector",
  "objectTypes": [
    "__GROUP__",
    "__POPULATION__",
    "__ACCOUNT__",
    "__ALL__",
    "__AVAILABLE_ROLE__",
    "__ROLE__"
  ],
  "ok": true
}
```

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

PingOne remote connector

If you want to run this connector outside of PingOne Advanced Identity Cloud or IDM, you can configure the PingOne connector as a remote connector. Java Connectors installed remotely on a Java Connector Server function identically to those bundled locally within PingOne Advanced Identity Cloud or installed locally on IDM.

You can download the PingOne connector from here.

Refer to Remote connectors for configuring the PingOne remote connector.

Configure connection pooling

The PingOne connector supports HTTP pooling, which can substantially improve the performance of the connector. Learn more about the basic connection pooling configuration and different pooling mechanisms described in Connection pooling configuration.

Implementation specifics

Any sync mapping that uses PingOne as the source or target must include the following in the mapping configuration:

Source

```
{
  "reconSourceQueryPaging" : true
}
```

Target

```
{
  "reconTargetQueryPaging" : true
}
```

The default recon query page size for IDM and Advanced Identity Cloud is too large for PingOne resources. You must set the source and target to `1000` or less. For example:

```
{
  "reconSourceQueryPageSize" : 1000,
  "reconTargetQueryPageSize" : 1000,
  ...
}
```

For more information about sync mappings, refer to:

- Advanced Identity Cloud: Synchronization reference
- IDM: Synchronization reference

Use the PingOne connector

You can use the PingOne connector to perform various actions on the following PingOne resources:

Connector resource	PingOne resource type
`__ACCOUNT__`	Users
`__GROUP__`	Groups
`__POPULATION__`	Populations
`__ROLE__`	Roles
`__AVAILABLE_ROLE__`	Available roles

Users

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request POST \
--data '{
  "username": "A_20240223183659857",
  "nickname": "A_2024022",
  "email": "A_20240223183659857@example.com",
  "primaryPhone": "123 123 123"
}' \
"http://localhost:8080/openidm/system/pingone/__ACCOUNT__?_action=create"
{
  "_id": "dc3c02ab-5a22-4537-b5cb-7f3a0c164ab1",
  "createdAt": "2024-02-24T02:37:01.889Z",
  "email": "A_20240223183659857@example.com",
  "__NAME__": "A_20240223183659857",
  "identityProvider": {
    "type": "PING_ONE"
  },
  "primaryPhone": "123 123 123",
  "enabled": true,
  "lifecycle": {
    "status": "ACCOUNT_OK"
  },
  "verifyStatus": "NOT_INITIATED",
  "username": "A_20240223183659857",
  "updatedAt": "2024-02-24T02:37:01.889Z",
  "mfaEnabled": true,
  "nickname": "A_2024022",
  "account": {
    "canAuthenticate": true,
    "status": "OK"
  },
  "population": {
    "id": "1a0348b5-c6f5-41b0-a71a-194ed76f703f"
  }
}
```

Note

When you create a new user, you must specify *at least* the `username` attribute.

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request POST \
--data '{
  "username": "20240223183659857_gm",
  "nickname": "2024022318",
  "email": "20240223183659857_gm@example.com",
  "primaryPhone": "123 123 123",
  "__groupMemberships__": [
    "336c9e3e-5480-4f32-8706-ccd944531286",
    "aa4c33a9-6726-4526-b70f-7cd63fce5c28"
  ],
  "__roleAssignments__": [
    "9da69b4c-8101-4db8-8cef-66a9a167b02e:ENVIRONMENT:0bd9c966-7664-4ac1-b059-0ff9293908e2",
    "9da69b4c-8101-4db8-8cef-66a9a167b02e:ENVIRONMENT:ce00e15f-f845-4df1-abf3-fdc4ff4e176c"
  ]
}' \
"http://localhost:8080/openidm/system/pingone/__ACCOUNT__?_action=create"
{
  "_id": "e9c4a878-d26d-4a2e-9da7-3d83b5d6a8b9",
  "__groupMemberships__": [
    "336c9e3e-5480-4f32-8706-ccd944531286",
    "aa4c33a9-6726-4526-b70f-7cd63fce5c28"
  ],
  "__roleAssignments__": [
    "9da69b4c-8101-4db8-8cef-66a9a167b02e:ENVIRONMENT:0bd9c966-7664-4ac1-b059-0ff9293908e2",
    "9da69b4c-8101-4db8-8cef-66a9a167b02e:ENVIRONMENT:ce00e15f-f845-4df1-abf3-fdc4ff4e176c"
  ],
  "createdAt": "2024-02-24T02:37:09.365Z",
  "email": "20240223183659857_gm@example.com",
  "__NAME__": "20240223183659857_gm",
  "identityProvider": {
    "type": "PING_ONE"
  },
  "primaryPhone": "123 123 123",
  "enabled": true,
  "lifecycle": {
    "status": "ACCOUNT_OK"
  },
  "verifyStatus": "NOT_INITIATED",
  "username": "20240223183659857_gm",
  "updatedAt": "2024-02-24T02:37:09.967Z",
  "mfaEnabled": true,
  "nickname": "2024022318",
  "account": {
    "canAuthenticate": true,
    "status": "OK"
  },
  "population": {
    "id": "1a0348b5-c6f5-41b0-a71a-194ed76f703f"
  },
  "groupMemberships": [
      {
        "name": "20240223183659857_A",
        "environment": {
          "id": "d182d341-2739-4082-975f-bc94396a9651"
        },
        "id": "336c9e3e-5480-4f32-8706-ccd944531286",
        "isExternal": false,
        "type": "DIRECT"
      },
      {
        "name": "20240223183659857_B",
        "environment": {
          "id": "d182d341-2739-4082-975f-bc94396a9651"
        },
        "id": "aa4c33a9-6726-4526-b70f-7cd63fce5c28",
        "isExternal": false,
        "type": "DIRECT"
      }
    ],
    "roleAssignments": [
      {
        "environment": {
          "id": "d182d341-2739-4082-975f-bc94396a9651"
        },
        "role": {
          "id": "0bd9c966-7664-4ac1-b059-0ff9293908e2"
        },
        "scope": {
          "id": "9da69b4c-8101-4db8-8cef-66a9a167b02e",
          "type": "ENVIRONMENT"
        },
        "readOnly": false,
        "id": "2c48fdd6-fec3-47f6-80d7-c993bafc8802",
        "type": "DIRECT",
        "user": {
          "id": "fd5e0c14-ba9b-479b-85f8-80ac9222cc97"
        }
      },
      {
        "environment": {
          "id": "d182d341-2739-4082-975f-bc94396a9651"
        },
        "role": {
          "id": "ce00e15f-f845-4df1-abf3-fdc4ff4e176c"
        },
        "scope": {
          "id": "9da69b4c-8101-4db8-8cef-66a9a167b02e",
          "type": "ENVIRONMENT"
        },
        "readOnly": false,
        "id": "c5eda229-fdc7-4e4a-b887-833b53b3ec87",
        "type": "DIRECT",
        "user": {
```

```
        "id": "fd5e0c14-ba9b-479b-85f8-80ac9222cc97"
      }
    }
  ]
}
```

To modify an existing user, include all user attributes in the PUT request. The following user attributes can’t be modified:

- `population`
- `mfaEnabled`
- `verifyStatus`
- `identityProvider`
- `linkedAccounts`

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "If-Match:*" \
--request PUT \
--data '{
  "username": "A_20240223183659857",
  "nickname": "A_2024022",
  "email": "update@email.com",
  "primaryPhone": "123 123 123"
}' \
"http://localhost:8080/openidm/system/pingone/__ACCOUNT__/4da38ae8-50dc-4555-9c76-6d2f9a984007"
{
  "_id": "4da38ae8-50dc-4555-9c76-6d2f9a984007",
  "createdAt": "2024-02-24T02:37:04.668Z",
  "email": "update@email.com",
  "__NAME__": "A_20240223183659857",
  "identityProvider": {
    "type": "PING_ONE"
  },
  "primaryPhone": "123 123 123",
  "enabled": true,
  "lifecycle": {
    "status": "ACCOUNT_OK"
  },
  "verifyStatus": "NOT_INITIATED",
  "username": "C_20240223183659857",
  "updatedAt": "2024-02-24T02:37:05.866Z",
  "mfaEnabled": true,
  "nickname": "A_2024022",
  "account": {
    "canAuthenticate": true,
    "status": "OK"
  },
  "population": {
    "id": "1a0348b5-c6f5-41b0-a71a-194ed76f703f"
  }
}
```

The following example queries all PingOne users:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/pingone/__ACCOUNT__?_queryId=query-all-ids"
{
  "result": [
    {
      "_id": "ffc979b8-6279-4cb9-b327-98db745ff60b"
    },
    {
      "_id": "e0e0258a-87de-43c5-8e25-91915959317d"
    },
    ...
  ],
  "resultCount": 5,
  "pagedResultsCookie": null,
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

The following example reads all PingOne users:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/pingone/__ACCOUNT__?_queryFilter=true"
{
  "result": [
    {
      "_id": "ffc979b8-6279-4cb9-b327-98db745ff60b",
      "createdAt": "2024-02-24T02:35:33.304Z",
      "email": "OCPSGYHOARDL@example.com",
      "__NAME__": "OCPSGYHOARDL",
      "identityProvider": {
        "type": "PING_ONE"
      },
      "primaryPhone": "123 123 123",
      "enabled": true,
      "lifecycle": {
        "status": "ACCOUNT_OK"
      },
      "verifyStatus": "NOT_INITIATED",
      "username": "OCPSGYHOARDL",
      "updatedAt": "2024-02-24T02:35:33.304Z",
      "mfaEnabled": true,
      "nickname": "OCPSGY",
      "account": {
        "canAuthenticate": true,
        "status": "OK"
      },
      "population": {
        "id": "1a0348b5-c6f5-41b0-a71a-194ed76f703f"
      }
    },
    {
      "_id": "e0e0258a-87de-43c5-8e25-91915959317d",
      "createdAt": "2024-02-24T02:35:34.219Z",
      "email": "FQHCIODRAYIK@example.com",
      "__NAME__": "FQHCIODRAYIK",
      "identityProvider": {
        "type": "PING_ONE"
      },
      "primaryPhone": "123 123 123",
      "enabled": true,
      "lifecycle": {
        "status": "ACCOUNT_OK"
      },
      "verifyStatus": "NOT_INITIATED",
      "username": "FQHCIODRAYIK",
      "updatedAt": "2024-02-24T02:35:34.219Z",
      "mfaEnabled": true,
      "nickname": "FQHCIO",
      "account": {
        "canAuthenticate": true,
        "status": "OK"
      },
      "population": {
        "id": "1a0348b5-c6f5-41b0-a71a-194ed76f703f"
      }
    },
    ...
  ],
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

The following example reads PingOne user `dfdb1068-174c-4c05-8a73-8c91e3f20f83`:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/pingone/__ACCOUNT__/dfdb1068-174c-4c05-8a73-8c91e3f20f83"
{
  "_id": "dfdb1068-174c-4c05-8a73-8c91e3f20f83",
  "createdAt": "2024-02-24T02:37:06.471Z",
  "email": "D_20240223183659857@example.com",
  "__NAME__": "D_20240223183659857",
  "identityProvider": {
    "type": "PING_ONE"
  },
  "primaryPhone": "123 123 123",
  "enabled": true,
  "lifecycle": {
    "status": "ACCOUNT_OK"
  },
  "verifyStatus": "NOT_INITIATED",
  "username": "D_20240223183659857",
  "updatedAt": "2024-02-24T02:37:06.471Z",
  "mfaEnabled": true,
  "nickname": "D_2024022",
  "account": {
    "canAuthenticate": true,
    "status": "OK"
  },
  "population": {
    "id": "1a0348b5-c6f5-41b0-a71a-194ed76f703f"
  }
}
```

The following example deletes a PingOne account:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "If-Match: *" \
--request DELETE \
"http://localhost:8080/openidm/system/pingone/__ACCOUNT__/dfdb1068-174c-4c05-8a73-8c91e3f20f83"
{
  "_id": "dfdb1068-174c-4c05-8a73-8c91e3f20f83",
  "createdAt": "2024-02-24T02:37:06.471Z",
  "email": "D_20240223183659857@example.com",
  "__NAME__": "D_20240223183659857",
  "identityProvider": {
    "type": "PING_ONE"
  },
  "primaryPhone": "123 123 123",
  "enabled": true,
  "lifecycle": {
    "status": "ACCOUNT_OK"
  },
  "verifyStatus": "NOT_INITIATED",
  "username": "D_20240223183659857",
  "updatedAt": "2024-02-24T02:37:06.471Z",
  "mfaEnabled": true,
  "nickname": "D_2024022",
  "account": {
    "canAuthenticate": true,
    "status": "OK"
  },
  "population": {
    "id": "1a0348b5-c6f5-41b0-a71a-194ed76f703f"
  }
}
```

Groups

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request POST \
--data '{
  "name": "A_20240223183647490",
  "description": "standard description"
}' \
'http://localhost:8080/openidm/system/pingone/__GROUP__?_action=create'
{
  "_id": "bb6534d0-39a0-4e8e-b4b1-efe595063540",
  "directMemberCounts": {
    "users": 0,
    "groups": 0
  },
  "__NAME__": "A_20240223183647490",
  "name": "A_20240223183647490",
  "description": "standard description"
}
```

Note

When you create a new group, you must specify *at least* the `name` attribute.

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/pingone/__GROUP__?_queryId=query-all-ids"
{
  "result": [
    {
      "_id": "c42b514f-56c9-4c63-a91c-679d006ac248"
    },
    {
      "_id": "bb6534d0-39a0-4e8e-b4b1-efe595063540"
    },
    {
      "_id": "1ce940ce-dde6-4066-8869-9435072c4089"
    },
    ...
  ],
  "resultCount": 14,
  "pagedResultsCookie": null,
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/pingone/__GROUP__/c42b514f-56c9-4c63-a91c-679d006ac248"
{
  "_id": "c42b514f-56c9-4c63-a91c-679d006ac248",
  "directMemberCounts": {
    "users": 0,
    "groups": 0
  },
  "__NAME__": "PY_GROUP_20240223183647490",
  "name": "PY_GROUP_20240223183647490",
  "description": "standard description"
}
```

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'If-Match: *' \
--header 'Content-Type: application/json' \
--request PUT \
--data '{
  "name": "C_20240223183647490",
  "description": "updated description"
}' \
'http://localhost:8080/openidm/system/pingone/__GROUP__/bb6534d0-39a0-4e8e-b4b1-efe595063540'
{
  "_id": "bb6534d0-39a0-4e8e-b4b1-efe595063540",
  "directMemberCounts": {
    "users": 0,
    "groups": 0
  },
  "__NAME__": "A_20240223183647490",
  "name": "A_20240223183647490",
  "description": "updated description"
}
```

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "If-Match: *" \
--header 'Accept-API-Version: resource=1.0' \
--request DELETE \
'http://localhost:8080/openidm/system/pingone/__GROUP__/1ce940ce-dde6-4066-8869-9435072c4089'
{
  "_id": "1ce940ce-dde6-4066-8869-9435072c4089",
  "directMemberCounts": {
    "users": 0,
    "groups": 0
  },
  "__NAME__": "D_20240223183647490",
  "name": "D_20240223183647490",
  "description": "standard description"
}
```

Populations

A PingOne population defines a particular set of users.

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request POST \
--data '{
  "name": "C_20240223183654888",
  "description": "standard description"
}' \
'http://localhost:8080/openidm/system/pingone/__POPULATION__?_action=create'
{
  "_id": "e40936b4-36ec-4250-96dd-1c37ee8773e5",
  "createdAt": "2024-02-24T02:36:56.710Z",
  "environment": {
    "id": "d182d341-2739-4082-975f-bc94396a9651"
  },
  "description": "standard description",
  "default": "false",
  "updatedAt": "2024-02-24T02:36:56.710Z",
  "name": "C_20240223183654888",
  "userCount": 0,
  "__NAME__": "C_20240223183654888"
}
```

> **Note**
>
> When you create a new population, you must specify *at least* the `name` attribute.

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/pingone/__POPULATION__/e40936b4-36ec-4250-96dd-1c37ee8773e5"
{
  "_id": "e40936b4-36ec-4250-96dd-1c37ee8773e5",
  "createdAt": "2024-02-24T02:36:56.710Z",
  "environment": {
    "id": "d182d341-2739-4082-975f-bc94396a9651"
  },
  "description": "standard description",
  "default": "false",
  "updatedAt": "2024-02-24T02:36:56.710Z",
  "name": "C_20240223183654888",
  "userCount": 0,
  "__NAME__": "C_20240223183654888"
}
```

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'If-Match: *' \
--header 'Content-Type: application/json' \
--request PUT \
--data '{
  "name": "C_20240223183654888",
  "description": "updated description"
}' \
'http://localhost:8080/openidm/system/pingone/__POPULATION__/e40936b4-36ec-4250-96dd-1c37ee8773e5'
{
  "_id": "e40936b4-36ec-4250-96dd-1c37ee8773e5",
  "createdAt": "2024-02-24T02:36:56.710Z",
  "environment": {
    "id": "d182d341-2739-4082-975f-bc94396a9651"
  },
  "description": "updated description",
  "default": "false",
  "updatedAt": "2024-02-24T02:36:57.125Z",
  "name": "C_20240223183654888",
  "userCount": 0,
  "__NAME__": "C_20240223183654888"
}
```

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'If-Match: *' \
--header 'Content-Type: application/json' \
--request DELETE \
'http://localhost:8080/openidm/system/pingone/__POPULATION__/93c83bc9-a546-4728-bd1f-f75a6436616c'
{
  "_id": "93c83bc9-a546-4728-bd1f-f75a6436616c",
  "createdAt": "2024-02-24T02:36:57.463Z",
  "environment": {
    "id": "d182d341-2739-4082-975f-bc94396a9651"
  },
  "description": "standard description",
  "default": "false",
  "updatedAt": "2024-02-24T02:36:57.463Z",
  "name": "D_20240223183654888",
  "userCount": 0,
  "__NAME__": "D_20240223183654888"
}
```

Roles

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/pingone/__ROLE__?_queryFilter=true"
{
  "result": [
    {
      "_id": "1813bc13-8d13-4e88-a825-d40bfe82777b",
      "applicableTo": [
        "ORGANIZATION"
      ],
      "name": "Organization Admin",
      "description": "Organization Admin",
      "permissions": [
        {
          "id": "advancedservices:read:config",
          "classifier": "config",
          "description": "Retrieve PingOne Advanced Services customer configuration"
        },
        {
          "id": "authz:update:entity",
          "classifier": "entity",
          "description": "Update PingOne Authorize Entity"
        },
        {
          "id": "pingintelligence:create:orchestration",
          "classifier": "orchestration",
          "description": "Creates a Orchestration flow for Ping Intelligence deployment"
        },
        {
          "id": "authz:read:entity",
          "classifier": "entity",
          "description": "Read PingOne Authorize Entity"
        },
        {
          "id": "authz:create:condition",
          "classifier": "condition",
          "description": "Create a PingOne Authorize Condition"
        },
        {
          "id": "authz:test:condition",
          "classifier": "condition",
          "description": "Test a PingOne Authorize Condition"
        },
        {
          "id": "authz:create:service",
          "classifier": "service",
          "description": "Create a PingOne Authorize Service"
        },
        {
          "id": "authz:read:recentdecisions",
          "classifier": "recentdecisions",
          "description": "Read Recent Decisions of a PingOne Authorize Decision Endpoint"
        },
        {
          "id": "orgmgt:read:organization",
          "classifier": "organization",
          "description": "Read organizations"
        },
        {
          "id": "pingenterprise:update:orchestration",
          "classifier": "orchestration",
          "description": "Updates Orchestration flow for Ping Enterprise deployment"
        },
        ...
      ],
      "__NAME__": "Organization Admin"
    },
    ...
  ],
  "resultCount": 13,
  "pagedResultsCookie": null,
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/pingone/__ROLE__/1813bc13-8d13-4e88-a825-d40bfe82777b"
{
  "_id": "1813bc13-8d13-4e88-a825-d40bfe82777b",
  "applicableTo": [
    "ORGANIZATION"
  ],
  "name": "Organization Admin",
  "description": "Organization Admin",
  "permissions": [
    {
      "id": "advancedservices:read:config",
      "classifier": "config",
      "description": "Retrieve PingOne Advanced Services customer configuration"
    },
    {
      "id": "authz:update:entity",
      "classifier": "entity",
      "description": "Update PingOne Authorize Entity"
    },
    {
      "id": "pingintelligence:create:orchestration",
      "classifier": "orchestration",
      "description": "Creates a Orchestration flow for Ping Intelligence deployment"
    },
    {
      "id": "authz:read:entity",
      "classifier": "entity",
      "description": "Read PingOne Authorize Entity"
    },
    {
      "id": "authz:create:condition",
      "classifier": "condition",
      "description": "Create a PingOne Authorize Condition"
    },
    {
      "id": "authz:test:condition",
      "classifier": "condition",
      "description": "Test a PingOne Authorize Condition"
    },
    {
      "id": "authz:create:service",
      "classifier": "service",
      "description": "Create a PingOne Authorize Service"
    },
    {
      "id": "authz:read:recentdecisions",
      "classifier": "recentdecisions",
      "description": "Read Recent Decisions of a PingOne Authorize Decision Endpoint"
    },
    {
      "id": "orgmgt:read:organization",
      "classifier": "organization",
      "description": "Read organizations"
    },
    {
      "id": "pingenterprise:update:orchestration",
      "classifier": "orchestration",
      "description": "Updates Orchestration flow for Ping Enterprise deployment"
    },
    {
      "id": "authz:read:condition",
      "classifier": "condition",
      "description": "Read a PingOne Authorize Condition"
    },
    {
      "id": "authz:test:service",
      "classifier": "service",
      "description": "Test a PingOne Authorize Service"
    },
    {
      "id": "pingenterprise:create:orchestration",
      "classifier": "orchestration",
      "description": "Creates a Orchestration flow for Ping Enterprise deployment"
    },
    {
      "id": "orgmgt:read:environment",
      "classifier": "environment",
      "description": "Read environments"
    },
    {
      "id": "authz:update:service",
      "classifier": "service",
      "description": "Update a PingOne Authorize Service"
    },
    {
      "id": "pingintelligence:update:orchestration",
      "classifier": "orchestration",
      "description": "Updates Orchestration flow for Ping Intelligence deployment"
    },
    {
      "id": "orgmgt:read:deployment",
      "classifier": "deployment",
      "description": "Read deployment resource"
    },
    {
      "id": "authz:update:tag",
      "classifier": "tag",
      "description": "Create and Update PingOne Authorize Policy Version Tag"
    },
    {
      "id": "authz:create:decisionendpoint",
      "classifier": "decisionendpoint",
      "description": "Create PingOne Authorize Decision Endpoint"
    },
    {
      "id": "advancedservices:delete:environment",
      "classifier": "environment",
      "description": "Delete PingOne Advanced Services environment"
    },
    {
      "id": "authz:create:entity",
      "classifier": "entity",
      "description": "Create PingOne Authorize Entity"
    },
    {
      "id": "authz:test:entity",
      "classifier": "entity",
      "description": "Test PingOne Authorize Entity"
    },
    {
      "id": "authz:delete:condition",
      "classifier": "condition",
      "description": "Delete a PingOne Authorize Condition"
    },
    {
      "id": "authz:update:decisionendpoint",
      "classifier": "decisionendpoint",
      "description": "Update PingOne Authorize Decision Endpoint"
    },
    {
      "id": "authz:delete:entity",
      "classifier": "entity",
      "description": "Delete PingOne Authorize Entity"
    },
    {
      "id": "orgmgt:update:environment",
      "classifier": "environment",
      "description": "Update environment"
    },
    {
      "id": "integrations:read:integration",
      "classifier": "integration",
      "description": "Read an integration in integration catalog"
    },
    {
      "id": "authz:read:service",
      "classifier": "service",
      "description": "Read a PingOne Authorize Service"
    },
    {
      "id": "authz:authorize:decisionendpoint",
      "classifier": "decisionendpoint",
      "description": "Request Decision from a PingOne Authorize Decision Endpoint"
    },
    {
      "id": "visualization:create:exploration",
      "classifier": "exploration",
      "description": "Create data exploration"
    },
    {
      "id": "bootstrap:read:bootstrap",
      "classifier": "bootstrap",
      "description": "Read bootstrap"
    },
    {
      "id": "authz:delete:service",
      "classifier": "service",
      "description": "Delete a PingOne Authorize Service"
    },
    {
      "id": "visualization:read:template",
      "classifier": "template",
      "description": "Read data exploration template"
    },
    {
      "id": "pingenterprise:read:orchestration",
      "classifier": "orchestration",
      "description": "Retrieve Orchestration flow for Ping Enterprise deployment"
    },
    {
      "id": "ratelimiting:read:rateLimits",
      "classifier": "rateLimits",
      "description": "Read rate limits"
    },
    {
      "id": "authz:delete:attribute",
      "classifier": "attribute",
      "description": "Delete a PingOne Authorize Attribute"
    },
    {
      "id": "authz:delete:adaptiveTrustPolicy",
      "classifier": "adaptiveTrustPolicy",
      "description": "Delete a PingOne Adaptive Trust Policy"
    },
    {
      "id": "pingenterprise:delete:orchestration",
      "classifier": "orchestration",
      "description": "Deletes Orchestration flow for Ping Enterprise deployment"
    },
    {
      "id": "visualization:read:exploration",
      "classifier": "exploration",
      "description": "Read data exploration"
    },
    {
      "id": "authz:delete:processor",
      "classifier": "processor",
      "description": "Delete a PingOne Authorize Processor"
    },
    {
      "id": "orgmgt:delete:environment",
      "classifier": "environment",
      "description": "Delete environment"
    },
    {
      "id": "orgmgt:promote:environment",
      "classifier": "environment",
      "description": "Promote environment"
    },
    ...
  ],
  "__NAME__": "Organization Admin"
}
```

Available roles

Available roles is a virtual object class, not a native PingOne resource. The `_id` is a concatenation of two other GUIDs and the level of access the role grants in the following format:

```
scopeId:scopeType:roleId
```

Available roles definitions

`scopeId`	The id for the associated scope type
`scopeType`	`ORGANIZATION`, `ENVIRONMENT`, or `POPULATION`
`roleId`	The role id

Example available role:

```
{
  "_id": "9da69b4c-8101-4db8-8cef-66a9a167b02e:ENVIRONMENT:0bd9c966-7664-4ac1-b059-0ff9293908e2",
  "role": {
    "id": "0bd9c966-7664-4ac1-b059-0ff9293908e2" (1)
  },
  "name": "ENVIRONMENT - Administrators: Identity Data Admin",
  "scope": {
    "id": "9da69b4c-8101-4db8-8cef-66a9a167b02e", (2)
    "type": "ENVIRONMENT" (3)
  },
  "environment": {
    "id": "d182d341-2739-4082-975f-bc94396a9651"
  },
  "__NAME__": "ENVIRONMENT - Administrators: Identity Data Admin"
}
```

1 `roleId`

2 `scopeId`

3 `scopeType`

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
'http://localhost:8080/openidm/system/pingone/__AVAILABLE_ROLE__?_queryFilter=true'
{
  "result": [
    {
      "_id": "9da69b4c-8101-4db8-8cef-66a9a167b02e:ENVIRONMENT:0bd9c966-7664-4ac1-b059-0ff9293908e2",
      "role": {
        "id": "0bd9c966-7664-4ac1-b059-0ff9293908e2"
      },
      "name": "ENVIRONMENT - Administrators: Identity Data Admin",
      "scope": {
        "id": "9da69b4c-8101-4db8-8cef-66a9a167b02e",
        "type": "ENVIRONMENT"
      },
      "environment": {
        "id": "d182d341-2739-4082-975f-bc94396a9651"
      },
      "__NAME__": "ENVIRONMENT - Administrators: Identity Data Admin"
    },
    {
      "_id": "9da69b4c-8101-4db8-8cef-66a9a167b02e:ENVIRONMENT:ce00e15f-f845-4df1-abf3-fdc4ff4e176c",
      "role": {
        "id": "ce00e15f-f845-4df1-abf3-fdc4ff4e176c"
      },
      "name": "ENVIRONMENT - Administrators: Identity Data Read Only",
      "scope": {
        "id": "9da69b4c-8101-4db8-8cef-66a9a167b02e",
        "type": "ENVIRONMENT"
      },
      "environment": {
        "id": "d182d341-2739-4082-975f-bc94396a9651"
      },
      "__NAME__": "ENVIRONMENT - Administrators: Identity Data Read Only"
    },
    ...
  ],
  "resultCount": 10,
  "pagedResultsCookie": null,
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
'http://localhost:8080/openidm/system/pingone/__AVAILABLE_ROLE__/
9da69b4c-8101-4db8-8cef-66a9a167b02e:ENVIRONMENT:0bd9c966-7664-4ac1-b059-0ff9293908e2'
{
  "_id": "9da69b4c-8101-4db8-8cef-66a9a167b02e:ENVIRONMENT:0bd9c966-7664-4ac1-b059-0ff9293908e2",
  "role": {
    "id": "0bd9c966-7664-4ac1-b059-0ff9293908e2"
  },
  "name": "ENVIRONMENT - Administrators: Identity Data Admin",
  "scope": {
    "id": "9da69b4c-8101-4db8-8cef-66a9a167b02e",
    "type": "ENVIRONMENT"
  },
  "environment": {
    "id": "d182d341-2739-4082-975f-bc94396a9651"
  },
  "__NAME__": "ENVIRONMENT - Administrators: Identity Data Admin"
}
```

Supported search filters

The PingOne connector supports filtered searches against PingOne resources. However, certain limitations imposed by the PingOne APIs prevent the filtering of resource types based on arbitrary attributes and values. For a complete list, refer to Limiting and Filtering data in the PingOne documentation.

OpenICF Interfaces Implemented by the PingOne Connector

The PingOne Connector implements the following OpenICF interfaces. For additional details, see ICF 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.

PingOne Connector Configuration

The PingOne Connector has the following configurable properties:

Configuration properties

Property	Type	Default	Encrypted[1]	Required[2]
`environmentId`	`String`	`null`		✓ Yes
The environment identifier.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

Basic Configuration Properties

Property	Type	Default	Encrypted[1]	Required[2]
`serviceUri`	`String`	`null`		✓ Yes
The service endpoint URI.				
`login`	`String`	`null`		✓ Yes
The service login name.				
`password`	`GuardedString`	`null`	🔒 Yes	✗ No
The service user password.				
`authenticationMethod`	`String`	`OAUTH`		✓ Yes
Defines which method is to be used to authenticate on the remote server. Options are BASIC (username/password), OAUTH (Client id/secret) or TOKEN (static token).				
`tokenEndpoint`	`String`	`null`		✗ No
When using OAUTH as authentication method, this property defines the endpoint where a new access token should be queried for (https://myserver.com/oauth2/token).				
`clientId`	`String`	`null`		✓ Yes
The client identifier for OAuth2.				
`clientSecret`	`GuardedString`	`null`	🔒 Yes	✗ No
Secure client secret for OAuth2.				
`authToken`	`GuardedString`	`null`	🔒 Yes	✗ No
Static authentication token.				
`acceptSelfSignedCertificates`	`boolean`	`false`		✓ Yes
To be used for debug/test purposes. To be avoided in production.				
`disableHostNameVerifier`	`boolean`	`false`		✓ Yes
To be used for debug/test purposes. To be avoided in production.				
`disableHttpCompression`	`boolean`	`false`		✓ Yes
Content compression is enabled by default. Set this property to true to disable it.				

Property	Type	Default	Encrypted[1]	Required[2]
`clientCertAlias`	`String`	`null`		✓ Yes
If TLS Mutual Auth is needed, set this to the certificate alias from the keystore.				
`clientCertPassword`	`GuardedString`	`null`	🔒 Yes	✓ Yes
If TLS Mutual Auth is needed and the client certificate (private key) password is different from the keystore password, set this to the client private key password.				
`maximumConnections`	`Integer`	`10`		✓ Yes
Defines the max size of the HTTP connection pool used.				
`httpProxyHost`	`String`	`null`		✓ Yes
Defines the Hostname if an HTTP proxy is used between the connector and the service.				
`httpProxyPort`	`Integer`	`null`		✓ Yes
Defines the Port if an HTTP proxy is used between the connector and the service.				
`httpProxyUsername`	`String`	`null`		✓ Yes
Defines Proxy Username if an HTTP proxy is used between the connector and the service.				
`httpProxyPassword`	`GuardedString`	`null`	🔒 Yes	✓ Yes
Defines Proxy Password if an HTTP proxy is used between the connector and the service.				
`connectionTimeout`	`int`	`30`		✕ No
Defines a timeout for the underlying HTTP connection in seconds.				
`refreshToken`	`GuardedString`	`null`		✕ No
Used by the refresh_token grant type.				
`grantType`	`String`	`null`		✕ No
The OAuth2 grant type to use (client_credentials, refresh_token, or jwt_bearer).				
`scope`	`String`	`null`		✕ No
The OAuth2 scope to use.				
`authorizationTokenPrefix`	`String`	`Bearer`		✕ No
The prefix to be used in the Authorization HTTP header for Token authentication.				

Property	Type	Default	Encrypted[1]	Required[2]
`useBasicAuthForOauthTokenNeg`	`boolean`	`true`		✓ Yes
The Authentication method for refresh token (Basic Authentication or Sending the ClientId and Client Secret in the Header).				
`jwtKey`	`String`	`null`		✕ No
The JWT data structure that represents a cryptographic key.				
`jwtExpiration`	`Integer`	`null`		✕ No
Defines the JWT expiration time in seconds.				
`jwtAlgorithm`	`String`	`null`		✕ No
The Algorithm type to sign payload.				
`jwtClaims`	`Map`	`null`		✕ No
JWT Claims to be included in the payload				
`jwtPem`	`String`	`null`		✕ No
The contents of the private key of the PEM file				
`jwtCert`	`String`	`null`		✕ No
The contents of the certificate of the PEM file				
`keyAlgorithm`	`String`	`null`		✕ No
Indicates the type of key (such as RSA, DSA or EC) used to sign from the PEM.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

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 your Microsoft Windows ecosystem requirements. You can use the PowerShell connector toolkit to create connectors that can provision any Microsoft system, including, but not limited to, Active Directory, Microsoft SQL, MS Exchange, SharePoint, Azure, and Office365. Any task you can perform with PowerShell can be executed through connectors based on this toolkit.

The PowerShell connector toolkit is available from the BackStage download site. It is also bundled with the .NET remote connector server.

To use this connector, you must write a PowerShell script for each operation that you want the connector to perform (create, read, update, delete, authenticate, and so on). Sample scripts for core operations are included with the .NET RCS.

Before you start

To implement a scripted PowerShell connector, you must install the following:

- Microsoft .NET Framework 4.6.2 or later. Connectors created with the PowerShell connector toolkit run on the .NET platform and require the installation of a .NET connector server on the Windows system. To install the .NET connector server, follow the instructions in Install .NET RCS.
- PowerShell version 4.0 or above.
- The PowerShell connector toolkit.

Install the PowerShell connector

To run the commands in this procedure, start with the PowerShell command line. Some commands require administrative privileges.

1. Install, configure, and start the .NET connector server on a Windows host. If you are running an Active Directory Domain Controller, install the .NET connector server on the same host where the Windows PowerShell module is installed.
2. Configure IDM to connect to the .NET connector server.
3. The PowerShell connector toolkit comes bundled with the .NET RCS. To confirm it is installed, check the .NET RCS installation directory for a `MsPowerShell.Connector.dll` file.
4. Sample scripts are provided with the .NET remote connector server.

   Reference the full path to the scripts in your connector configuration, for example:

   ```
   "CreateScriptFileName" : "C:/Program Files (x86)/ForgeRock/OpenICF/samples/ADCreate.ps1",
   ...
   ```

Configure the PowerShell connector

1. You cannot configure a PowerShell connector through the UI. Configure the connector over REST, as described in Configure Connectors Over REST.
2. Alternatively, copy the sample connector configuration file (`provisioner.openicf-adpowershell.json`) from the `samples\example-configurations\provisioners` directory to your project's `conf` directory.

   > **Note**
   >
   > Paths in these files must use forward slash characters and not the backslash characters that you would expect in a Windows path.

3. Verify that at least the path to the scripts and the connection and authentication details are correct for your deployment.

PowerShell Connector Configuration Properties

<table>
<tr><th>Property</th><th>Type</th><th>Example</th><th>Encrypted[1]</th><th>Required[2]</th></tr>
<tr><td>operationScriptFileName</td><td>String</td><td>C:/openidm/AD/ADCreate.ps1,</td><td>✗</td><td>✓</td></tr>
<tr><td colspan="5">The full path to the script that implements the corresponding OpenICF operation.</td></tr>
<tr><td>VariablesPrefix</td><td>String</td><td>Connector</td><td>✗</td><td>✗</td></tr>
<tr><td colspan="5">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.</td></tr>
<tr><td>QueryFilterType</td><td>String</td><td>AdPsModule (for Active Directory)</td><td>✗</td><td>✓</td></tr>
<tr><td colspan="5">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</td></tr>
<tr><td>ReloadScriptOnExecution</td><td>Boolean</td><td>true</td><td>✗</td><td>✗</td></tr>
<tr><td colspan="5">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.</td></tr>
<tr><td>UseInterpretersPool</td><td>Boolean</td><td>true</td><td>✗</td><td>✗</td></tr>
<tr><td colspan="5">If true, the connector leverages the PowerShell RunSpace Pool.</td></tr>
<tr><td>MaxInterpretersPoolSize</td><td>Integer</td><td>5</td><td>✗</td><td>✗</td></tr>
<tr><td colspan="5">The maximum size of the interpreter pool.</td></tr>
<tr><td>MinInterpretersPoolSize</td><td>Integer</td><td>1</td><td>✗</td><td>✗</td></tr>
<tr><td colspan="5">The minimum size of the interpreter pool.</td></tr>
<tr><td>PoolCleanupInterval</td><td>Double</td><td>60</td><td>✗</td><td>✗</td></tr>
<tr><td colspan="5">Specifies the interval (in minutes) at which unused interpreter instances are discarded. To avoid cleaning up unused interpreter instances, set this property to 0.</td></tr>
</table>

Property	Type	Example	Encrypted[1]	Required[2]
`SubstituteUidAndNameInQueryFilter`	`Boolean`	`true`	✗	✗
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`	✗	✗
The attribute on the resource that contains the object `__UID__`.				
`NameAttributeName`	`String`	`DistinguishedName`	✗	✗
The attribute on the resource that contains the object `__NAME__`.				
`PsModulesToImport`	`Array`	`["ActiveDirectory","C:/openidm/samples/scripted-powershell-with-ad/tools/ADSISearch.psm1"]`	✗	✗
An array of additional PowerShell modules that the connector must import.				
`Host`	`String`	`ad.example.com`	✗	✓
The host name or IP address of the Active Directory server.				
`Port`	`Integer`	`null`	✗	✓
The port number on which the remote resource listens for connections.				
`Login`	`String`	`""`	✗	✗
The user account in the remote resource that is used for the connection.				
`Password`	`String`	`null`	🔒	✓
The password of the user account that is used for the connection.				
`CustomProperties`	`Array`	`[ ]`	✗	✗

Property	Type	Example	Encrypted[1]	Required[2]

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
```

Test the PowerShell connector

These examples show you how to test the connector is configured correctly and operating as expected.

Check the connector configuration

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system?_action=test"
{
  "name" : "adpowershell",
  "enabled" : true,
  "config" : "config/provisioner.openicf/adpowershell",
  "objectTypes" : [ "__ALL__", "group", "account" ],
  "connectorRef" : {
    "connectorName" : "Org.Forgerock.OpenICF.Connectors.MsPowerShell.MsPowerShellConnector",
    "bundleName" : "MsPowerShell.Connector",
    "bundleVersion" : "[1.4.3.0,1.5.0.0)"
  },
  "displayName" : "PowerShell Connector",
  "ok" : true
}
```

When you run this test, a log entry associated with the .NET connector server should be created in the `logs/` directory of that server.

Search user entries

You can use the connector, with a PowerShell search script, to retrieve information from a target system. The PowerShell search script accepts IDM queries, including `query-all-ids` and `_queryFilter`.

The following command retrieves a list of users in an Active Directory server. You can also use any system-enabled filter, such as those described in Presence Expressions:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
"http://localhost:8080/openidm/system/adpowershell/account?_queryId=query-all-ids"
```

Create users or groups

This command creates a new user in Active Directory:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
--header "content-type: application/json" \
--data '{
  "distinguishedName" : "CN=Robert Smith,CN=Users,DC=EXAMPLE,DC=COM",
  "sAMAccountName" : "robert.smith",
  "sn" : "Smith",
  "cn" : "Robert Smith",
  "userPrincipalName": "Robert.Smith@example.com",
  "enabled" : true,
  "password" : "Passw0rd",
  "telephoneNumber" : "0052-611-091"
}' \
"http://localhost:8080/openidm/system/adpowershell/account?_action=create"
```

Update entries

You can update the following properties with the sample scripts:

- Password
- Principal Name
- License
- Common user attributes

This command changes the password for the user with the specified `_id`:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request PATCH \
--header "content-type: application/json" \
--data '{
  "operation": "replace",
  "Field": "password",
  "value": "Passw1rd"
}' \
"http://localhost:8080/openidm/system/adpowershell/account/1d4c9276-6937-4d9e-9c60-67e8b4207f4e"
```

Delete Users and Groups

This command deletes an Active Directory user entry with the specified `_id`:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request DELETE \
"http://localhost:8080/openidm/system/adpowershell/account/1d4c9276-6937-4d9e-9c60-67e8b4207f4e"
```

Run Scripts Through the Connector

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

`configuration`

A handler to the connector’s configuration object.

`options`

A handler to the Operation Options.

`operation`

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

`log`

A handler to the log.

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

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

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

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

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

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system/adpowershell?
_action=script&scriptId=MyScript&param1=value1&param2=value2"
```

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

```
openidm.action("/system/adpowershell", "script", {}, {"scriptId": "MyScript", "param1": "value1", "param2":
"value2"})
```

Important

Because the action script is stored locally with IDM, it must be transmitted across the network every time it is called. An alternative approach is to write a PowerShell module and to load it using the `PsModulesToImport` option of the PowerShell connector. In this case, the action script is limited to a function call, and you do not need a script file on the IDM side.

The following example uses the `actionSource` property in the provisioner, instead of the `actionFile` property, to call the action. The example calls a custom `Set-Exchange` function from a module loaded on the .Net connector server by the PowerShell connector:

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

Configure connection pooling

The PowerShell connector uses ICF pooling to manage connections. Learn more about the different pooling mechanisms in Connectors by pooling mechanism.

Manage Azure AD Objects With the PowerShell Connector

Ping provides two sets of sample scripts to let you manage objects in Azure AD with the PowerShell connector:

- **Version 1**: These scripts are based on the older Microsoft Online (MSOL) V1 PowerShell module. For information on connecting to your Azure AD with this module, refer to the corresponding Microsoft documentation. Microsoft has expressed its intention to deprecate this module when its functionality has been completely migrated to the newer Azure Active Directory PowerShell for Graph Module. These scripts are supported only up to Windows 2012 R2.

  The Version 1 scripts can manage security groups but not dynamic groups.

- **Version 2**: These scripts are based on the Azure Active Directory PowerShell for Graph Module. For information on connecting to your Azure AD with this module, refer to the corresponding Microsoft documentation. The cmdlets in this module let you perform CRUD operations on an Azure AD instance, and configure the directory and its features.

  The Version 2 scripts can manage user password policies, security and mail groups, dynamic groups, and devices.

Follow these procedures to use the sample Azure AD scripts with the PowerShell connector:

Set Up a Remote Connector Server

1. Install a .NET connector server on your Windows host. These steps assume a Windows hostname of `windows-host.example.com`.

2. On `windows-host.example.com`, install the PowerShell connector.

   When you have installed the PowerShell connector, make sure that the ICF .NET connector server is still running. If it is not running, restart the connector server and check the logs. In some cases, Windows blocks the PowerShell connector .dll files. If the connector server fails to start, right-click on `MsPowerShell.Connector.dll` and select **Properties > Security**

3. If the following text displays:

   ```
   This file came from another computer and might be blocked to help protect this computer.
   ```

   Click **Unblock** to unblock the connector .dll file. Then, restart the connector server.

4. On `windows-host.example.com`, install the Windows Azure AD Module that corresponds to the version of the scripts you are using.

   - For Version 1 scripts, install the MSOnline module.

   - For Version 2 scripts, install the Azure AD module.

5. These instructions assume that you have an existing Azure AD instance.

   Create a specific administrative account in Azure AD, to run the PowerShell connector scripts.

6. In a PowerShell window on `windows-host.example.com`, verify that your Windows host can connect to your Azure AD tenant:
   - For Version 1 scripts, run `Connect-MsolService`.
   - For Version 2 scripts, run `Connect-AzureAD`.

Set Up the PowerShell Azure AD Scripts

When all your systems are installed and running, and you have verified that your Windows host can connect to your Azure AD, set up the sample scripts as follows:

1. On `windows-host.example.com`, create a directory for the PowerShell scripts, for example:

   ```
   PS C:\> mkdir -Path openidm\scripted-powershell-with-azure-ad\scripts
   ```

   Whatever location you choose for the scripts will be referenced in your connector configuration (provisioner file).

2. Download the Azure AD scripts from the ForgeRock stash repository.

   Download either the V1 or V2 scripts, depending on your Azure AD module, and place them in the **scripts** directory you created in the previous step:

   ```
   ls C:\openidm\scripted-powershell-with-azure-ad\scripts
   Directory: C:\openidm\scripted-powershell-with-azure-ad\scripts

   Mode                LastWriteTime     Length Name
   ----                -------------     ------ ----
   -a---          7/21/2020   4:00 AM      10965 AzureADCreate.ps1
   -a---          7/21/2020   4:00 AM       3547 AzureADDelete.ps1
   -a---          7/21/2020   4:00 AM       6952 AzureADSchema.ps1
   -a---          7/21/2020   4:00 AM       8149 AzureADSearch.ps1
   -a---          7/21/2020   4:00 AM       2465 AzureADTest.ps1
   -a---          7/21/2020   4:00 AM      10840 AzureADUpdate.ps1
   ```

   > **Note**
   >
   > By default, Windows does not trust downloaded scripts. To be able to run the scripts, you might need to do the following:
   >
   > - Run the Unblock-File cmdlet. This cmdlet unblocks PowerShell script files that were downloaded from the Internet so that you can run them, regardless of the PowerShell execution policy.
   > - Change the PowerShell execution policy to let you run the scripts.

3. In IDM, configure the connection to the .NET connector server.

4. In IDM, configure the PowerShell connector.

   The ForgeRock stash repository includes a sample provisioner file for both versions of the scripts. Use those files as a starting point. Set at least the following properties:

   `connectorHostRef` The name of the connector server referenced in the previous step.

`*ScriptFileName` Set the path to the script directory that you created on `windows-host.example.com` .

Test the PowerShell Connector With Azure AD

1. Test that the connector has been configured correctly and can reach the Azure AD:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system/azureadpowershell?_action=test"
{
  "name": "azureadpowershell",
  "enabled": true,
  "config": "config/provisioner.openicf/azureadpowershell",
  "objectTypes": [
    "__ALL__",
    "account",
    "group"
  ],
  "connectorRef": {
    "bundleName": "MsPowerShell.Connector",
    "connectorName": "Org.ForgeRock.OpenICF.Connectors.MsPowerShell.MsPowerShellConnector",
    "bundleVersion": "[1.4.3.0,1.5.0.0)"
  },
  "displayName": "PowerShell Connector ",
  "ok": true
}
```

If there is no response from this connector test, check your connector configuration and the connection to the .NET connector server.

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

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

SaaS REST Connector

Tip

This is a SaaS common connector.

Important

You can only use SaaS REST connector version 1.5.20.29 and later with:

- Connector framework 1.5.20.24 or later
- RCS 1.5.20.24 or later

Learn more in Changed functionality.

The SaaS REST Connector allows you to interact with most REST APIs.

Install the SaaS REST connector

> **Tip**
>
> To check for an Advanced Identity Cloud application for this connector, refer to:
>
> - Application management
> - App catalog

You can download any connector from Backstage, but some are included in the default deployment for Advanced Identity Cloud, IDM, or RCS. When using an included connector, you can skip installing it and move directly to configuration.

Connector included in default deployment

Connector	IDM	RCS
SaaS REST	✕ No	✕ No

Download the connector .jar file from Backstage.

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

  ```
  mv ~/Downloads/rest-connector-1.5.20.30.jar /path/to/openidm/connectors/
  ```

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

Configure the SaaS REST connector

You cannot configure the SaaS REST connector through the UI. Configure the connector over REST, as described in Configure Connectors Over REST.

- `serviceUri` : The service endpoint URI.
- `useBasicAuthForOauthTokenNeg` : The Authentication method for refresh token (Basic Authentication or Sending the ClientId and Client Secret in the Header). `true` | `false`
- `tokenEndpoint` : Your URL to get the token.
- `authenticationMethod` : Defines which method is to be used to authenticate on the remote server. Options are BASIC (username/password), OAUTH (Client id/secret) or TOKEN (static token).
- `authorizationTokenPrefix` : The prefix to be used in the Authorization HTTP header for Token authentication.
- `clientId` : The client identifier for OAuth2.
- `clientSecret` : Secure client secret for OAuth2.
- `refreshToken` : Used by the `refresh_token` grant type.
- `authToken` : Static authentication token.

- `grantType` : Your grant type. `client_credentials` | `refresh_token` | `jwt_bearer`
- `login` : Your service login name.
- `password` : Your service user password.
- `scope` : The OAuth2 scope to use.
- `defaultHeaders:` Http headers sent by default.
- `maximumConnections` : Defines the max size of the http connection pool used. Defaults to `10` .
- `connectionTimeout` : Defines a timeout for the underlying http connection in seconds. Defaults to `30` .
- `disableHttpCompression:` Content compression is enabled by default. Set this property to true to disable it.

Certification Details

- `clientCertAlias:` If TLS Mutual Auth is needed, set this to the certificate alias from the keystore.
- `clientCertPassword:` If TLS Mutual Auth is needed and the client certificate (private key) password is different from the keystore password, set this to the client private key password.

Proxy Connection Details

- `httpProxyHost` : Defines the Hostname if an HTTP proxy is used between the connector and the service.
- `httpProxyPort` : Defines the Port if an HTTP proxy is used between the connector and the service.
- `httpProxyUsername` : Defines Proxy Username if an HTTP proxy is used between the connector and the service.
- `httpProxyPassword` : Defines Proxy Password if an HTTP proxy is used between the connector and the service.

JWT signer

- `jwtkey` : The Jwt data structure that represents a cryptographic key.
- `jwtExpiration` : Jwt Expiration.
- `jwtClaims` : Jwt Claims to be included in the payload.
- `jwtAlgorithm` : The Algorithm type to sign payload.

If configuring the connector over REST or through the filesystem, specify the connection details to the SasS Rest resource provider in the `configurationProperties` for the connector. If you are using OAuth for your connection, the minimum required property is `serviceUri` .

```
{
  "configurationProperties" : {
    "objectTypes" : {
      "__ACCOUNT__" : {
        "schema" : [
          {
            "fieldName" : null,
            "type" : null,
            "flags" : []
          }
        ],
        "operations" : {
          "GET" : {
            "path" : null,
            "method" : null,
            "idPath" : null,
            "namePath" : null,
            "headers" : {},
            "queryParams" : {},
            "responseMapping" : {},
            "requestBody" : {},
            "additionalStep" : [
              {
                "resourceName" : null,
                "method" : null,
                "path" : null,
                "valuePath" : null
              }
            ]
          },
          "QUERY" : {
            "path" : null,
            "method" : null,
            "idPath" : null,
            "namePath" : null,
            "headers" : {},
            "queryParams" : {},
            "responseMapping" : {},
            "pagination" : {
              "offSetPagination" : {
                "type" : null,
                "param" : null,
                "path" : null,
                "requestBody" : null
              },
              "cookiePagination" : {
                "type" : null,
                "param" : null,
                "path" : null,
                "requestBody": null
              },
              "pageSizePagination" : {
                "type" : null,
                "param" : null,
                "path" : null,
                "requestBody" : null
              },
              "pagedResultsCookie" : {
                "type" : null,
                "path" : null,
                "regularExpression" : null
              }
            },
            "additionalStep" : [
              {
                "resourceName" : null,
                "method" : null,
                "path" : null,
                "valuePath" : null
              }
            ]
          },
          "CREATE" : {
            "path" : null,
            "method" : null,
            "idPath" : null,
            "headers" : {},
            "queryParams" : {},
            "requestMapping" : {},
            "unflattenAttributes" : [],
            "additionalStep" : {
              "method" : null,
              "path" : null,
              "requestMapping" : null,
              "requestBody" : null
            }
          },
          "UPDATE" : {
            "path" : null,
            "method" : null,
            "idPath" : null,
            "headers" : {},
            "queryParams" : {},
            "requestMapping" : {},
            "unflattenAttributes" : [
              {
                "attributeName" : null,
                "attributeValue" : null
              },
              {
                "attributeName" : null,
                "attributeValue" : null
              }
            ],
            "additionalStep" : [
              {
                "method" : null,
                "path" : null,
                "requestMapping" : {},
                "requestBody" : {}
              }
            ]
          }
        },
        "DELETE" : {
          "path" : null,
          "method" : null,
          "headers" : {},
          "queryParams" : {},
          "requestBody" : {},
          "additionalStep" : {}
        }
```

```
      }
    },
    "exceptions" : {
      "BAD_REQUEST" : [
        {
          "code" : null,
          "regularExpression" : null,
          "messageErrorPath" : null
        },
        {
          "code" : null,
          "regularExpression" : null,
          "messageErrorPath" : null
        }
      ]
    },
    "tokenExpiration" : null,
    "accessToken" : null,
    "serviceUri" : "https://api.exampleapi.com/v1",
    "login" : null,
    "password" : null,
    "authenticationMethod" : "OAUTH",
    "tokenEndpoint" : "https://api.exampleapi.com/oauth2/token",
    "clientId" : "k3..........5g",
    "clientSecret" : "xxxxxxxxxxxxxxxxxx",
    "refreshToken" : "xxxxxxxxxxxxxxxxxx",
    "authToken" : null,
    "acceptSelfSignedCertificates" : false,
    "disableHostNameVerifier" : false,
    "disableHttpCompression" : false,
    "clientCertAlias" : null,
    "clientCertPassword" : null,
    "maximumConnections" : "10",
    "httpProxyHost" : null,
    "httpProxyPort" : null,
    "httpProxyUsername" : null,
    "httpProxyPassword" : null,
    "connectionTimeout" : "30",
    "grantType" : "refresh_token",
    "scope" : null,
    "authorizationTokenPrefix" : "Bearer",
    "useBasicAuthForOauthTokenNeg" : true,
    "jwtKey" : null,
    "jwtExpiration" : null,
    "customGrantType" : null,
    "jwtClaims" : null,
    "defaultHeader" : null
  }
}
```

Note

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

SCHEMA

```
{
  "schema" : [
    {
      "fieldName" : null,
      "type" : null,
      "flags" : []
    }
  ]
}
```

For the creation of the schemas it is necessary to provide name and type of value.

`fieldName:` The attribute name.

`type:` The type of value for the attribute. Available values:

- `BINARY`
- `BOOLEAN`
- `COMPLEX`
- `DECIMAL`
- `INTEGER`
- `REFERENCE`
- `STRING`
- `DATETIME`

  `flags:` determines if is required, readable, writable, updatable, multivalued, or nullable.

- `NOT_UPDATEABLE`
- `NOT_CREATABLE`
- `NOT_READABLE`
- `NOT_RETURNED_BY_DEFAULT`
- `MULTIVALUED`
- `REQUIRED`

OPERATIONS

GET

```
{
  "GET" : {
    "path" : null,
    "method" : null,
    "idPath" : null,
    "namePath" : null,
    "headers" : {},
    "responseMapping" : {},
    "requestBody" : {},
    "additionalStep" : [
      {
        "resourceName" : null,
        "method" : null,
        "path" : null,
        "valuePath" : null
      }
    ]
  }
}
```

- `path:` Resource Path. If you want to include the uid in the path, you must set the variable {uid}. Required.

- `method:` HTTP method. Required.

- `idPath:` Path of the response to assign the resource UID. Required.

- `namePath:` Path of the response to assign the resource __NAME__. Required.

- `headers:` HTTP headers sent in the request, indicated by key-value. Optional.

- `queryParams:` Query parameters to add to the request url. Optional.

- `responseMapping:` It should be configured based on the API response if the response is complex. The keys represent the attributes of the schemas, while the values indicate the paths to the resources within the response. If no mapping is required, you should set both the key and value as /*.

- `requestBody:` Represents the JSON structure of the request for the query. You must provide the body required by the API. If no body is needed, you don't need to send one. The `requestBody` defines where the attributes are to be assigned within the request body. Optional. If you want to insert the `uid` in the body, you must insert the variable `{uid}` as the value of a field.

- `additionalStep:` If you need to make additional queries, you can add them to the schema attributes. To make an extra query, you must include the following properties. Optional.

  - `path:` Required.

  - `method:` Required.

  - `requestBody:` Must be specified if necessary Optional.

  - `resourceName:` Attribute name, must be in the schema. Required.

  - `valuePath:` Path of the attribute to retrieve from the response. If you want to get an array of elements at the end of the path, you must specify [*]. The square brackets indicate that it is an array, and the asterisk indicates that all the elements will be fetched. If the elements are a list of objects and you only want to fetch a particular field from each one, you must use [*].fieldName. If the element is an object and you only want to get a specific field from that object, you must indicate the object's name followed by fieldName. Required.

QUERY

```
 {
  "QUERY" : {
    "path" : null,
    "method" : null,
    "idPath" : null,
    "namePath" : null,
    "headers" : {},
    "responseMapping" : {},
    "pagination" : {
        "offSetPagination": {
            "type": null,
            "param": null,
            "path" : null,
            "requestBody" : null
         },
         "cookiePagination" : {
            "type" : null,
            "param" : null,
            "path" : null,
            "requestBody" : null
          },
          "pageSizePagination" : {
              "type" : null,
              "param" : null,
              "path" : null,
              "requestBody" : null
          },
          "pagedResultsCookie" : {
              "type" : null,
              "path" : null,
              "regularExpression" : null
          }
    },
    "additionalStep" : [
      {
        "resourceName" : null,
        "method" : null,
        "path" : null,
        "valuePath" : null,
        "requestBody" : null
      }
    ]
  }
}
```

- `path:` Resource Path. If you want to include the uid in the path, you must set the variable {uid}. Required.
- `method:` HTTP method. Required.
- `idPath:` Path of the response to assign the resource UID. Required.

- `namePath:` Path of the response to assign the resource __NAME__. Required.
- `headers:` HTTP headers sent in the request, indicated by key-value. Optional.
- `queryParams:` Query parameters to add to the request url. Optional.
- `responseMapping:` It should be configured based on the API response if the response is complex. The keys represent the attributes of the schemas, while the values indicate the paths to the resources within the response. If no mapping is required, you should set both the key and value as /*. Optional.
- `requestBody:` Represents the JSON structure of the request for the query. You must provide the body required by the API. If no body is needed, you don't need to send one. The `requestBody` defines where the attributes are to be assigned within the request body. Optional. In case you want a body in the API, you can specify it.
- `additionalStep:` If you need to make additional queries, you can add them to the schema attributes. To make an extra query, you must include the following properties. Optional.
  - `path:` Required.
  - `method:` Required.
  - `requestBody:` must be specified if necessary Optional.
  - `resourceName:` Attribute name, must be in the schema. Required.
  - `valuePath:` Path of the attribute to retrieve from the response. If you want to get an array of elements at the end of the path, you must specify [*]. The square brackets indicate that it is an array, and the asterisk indicates that all the elements will be fetched. If the elements are a list of objects and you only want to fetch a particular field from each one, you must use [*].fieldName. If the element is an object and you only want to get a specific field from that object, you must indicate the object's name followed by fieldName. Required.

  `pagination:` There are 3 types of pagination: `CookiePagination`, `OffSetPagination`, `PageSizePagination`.

  - `type:` An HTTP method. Can be body or param. Optional.
  - `param:` To set the params is a key-value string, where you must provide the key followed by an equal sign '=' and the available values of the following 3 variables `_pagedResultsCookie`, `_pagedResultsOffSet`, `_pageSize` if you want to add more than one key-value set, they must be separated by a `&`. Optional.
  - `requestBody:` In case you need to perform pagination through the body, you need to specify the content required by the API. To insert the pagination values, you must use variables inside the requestBody. Available variables are `_pagedResultsCookie`, `_pagedResultsOffSet`, or `_pageSize`.
  - `path:` In case you need a different URL than the one in the operation. Optional.
  - `regularExpression:` Extracts the value of the cookie. Optional.

  `pagedResultsCookie:` Indicates how to get the cookie or cursor for the next page of results.

- `type:` Pagination type. Available values are `body` or `header`. Required.
- `path:` Indicates the path or name of the header containing the cookie. Required.
- `regularExpression:` Extracts the value of the cookie. The `path` and `method` must be specified. The `requestBody` and `requestMapping` are optional.

> **Important**
>
> It is important to make sure that you use the exact variable name and enclose it in braces.

CREATE

```
{
  "CREATE" : {
    "idPath" : null,
    "path" : null,
    "method" : null,
    "headers" : {},
    "queryParams" : {},
    "requestMapping" : {},
    "unflattenAttributes" : [],
    "additionalStep" : [
      {
        "method" : null,
        "path" : null,
        "requestMapping" : null,
        "requestBody" : null
      }
    ]
  }
}
```

- `idPath:` Path of the response to assign the resource UID. Required.
- `path:` Resource Path. If you want to include the uid in the path, you must set the variable {uid}. Required.
- `method:` HTTP method. Required.
- `headers:` HTTP headers sent in the request, indicated by key-value. Optional.
- `queryParams:` Query parameters to add to the request url. Optional.
- `requestMapping:` The keys correspond to the attribute names, and the values represent how they will be sent in the request. If you need a complex `requestBody`, you must define the `requestBody`, property using the appropriate API format . The `requestMapping` values will then be mapped to this `requestBody`. Required.
- `requestBody:` Represents the JSON structure of the request for the query. You must provide the body required by the API. If no body is needed, you don't need to send one. The `requestBody` defines where the attributes are to be assigned within the request body. Optional.
- `unflattenAttributes` It is used to transform multivalued attributes into an array of objects. The `attributeName` represents the name of the attribute, and `attributeValue` represents the value of the field that the object will have. Optional.
- `additionalStep:` If you need to make additional queries, you can add them to the schema attributes. To make an extra query, you must include the following properties. Optional. The `path` and `method` must be specified. The `requestBody` and `requestMapping` are optional.

UPDATE

```
{
  "UPDATE" : {
    "path" : null,
    "method" : null,
    "idPath" : null,
    "headers" : {},
    "queryParams" : {},
    "requestMapping" : {},
    "unflattenAttributes" : [],
    "additionalStep" : [
      {
        "method" : null,
        "path" : null,
        "requestMapping" : null,
        "requestBody" : null
      }
    ]
  }
}
```

- `path:` Resource Path. If you want to include the uid in the path, you must set the variable {uid}. Required.
- `method:` HTTP method. Required.
- `idPath:` Path of the response to assign the resource UID. Required.
- `headers:` HTTP headers sent in the request, indicated by key-value. Optional.
- `queryParams:` Query parameters to add to the request url. Optional.
- `requestMapping:` The keys correspond to the attribute names, and the values represent how they will be sent in the request. If you need a complex `requestBody`, you must define the `requestBody`, property using the appropriate API format . The `requestMapping` values will then be mapped to this `requestBody`. Required.
- `requestBody:` Represents the JSON structure of the request for the query. You must provide the body required by the API. If no body is needed, you don’t need to send one. The `requestBody` defines where the attributes are to be assigned within the request body. Optional.
- `unflattenAttributes:` It is used to transform multivalued attributes into an array of objects. The `attributeName` represents the name of the attribute, and `attributeValue` represents the value of the field that the object will have. Optional.
- `additionalStep:` If you need to make additional queries, you can add them to the schema attributes. To make an extra query, you must include the following properties. Optional. The `path` and `method` must be specified. The `requestBody` and `requestMapping` are optional.

DELETE

```
{
  "DELETE" : {
    "path" : null,
    "method" : null,
    "queryParams" : null,
    "headers" : {},
    "requestBody" : {},
    "additionalStep" : {}
  }
}
```

- `path:` Resource Path. If you want to include the uid in the path, you must set the variable {uid}. Required.
- `method:` HTTP method. Required.
- `headers:` HTTP headers sent in the request, indicated by key-value. Optional.
- `QueryParams:` Query parameters to add to the request url. Optional.
- `requestBody:` Represents the JSON structure of the request for the query. You must provide the body required by the API. If no body is needed, you don’t need to send one. The `requestBody` defines where the attributes are to be assigned within the request body. Optional. The `path` and `method` must be specified. The `requestBody` and `requestMapping` are optional.

Exceptions

This JSON represents a configuration structure for an exception and its standardised error types.

```
{
  "exceptions" : {
    "BAD_REQUEST" : [
      {
        "code" : null,
        "regularExpression" : null,
        "messageErrorPath" : null
      },
      {
        "code" : null,
        "regularExpression" : null,
        "messageErrorPath" : null
      }
    ]
  }
}
```

Exceptions types:

- `BAD_REQUEST`
- `NOT_FOUND`
- `ALREADY_EXIST`

- `FORBIDDEN`
- `UNAUTHORIZED`

Each element of the array has the following fields:

- `code` : Error code (String). Required.
- `messageErrorPath` : Path where the error field is contained in the response. Required.
- `regularExpression` : Value contained in `messageErrorPath` . Optional.

Examples

Example with unflatten attributes:

```
{
  "unflattenAttributes" : [
    {
      "attributeName" : "role_ids",
      "attributeValue" : "id"
    },
    {
      "attributeName" : "group_ids",
      "attributeValue" : "id"
    }
  ]
}
```

Example with response mapping:

```
{
  "responseMapping" : {
    "users" : {
      "roles" : "roles[*].role_id",
      "first_name" : "first_name",
      "created_at" : "created_at",
      "active" : "active"
    }
  }
}

{
  "responseMapping" : {
    "/*": "/*"
  }
}

{
  "responseMapping" : {
    "users" : "users"
  }
}
```

Example with Request mapping:

```
{
  "requestMapping" : {
    "userName" : "userName",
    "sn" : "sn",
    "givenName" : "givenName",
    "mail" : "mail",
    "telephoneNumber" : "telephoneNumber",
    "description" : "description"
  }
}
```

Example with request mapping and request body:

```
{
  "requestMapping" : {
    "userName" : "users[0].userName",
    "sn" : "users[0].sn",
    "givenName" : "users[0].givenName"
  },
  "requestBody" : {
    "users" : [
      {
        "active" : true
      }
    ]
  }
}
```

Example with request body pagination:

```
{
  {
    "limit" : "{_pageSize}"
  }
}
```

```
{
  {
    "limit" : "{_pageSize}",
    "offset" : "{_pagedResultsOffSet}"
  }
}
```

```
{
  {
    "cookie" : "{_pagedResultsCookie}"
  }
}
```

Example with param pagination:

```
  limit={_pageSize}&offSet={_pagedResultsOffSet}
  cursor={_pagedResultsCookie}
```

Test the SaaS REST connector

Test that the connector was configured correctly:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Accept-API-Version: resource=1.0' \
--request POST  'http://localhost:8080/system/rest?_action=test'
{
  "name" : rest,
  "enabled" : true,
  "config" : "config/provisioner.openicf/rest",
  "connectorRef" : {
    "bundleVersion" : 1.5.20.30,
    "bundleName" : "org.forgerock.openicf.connectors.rest-connector",
    "connectorName" : "org.forgerock.openicf.connectors.rest.RestConnector"
  },
  "displayName" : "Rest Connector",
  "objectTypes" : [
    "__ACCOUNT__",
    "__ALL__"
  ],
  "ok" : true
}
```

Create user

Example of Advanced Identity Cloud

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request POST \
--data '{
  "userName" : "johndoe",
  "sn" : "doe",
  "givenName" : "john",
  "mail" : "john.doe@example.com",
  "telephoneNumber" : "0101010101",
  "description" : "some description"
}' \
'http://localhost:8080/openidm/system/rest/__ACCOUNT__?_action=create'
{
  "_id" : "user_ID",
  "_rev" : "rev_ID",
  "__NAME__" : "john",
  "sn" : "doe",
  "givenName" : "john",
  "mail" : "john.doe@example.com",
  "telephoneNumber" : "0101010101",
  "description" : "some description",
  "accountStatus" : "active"
}
```

Update user

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request PUT \
--data '{
  "userName" : "updatedUsername",
  "sn" : "updatedSn",
  "givenName" : "updatedGivenName",
  "mail" : "john.doe123@example.com",
  "telephoneNumber" : "11110000",
  "description" : "updated description"
}' \
'http://localhost:8080/openidm/system/rest/__ACCOUNT__/USER_ID'
{
  "_id" : "ID",
  "_rev" : "redID",
  "__NAME__" : "updatedUsername",
  "sn" : "updatedSn",
  "givenName" : "updatedGivenName",
  "mail" : "john.doe123@example.com",
  "telephoneNumber" : "11110000",
  "description" : "updated description",
  "accountStatus" : "active"
}
```

Get user

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request GET \
'http://localhost:8080/openidm/system/rest/__ACCOUNT__/USER_ID'
{
  "_id" : "user_ID",
  "_rev" : "rev_ID",
  "__NAME__" : "john",
  "sn" : "doe",
  "givenName" : "john",
  "mail" : "john.doe@example.com",
  "telephoneNumber" : "0101010101",
  "description" : "some description",
  "accountStatus" : "active"
}
```

List users

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request GET \
'http://localhost:8080/openidm/system/rest/__ACCOUNT__?_queryFilter=true'
{
  "result" : [
    {
      "_id" : "user_ID_1",
      "_rev" : "rev_ID_1",
      "__NAME__" : "johndoe",
      "accountStatus" : "active",
      "mail" : "john.doe@example.com",
      "givenName" : "john",
      "sn" : "doe",
      "description" : "some description"
    },
    {
      "_id" : "user_ID_2",
      "_rev" : "rev_ID_2",
      "__NAME__" : "testuser",
      "accountStatus" : "active",
      "mail" : "test.user@example.com",
      "givenName" : "test",
      "sn" : "user",
      "description": "some description"
    }
    ...
  ],
  "resultCount" : 999,
  "pagedResultsCookie" : "Cookie",
  "totalPagedResultsPolicy" : "NONE",
  "totalPagedResults" : -1,
  "remainingPagedResults" : -1
}
```

OpenICF Interfaces Implemented by the Rest Connector

The Rest Connector implements the following OpenICF interfaces. For additional details, see ICF 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.

Rest Connector Configuration

The Rest Connector has the following configurable properties:

Configuration properties

Property	Type	Default	Encrypted[1]	Required[2]
`tokenExpiration`	`Long`	`null`		✗ No
Description is not available				

Property	Type	Default	Encrypted[1]	Required[2]
`accessToken`	`String`	`null`		✕ No
Description is not available				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

Basic Configuration Properties

Property	Type	Default	Encrypted[1]	Required[2]
`serviceUri`	`String`	`null`		✓ Yes
The service endpoint URI.				
`login`	`String`	`null`		✓ Yes
The service login name.				
`password`	`GuardedString`	`null`	🔒 Yes	✕ No
The service user password.				
`authenticationMethod`	`String`	`OAUTH`		✓ Yes
Defines which method is to be used to authenticate on the remote server. Options are BASIC (username/password), OAUTH (Client id/secret) or TOKEN (static token).				
`tokenEndpoint`	`String`	`null`		✕ No
When using OAUTH as authentication method, this property defines the endpoint where a new access token should be queried for (https://myserver.com/oauth2/token).				
`clientId`	`String`	`null`		✓ Yes
The client identifier for OAuth2.				
`clientSecret`	`GuardedString`	`null`	🔒 Yes	✕ No
Secure client secret for OAuth2.				
`authToken`	`GuardedString`	`null`	🔒 Yes	✕ No

Property	Type	Default	Encrypted[1]	Required[2]
Static authentication token.				
`acceptSelfSignedCertificates`	`boolean`	`false`		✓ Yes
To be used for debug/test purposes. To be avoided in production.				
`disableHostNameVerifier`	`boolean`	`false`		✓ Yes
To be used for debug/test purposes. To be avoided in production.				
`disableHttpCompression`	`boolean`	`false`		✓ Yes
Content compression is enabled by default. Set this property to true to disable it.				
`clientCertAlias`	`String`	`null`		✓ Yes
If TLS Mutual Auth is needed, set this to the certificate alias from the keystore.				
`clientCertPassword`	`GuardedString`	`null`	🔒 Yes	✓ Yes
If TLS Mutual Auth is needed and the client certificate (private key) password is different from the keystore password, set this to the client private key password.				
`maximumConnections`	`Integer`	`10`		✓ Yes
Defines the max size of the HTTP connection pool used.				
`httpProxyHost`	`String`	`null`		✓ Yes
Defines the Hostname if an HTTP proxy is used between the connector and the service.				
`httpProxyPort`	`Integer`	`null`		✓ Yes
Defines the Port if an HTTP proxy is used between the connector and the service.				
`httpProxyUsername`	`String`	`null`		✓ Yes
Defines Proxy Username if an HTTP proxy is used between the connector and the service.				
`httpProxyPassword`	`GuardedString`	`null`	🔒 Yes	✓ Yes

Property	Type	Default	Encrypted[1]	Required[2]
Defines Proxy Password if an HTTP proxy is used between the connector and the service.				
`connectionTimeout`	`int`	`30`		✕ No
Defines a timeout for the underlying HTTP connection in seconds.				
`refreshToken`	`GuardedString`	`null`		✕ No
Used by the refresh_token grant type.				
`grantType`	`String`	`null`		✕ No
The OAuth2 grant type to use (client_credentials, refresh_token, or jwt_bearer).				
`scope`	`String`	`null`		✕ No
The OAuth2 scope to use.				
`authorizationTokenPrefix`	`String`	`Bearer`		✕ No
The prefix to be used in the Authorization HTTP header for Token authentication.				
`useBasicAuthForOauthTokenNeg`	`boolean`	`true`		✓ Yes
The Authentication method for refresh token (Basic Authentication or Sending the ClientId and Client Secret in the Header).				
`jwtKey`	`String`	`null`		✕ No
The JWT data structure that represents a cryptographic key.				
`jwtExpiration`	`Integer`	`null`		✕ No
Defines the JWT expiration time in seconds.				
`jwtAlgorithm`	`String`	`null`		✕ No
The Algorithm type to sign payload.				
`jwtClaims`	`Map`	`null`		✕ No
JWT Claims to be included in the payload				
`jwtPem`	`String`	`null`		✕ No

Property	Type	Default	Encrypted(1)	Required(2)
The contents of the private key of the PEM file				
`jwtCert`	`String`	`null`		✕ No
The contents of the certificate of the PEM file				
`objectTypes`	`Map`	`{ACCOUNT={schema=[{fieldName=null, flags=[], type=null}], operations={DELETE={path=null, headers={}, additionalStep=[{path=null, method=null, requestBody={}, requestMapping={}}], method=null, queryParams={}, requestBody=null}, CREATE={unflattenAttributes=[{attributeValue=null, attributeName=null}], path=null, headers={}, additionalStep=[{path=null, method=null, requestBody={}, requestMapping={}, valuePath=null}], idPath=null, method=null, queryParams={}, requestMapping={}}, QUERY={path=null, headers={}, additionalStep=[{path=null, method=null, requestBody=null, resourceName=null, valuePath=null}], idPath=null, namePath=null, pagination={cookiePagination=null, offSetPagination=null, pagedResultsCookie=null, pageSizePagination=null}, method=null, queryParams={}, responseMapping={}}, GET={path=null, headers={}, additionalStep=[{path=null, method=null, requestBody=null, resourceName=null, valuePath=null}], idPath=null, namePath=null, method=null, queryParams={}, requestBody={}, responseMapping={}}, UPDATE={unflattenAttributes=[{attributeValue=null, attributeName=null}], path=null, headers={}, additionalStep=[{path=null, method=null, requestBody={}, requestMapping={}, valuePath=null}], idPath=null, method=null, queryParams={}, requestMapping={}}}, exceptions={}}}`		✓ Yes
Defines the configuration for operations(GET/QUERY/CREATE/UPDATE/DELETE) and schemas.				
`keyAlgorithm`	`String`	`null`		✕ No
Indicates the type of key (such as RSA, DSA or EC) used to sign from the PEM.				
`defaultHeaders`	`Map`	`null`		✓ Yes

Property	Type	Default	Encrypted[1]	Required[2]
Headers to be sent by default.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

Salesforce connector

The Salesforce connector lets you provision, reconcile, and synchronize users between Salesforce and the IDM managed user repository.

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

For a complete example that includes the configuration required to synchronize users with this connector, refer to Synchronize users between Salesforce and IDM.

Before you configure the Salesforce connector

The instructions in this topic assume that you have an existing Salesforce organization, a Salesforce administrative account, and a Connected App with OAuth enabled.

For instructions on setting up a Connected App, refer to the corresponding Salesforce documentation. When you have set up the Connected App, locate the *Consumer Key* and *Consumer Secret*. You will need these details to configure the connector.

The Salesforce connector is bundled with IDM and Advanced Identity Cloud and has no specific installation requirements.

Install the Salesforce connector

Tip

To check for an Advanced Identity Cloud application for this connector, refer to:

- Application management
- App catalog

You can download any connector from Backstage, but some are included in the default deployment for Advanced Identity Cloud, IDM, or RCS. When using an included connector, you can skip installing it and move directly to configuration.

Connector included in default deployment

Connector	IDM	RCS
Salesforce	✓ Yes	✗ No

Download the connector .jar file from Backstage.

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

```
mv ~/Downloads/salesforce-connector-1.5.20.30.jar /path/to/openidm/connectors/
```

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

Configure the Salesforce connector

Create a connector configuration using the IDM admin UI:

1. From the navigation bar, click **Configure > Connectors**.

2. On the **Connectors** page, click **New Connector**.

3. On the **New Connector** page, type a **Connector Name**.

4. From the **Connector Type** drop-down list, select **Salesforce Connector - 1.5.20.30**.

   The Login URL is the OAuth endpoint used to make the OAuth authentication request to Salesforce.

   > **Note**
   >
   > When you create your connected app, you are instructed to wait 2-10 minutes for the settings to propagate across all the Salesforce data centers. If you are using a Salesforce test tenant, such as `https://eu26.lightning.force.com`, you can specify a custom URL here and enter the FQDN of the test tenant. This lets you test the connector without waiting for the new app settings to be propagated.

5. Complete the **Base Connector Details**.

   > **Tip**
   >
   > For a list of all configuration properties, refer to Salesforce Connector Configuration

   The connector attempts to access your Salesforce organization.

6. Enter your Salesforce login credentials.

7. On the permission request screen, click **Allow** to enable IDM to access your Salesforce Connected App.

8. Click **Save**.

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

Refer to this procedure to create a connector configuration over REST.

Configure the Salesforce connector with a configuration file

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

Copy this sample file (`provisioner.openicf-salesforce.json`) to your project's `conf` directory and set at least the following properties, based on the grant type:

Refresh token

```
"configurationProperties" : {
    "grantType" : "refresh_token",
    "refreshToken" : "refreshToken",
    "loginUrl" : "loginURL",
    "clientSecret" : "clientSecret",
    "clientId" : "clientId",
    "instanceUrl" : "instanceURL"
}
```

Client credentials

```
"configurationProperties" : {
    "grantType" : "client_credentials",
    "loginUrl" : "loginURL",
    "clientSecret" : "clientSecret",
    "clientId" : "clientId",
    "instanceUrl" : "instanceURL"
}
```

`loginUrl`

The OAuth endpoint that will be used to make the OAuth authentication request to Salesforce.

The default endpoint for a production system is `https://login.salesforce.com/services/oauth2/token` . The default endpoint for a sandbox (test) system is `https://test.salesforce.com/services/oauth2/token` .

`clientSecret`

The Consumer Secret associated with your Connected App.

`clientId`

The Consumer Key associated with your Connected App.

grantType

The OAuth2 grant type to use (`client_credentials` or `refresh_token`).

client_credentials

This grant type requires `clientId` , `clientSecret` , `loginURL` , and `instanceURL` .

refresh_token

This grant type requires `clientId` , `clientSecret` , `loginURL` , and `instanceURL` , and `refreshToken` .

refreshToken

The admin UI gets the `refreshToken` automatically during connector configuration. Required when the grant type is `refresh_token` . To manually get the value, refer to Get the `refreshToken` and `instanceURL` .

instanceURL

The admin UI gets the `instanceURL` automatically during connector configuration when using the `refresh_token` grant type. For the `client_credentials` grant type, you must provide the value. This is typically a well-known value, but to manually get the value, refer to Get the `refreshToken` and `instanceURL` .

Get the `refreshToken` and `instanceURL`

1. Browse to the following URL:

   ```
   SALESFORCE_URL/services/oauth2/authorize?
   response_type=code&client_id=CONSUMER_KEY&redirect_uri=REDIRECT_URI&scope=id+api+refresh_token
   ```

   - SALESFORCE_URL is one of the following:
     - A production URL (`https://login.salesforce.com`)
     - A sandbox URL (`https://test.salesforce.com`)
     - A custom Salesforce MyDomain URL, such as `https://ic-example-com--SUP1.cs21.my.salesforce.com`
   - CONSUMER_KEY is the Consumer Key associated with the Connected App that you created within your Salesforce organization.
   - REDIRECT_URI is the IDM URI Salesforce should redirect to during authentication. It must match the Redirect URI specified within your Salesforce Connect App configuration, for example:

     ```
     https://localhost:8443/
     ```

2. You are redirected to Salesforce, and prompted to give this application access to your Salesforce account. When you have given consent, you should receive a response URL that looks similar to the following:

   ```
   https://localhost:8443/admin/index.html#connectors/edit//&code=aPrxJZTK7Rs03PU634VK8Jn9o_U3ZY1ERxM7IiklF...
   ```

The `&code` part of this URL is an authorization code, that you need for the following step.

> **Caution**
>
> This authorization code expires after 10 minutes. If you do not complete the OAuth flow within that time, you will need to start this process again.

3. Copy the authorization code from the response URL and use it as the value of the `code` parameter in the following REST call. The consumer-key, redirect-uri, and SALESFORCE_URL must match what you used in the first step of this procedure:

```
curl \
--verbose \
--data "grant_type=authorization_code" \
--data "client_id=consumer-key" \
--data "client_secret=consumer-secret" \
--data "redirect_uri=https://localhost:8443/" \
--data "code=access-token-code" \
"SALESFORCE_URL/services/oauth2/token"
{
  "access_token": "00DS0000003K4fU!AQMAQOzEU.8tCjg8Wk79yKPKCtrtaszX5jrHtoT4NBpJ8x...",
  "signature": "2uREX1lseXdg3Vng/2+Hrlo/KHOWYoim+poj74wKFtw=",
  "refresh_token": "5Aep861KIwKdekr90I4iHdtDgWwRoG7O_6uHrgJ.yVtMS0UaGxRqE6WFM77W7...",
  "token_type": "Bearer",
  "instance_url": "https://example-com.cs1.my.salesforce.com",
  "scope": "id api refresh_token",
  "issued_at": "1417182949781",
  "id": "https://login.salesforce.com/id/00DS0000003K4fUMAS/00530000009hWLcAAM"
}
```

The output includes the `refresh_token` and `instance_url` you need to configure the connector.

4. Set `"enabled" : true` to enable the connector.

5. Save the connector configuration.

Use the Salesforce connector with a proxy server

If the IDM server is hosted behind a firewall and requests to the resource server are routed through a proxy, you must specify the proxy information in the connector configuration. The Salesforce connector supports two mutually exclusive configuration properties for specifying the proxy: `proxyHost` and `proxyUri`.

To specify the proxy server details using an HTTP scheme, set the `proxyHost` and `proxyPort` properties in the connector configuration. If the proxy requires authentication, set the `proxyUsername` and `proxyPassword` properties. For example:

```
"configurationProperties": {
    ...
    "proxyHost": "myproxy.example.com",
    "proxyPort": 8080,
    "proxyUsername": "hgale815",
    "proxyPassword": "password123",
    ...
}
```

To specify the proxy server details using a URI, set the `proxyUri` property in the connector configuration. The `proxyUri` property contains the scheme, host, and port. If the proxy requires authentication, set the `proxyUsername` and `proxyPassword` properties. For example:

```
"configurationProperties": {
    ...
    "proxyUri": "https://myproxy.example.com:8080",
    "proxyUsername": "hgale815",
    "proxyPassword": "password123",
    ...
}
```

Test the Salesforce connector

Test that the configuration is correct by running the following command:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system/salesforce?_action=test"
{
  "name": "salesforce",
  "enabled": true,
  "config": "config/provisioner.openicf/salesforce",
  "connectorRef": {
    "bundleVersion": "[1.5.0.0,1.6.0.0)",
    "bundleName": "org.forgerock.openicf.connectors.salesforce-connector",
    "connectorName": "org.forgerock.openicf.connectors.salesforce.SalesforceConnector"
  },
  "displayName": "Salesforce Connector",
  "objectTypes": [
    "__ALL__",
    "User"
  ],
  "ok": true
}
```

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

Salesforce remote connector

If you want to run this connector outside of PingOne Advanced Identity Cloud or IDM, you can configure the Salesforce connector as a remote connector. Java Connectors installed remotely on a Java Connector Server function identically to those bundled locally within PingOne Advanced Identity Cloud or installed locally on IDM.

You can download the Salesforce connector from here.

Refer to Remote connectors for configuring the Salesforce remote connector.

Configure connection pooling

The Salesforce connector supports HTTP pooling, which can substantially improve the performance of the connector. Learn more about the basic connection pooling configuration and different pooling mechanisms described in Connection pooling configuration.

Implementation Specifics

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

- Attributes themselves cannot be removed from Salesforce. The connector therefore performs an update with `""` as the value of the attribute being removed. This sets the value of the removed attribute to `null`.

  > **Note**
  >
  > Salesforce does not support application user DELETE requests.

- The Salesforce connector supports any Salesforce object that is available to the API. To check which objects are available, log in to Salesforce Workbench to access the API explorer. This URL points to Version 49 of the API. Adjust the URL for the latest API version.

  Because the number of Salesforce objects is potentially very large, the Salesforce connector configuration includes a `supportedObjectTypes` property that lets you specify the objects you want to support. The connector checks the metadata in Salesforce for each of the objects you list in this property, and dynamically builds the required schema. The sample connector configuration file (`provisioner.openicf-salesforce.json`) generates the schema only for the User object:

  ```
  {
      ...
      "configurationProperties": {
          ...
          "supportedObjectTypes": [
              "User"
          ]
      },
  }
  ```

  You can add any object to the list of `supportedObjectTypes`, and the connector will build the schema for that object.

- The Salesforce API restricts how query results can be paged. The default, and maximum page size is `2000`. The minimum page size is `200`. The Salesforce API does not guarantee that the requested page size is the actual page size. Returned results might vary, to maximize performance.

  For example, the following query (with `"pageSize=1"`) might return more than one user if more than one user exists in Salesforce:

  ```
  http://localhost:8080/openidm/system/salesforce/user?_queryId=query-all-ids&_pageSize=1
  ```

For more information, refer to the Salesforce documentation.

Permission sets and groups

The Salesforce connector can display and allow you to interact with a user’s groups and permissions on the user’s data model. This is done through the `__PermissionSetIds__` and `__GroupIds__` attributes.

Configure permission sets and groups

To configure these attributes, add the following to the user object in your provisioner file:

```
[...]
    "__PermissionSetIds__" : {
        "type" : "array",
        "items" : {
            "type" : "string",
            "nativeType" : "string"
        },
        "nativeName" : "__PermissionSetIds__",
        "nativeType" : "string",
        "absentIfEmpty" : true,
        "flags" : [
            "NOT_RETURNED_BY_DEFAULT"
        ]
    },
    "__GroupIds__" : {
        "type" : "array",
        "items" : {
            "type" : "string",
            "nativeType" : "string"
        },
        "nativeName" : "__GroupIds__",
        "nativeType" : "string",
        "absentIfEmpty" : true,
        "flags" : [
            "NOT_RETURNED_BY_DEFAULT"
        ]
    }
```

Important

Both `__PermissionSetIds__` and `__GroupIds__` should have the `NOT_RETURNED_BY_DEFAULT` flag enabled.

Get a user’s permission sets and groups

When getting a user object, you must specifically request that the system return the `__PermissionSetIds__` and `__GroupIds__` fields. For example:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
"https://localhost:8443/openidm/system/salesforce/User?
_queryFilter=Alias+eq+"bjensen"&_fields=_id,__PermissionSetIds__,__GroupIds__"
```

> **Note**
>
> Salesforce assigns a default permission set to each created user.

Create a user with permissions sets and groups

To create a user object with associated permissions and groups, include the `__PermissionSetIds__` and `__GroupIds__` fields in the POST request. These fields are an array of strings.

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
--data '{
  "Username": "bjensen@example.com",
  "LastName": "Jensen",
  "Email": "bjensen@example.com",
  "Alias": "bjensen",
  "LocaleSidKey": "en_US",
  "LanguageLocaleKey": "en_US",
  "FirstName": "Barbara"
  "TimeZoneSidKey": "America/Los_Angeles",
  "CommunityNickname": "bjensen"
  "ProfileId": "00e200000001ehhP"
  "__PermissionSetIds__": ["0P50300000QIbGAM"],
  "__GroupIds__": ["0J40700000DIg3Hm"]
}' \
"https://localhost:8443/openidm/system/salesforce/User?_action=create"
```

Update a user’s permission sets and groups

You can update a user’s permission sets and groups through the API using a PUT call. The following request adds a permission set and group to `scarter`, a user with id `00503000002XmQqAAK`.

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request PUT \
--data '{
    "OutOfOfficeMessage": "",
    "LastName": "Carter",
    "UserPreferencesSortFeedByComment": true,
    "UserPreferencesPathAssistantCollapsed": false,
    "BannerPhotoUrl": "/profilephoto/005/B",
    "EmailPreferencesAutoBcc": true,
    "TimeZoneSidKey": "America/Los_Angeles",
    "UserPreferencesHideLightningMigrationModal": false,
    "UserPreferencesFavoritesWTShown": false,
    "MediumPhotoUrl": "https://example.force.com/profilephoto/005/M",
    "UserPreferencesHideS1BrowserUI": false,
    "UserType": "Standard",
    "UserPreferencesDisableSharePostEmail": false,
    "UserPreferencesCreateLEXAppsWTShown": false,
    "UserPreferencesApexPagesDeveloperMode": false,
    "EmailPreferencesStayInTouchReminder": true,
    "UserPreferencesDisMentionsCommentEmail": false,
    "UserPreferencesShowMobilePhoneToExternalUsers": false,
    "Username": "scarter@example.com",
    "EmailEncodingKey": "UTF-8",
    "UserPreferencesShowFaxToExternalUsers": false,
    "UserPreferencesSuppressTaskSFXReminders": false,
    "UserPreferencesShowStateToGuestUsers": false,
    "UserPreferencesHideSfxWelcomeMat": true,
    "LanguageLocaleKey": "en_US",
    "UserPreferencesShowManagerToExternalUsers": false,
    "LastViewedDate": "2022-11-09T17:15:08.000+0000",
    "UserPermissionsAvantgoUser": false,
    "UserPreferencesHideCSNGetChatterMobileTask": false,
    "UserPreferencesPipelineViewHideHelpPopover": false,
    "UserPreferencesDisableLikeEmail": true,
    "UserPermissionsSupportUser": false,
    "UserPermissionsChatterAnswersUser": false,
    "FirstName": "Sam",
    "ForecastEnabled": false,
    "ReceivesAdminInfoEmails": false,
    "UserPreferencesDisableAllFeedsEmail": false,
    "LocaleSidKey": "en_US",
    "UserPreferencesHideCSNDesktopTask": false,
    "SystemModstamp": "2022-11-09T17:14:58.000+0000",
    "Id": "00503000002XmQqAAK",
    "UserPreferencesLightningExperiencePreferred": true,
    "UserPreferencesDisableBookmarkEmail": false,
    "UserPreferencesEventRemindersCheckboxDefault": true,
    "UserPreferencesDisableMentionsPostEmail": false,
    "CommunityNickname": "scarter",
    "UserPreferencesShowCityToExternalUsers": false,
    "UserPreferencesShowStreetAddressToExternalUsers": false,
    "IsProfilePhotoActive": false,
    "UserPreferencesDisProfPostCommentEmail": false,
    "CreatedById": "00530000009hWLcAAM",
    "UserPermissionsMobileUser": false,
    "UserPreferencesDisableLaterCommentEmail": false,
    "UserPreferencesShowMobilePhoneToGuestUsers": false,
    "UserPreferencesPreviewCustomTheme": false,
    "UserPermissionsCallCenterAutoLogin": false,
    "UserPreferencesPreviewLightning": false,
    "IsActive": true,
    "UserPreferencesDisableFileShareNotificationsForApi": false,
    "BadgeText": "",
    "UserPreferencesShowFaxToGuestUsers": false,
    "UserPreferencesHasCelebrationBadge": false,
    "UserPreferencesTaskRemindersCheckboxDefault": true,
    "LastReferencedDate": "2022-11-09T17:15:08.000+0000",
    "UserPreferencesDisableChangeCommentEmail": false,
    "FullPhotoUrl": "https://example.force.com/profilephoto/005/F",
    "UserPreferencesDisableEndorsementEmail": false,
    "UserPreferencesActivityRemindersPopup": true,
    "UserPreferencesEnableAutoSubForFeeds": false,
    "UserPreferencesShowStateToExternalUsers": false,
    "UserPermissionsOfflineUser": false,
    "UserPreferencesDisCommentAfterLikeEmail": false,
    "UserPreferencesShowTitleToGuestUsers": false,
    "SmallBannerPhotoUrl": "/profilephoto/005/D",
    "UserPreferencesShowWorkPhoneToExternalUsers": false,
    "MediumBannerPhotoUrl": "/profilephoto/005/E",
    "UserPreferencesShowProfilePicToGuestUsers": false,
    "UserPreferencesShowCountryToGuestUsers": false,
    "EmailPreferencesAutoBccStayInTouch": false,
    "UserPreferencesShowEmailToExternalUsers": false,
    "UserPreferencesShowWorkPhoneToGuestUsers": false,
    "UserPreferencesReminderSoundOff": false,
    "UserPreferencesUserDebugModePref": false,
    "LastModifiedById": "00530000009hWLcAAM",
    "UserPreferencesDisableFollowersEmail": false,
    "UserPreferencesHideEndUserOnboardingAssistantModal": false,
    "UserPreferencesShowStreetAddressToGuestUsers": false,
    "ReceivesInfoEmails": false,
    "UserPreferencesShowManagerToGuestUsers": false,
    "UserPreferencesExcludeMailAppAttachments": false,
    "UserPreferencesShowEmailToGuestUsers": false,
    "DigestFrequency": "D",
    "UserPreferencesRecordHomeReservedWTShown": false,
    "UserPermissionsMarketingUser": false,
    "UserPreferencesHideBiggerPhotoCallout": false,
    "UserPreferencesDisableProfilePostEmail": false,
    "UserPreferencesShowPostalCodeToExternalUsers": false,
    "LastModifiedDate": "2022-11-09T17:14:57.000+0000",
    "ProfileId": "00e30000001ehhPAAQ",
    "Name": "Sam Carter",
    "UserPreferencesNewLightningReportRunPageEnabled": false,
    "UserPermissionsSFContentUser": false,
    "UserPreferencesShowTitleToExternalUsers": true,
    "UserPreferencesGlobalNavBarWTShown": false,
    "UserPreferencesShowCityToGuestUsers": false,
    "UserPreferencesRecordHomeSectionCollapseWTShown": false,
    "Alias": "scarter",
    "DefaultGroupNotificationFrequency": "N",
    "__NAME__": "00503000002XmQqAAK",
    "UserPreferencesShowPostalCodeToGuestUsers": false,
    "UserPreferencesGlobalNavGridMenuWTShown": false,
    "UserPreferencesHideChatterOnboardingSplash": false,
    "UserPreferencesSuppressEventSFXReminders": false,
    "UserPreferencesFavoritesShowTopFavorites": false,
    "UserPreferencesDisableMessageEmail": false,
    "UserPreferencesCacheDiagnostics": false,
    "SmallPhotoUrl": "https://example.force.com/profilephoto/005/T",
    "IsExtIndicatorVisible": false,
    "Email": "scarter@example.com",
    "UserPreferencesShowCountryToExternalUsers": false,
    "CreatedDate": "2022-11-09T17:14:57.000+0000",
    "UserPreferencesHideSecondChatterOnboardingSplash": false,
    "__PermissionSetIds__": [
        "0PS30000000LFR9GAO"
    ],
    "__GroupIds__": [
        "00G03000001V8p5EAC"
    ]
}' \
"https://localhost:8443/openidm/system/salesforce/User/00503000002XmQqAAK"
```

Feature licenses

The Salesforce connector can query all supported feature licenses and return details of individual feature licenses. The feature licenses available to your organization depend on your Salesforce instance. For more information, refer to View Your Organization's Feature Licenses in the Salesforce help site.

Configure supported feature licenses

To configure supported feature licenses, make the following changes to your connector configuration (`provisioner.openicf-salesforce.json`):

1. Add the supported feature licenses. For a complete list of licenses, refer to `supportedFeatureLicenses` in the Salesforce Connector Configuration.

   For example, to add `UserPermissionsOfflineUser` , `UserPermissionsMarketingUser` , and `UserPermissionsWorkDotComUserFeature` as supported feature licenses:

   ```
   "supportedFeatureLicenses": [
       "UserPermissionsOfflineUser",
       "UserPermissionsMarketingUser",
       "UserPermissionsWorkDotComUserFeature"
   ]
   ```

2. Add the `__FeatureLicenses__` property to the `User` object:

```
"User" : {
    ...
    "properties" : {
        "__NAME__" : {
            "type" : "string",
            "nativeName" : "__NAME__",
            "nativeType" : "string"
        },
        "Id" : {
            "type" : "string",
            "nativeName" : "Id",
            "nativeType" : "string"
        },
        "__FeatureLicenses__" : {
            "type" : "array",
            "items" : {
                "type" : "string",
                "nativeType" : "string"
            },
            "nativeName" : "__FeatureLicenses__",
            "nativeType" : "string",
            "absentIfEmpty" : true,
            "flags" : [
                "NOT_RETURNED_BY_DEFAULT"
            ]
        },
        ...
```

Get a user's feature licenses

When getting a user object, you must specifically request the system return the `__FeatureLicenses__` field. For example:

Request

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
"https://localhost:8443/openidm/system/salesforce/User?
_queryFilter=Alias+eq+"birdch"&_fields=_id,__FeatureLicenses__"
```

Return

```
{
  "result": [
    {
      "_id": "00503000004PDoeAAG",
      "__FeatureLicenses__": [
        "UserPermissionsOfflineUser"
      ]
    }
  ],
  "resultCount": 1,
  "pagedResultsCookie": null,
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

Create a user with feature licenses

To create a user object with associated feature licenses, include the `__FeatureLicenses__` field in the POST request. This field is an array of strings.

Request

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
--data '{
  "Username": "birdyjbirch@example.com",
  "LastName": "birdyjbirch",
  "Email": "birdyjbirch@example.com",
  "Alias": "birdch",
  "LocaleSidKey": "en_US",
  "LanguageLocaleKey": "en_US",
  "EmailEncodingKey": "UTF-8",
  "FirstName": "birdyjbirch",
  "TimeZoneSidKey": "America/Los_Angeles",
  "CommunityNickname": "birdyjbirch",
  "ProfileId": "00e30000001ehhPAAQ",
  "__FeatureLicenses__": ["UserPermissionsOfflineUser"]
}' \
"https://localhost:8443/openidm/system/salesforce/User?_action=create"
```

Return

```
{
  "_id": "00503000004PDoeAAG",
  "UserPreferencesShowStreetAddressToExternalUsers": false,
  "ForecastEnabled": false,
  "DigestFrequency": "D",
  "UserPreferencesPipelineViewHideHelpPopover": false,
  "UserPermissionsMarketingUser": false,
  "UserPreferencesPreviewCustomTheme": false,
  "UserPreferencesDisMentionsCommentEmail": false,
  "UserPreferencesShowMobilePhoneToGuestUsers": false,
  "UserPermissionsCallCenterAutoLogin": false,
  "UserPreferencesApexPagesDeveloperMode": false,
  "UserPreferencesShowCountryToGuestUsers": false,
  "Id": "00503000004PDoeAAG",
  "UserPreferencesHasCelebrationBadge": false,
  "UserPreferencesShowPostalCodeToExternalUsers": false,
  "IsExtIndicatorVisible": false,
  "BadgeText": "",
  "UserPreferencesDisableProfilePostEmail": false,
  "Username": "birdyjbirch@example.com",
  "UserPreferencesLightningExperiencePreferred": true,
  "UserPreferencesShowWorkPhoneToExternalUsers": false,
  "EmailPreferencesStayInTouchReminder": true,
  "UserPreferencesHideLightningMigrationModal": false,
  "UserPreferencesGlobalNavGridMenuWTShown": false,
  "EmailEncodingKey": "UTF-8",
  "UserPreferencesDisProfPostCommentEmail": false,
  "UserPreferencesHideCSNDesktopTask": false,
  "LastModifiedById": "00530000009hWLcAAM",
  "UserPreferencesShowStreetAddressToGuestUsers": false,
  "UserPreferencesHideBiggerPhotoCallout": false,
  "UserPreferencesShowWorkPhoneToGuestUsers": false,
  "UserPreferencesSuppressEventSFXReminders": false,
  "UserPreferencesShowManagerToExternalUsers": false,
  "UserPreferencesShowPostalCodeToGuestUsers": false,
  "UserPermissionsMobileUser": false,
  "CreatedDate": "2023-07-21T23:39:02.000+0000",
  "UserPreferencesFavoritesWTShown": false,
  "UserPreferencesDisableLikeEmail": true,
  "UserPreferencesShowMobilePhoneToExternalUsers": false,
  "UserPreferencesDisableSharePostEmail": false,
  "IsActive": true,
  "LastName": "birdyjbirch",
  "FirstName": "birdyjbirch",
  "DefaultGroupNotificationFrequency": "N",
  "UserPermissionsAvantgoUser": false,
  "UserPreferencesHideCSNGetChatterMobileTask": false,
  "CommunityNickname": "birdyjbirch",
  "UserPreferencesShowCityToGuestUsers": false,
  "UserPreferencesShowEmailToGuestUsers": false,
  "UserPreferencesDisableChangeCommentEmail": false,
  "UserPreferencesHideSfxWelcomeMat": true,
  "EmailPreferencesAutoBcc": true,
  "BannerPhotoUrl": "/profilephoto/005/B",
  "UserPreferencesCacheDiagnostics": false,
  "UserPreferencesHideEndUserOnboardingAssistantModal": false,
  "UserPreferencesShowCityToExternalUsers": false,
  "UserPermissionsSupportUser": false,
  "UserPreferencesShowProfilePicToGuestUsers": false,
  "SmallPhotoUrl": "https://example.force.com/profilephoto/005/T",
  "UserPreferencesShowTitleToGuestUsers": false,
  "SystemModstamp": "2023-07-21T23:39:02.000+0000",
  "UserPreferencesDisableLaterCommentEmail": false,
  "LocaleSidKey": "en_US",
  "UserPreferencesSuppressTaskSFXReminders": false,
  "UserPreferencesDisableBookmarkEmail": false,
  "__NAME__": "00503000004PDoeAAG",
  "UserPermissionsOfflineUser": true, (1)
  "UserPreferencesActivityRemindersPopup": true,
  "UserPreferencesEnableAutoSubForFeeds": false,
  "UserPreferencesShowEmailToExternalUsers": false,
  "UserPreferencesReminderSoundOff": false,
  "UserPreferencesCreateLEXAppsWTShown": false,
  "UserPreferencesDisableMentionsPostEmail": false,
  "UserPreferencesExcludeMailAppAttachments": false,
  "OutOfOfficeMessage": "",
  "SmallBannerPhotoUrl": "/profilephoto/005/D",
  "UserPreferencesDisableFileShareNotificationsForApi": false,
  "UserPreferencesEventRemindersCheckboxDefault": true,
  "UserPreferencesRecordHomeSectionCollapseWTShown": false,
  "MediumBannerPhotoUrl": "/profilephoto/005/E",
  "UserPreferencesUserDebugModePref": false,
  "UserPreferencesHideS1BrowserUI": false,
  "MediumPhotoUrl": "https://example.force.com/profilephoto/005/M",
  "UserPreferencesPathAssistantCollapsed": false,
  "Email": "birdyjbirch@example.com",
  "UserType": "Standard",
  "UserPreferencesDisableMessageEmail": false,
  "ReceivesInfoEmails": false,
  "EmailPreferencesAutoBccStayInTouch": false,
  "UserPreferencesShowManagerToGuestUsers": false,
  "IsProfilePhotoActive": false,
  "LastModifiedDate": "2023-07-21T23:39:02.000+0000",
  "UserPreferencesTaskRemindersCheckboxDefault": true,
  "UserPreferencesDisableEndorsementEmail": false,
  "FullPhotoUrl": "https://example.force.com/profilephoto/005/F",
  "Name": "birdyjbirch birdyjbirch",
  "UserPreferencesDisCommentAfterLikeEmail": false,
  "UserPreferencesShowFaxToExternalUsers": false,
  "UserPreferencesSortFeedByComment": true,
  "UserPreferencesRecordHomeReservedWTShown": false,
  "UserPreferencesGlobalNavBarWTShown": false,
  "TimeZoneSidKey": "America/Los_Angeles",
  "UserPreferencesShowFaxToGuestUsers": false,
  "ProfileId": "00e30000001ehhPAAQ",
  "UserPreferencesHideSecondChatterOnboardingSplash": false,
  "UserPreferencesShowTitleToExternalUsers": true,
  "UserPreferencesNewLightningReportRunPageEnabled": false,
  "UserPermissionsChatterAnswersUser": false,
  "LanguageLocaleKey": "en_US",
  "ReceivesAdminInfoEmails": false,
  "UserPreferencesShowStateToGuestUsers": false,
  "Alias": "birdch",
  "UserPreferencesPreviewLightning": false,
  "UserPreferencesHideChatterOnboardingSplash": false,
  "UserPreferencesShowStateToExternalUsers": false,
  "CreatedById": "00530000009hWLcAAM",
  "UserPreferencesDisableFollowersEmail": false,
  "UserPreferencesDisableAllFeedsEmail": false,
  "UserPermissionsSFContentUser": false,
  "UserPreferencesShowCountryToExternalUsers": false,
  "UserPreferencesFavoritesShowTopFavorites": false
}
```

1 `UserPermissionsOfflineUser` is now `true`.

Update a user’s feature licenses

You can update a user’s feature licenses through the API using a PUT call. The following request adds a feature license to `birdj`, a user with id `00503000004PDmsAAG`:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request PUT \
--data '{
  "UserPreferencesShowStreetAddressToExternalUsers": false,
  "ForecastEnabled": false,
  "DigestFrequency": "D",
  "UserPreferencesPipelineViewHideHelpPopover": false,
  "UserPermissionsMarketingUser": false,
  "LastReferencedDate": "2023-07-21T23:29:42.000+0000",
  "UserPreferencesPreviewCustomTheme": false,
  "UserPreferencesDisMentionsCommentEmail": false,
  "UserPreferencesShowMobilePhoneToGuestUsers": false,
  "UserPermissionsCallCenterAutoLogin": false,
  "UserPreferencesApexPagesDeveloperMode": false,
  "UserPreferencesShowCountryToGuestUsers": false,
  "Id": "00503000004PDmsAAG",
  "UserPreferencesHasCelebrationBadge": false,
  "UserPreferencesShowPostalCodeToExternalUsers": false,
  "IsExtIndicatorVisible": false,
  "BadgeText": "",
  "UserPreferencesDisableProfilePostEmail": false,
  "Username": "birdyjbird@example.com",
  "UserPreferencesLightningExperiencePreferred": true,
  "UserPreferencesShowWorkPhoneToExternalUsers": false,
  "EmailPreferencesStayInTouchReminder": true,
  "UserPreferencesHideLightningMigrationModal": false,
  "UserPreferencesGlobalNavGridMenuWTShown": false,
  "EmailEncodingKey": "UTF-8",
  "UserPreferencesDisProfPostCommentEmail": false,
  "UserPreferencesHideCSNDesktopTask": false,
  "LastModifiedById": "00530000009hWLcAAM",
  "UserPreferencesShowStreetAddressToGuestUsers": false,
  "UserPreferencesHideBiggerPhotoCallout": false,
  "UserPreferencesShowWorkPhoneToGuestUsers": false,
  "UserPreferencesSuppressEventSFXReminders": false,
  "UserPreferencesShowManagerToExternalUsers": false,
  "UserPreferencesShowPostalCodeToGuestUsers": false,
  "UserPermissionsMobileUser": false,
  "CreatedDate": "2023-07-21T23:29:41.000+0000",
  "UserPreferencesFavoritesWTShown": false,
  "UserPreferencesDisableLikeEmail": true,
  "UserPreferencesShowMobilePhoneToExternalUsers": false,
  "UserPreferencesDisableSharePostEmail": false,
  "IsActive": true,
  "LastName": "birdyjbird",
  "FirstName": "birdyjbird",
  "DefaultGroupNotificationFrequency": "N",
  "UserPermissionsAvantgoUser": false,
  "UserPreferencesHideCSNGetChatterMobileTask": false,
  "CommunityNickname": "birdyjbird",
  "UserPreferencesShowCityToGuestUsers": false,
  "UserPreferencesShowEmailToGuestUsers": false,
  "UserPreferencesDisableChangeCommentEmail": false,
  "UserPreferencesHideSfxWelcomeMat": true,
  "EmailPreferencesAutoBcc": true,
  "LastViewedDate": "2023-07-21T23:29:42.000+0000",
  "BannerPhotoUrl": "/profilephoto/005/B",
  "UserPreferencesCacheDiagnostics": false,
  "UserPreferencesHideEndUserOnboardingAssistantModal": false,
  "UserPreferencesShowCityToExternalUsers": false,
  "UserPermissionsSupportUser": false,
  "UserPreferencesShowProfilePicToGuestUsers": false,
  "SmallPhotoUrl": "https://example.force.com/profilephoto/005/T",
  "UserPreferencesShowTitleToGuestUsers": false,
  "SystemModstamp": "2023-07-21T23:29:41.000+0000",
  "UserPreferencesDisableLaterCommentEmail": false,
  "LocaleSidKey": "en_US",
  "UserPreferencesSuppressTaskSFXReminders": false,
  "UserPreferencesDisableBookmarkEmail": false,
  "__NAME__": "00503000004PDmsAAG",
  "UserPermissionsOfflineUser": false,
  "UserPreferencesActivityRemindersPopup": true,
  "UserPreferencesEnableAutoSubForFeeds": false,
  "UserPreferencesShowEmailToExternalUsers": false,
  "UserPreferencesReminderSoundOff": false,
  "UserPreferencesCreateLEXAppsWTShown": false,
  "UserPreferencesDisableMentionsPostEmail": false,
  "UserPreferencesExcludeMailAppAttachments": false,
  "OutOfOfficeMessage": "",
  "SmallBannerPhotoUrl": "/profilephoto/005/D",
  "UserPreferencesDisableFileShareNotificationsForApi": false,
  "UserPreferencesEventRemindersCheckboxDefault": true,
  "UserPreferencesRecordHomeSectionCollapseWTShown": false,
  "MediumBannerPhotoUrl": "/profilephoto/005/E",
  "UserPreferencesUserDebugModePref": false,
  "UserPreferencesHideS1BrowserUI": false,
  "MediumPhotoUrl": "https://example.force.com/profilephoto/005/M",
  "UserPreferencesPathAssistantCollapsed": false,
  "Email": "birdyjbird@example.com",
  "UserType": "Standard",
  "UserPreferencesDisableMessageEmail": false,
  "ReceivesInfoEmails": false,
  "EmailPreferencesAutoBccStayInTouch": false,
  "UserPreferencesShowManagerToGuestUsers": false,
  "IsProfilePhotoActive": false,
  "LastModifiedDate": "2023-07-21T23:29:41.000+0000",
  "UserPreferencesTaskRemindersCheckboxDefault": true,
  "UserPreferencesDisableEndorsementEmail": false,
  "FullPhotoUrl": "https://example.force.com/profilephoto/005/F",
  "Name": "birdyjbird birdyjbird",
  "UserPreferencesDisCommentAfterLikeEmail": false,
  "UserPreferencesShowFaxToExternalUsers": false,
  "UserPreferencesSortFeedByComment": true,
  "UserPreferencesRecordHomeReservedWTShown": false,
  "UserPreferencesGlobalNavBarWTShown": false,
  "TimeZoneSidKey": "America/Los_Angeles",
  "UserPreferencesShowFaxToGuestUsers": false,
  "ProfileId": "00e30000001ehhPAAQ",
  "UserPreferencesHideSecondChatterOnboardingSplash": false,
  "UserPreferencesShowTitleToExternalUsers": true,
  "UserPreferencesNewLightningReportRunPageEnabled": false,
  "UserPermissionsChatterAnswersUser": false,
  "LanguageLocaleKey": "en_US",
  "ReceivesAdminInfoEmails": false,
  "UserPreferencesShowStateToGuestUsers": false,
  "Alias": "birdj",
  "UserPreferencesPreviewLightning": false,
  "UserPreferencesHideChatterOnboardingSplash": false,
  "UserPreferencesShowStateToExternalUsers": false,
  "CreatedById": "00530000009hWLcAAM",
  "UserPreferencesDisableFollowersEmail": false,
  "UserPreferencesDisableAllFeedsEmail": false,
  "UserPermissionsSFContentUser": false,
  "UserPreferencesShowCountryToExternalUsers": false,
  "UserPreferencesFavoritesShowTopFavorites": false,
  "__FeatureLicenses__": [ (1)
    "UserPermissionsOfflineUser"
  ]
}' \
"https://localhost:8443/openidm/system/salesforce/User/00503000004PDmsAAG"
```

1 The `__FeatureLicenses__` array of strings to add to the user.

Query all supported feature licenses

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
"http://localhost:8080/openidm/system/salesforce/FeatureLicense?_queryFilter=true"
{
  "result": [
    {
      "_id": "UserPermissionsOfflineUser",
      "__NAME__": "UserPermissionsOfflineUser"
    },
    {
      "_id": "UserPermissionsMarketingUser",
      "__NAME__": "UserPermissionsMarketingUser"
    },
    {
      "_id": "UserPermissionsInteractionUser",
      "__NAME__": "UserPermissionsInteractionUser"
    }
  ],
  "resultCount": 3,
  "pagedResultsCookie": null,
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

Read a feature license

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
"http://localhost:8080/openidm/system/salesforce/FeatureLicense/UserPermissionsMarketingUser"
{
  "_id": "UserPermissionsMarketingUser",
  "__NAME__": "UserPermissionsMarketingUser"
}
```

OpenICF Interfaces Implemented by the Salesforce Connector

The Salesforce Connector implements the following OpenICF interfaces. For additional details, see ICF interfaces:

Create

Creates an object and its `uid`.

Delete

Deletes an object, referenced by its `uid`.

Schema

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

Script on Connector

Enables an application to run a script in the context of the connector.

Any script that runs on the connector has the following characteristics:

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

Search

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

Sync

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

Test

Tests the connector configuration.

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

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

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

Update

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

Salesforce Connector Configuration

The Salesforce Connector has the following configurable properties:

Basic Configuration Properties

Property	Type	Default	Encrypted(1)	Required(2)
`clientId`	`String`	`null`		✓ Yes
The client identifier.				
`clientSecret`	`GuardedString`	`null`	🔒 Yes	✓ Yes
The secure client secret for OAUTH.				
`grantType`	`String`	`refresh_token`		✓ Yes
The OAuth2 grant type to use (client_credentials or refresh_token).				
`refreshToken`	`GuardedString`	`null`	🔒 Yes	✕ No
The refresh token for the application. Required for refresh_token grant type.				
`loginUrl`	`String`	`https://login.salesforce.com/services/oauth2/token`		✓ Yes
The endpoint from which a new access token should be queried (https://login.salesforce.com/services/oauth2/token).				
`instanceUrl`	`String`	`null`		✓ Yes
The URL of the Salesforce instance (such as https://example-com.cs1.my.salesforce.com).				
`version`	`double`	`48.0`		✕ No
The Salesforce API version.				
`connectTimeout`	`long`	`120000`		✕ No
The maximum connection timeout.				
`proxyHost`	`String`	`null`		✕ No
The hostname of an HTTP proxy used between the connector and the Salesforce service provider. Mutually exclusive with proxyUri and uses http scheme.				
`proxyPort`	`Integer`	`3128`		✕ No
The proxy port number, if an HTTP proxy is used between the connector and the Salesforce service provider.				
`maximumConnections`	`int`	`10`		✕ No

Property	Type	Default	Encrypted[1]	Required[2]
The maximum size of the HTTP connection pool.				
`supportedObjectTypes`	`String[]`	`[]`		✕ No
Defines a list of Salesforce objects that will be used to dynamically build the provisioner schema.				
`proxyUri`	`String`	`null`		✕ No
The URI of an HTTP proxy that contains the scheme, host, and port number for that proxy. Mutually exclusive with proxyHost.				
`proxyUsername`	`String`	`null`		✕ No
The proxy username to use with a proxy that requires authentication.				
`proxyPassword`	`GuardedString`	`null`	🔒 Yes	✕ No
The proxy user password to use with a proxy that requires authentication.				

Property	Type	Default	Encrypted[1]	Required[2]
`supportedFeatureLicenses`	`String[]`	`['UserPermissionsChatterAnswersUser', 'UserPermissionsInteractionUser', 'UserPermissionsKnowledgeUser', 'UserPermissionsLiveAgentUser', 'UserPermissionsMarketingUser', 'UserPermissionsOfflineUser', 'UserPermissionsSFContentUser', 'UserPermissionsSupportUser', 'UserPermissionsSiteforceContributorUser', 'UserPermissionsSiteforcePublisherUser', 'UserPermissionsWorkDotComUserFeature']`		✕ No
Defines a list of Salesforce feature licenses that can be assigned.				
`initialSyncTokenOffset`	`long`	`0`		✕ No
Defines the number of hours to offset the initial sync token. This is used to ensure that the initial sync does not miss any changes that occurred just before the sync started.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

SAP connector

The SAP connector is an implementation of the Scripted Groovy connector toolkit that connects to any SAP system using the SAP JCo Java libraries. This topic describes how to install and configure the scripted SAP connector and how to test the sample scripts bundled with the connector.

> **Note**
>
> You can configure the SAP connector to work with either SAP HR or SAP ERP systems.

The sample scripts illustrate the following scenarios:

- Synchronization of users between an SAP HR module and IDM
- Synchronization of users between IDM and an SAP (R/3) system

Install the SAP connector

> **Tip**
>
> To check for an Advanced Identity Cloud application for this connector, refer to:
>
> - Application management
> - App catalog

You can download any connector from Backstage, but some are included in the default deployment for Advanced Identity Cloud, IDM, or RCS. When using an included connector, you can skip installing it and move directly to configuration.

Connector included in default deployment

Connector	IDM	RCS
SAP	✕ No	✓ Yes

Download the connector .jar file from Backstage.

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

  ```
  mv ~/Downloads/sap-connector-1.5.20.28.jar /path/to/openidm/connectors/
  ```

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

Download the connector dependencies.

- The SAP connector requires the SAP Java Connector (JCo) libraries, version 3.0.12 or later. Ping distributes the SAP connector without these JCo libraries. Before you can use the SAP connector, you must obtain the JCo libraries that correspond to your architecture.

  Copy the required SAP JCo libraries to the `/path/to/openidm/lib` directory. For example:

```
cp sapjco3.jar /path/to/openidm/lib
cp libsapjco3.so /path/to/openidm/lib
```

Change your IDM logging configuration to log messages from the SAP connector. By default, IDM logs nothing for the SAP connector. To troubleshoot any issues with the connector, set the following properties in your project's `conf/logging.properties` file:

```
# SAP Connector Logging
org.forgerock.openicf.connectors.sap.level=FINER
scripts.sap.r3.level=FINER
scripts.sap.hr.level=FINER
scripts.sap.level=FINER
```

Configure connection pooling

The SAP connector supports HTTP pooling, which can substantially improve the performance of the connector. Learn more about the basic connection pooling configuration and different pooling mechanisms described in Connection pooling configuration.

Using the SAP connector with an SAP HR system

The SAP HR sample scripts let you manage the email address and global employee UID of records in an SAP HR system.

The following sections explain how to configure IDM to use these sample scripts, how to test the connection to the SAP HR system, and how to update user records.

Setting up IDM for the SAP HR samples

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

   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, port, and optional password of the SAP router, if applicable. The syntax is `/H/host/S/port/W/optionalPassword`. For example:

```
/H/203.0.113.0/S/3299/W/48npb_hg815.77rr62.hdj
```

`poolCapacity`

The maximum number of idle connections kept open by the destination. If there is no connection pooling, set this to `0`. The default value is `1`.

For optimum performance, set this value to an integer between `5` and `10`.

2. The connector bundles a number of sample Groovy scripts:

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

   If necessary, you can customize these scripts to suit your deployment by extracting them from the connector JAR and updating the connector configuration to point to the new file path.

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

```
"testScriptFileName" : "TestSAP.groovy",
"searchScriptFileName" : "hr/SearchSAPHR.groovy",
"updateScriptFileName" : "hr/UpdateSAPHR.groovy",
"schemaScriptFileName" : "hr/SchemaSAPHR.groovy",
```

   You must place the `EmplComm.groovy` in the same location as the Search, Update, and Schema scripts.

> **Important**
>
> The Groovy scripts belong to a specific package. The parent directory where the scripts are located must be the same as the package name. So the `TestSAP.groovy` script must be under a `scripts/sap` directory (because it belongs to the `scripts/sap` package) and the remaining HR scripts must be under a `scripts/sap/hr` directory (because they belong to the `hr` package).

Testing the connection to the SAP HR system

1. Start IDM with the configuration for your SAP connector project.

   This procedure assumes that the configuration is in the default `path/to/openidm` directory. If your SAP project is in a different directory, use the `-p` option with the startup command to point to that directory:

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

2. Test that the connector has been configured correctly and that the SAP HR system can be reached:

   ```
   curl \
   --header "X-OpenIDM-Username: openidm-admin" \
   --header "X-OpenIDM-Password: openidm-admin" \
   --header "Accept-API-Version: resource=1.0" \
   --request POST \
   "http://localhost:8080/openidm/system/saphr/?_action=test"
   {
     "name" : "saphr",
     "enabled" : true,
     "config" : "config/provisioner.openicf/saphr2",
     "objectTypes" : [ "__ALL__", "employee" ],
     "connectorRef" : {
       "connectorName" : "org.forgerock.openicf.connectors.sap.SapConnector",
       "bundleName" : "org.forgerock.openicf.connectors.sap-connector",
       "bundleVersion" : "[1.5.0.0,1.6.0.0)"
     },
     "displayName" : "Sap Connector",
     "ok" : true
   }
   ```

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

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--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" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
"http://localhost:8080/openidm/system/saphr/employee/55099307"
{
  "_id" : "55099307",
  "PERSONAL_DATA" : {
    "PERNO" : "55099307",
    "INFOTYPE" : "0002",
    "TO_DATE" : "Fri Dec 31 00:00:00 CET 9999",
    "FROM_DATE" : "Tue Mar 30 00:00:00 CET 1954",
    "SEQNO" : "000",
    "CH_ON" : "Thu Mar 27 00:00:00 CET 2003",
    "CHANGED_BY" : "MAYROCK",
    "LAST_NAME" : "Gonzales",
    "FIRSTNAME" : "Maria",
    "NAME_FORM" : "00",
    "FORMOFADR" : "2",
    "GENDER" : "2",
    "BIRTHDATE" : "Tue Mar 30 00:00:00 CET 1954",
    "LANGU" : "D",
    "NO_O_CHLDR" : "0",
    "BIRTHYEAR" : "1954",
    "BIRTHMONTH" : "03",
    "BIRTHDAY" : "30",
    "LASTNAME_M" : "GONZALES",
    "FSTNAME_M" : "MARIA"
  },
  ...
}
```

Using the SAP connector to manage employee information (SAP HR)

The following sample commands show how you can use the SAP connector to manage the email account of user Maria Gonzales, retrieved in the previous step. Management of the global UID (`SYS-UNAME`) works in the same way.

1. Check if Maria Gonzales already has an email account on the SAP HR system by filtering a query on her user account for the `EMAIL` field:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
"http://localhost:8080/openidm/system/saphr/employee/55099307?_fields=EMAIL"
{
  "_id" : "55099307",
}
```

No email account is found for Maria Gonzales.

2. Add an email account by sending a PUT request. The JSON payload should include the email address as the value of the `ID` property:

   ```
   curl \
   --header "X-OpenIDM-Username: openidm-admin" \
   --header "X-OpenIDM-Password: openidm-admin" \
   --header "Accept-API-Version: resource=1.0" \
   --header "Content-Type: application/json" \
   --request PUT \
   --data '{
     "EMAIL": { "ID": "maria.gonzales@example.com" }
   }' \
   "http://localhost:8080/openidm/system/saphr/employee/55099307"
   {
     "_id" : "55099307",
     "EMAIL" : [ {
       "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 "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request PUT \
--data '{
  "EMAIL": {
    "ID": "maria.gonzales-admin@example.com",
    "RECORDNR" : "000",
    "VALIDEND" : "99991231",
    "VALIDBEGIN" : "20000101"
  }
}' \
"http://localhost:8080/openidm/system/saphr/employee/55099307"
```

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

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

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request PUT \
--data '{
  "EMAIL": {
    "ID": "maria.gonzales-admin@example.com",
    "RECORDNR" : "000",
    "VALIDEND" : "99991231",
    "VALIDBEGIN" : "20000101",
    "DELIMIT_DATE": "20161231"
  }
}' \
"http://localhost:8080/openidm/system/saphr/employee/55099307"
```

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

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

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request PUT \
--data '{
  "EMAIL": {
    "RECORDNR" : "000",
    "VALIDEND" : "99991231",
    "VALIDBEGIN" : "20000101"
  }
}' \
"http://localhost:8080/openidm/system/saphr/employee/55099307"
```

Using the SAP connector to manage SAP Basis System (R/3) users

The SAP connector enables you to perform the following operations on SAP system user accounts:

- List all users
- List all activity groups (roles)
- List all user profiles
- List all user companies
- List all user groups
- 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

Setting up IDM for the SAP R/3 samples

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

   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/W/optionalPassword`. For example:

```
/H/203.0.113.0/S/3299/W/48npb_hg815.77rr62.hdj
```

`poolCapacity`

The maximum number of idle connections kept open by the destination. If there is no connection pooling, set this to `0`. The default value is `1`.

For optimum performance, set this value to an integer between `5` and `10`.

2. The connector bundles a number of sample Groovy scripts:

   - `TestSAP.groovy`
   - `SearchSAPR3.groovy`
   - `CreateSAPR3.groovy`
   - `UpdateSAPR3.groovy`
   - `DeleteSAPR3.groovy`
   - `SyncSAPR3.groovy`
   - `SchemaSAPR3.groovy`

- `ListR3Objects.groovy`
- `R3User.groovy`
- `R3UserActivityGroup.groovy`
- `R3UserAddress.groovy`
- `R3UserGroup.groovy`
- `R3UserLogonData.groovy`
- `R3UserProfile.groovy`
- `R3Config.groovy`
- `ListTablesR3Objects.groovy`
- `QueryTable.groovy`

If necessary, you can customize these scripts to suit your deployment by extracting them from the connector JAR and updating the connector configuration to point to the new file path.

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

```
"testScriptFileName" : "TestSAP.groovy",
"searchScriptFileName" : "r3/SearchSAPR3.groovy",
"createScriptFileName" : "r3/CreateSAPR3.groovy",
"updateScriptFileName" : "r3/UpdateSAPR3.groovy",
"deleteScriptFileName" : "r3/DeleteSAPR3.groovy",
"syncScriptFileName" : "r3/SyncSAPR3.groovy",
"schemaScriptFileName" : "r3/SchemaSAPR3.groovy",
```

> **Important**
>
> The Groovy scripts belong to a specific package. The parent directory where the scripts are located must be the same as the package name. So the `TestSAP.groovy` script must be under a `scripts/sap` directory (because it belongs to the `scripts/sap` package) and the R/3 scripts must be under a `scripts/sap/r3` directory (because they belong to the `r3` package).

Testing the connection to the SAP R/3 system

1. Start IDM with the configuration for your SAP R/3 project.

   This procedure assumes that the configuration is in the default `path/to/openidm` directory. If your SAP project is in a different directory, use the `-p` option with the startup command to point to that directory:

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

2. Test that the connector has been configured correctly and that the SAP R/3 system can be reached:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system/mysap/?_action=test"
{
  "name": "mysap",
  "enabled": true,
  "config": "config/provisioner.openicf/mysap",
  "objectTypes": [
    "__ALL__",
    "user",
    "activity_group",
    "company",
    "profile",
    "group"
  ],
  "connectorRef": {
    "connectorName": "org.forgerock.openicf.connectors.sap.SapConnector",
    "bundleName": "org.forgerock.openicf.connectors.sap-connector",
    "bundleVersion": "[1.5.0.0,1.6.0.0)"
  },
  "displayName": "Sap Connector",
  "ok": true
}
```

Using the SAP connector to manage SAP R/3 users

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

Listing the users in the SAP system

The following command returns a list of the existing users in the SAP system, with their IDs:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
"http://localhost:8080/openidm/system/mysap/user?_queryId=query-all-ids"
{
  "result": [
    {
      "_id": "BJENSEN",
      "__NAME__": "BJENSEN"
    },
    {
      "_id": "DDIC",
      "__NAME__": "DDIC"
    },
    ...
    {
      "_id": "USER4",
      "__NAME__": "USER4"
    },
    {
      "_id": "USER6",
      "__NAME__": "USER6"
    },
    {
      "_id": "USER7",
      "__NAME__": "USER7"
    }
  ],
  "resultCount": 9,
  "pagedResultsCookie": null,
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

Obtaining the details of an SAP user

The following command uses the SAP connector to obtain a user’s details from a target SAP system:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
"http://localhost:8080/openidm/system/mysap/user/BJENSEN"
{
  "__NAME__": "BJENSEN",
  "__ENABLE__": true,
  "__ENABLE_DATE__": "2015-09-01",
  "__DISABLE_DATE__": "2016-09-01",
  "__LOCK_OUT__": false,
  "ADDTEL": [
    {
      "COUNTRY": "DE",
      "TELEPHONE": "19851444",
      ...
    },
    ...
  ],
  "PROFILES": [
    "T_ALM_CONF",
    ...
  ],
  "ISLOCKED": {
    "WRNG_LOGON": "U",
    ...
  },
  "ACTIVITYGROUPS": [
    {
      "AGR_NAME": "MW_ADMIN",
      "FROM_DAT": "2015-07-15",
      "TO_DAT": "9999-12-31"
    },
    ...
  ],
  "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",
    ...
  },
  "GROUPS": {
    "USERGROUP": "SUPER"
    ...
  },
  "_id": "BJENSEN"
}
```

When using Central User Administration (CUA), the system also returns the `SUBSYSTEMS` attribute. Additionally, `PROFILES` and `ACTIVITYGROUPS` have a different definition:

```
"SYSTEMS": [
  "TE9CLNT200",
  "TE9CLNT300",
  \...
],
"PROFILES": [
  {
    "BAPIPROF": "T_ALM_CONF",
    "SUBSYSTEM: "TE9CLNT200"
  },
  {
    "BAPIPROF": "T_ALM_CONF",
    "SUBSYSTEM: "TE9CLNT300"
  },
  \...
],
"ACTIVITYGROUPS": [
  {
    "AGR_NAME": "MW_ADMIN",
    "FROM_DAT": "2015-07-15",
    "TO_DAT": "9999-12-31",
    "SUBSYSTEM": "TE9CLNT200"
  },
  {
    "AGR_NAME": "MW_ADMIN",
    "FROM_DAT": "2015-07-15",
    "TO_DAT": "9999-12-31",
    "SUBSYSTEM": "TE9CLNT300"
  },
  \...
]
...
```

In addition to the standard user attributes, the GET request returns the following ICF operational attributes:

- `__ENABLE__` - indicates whether the account is enabled, based on the value of the `LOGONDATA` attribute
- `__ENABLE_DATE__` - set to the value of `LOGONDATA/GLTGV` (date from which the user account is valid)
- `__DISABLE_DATE__` - set to the value of `LOGONDATA/GLTGB` (date to which the user account is valid)

- `__LOCK_OUT__` - indicates whether the account is locked

Creating SAP user accounts

To create a user, you must supply *at least* a username and password. If you do not provide a lastname, the connector uses the value of the username.

The following command creates a new SAP user, `SCARTER`:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request POST \
--data '{
  "__NAME__" : "SCARTER",
  "__PASSWORD__": "Passw0rd"
}' \
"http://localhost:8080/openidm/system/mysap/user/?_action=create"
{
  "_id": "SCARTER",
  "COMPANY": {
    "COMPANY": "SAP AG"
  },
  "__LOCK_OUT__": false,
  "ADDRESS": {
    ...
  },
  "__NAME__": "SCARTER",
  "LASTMODIFIED": {
    "MODDATE": "2016-04-20",
    "MODTIME": "04:14:29"
  },
  "UCLASS": {
    "COUNTRY_SURCHARGE": "0",
    "SUBSTITUTE_FROM": "0000-00-00",
    "SUBSTITUTE_UNTIL": "0000-00-00"
  },
  "__ENABLE__": true,
  "DEFAULTS": {
    "SPDB": "H",
    "SPDA": "K",
    "DATFM": "1",
    "TIMEFM": "0"
  },
  "LOGONDATA": {
    ...
  },
  "ISLOCKED": {
    "WRNG_LOGON": "U",
    "LOCAL_LOCK": "U",
    "GLOB_LOCK": "U",
    "NO_USER_PW": "U"
  }
}
```

The SAP account that is created is valid and enabled, but the password is expired by default. To log in to the SAP system, the newly created user must first provide a new password.

To create a user with a valid (non-expired) password, include the `__PASSWORD_EXPIRED__` attribute in the JSON payload with a value of `false`. For example:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request POST \
--data '{
  "__NAME__" : "SCARTER",
  "__PASSWORD__": "Passw0rd",
  "__PASSWORD_EXPIRED__": false
}' \
"http://localhost:8080/openidm/system/mysap/user/?_action=create"
```

To create an account that is locked by default, include the `__LOCK_OUT__` attribute in the JSON payload with a value of `true`. For example:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request POST \
--data '{
  "__NAME__" : "SCARTER",
  "__PASSWORD__": "Passw0rd",
  "__LOCK_OUT__": true
}' \
"http://localhost:8080/openidm/system/mysap/user/?_action=create"
{
  "__NAME__": "SCARTER",
  "__ENABLE__": false,
  "__LOCK_OUT__": true,
  "LOGONDATA": {
    "GLTGV": "0000-00-00",
    "GLTGB": "0000-00-00",
    "USTYP": "A",
    "LTIME": "00:00:00"
  },
  "LASTMODIFIED": {
    "MODDATE": "2015-10-01",
    "MODTIME": "15:25:18"
  },
  "ISLOCKED": {
    "WRNG_LOGON": "U",
    "LOCAL_LOCK": "L",     (1)
    "GLOB_LOCK": "U",     (2)
    "NO_USER_PW": "U"
  },
  ...
}
```

1 "L" indicates that the user is locked on the local system.

2 On CUA Systems "GLOB_LOCK" will be marked as locked instead of "LOCAL_LOCK".

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 (Roles)
- `ADDTEL` - telephone numbers assigned to the user
- `GROUPS` - groups assigned to the user
- `SYSTEMS` - subsystems assigned to the user (Only on CUA systems)

In addition, the SAP connector supports the following ICF operational attributes for CREATE requests:

- `LOCK_OUT`
- `PASSWORD`
- `PASSWORD_EXPIRED`

The following example creates a user, KVAUGHAN, with all the standard attributes:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request POST \
--data '{
  "__NAME__" : "KVAUGHAN",
  "__PASSWORD__": "Passw0rd",
  "__PASSWORD_EXPIRED__": false,
  "LOGONDATA": {
    "GLTGV": "2016-04-01",
    "GLTGB": "2016-12-01",
    "USTYP": "A"
  },
  "ADDRESS": {
    "FIRSTNAME": "Katie",
    "LASTNAME": "Vaughan",
    "TEL1_NUMBR": "33297603177",
    "E_MAIL": "katie.vaughan@example.com",
    "FUNCTION": "Test User"
  },
  "COMPANY": {
    "COMPANY": "EXAMPLE.COM"
  },
  "ALIAS": {
    "USERALIAS": "KVAUGHAN"
  }
}' \
"http://localhost:8080/openidm/system/mysap/user/?_action=create"
{
  "_id": "KVAUGHAN",
  "ADDRESS": {
    "PERS_NO": "0000010923",
    "ADDR_NO": "0000010765",
    "FIRSTNAME": "Katie",
    "LASTNAME": "Vaughan",
    "FULLNAME": "Katie Vaughan",
    ...
    "E_MAIL": "katie.vaughan@example.com",
    "LANGU_CR_P": "E",
    "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",      (1)
  "DEFAULTS": {
    "SPDB": "H",
    "SPDA": "K",
    "DATFM": "1",
    "TIMEFM": "0"
  },
  "__DISABLE_DATE__": "2016-12-01"     (2)
}
```

1 Value of LOGONDATA/GLTGV.

2 Value of LOGONDATA/GLTGB.

Updating SAP user accounts

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

Locking and unlocking an account

To lock or unlock a user’s account, send a PUT request, and set the value of the user’s `__LOCK_OUT__` attribute to `true` or `false`.

The following example locks user KVAUGHAN’s account:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--header "If-Match: *" \
--request PUT \
--data '{
  "__LOCK_OUT__": true
}' \
"http://localhost:8080/openidm/system/mysap/user/KVAUGHAN"
```

The following example unlocks KVAUGHAN's account:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--header "If-Match: *" \
--request PUT \
--data '{
  "__LOCK_OUT__": false
}' \
"http://localhost:8080/openidm/system/mysap/user/KVAUGHAN"
```

Updating the standard attributes of a user's account

To update a user's standard attributes, send a PUT request to the user ID. The JSON payload must respect the structure for each attribute, as indicated in Schema used by the SAP connector for user accounts.

The following command updates the `ADDRESS` attribute of user KVAUGHAN:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--header "If-Match: *" \
--request PUT \
--data '{
  "ADDRESS": {
    "FIRSTNAME": "Katie",
    "LASTNAME": "Vaughan",
    "FULLNAME": "Katie Vaughan",
    "FUNCTION": "Administrator",
    "TITLE": "Company",
    "NAME": "EXAMPLE.COM",
    "CITY": "San Francisco",
    "POSTL_COD1": "94105",
    "STREET": "Sacramento St",
    "HOUSE_NO": "2912",
    "COUNTRY": "US",
    "COUNTRYISO": "US",
    "LANGU": "E",
    "LANGU_ISO": "EN",
    "REGION": "CA",
    "TIME_ZONE": "PST",
    "TEL1_NUMBR": "33297603177",
    "E_MAIL": "katie.vaughan@example.com",
    "LANGU_CR_P": "E",
    "LANGUCPISO": "EN"
  }
}' \
"http://localhost:8080/openidm/system/mysap/user/KVAUGHAN"
```

Resetting a user’s password

To reset the user’s password, provide the new password as the value of the `__PASSWORD__` attribute in a PUT request. The following command resets KVAUGHAN’s password to `MyPassw0rd`:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--header "If-Match: *" \
--request PUT \
--data '{
  "__PASSWORD__": "MyPassw0rd"
}' \
"http://localhost:8080/openidm/system/mysap/user/KVAUGHAN"
```

Note that unless you set the `__PASSWORD_EXPIRED__` attribute to `false`, the user will be required to reset her password the next time she logs into the SAP system.

The following command resets KVAUGHAN's password to `MyPassw0rd`, and ensures that she does not have to reset her password the next time she logs in:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request PUT \
--data '{
  "__PASSWORD__": "MyPassw0rd",
  "__PASSWORD_EXPIRED__": false
}'
"http://localhost:8080/openidm/system/mysap/user/KVAUGHAN"
```

Deleting user accounts

To delete a user account, send a DELETE request to the user ID. The following example deletes KVAUGHAN:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request DELETE \
"http://localhost:8080/openidm/system/mysap/user/KVAUGHAN"
```

The command returns the complete user object that was deleted.

Get the latest changes with LiveSync

The following example updates the user's data since the last synchronization:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system?_action=liveSync&source=system/mysap/__ACCOUNT__"
{
  "connectorData": {
    "nativeType": "string",
    "syncToken": "20230707160932"
  },
  "_rev": "b69ca221-6610-484a-983f-142e8544e519-101",
  "_id": "SYSTEMMYSAP+ACCOUNT+"
}
```

Managing user profiles

An SAP system uses *profiles* to manage authorization. The following examples demonstrate how to add, change, and remove a user’s profiles.

Creating a user with one or more profiles

Profiles are added as an array of one or more strings.

The following command creates a user BJENSEN with the system administrator profile (`S_A.SYSTEM`):

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request POST \
--data '{
  "__NAME__" : "BJENSEN",
  "__PASSWORD__": "Passw0rd",
  "__PASSWORD_EXPIRED__": false,
  "PROFILES": [
    "S_A.SYSTEM"
  ]
}' \
"http://localhost:8080/openidm/system/mysap/user/?_action=create"
{
  "_id": "BJENSEN",
  "PROFILES": [
    "S_A.SYSTEM"
  ],
  ...
  "__NAME__": "BJENSEN"
}
```

Note that the additional information regarding that profile is added to the user account automatically.

Updating a user’s profiles

To update a user’s profiles, send a PUT request to the user’s ID, specifying the new profiles as an array of values for the `PROFILES` attribute. The values provided in the PUT request will replace the current profiles, so you must include the existing profiles in the request.

The following example adds the `SAP_ALL` profile to user BJENSEN’s account:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--header "If-Match: *" \
--request PUT \
--data '{
  "PROFILES": [
    "S_A.SYSTEM",
    "SAP_ALL"
  ]
}' \
"http://localhost:8080/openidm/system/mysap/user/BJENSEN"
{
  "_id": "BJENSEN",
  "PROFILES": [
    "SAP_ALL",
    "S_A.SYSTEM"
  ],
  ...
  "__NAME__": "BJENSEN"
}
```

Removing all profiles from a user account

To remove all profiles from a user's account, update the account with an empty array. The following example removes all profiles from BJENSEN's account:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--header "If-Match: *" \
--request PUT \
--data '{
  "PROFILES": []
}' \
"http://localhost:8080/openidm/system/mysap/user/BJENSEN"
{
  "_id": "BJENSEN",
  ...
  "__NAME__": "BJENSEN"
}
```

The output shows no `PROFILES` attribute, as this attribute is now empty for this user.

Note

When using CUA systems, each `PROFILE` has to be specified with a `SUBSYSTEM` like the next example:

```
"PROFILES": [
  {
    "BAPIPROF": "SAP_ALL",
    "SUBSYSTEM: "TE9CLNT200"
  },
  {
    "BAPIPROF": "S_A.SYSTEM",
    "SUBSYSTEM: "TE9CLNT200"
  },
],
...
```

Managing user roles

SAP user roles (or *activity groups*) are an alternative mechanism to grant authorization to an SAP system. Essentially, a role encapsulates a set of one or more profiles.

Roles can be granted with *temporal constraints*, that is, a period during which the role is valid. If no temporal constraints are specified, the SAP connector sets the FROM date to the current date and the TO date to 9999-12-31.

Creating a user with one or more profiles

Roles are added as an array of one or more objects.

The following command creates a user SCARTER, with two roles: `SAP_AUDITOR_SA_CCM_USR` and `SAP_ALM_ADMINISTRATOR`. The auditor role includes a temporal constraint, which is only valid from May 1st, 2016 to April 30th, 2017. The format of the temporal constraint is `YYYY-mm-dd`:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request POST \
--data '{
  "__NAME__" : "SCARTER",
  "__PASSWORD__": "Passw0rd",
  "__PASSWORD_EXPIRED__": false,
  "ACTIVITYGROUPS": [
    {
      "AGR_NAME": "SAP_AUDITOR_SA_CCM_USR",
      "FROM_DAT": "2016-05-01",
      "TO_DAT": "2017-04-30"
    },
    {
      "AGR_NAME": "SAP_ALM_ADMINISTRATOR"
    }
  ]
}' \
"http://localhost:8080/openidm/system/mysap/user/?_action=create"
{
  "_id": "SCARTER",
  "PROFILES": [
    "T_ALM_CONF"
  ],
  ...
  "ACTIVITYGROUPS": [
    {
      "AGR_NAME": "SAP_ALM_ADMINISTRATOR",
      "FROM_DAT": "2016-04-20",
      "TO_DAT": "9999-12-31"
    },
    {
      "AGR_NAME": "SAP_AUDITOR_SA_CCM_USR",
      "FROM_DAT": "2016-05-01",
      "TO_DAT": "2017-04-30"
    }
  ],
  "__NAME__": "SCARTER"
}
```

When a role is granted, the corresponding profiles are attached to the user account automatically.

Updating a user’s roles

To update a user’s roles, send a PUT request to the user’s ID specifying the new roles as an array of values of the `ACTIVITYGROUPS` attribute. The values provided in the PUT request will replace the current `ACTIVITYGROUPS`.

The following example removes the `SAP_AUDITOR_SA_CCM_USR` role and changes the temporal constraints on the `SAP_ALM_ADMINISTRATOR` role for SCARTER’s account:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--header "If-Match: *" \
--request PUT \
--data '{
  "ACTIVITYGROUPS": [
    {
      "AGR_NAME": "SAP_ALM_ADMINISTRATOR",
      "FROM_DAT": "2015-06-02",
      "TO_DAT": "2016-06-02"
    }
  ]
}' \
"http://localhost:8080/openidm/system/mysap/user/SCARTER"
{
  "_id": "SCARTER",
  "PROFILES": [
    "T_ALM_CONF"
  ],
  ...
  "ACTIVITYGROUPS": [
    {
      "AGR_NAME": "SAP_ALM_ADMINISTRATOR",
      "FROM_DAT": "2015-06-02",
      "TO_DAT": "2016-06-02"
    }
  ],
  "__NAME__": "SCARTER"
}
```

Removing all roles from a user account

To remove all roles from a user’s account, update the value of the `ACTIVITYGROUPS` attribute with an empty array. The following example removes all roles from SCARTER’s account:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--header "If-Match: *" \
--request PUT \
--data '{
  "ACTIVITYGROUPS": []
}' \
"http://localhost:8080/openidm/system/mysap/user/SCARTER"
{
  "_id": "SCARTER",
  ...
  "LASTMODIFIED": {
    "MODDATE": "2016-04-21",
    "MODTIME": "04:27:00"
  },
  "__NAME__": "SCARTER"
}
```

The output shows no `ACTIVITYGROUPS` attribute, as this attribute is now empty.

> **Note**
>
> On CUA systems, each `ACTIVITYGROUP` has to be specified with a `SUBSYSTEM` like the next example:
>
> ```
> "ACTIVITYGROUPS": [
>   {
>     "AGR_NAME": "SAP_ALM_ADMINISTRATOR",
>     "SUBSYSTEM": "T9CLNT200",
>     "FROM_DAT": "2015-06-02",
>     "TO_DAT": "2016-06-02"
>   }
> ],
> ...
> ```

Managing user groups

One of the Primary uses of user groups is to sort users into logical groups. This allows users to be categorized in a method that is not dependent on roles, AG’s, Responsibilities, Profiles, and so on.

User Groups also allow segregation of user maintenance, this is especially useful in a large organisation as you can control who your user admin team can maintain - an example would be giving a team leader the authority to change passwords for users in their team.

Creating a user with one or more groups

You add groups as an array of one or more strings.

The following command creates a user SCARTER, with two groups, `SUPER` and `TEST_GROUP`:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request POST \
--data '{
  "__NAME__" : "SCARTER",
  "__PASSWORD__": "Passw0rd",
  "__PASSWORD_EXPIRED__": false,
  "GROUPS": [
    "SUPER",
    "TEST_GROUP"
  ]
}' \
"http://localhost:8080/openidm/system/mysap/user/?_action=create"
{
  "_id": "SCARTER",
  "GROUPS": [
    "SUPER",
    "TEST_GROUP"
  ],
  ...
  "__NAME__": "SCARTER"
}
```

Updating a user’s groups

To update a user’s groups, send a PUT request to the user’s ID, specifying the new groups as an array of values of the `GROUPS` attribute. The values provided in the PUT request replaces the current `GROUPS`.

The following example removes the `SUPER` group for SCARTER’s account:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--header "If-Match: *" \
--request PUT \
--data '{
  "GROUPS": [
    "TEST_GROUP"
  ]
}' \
"http://localhost:8080/openidm/system/mysap/user/SCARTER"
{
  "_id": "SCARTER",
  ...
  "GROUPS": [
    "TEST_GROUP"
  ],
  "__NAME__": "SCARTER"
}
```

Removing all groups from a user account

To remove all groups from a user's account, update the value of the `GROUPS` attribute with an empty array. The following example removes all groups from SCARTER's account:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--header "If-Match: *" \
--request PUT \
--data '{
  "GROUPS": []
}' \
"http://localhost:8080/openidm/system/mysap/user/SCARTER"
{
  "_id": "SCARTER",
  ...
  "LASTMODIFIED": {
    "MODDATE": "2016-04-21",
    "MODTIME": "04:27:00"
  },
  "__NAME__": "SCARTER"
}
```

The output shows no `GROUPS` attribute, as this attribute is now empty.

Configuring the SAP connector for OpenIDM user interface for R3

Create a connector configuration using the IDM admin UI:

1. From the navigation bar, click **Configure > Connectors**.
2. On the **Connectors** page, click **New Connector**.
3. On the **New Connector** page, type a **Connector Name**.
4. From the **Connector Type** drop-down list, select **SAP Connector - 1.5.20.28**.
5. Complete the **Base Connector Details**.

> **Tip**
>
> For a list of all configuration properties, refer to SAP Connector Configuration

6. Click **Save**.

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

Refer to this procedure to create a connector configuration over REST.

To run the connector, you need the following minimum configuration:

- `Gateway Host` : Host name or IP address of the SAP Gateway Server
- `Application Server Host` : Host name or IP address of the SAP Application Server (or SAP Netweaver Gateway)
- `Client` : Name of the SAP logon client
- `Language` : Language of the remote SAP System
- `SAP Router`
- `User` : Logon used for authenticating on the remote SAP System
- `Password` : Password of the previously specified logon, used for authenticating on the remote SAP System
- `Scripts Root` : Full Path To Script Files
- `Search Script`
- `Create Script`
- `Update Script`
- `Delete Script`
- `Sync Script`
- `Test Script`
- `Schema Script`

You must add or edit your Object Types including the following four objects with the listed minimum properties:

PROPERTY NAME	TYPE	NATIVE TYPE	REQUIRED	ITEM TYPE	ITEM NATIVE TYPE
__NAME__	String	String	YES	-	-
__PASSWORD__	String	JAVA_TYPE_GUARDEDSTRING	YES	-	-
ALIAS	String	String	YES	-	-
LOGONDATA	Object	Object	NO	-	-
COMPANY	Object	Object	NO	-	-
ADDRESS	Object	Object	YES	-	-
__LOCK_OUT__	Boolean	JAVA_TYPE_PRIMITIVE_BOOLEAN	NO	-	-
ACTIVITYGROUPS	Array	Object	NO	Object	Object
PROFILES (2)	Array	String	NO	String	String
GROUPS	Array	String	NO	String	String
SYSTEMS (1)	Array	String	NO	String	String

(1) On CUA Systems only.

(2) On CUA Systems it works as an array of objects.

PROPERTY NAME	TYPE	NATIVE TYPE	REQUIRED
MANDT	String	String	NO
BAPIPTEXT	String	String	NO
PROFN	String	String	NO
TYP	String	String	NO
SUBSYSTEMS (1)	Array	String	NO
SUB_PROF (2)	Array	String	NO

(1) On CUA Systems only.

(2) Currently on non CUA Systems only.

PROPERTY NAME	TYPE	NATIVE TYPE	REQUIRED
AGR_NAME	String	String	NO
MANDT	String	String	NO
SUBSYSTEMS [1]	Array	String	NO
SUB_AGR [2]	Array	String	NO
PROFILES [2]	Array	String	NO
T_CODES [2]	Array	String	NO

[1] On CUA Systems only.

[2] Currently on non CUA Systems only.

PROPERTY NAME	TYPE	NATIVE TYPE	REQUIRED
USERGROUP	String	String	NO
MANDT	String	String	NO

This object type is available on CUA Systems only

PROPERTY NAME	TYPE	NATIVE TYPE	REQUIRED
SYSNAME	String	String	NO
SYSTEMTYPE	String	String	NO
RFCDEST	String	String	NO
RCVSYSTEM	String	String	NO
NEW_SYSTEM	String	String	NO
MODEL	String	String	NO
SYSCLIENT	String	String	NO
CLIENT	String	String	NO

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, refer to SAP Secure Network Connection Configuration Properties.

Implementation specifics

For PATCH requests, a connector can potentially add, remove, or replace an attribute value. The SAP connector implements the add, remove, and replace operations but the sample scripts provided with the connector implement only the replace operation. If you use these sample scripts, a PATCH request will therefore always replace the entire attribute value with the new value.

Setting productive passwords on the SAP system

Synchronization of passwords to the SAP system *requires* that you configure SNC and SSO. If you do not configure these two elements correctly, passwords that are updated by IDM are set as *initial* passwords rather than *productive* passwords, and users are forced to change their passwords on login.

1. To configure the SAP connector to use SNC, set the `sncMode` property to `"1"`.

   To configure the connector to use SSO with SNC, set the `sncSSO` property to `"1"`.

2. The logon session during which a productive password is set must be secured using the authentication method Single Sign-On (SSO) using Secure Network Communications (SNC). IDM must request and receive an SSO logon ticket from the SAP system to allow the `BAPI_USER_CHANGE` process to set a productive password. For more information, refer to SAP Note 1287410.

   To configure the connector to request this logon ticket, set the value of the `x509Cert` property as follows:

   - If you are using an X509 certificate to negotiate with the SAP server, set the `x509Cert` property to the base 64-encoded certificate.

     Note that the certificate must be a valid, CA-signed certificate. You cannot use a self-signed certificate here.

   - If you do not use an X509 certificate to negotiate with the SAP server, set the `x509Cert` property to `null`.

     In this case, the connector will use the `user` and `password` specified in the connector configuration to request the SSO logon ticket.

OpenICF Interfaces Implemented by the SAP Connector

The SAP Connector implements the following OpenICF interfaces. For additional details, see ICF interfaces:

Authenticate

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

Create

Creates an object and its `uid`.

Delete

Deletes an object, referenced by its `uid`.

Resolve Username

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

Schema

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

Script on Connector

Enables an application to run a script in the context of the connector.

Any script that runs on the connector has the following characteristics:

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

Script on Resource

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

Search

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

Sync

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

Test

Tests the connector configuration.

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

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

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

Update

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

SAP Connector Configuration

The SAP Connector has the following configurable properties:

Basic Configuration Properties

Property	Type	Default	Encrypted[1]	Required[2]
`gwHost`	`String`	`null`		✕ No
SAP gateway host name.				
`gwServ`	`String`	`null`		✕ No
SAP gateway service.				
`asHost`	`String`	`null`		✕ No
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
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`		✕ No
SAP router string to use for a system protected by a firewall. (/H/host[/S/port]).				
`cua`	`boolean`	`false`		✓ Yes

Property	Type	Default	Encrypted[1]	Required[2]
Description is not available				
`threadPoolSize`	`int`	`8`		✕ No
Description is not available				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

Advanced Configuration

Property	Type	Default	Encrypted[1]	Required[2]
`r3Name`	`String`	`null`		✕ No
Specifies the name of the SAP system, used when you log in to a logon group that uses load balancing.				
`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.				
`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.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

SAP Secure Network Connection Configuration

Property	Type	Default	Encrypted[1]	Required[2]
`x509Cert`	`String`	`null`	🔒 Yes	✕ No
The X509 certificate supplied for authentication.				
`sncPartnerName`	`String`	`null`		✕ No
Specifies the AS ABAP SNC name, for example, "p:CN=ABC, O=MyCompany, C=US". You can find the application server SNC name in the profile parameter snc/identity/as on the AS ABAP.				

Property	Type	Default	Encrypted[1]	Required[2]
`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 connector SNC name, for example, "p:CN=OpenIDM, O=MyCompany, C=US". This parameter is optional, but you should set it to make sure that the correct SNC name is used for the connection.				
`sncMode`	`String`	`0`		✓ Yes
Flag used to activate SNC. Possible values are 0 (OFF) and 1 (ON).				
`sncSSO`	`String`	`0`		✕ No
Specifies whether the connection should be configured for single sign-on (SSO). Possible values are 0 (OFF) and 1 (ON).				
`sncLibrary`	`String`	`null`		✕ No
Specifies the path to the external library that provides Secure Network Connection service. The default is the system-defined library as defined in the environment variable SNC_LIB.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

JCo Connection Pool Configuration

Property	Type	Default	Encrypted[1]	Required[2]
`peakLimit`	`String`	`0`		✕ No
Maximum number of active connections that can be created for a destination simultaneously. The value 0 means unlimited.				
`poolCapacity`	`String`	`1`		✕ No
Maximum number of idle connections kept open by the destination. 0 = no connection pooling.				
`expirationTime`	`String`	`60000`		✕ No
Time in ms after that a free connection can be closed.				
`expirationPeriod`	`String`	`60000`		✕ No
Period in ms after that the destination checks the released connections for expiration.				

Property	Type	Default	Encrypted[1]	Required[2]
`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.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

SAP Jco Logs Configuration

Property	Type	Default	Encrypted[1]	Required[2]
`cpicTrace`	`String`	`0`		✕ No
Enable/disable CPIC trace [0..3].				
`trace`	`String`	`0`		✕ No
Enable/disable RFC trace (0 or 1).				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

Groovy Engine configuration

Property	Type	Default	Encrypted[1]	Required[2]
`scriptRoots`	`String[]`	`['!/scripts/sap/']`		✓ Yes
The root folder to load the scripts from. If the value is null or empty the classpath value is used.				
`classpath`	`String[]`	`[]`		✕ No
Classpath for use during compilation.				
`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.				
`minimumRecompilationInterval`	`int`	`100`		✕ No

Property	Type	Default	Encrypted[1]	Required[2]
Sets the minimum of time after a script can be recompiled.				
`recompileGroovySource`	`boolean`	`false`		✕ No
If set to true recompilation is enabled.				
`scriptBaseClass`	`String`	`null`		✕ No
Base class name for scripts (must derive from Script).				
`scriptExtensions`	`String[]`	`['groovy']`		✕ No
Gets the extensions used to find groovy files.				
`sourceEncoding`	`String`	`UTF-8`		✕ No
Encoding for source files.				
`targetDirectory`	`File`	`null`		✕ No
Directory into which to write classes.				
`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.				
`verbose`	`boolean`	`false`		✕ No
If true, the compiler should produce action information.				
`warningLevel`	`int`	`1`		✕ No
Warning Level of the compiler.				
`customConfiguration`	`String`	`null`		✕ No
Custom Configuration script for Groovy ConfigSlurper.				
`customSensitiveConfiguration`	`GuardedString`	`null`	🔒 Yes	✕ No
Custom Sensitive Configuration script for Groovy ConfigSlurper.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

Operation Script Files

Property	Type	Default	Encrypted(1)	Required(2)
`authenticateScriptFileName`	`String`	`null`		- Authenticate
The name of the file used to perform the AUTHENTICATE operation.				
`createScriptFileName`	`String`	`null`		- Create
The name of the file used to perform the CREATE operation.				
`customizerScriptFileName`	`String`	`null`		✕ No
The script used to customize some function of the connector. Read the documentation for more details.				
`deleteScriptFileName`	`String`	`null`		- Delete
The name of the file used to perform the DELETE operation.				
`resolveUsernameScriptFileName`	`String`	`null`		- Resolve Username
The name of the file used to perform the RESOLVE_USERNAME operation.				
`schemaScriptFileName`	`String`	`null`		- Schema
The name of the file used to perform the SCHEMA operation.				
`scriptOnResourceScriptFileName`	`String`	`null`		- Script on Resource
The name of the file used to perform the RUNSCRIPTONRESOURCE operation.				
`searchScriptFileName`	`String`	`null`		- Read - Search
The name of the file used to perform the SEARCH operation.				

Property	Type	Default	Encrypted[1]	Required[2]
`syncScriptFileName`	`String`	`null`		• Sync
The name of the file used to perform the SYNC operation.				
`testScriptFileName`	`String`	`TestSAP.groovy`		• 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.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

SAP HANA Database connector

Before you start

These connector instructions require a SAP HANA Database account with elevated privileges to add roles, system, and application privileges. The following information is required to configure the connector:

Username

Your SAP HANA Database username.

Password

Your SAP HANA Database password.

JDBC Connection URL

The URL to establish the connection between the connector and the SAP HANA Database.

Driver class name

The class name driver path.

For more information, refer to the Connect to SAP HANA via JDBC documentation.

Install the SAP HANA Database connector

> **Tip**
>
> To check for an Advanced Identity Cloud application for this connector, refer to:
>
> - Application management
> - App catalog

You can download any connector from Backstage, but some are included in the default deployment for Advanced Identity Cloud, IDM, or RCS. When using an included connector, you can skip installing it and move directly to configuration.

Connector included in default deployment

Connector	IDM	RCS
SAP HANA Database	✕ No	✓ Yes

Download the connector .jar file from Backstage.

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

```
mv ~/Downloads/saphanadb-connector-1.5.20.26.jar /path/to/openidm/connectors/
```

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

Download the Sap Hana JDBC driver.

> **Note**
>
> The minimum required JDBC version is 2.16.14.

- If you are running the connector locally, place the library in the `/path/to/openidm/lib/` directory:

```
mv ~/Downloads/ngdbc-version.jar /path/to/openidm/lib/
```

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

Configure the SAP HANA Database connector

Create a connector configuration using the IDM admin UI:

1. From the navigation bar, click **Configure > Connectors**.
2. On the **Connectors** page, click **New Connector**.
3. On the **New Connector** page, type a **Connector Name**.
4. From the **Connector Type** drop-down list, select **SAP HANA Database Connector - 1.5.20.26**.

5. Complete the **Base Connector Details**.

> **Tip**
>
> For a list of all configuration properties, refer to SAP HANA Database Connector Configuration

6. Click **Save**.

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

Refer to this procedure to create a connector configuration over REST.

Base connector details

`username`	The username for logging in to the database.
`password`	The user password for logging in to the database.
`url`	The database connection string in the form of `jdbc:sap://<server>:<port>[/?<options>]`.
`driverClassName`	The file directory location of DBC driver files.
`pageSize`	Defines the page size to be displayed to users.
`ignoreUsers`	Database users to ignore. Typically, these are internal database users to avoid for security reasons. `SYS` and `SYSTEM` internal users are ignored by default.

Object types

You can add or edit the object type to obtain any of the following objects and their properties:

PROPERTY NAME	TYPE	NATIVE TYPE	REQUIRED
`USER_ID`	String	String	NO
`USER_NAME`	String	String	NO
`PASSWORD`	GuardedString	String	NO
`USER_MODE`	String	String	NO
`EMAIL_ADDRESS`	String	String	NO
`CLIENT`	String	String	NO
`TIME_ZONE`	String	String	NO
`CREATOR`	String	String	NO
`VALID_FROM`	String	String	NO
`VALID_UNTIL`	String	String	NO
`IS_RESTRICTED`	Boolean	Boolean	NO

PROPERTY NAME	TYPE	NATIVE TYPE	REQUIRED
`IS_CLIENT_CONNECT_ENABLED`	Boolean	Boolean	NO
`HAS_REMOTE_USERS`	Boolean	Boolean	NO
`PASSWORD_CHANGE_NEEDED`	Boolean	Boolean	NO
`IS_KERBEROS_ENABLED`	Boolean	Boolean	NO
`IS_SAML_ENABLED`	Boolean	Boolean	NO
`IS_PASSWORD_ENABLED`	Boolean	Boolean	NO
`SAML_PROVIDERS`	Array	Object	NO
`SYSTEM_PRIVILEGES`	Array	String	NO
`APPLICATION_PRIVILEGES`	Array	String	NO
`ROLES`	Array	String	NO
`EXTERNAL_IDENTITY`	Array	String	NO

PROPERTY NAME	TYPE	NATIVE TYPE	REQUIRED
`ROLE_ID`	String	String	NO
`ROLE_NAME`	String	String	NO
`GLOBAL_IDENTITY`	String	String	NO
`ROLE_SCHEMA_NAME`	String	String	NO
`ROLE_MODE`	String	String	NO

PROPERTY NAME	TYPE	NATIVE TYPE	REQUIRED
`__NAME__`	String	String	NO

PROPERTY NAME	TYPE	NATIVE TYPE	REQUIRED
`__NAME__`	String	String	NO

To configure the connector over REST or using the filesystem, specify the connection details to the SAP HANA Database resource provider in the `configurationProperties` for the connector. The minimum required properties are `username` , `password` , `url` and `driverClassName` .

Sample configuration

```
{
    "configurationProperties" : {
        "connectionProperties" : null,
        "propagateInterruptState" : false,
        "useDisposableConnectionFacade" : true,
        "defaultCatalog" : null,
        "validationInterval" : 3000,
        "ignoreExceptionOnPreLoad" : false,
        "jmxEnabled" : true,
        "commitOnReturn" : false,
        "logAbandoned" : false,
        "maxIdle" : 100,
        "testWhileIdle" : false,
        "removeAbandoned" : false,
        "abandonWhenPercentageFull" : 0,
        "minIdle" : 10,
        "defaultReadOnly" : null,
        "maxWait" : 30000,
        "logValidationErrors" : false,
        "name" : "Tomcat Connection Pool[1-20280544]",
        "useStatementFacade" : true,
        "initSQL" : null,
        "validationQueryTimeout" : -1,
        "validationQuery" : null,
        "rollbackOnReturn" : false,
        "alternateUsernameAllowed" : false,
        "dataSourceJNDI" : null,
        "validatorClassName" : null,
        "suspectTimeout" : 0,
        "useEquals" : true,
        "removeAbandonedTimeout" : 60,
        "defaultAutoCommit" : null,
        "testOnConnect" : false,
        "jdbcInterceptors" : null,
        "initialSize" : 10,
        "defaultTransactionIsolation" : -1,
        "numTestsPerEvictionRun" : 0,
        "url" : "jdbc:sap://HOST:PORT",
        "testOnBorrow" : false,
        "fairQueue" : true,
        "accessToUnderlyingConnectionAllowed" : true,
        "maxAge" : 0,
        "minEvictableIdleTimeMillis" : 60000,
        "timeBetweenEvictionRunsMillis" : 5000,
        "testOnReturn" : false,
        "useLock" : false,
        "maxActive" : 100,
        "username" : "USERNAME",
        "password" : "PASSWORD",
        "pageSize" : "50",
        "driverClassName" : "com.sap.db.jdbc.Driver",
        "ignoreUsers" : [
            "SYS",
            "SYSTEM"
        ]
    }
}
```

Configure connection pooling

The SAP HANA Database connector embeds the Apache Tomcat 9 JDBC Connection Pool. Learn more about the different pooling mechanisms in Connectors by pooling mechanism.

Mapping

Attributes

SOURCE	TARGET	TRANSFORMATION SCRIPT
`USER_ID`	`_id`	N/A
`USER_NAME`	`userName`	N/A
`PASSWORD`	`password`	N/A
`EMAIL_ADDRESS`	`mail`	N/A
`TIME_ZOME`	`timeZone`	N/A
`CLIENT`	`sessionClient`	N/A
`VALID_FROM`	`validFrom`	N/A
`VALID_UNTIL`	`validUntil`	N/A
`IS_RESTRICTED`	`isRestricted`	N/A
`ROLES`	`grantedRoles`	N/A
`APPLICATION_PRIVILEGES`	`applicationPrivileges`	N/A
`SYSTEM_PRIVILEGES`	`systemPrivileges`	N/A
`SAML_PROVIDERS`	`samlProviders`	N/A
`IS_SAML_ENABLED`	`isSamlEnabled`	N/A
`IS_KERBEROS_ENABLED`	`isKerberosEnabled`	N/A
`PASSWORD_CHANGE_NEEDED`	`passwordChangeNeeded`	N/A
`EXTERNAL_IDENTITY`	`externalIdentity`	N/A

Association>Association Rules>Correlation Queries

- Link Qualifier: default
- Any of the following fields: `USER_NAME`

Attributes

SOURCE	TARGET	TRANSFORMATION SCRIPT
`userName`	`USER_NAME`	N/A
`password`	`PASSWORD`	N/A
`mail`	`EMAIL_ADDRESS`	N/A
`timeZone`	`TIME_ZONE`	N/A
`sessionClient`	`CLIENT`	N/A
`validFrom`	`VALID_FROM`	N/A
`validUntil`	`VALID_UNTIL`	N/A
`isRestricted`	`IS_RESTRICTED`	N/A
`grantedRoles`	`ROLES`	N/A
`applicationPrivileges`	`APPLICATION_PRIVILEGES`	N/A
`systemPrivileges`	`SYSTEM_PRIVILEGES`	N/A
`samlProviders`	`SAML_PROVIDERS`	N/A
`isSamlEnabled`	`IS_SAML_ENABLED`	N/A
`isKerberosEnabled`	`IS_KERBEROS_ENABLED`	N/A
`passwordChangeNeeded`	`PASSWORD_CHANGE_NEEDED`	N/A
`externalIdentity`	`EXTERNAL_IDENTITY`	N/A

Association>Association Rules>Correlation Queries

- Link Qualifier: default
- Any of the following fields: `USER_NAME`

Test the SAP HANA Database connector

Test the connector configuration:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Accept-API-Version: resource=1.0' \
--request POST \
'http://localhost:8080/openidm/system/saphanadb?_action=test'
{
    "name": "saphanadb",
    "enabled": true,
    "config": "config/provisioner.openicf/saphanadb",
    "connectorRef": {
        "bundleVersion": [1.5.0.0,1.6.0.0),
        "bundleName": "org.forgerock.openicf.connectors.saphanadb-connector",
        "connectorName": "org.forgerock.openicf.connectors.saphanadb.SapHanaDBConnector"
    },
    "displayName": "SAP HANA Database Connector",
    "objectTypes": [
        "APPLICATION_PRIVILEGES",
        "__ACCOUNT__",
        "SYSTEM_PRIVILEGES",
        "ROLES",
        "__ALL__"
    ],
    "ok": true
}
```

Use the SAP HANA Database connector

Database user

Create a user

To create a new user, you must include at least the `USER_NAME` and `PASSWORD` fields. The default configuration requires passwords to have:

- a minimum of 8 characters.
- at least one number.
- at least one uppercase letter.
- at least one lowercase letter.

> **Note**
>
> Special characters are optional, but the default password setting (`Aa1`) only accepts underscores (_). For more information, refer to Password Policy Configuration Options.

If the `IS_RESTRICTED` field is true, a restricted user is created. A restricted user has no default roles and an unrestricted user has the default PUBLIC role.

The possible date format for the fields `VALID_FROM` and `VALID_UNTIL` is: `yyyy-MM-dd HH:mm AM/PM`.

When assigning SAML Providers to a User, only those providers that already exist within the database can be assigned during a create operation.

To grant and revoke roles, application or system privileges, some requirements are necessary, as detailed here.

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Accept-API-Version: resource=1.0' \
--header 'Content-Type: application/json' \
--request POST \
--data '{
    "USER_NAME" : "SAPHANADB_NEWUSER",
    "PASSWORD" : "Password123",
    "EMAIL_ADDRESS" : "SAPHANADB_NEWUSER@example.com",
    "CLIENT" : "001",
    "TIME_ZONE" : "GMT",
    "VALID_FROM" : "2024-12-12 12:30",
    "VALID_UNTIL" : "2025-12-12 15:00",
    "IS_SAML_ENABLED" : true,
    "IS_KERBEROS_ENABLED" : true,
    "IS_PASSWORD_ENABLED" : true,
    "IS_CLIENT_CONNECT_ENABLED": true,
    "ROLES" : [
      "MODELING"
    ],
    "APPLICATION_PRIVILEGES" : [
        "sap.hana.backup::Admin"
    ],
    "SYSTEM_PRIVILEGES" : [
        "REPO.EXPORT",
        "REPO.IMPORT",
        "REPO.MAINTAIN_DELIVERY_UNITS"
    ]
}' \
'http://localhost:8080/openidm/system/saphanadb/__ACCOUNT__'
{
    "USER_NAME" : "SAPHANADB_NEWUSER",
    "EMAIL_ADDRESS" : "SAPHANADB_NEWUSER@example.com",
    "IS_RESTRICTED" : false,
    "CLIENT" : "001",
    "TIME_ZONE" : "GMT",
    "USER_MODE" : "LOCAL"
    "VALID_FROM": "2024-12-12 12:30",
    "VALID_UNTIL": "2025-12-12 15:00",
    "IS_SAML_ENABLED" : true,
    "IS_KERBEROS_ENABLED" : true,
    "IS_PASSWORD_ENABLED" : true,
    "PASSWORD_CHANGE_NEEDED" : false,
    "IS_CLIENT_CONNECT_ENABLED": true,
    "HAS_REMOTE_USERS" : false,
    "EXTERNAL_IDENTITY" : false,
    "CREATOR" : "USER_CREATOR",
    "ROLES" : [
        "PUBLIC",
        "MODELING"
    ],
    "APPLICATION_PRIVILEGES" : [
        "sap.hana.backup::Admin"
    ],
    "SYSTEM_PRIVILEGES" : [
        "REPO.EXPORT",
        "REPO.IMPORT",
        "REPO.MAINTAIN_DELIVERY_UNITS"
    ]
}
```

Get users

Retrieve a list of database user ids from SAP HANA Database:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
'http://localhost:8080/openidm/system/saphanadb/__ACCOUNT__?_queryId=query-all-ids'
{
    "result": [
        {
            "_id" : "001"
        },
        {
            "_id" : "002"
        },
        {
            "_id" : "003"
        },
        ...
    ]
}
```

Get a user

Retrieve a user from SAP HANA Database. You must specify the id in the URI path:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
'http://localhost:8080/openidm/system/saphanadb/__ACCOUNT__/USER_ID'
{
    "USER_NAME" : "NEW_USER",
    "EMAIL_ADDRESS" : "NEW_USER@example.com",
    "IS_RESTRICTED" : false,
    "CLIENT" : "000",
    "TIME_ZONE" : "GMT",
    "USER_MODE" : "LOCAL",
    "VALID_FROM": "2023-09-06",
    "VALID_UNTIL": "2023-12-31",
    "IS_SAML_ENABLED" : fale,
    "IS_KERBEROS_ENABLED" : false,
    "IS_PASSWORD_ENABLED" : true,
    "PASSWORD_CHANGE_NEEDED" : false,
    "HAS_REMOTE_USERS" : false,
    "IS_CLIENT_CONNECT_ENABLED": true,
    "EXTERNAL_IDENTITY" : "999",
    "CREATOR" : "USER_CREATOR",
    "ROLES": [
        "PUBLIC",
        "MODELING"
    ],
    "APPLICATION_PRIVILEGES" : [
        "sap.hana.backup::Admin"
    ],
    "SYSTEM_PRIVILEGES" : [
        "REPO.EXPORT",
    ],
    "SAML_PROVIDERS" : [
        {
            "SAML_PROVIDER_NAME" : "PROVIDER_NAME",
            "EXTERNAL_IDENTITY" : null
        }
    ]
}
```

Update a user

Update a user from the database. You must specify the id in the URI path.

The roles field combines the catalog and repository roles. To grant and revoke roles, application or system privileges, some requirements are necessary, as detailed here.

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Accept-API-Version: resource=1.0' \
--request PUT \
--data '{
    "EMAIL_ADDRESS": "NEW_MAIL@EMAIL.COM",
    "CLIENT" : "002",
    "TIME_ZONE" : "PST",
    "VALID_FROM" : "2023-09-06",
    "VALID_UNTIL" : "2023-12-31",
    "IS_KERBEROS_ENABLED" : true,
    "IS_SAML_ENABLED" : true,
    "IS_PASSWORD_ENABLED" : true,
    "PASSWORD_CHANGE_NEEDED": true,
    "IS_CLIENT_CONNECT_ENABLED": true,
    "EXTERNAL_IDENTITY": "999",
    "ROLES": [
        "PUBLIC",
        "RESTRICTED_USER_JDBC_ACCESS"
    ]
}' \
'http://localhost:8080/openidm/system/saphanadb/__ACCOUNT__/USER_ID'
{
    "USER_NAME": "USERNAME",
    "EMAIL_ADDRESS": "NEW_MAIL@EMAIL.COM",
    "IS_RESTRICTED": false,
    "CLIENT": "002",
    "TIME_ZONE": "PST",
    "USER_MODE": "LOCAL",
    "VALID_FROM": "2023-09-06",
    "VALID_UNTIL": "2023-12-31",
    "IS_KERBEROS_ENABLED": true,
    "IS_SAML_ENABLED": true,
    "IS_PASSWORD_ENABLED": true,
    "PASSWORD_CHANGE_NEEDED": true,
    "IS_CLIENT_CONNECT_ENABLED": true,
    "EXTERNAL_IDENTITY": "999",
    "HAS_REMOTE_USERS": false,
    "CREATOR": "USER_CREATOR",
    "ROLES": [
        "PUBLIC",
        "RESTRICTED_USER_JDBC_ACCESS",
    ],
    "APPLICATION_PRIVILEGES": [],
    "SYSTEM_PRIVILEGES": [],
    "SAML_PROVIDERS" : []
}
```

Delete a user

Delete a user from a database. You must specify the id in the URI path:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Accept-API-Version: resource=1.0' \
--request DELETE \
'http://localhost:8080/openidm/system/saphanadb/__ACCOUNT__/USER_ID'
{
    "USER_NAME" : "NEW_USER",
    "EMAIL_ADDRESS" : "NEW_USER@EMAIL.COM",
    "IS_RESTRICTED" : false,
    "CLIENT" : "001",
    "TIME_ZONE" : "GMT",
    "USER_MODE" : "LOCAL"
    "VALID_FROM": "2024-12-12",
    "VALID_UNTIL": "2025-12-12",
    "IS_SAML_ENABLED" : false,
    "IS_KERBEROS_ENABLED" : false,
    "IS_PASSWORD_ENABLED" : true,
    "PASSWORD_CHANGE_NEEDED" : false,
    "IS_CLIENT_CONNECT_ENABLED": true,
    "HAS_REMOTE_USERS" : false,
    "EXTERNAL_IDENTITY" : "999",
    "CREATOR" : "USER_CREATOR",
    "ROLES": [
        "PUBLIC"
    ],
    "APPLICATION_PRIVILEGES" : [],
    "SYSTEM_PRIVILEGES" : [],
    "SAML_PROVIDERS" : []
}
```

Get roles

Retrieve roles from a SAP HANA Database:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Accept-API-Version: resource=1.0' \
--request GET \
'http://localhost:8080/openidm/system/saphanadb/ROLES?_queryFilter=true'
{
    "result": [
        {
            "_id": "2361418",
            "ROLE_NAME": "ABAP_READ",
            "ROLE_ID": "2361418",
            "ROLE_MODE": "LOCAL",
            "__NAME__": "ABAP_READ",
            "ROLE_SCHEMA_NAME": null,
            "GLOBAL_IDENTITY": null
        },
        ...
}
```

Get system privileges

Retrieve system privileges from a SAP HANA Database:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Accept-API-Version: resource=1.0' \
--request GET \
'http://localhost:8080/openidm/system/saphanadb/SYSTEM_PRIVILEGES?_queryFilter=true'
{
    "result": [
        {
            "_id": "ADAPTER ADMIN",
            "__NAME__": "ADAPTER ADMIN",
        },
        ...
}
```

Get application privileges

Retrieve application privileges from a SAP HANA Database:

```
curl
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Accept-API-Version: resource=1.0' \
--request GET 'http://localhost:8080/openidm/system/saphanadb/APPLICATION_PRIVILEGES/_queryFilter=true'
{
    "result": [
        {
            "_id": "sap.hana.ide::Catalog",
            "__NAME__": "sap.hana.ide::Catalog"
        },
        ...
}
```

OpenICF Interfaces Implemented by the SAP HANA Database Connector

The SAP HANA Database Connector implements the following OpenICF interfaces. For additional details, see ICF 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.

SAP HANA Database Connector Configuration

The SAP HANA Database Connector has the following configurable properties:

Configuration properties

Property	Type	Default	Encrypted[(1)]	Required[(2)]
`connectionProperties`	`String`	`null`		✕ No
Description is not available				
`propagateInterruptState`	`boolean`	`false`		✕ No
Description is not available				
`useDisposableConnectionFacade`	`boolean`	`true`		✕ No
Description is not available				
`defaultCatalog`	`String`	`null`		✕ No
Description is not available				
`validationInterval`	`long`	`3000`		✕ No
Description is not available				
`ignoreExceptionOnPreLoad`	`boolean`	`false`		✕ No
Description is not available				
`jmxEnabled`	`boolean`	`true`		✕ No
Description is not available				

Property	Type	Default	Encrypted[1]	Required[2]
`commitOnReturn`	`boolean`	`false`		✕ No
Description is not available				
`logAbandoned`	`boolean`	`false`		✕ No
Description is not available				
`maxIdle`	`int`	`100`		✕ No
Description is not available				
`testWhileIdle`	`boolean`	`false`		✕ No
Description is not available				
`removeAbandoned`	`boolean`	`false`		✕ No
Description is not available				
`abandonWhenPercentageFull`	`int`	`0`		✕ No
Description is not available				
`minIdle`	`int`	`10`		✕ No
Description is not available				
`defaultReadOnly`	`Boolean`	`null`		✕ No
Description is not available				
`maxWait`	`int`	`30000`		✕ No
Description is not available				
`logValidationErrors`	`boolean`	`false`		✕ No
Description is not available				
`driverClassName`	`String`	`null`		✕ No
Description is not available				
`name`	`String`	`Tomcat Connection Pool[7-351632824]`		✕ No

Property	Type	Default	Encrypted[1]	Required[2]
Description is not available				
`useStatementFacade`	`boolean`	`true`		✕ No
Description is not available				
`initSQL`	`String`	`null`		✕ No
Description is not available				
`validationQueryTimeout`	`int`	`-1`		✕ No
Description is not available				
`validationQuery`	`String`	`null`		✕ No
Description is not available				
`rollbackOnReturn`	`boolean`	`false`		✕ No
Description is not available				
`alternateUsernameAllowed`	`boolean`	`false`		✕ No
Description is not available				
`dataSourceJNDI`	`String`	`null`		✕ No
Description is not available				
`validatorClassName`	`String`	`null`		✕ No
Description is not available				
`suspectTimeout`	`int`	`0`		✕ No
Description is not available				
`useEquals`	`boolean`	`true`		✕ No
Description is not available				
`removeAbandonedTimeout`	`int`	`60`		✕ No
Description is not available				
`defaultAutoCommit`	`Boolean`	`null`		✕ No

Property	Type	Default	Encrypted[1]	Required[2]
Description is not available				
`testOnConnect`	`boolean`	`false`		✕ No
Description is not available				
`jdbcInterceptors`	`String`	`null`		✕ No
Description is not available				
`initialSize`	`int`	`10`		✕ No
Description is not available				
`defaultTransactionIsolation`	`int`	`-1`		✕ No
Description is not available				
`numTestsPerEvictionRun`	`int`	`0`		✕ No
Description is not available				
`url`	`String`	`null`		✕ No
Description is not available				
`testOnBorrow`	`boolean`	`false`		✕ No
Description is not available				
`fairQueue`	`boolean`	`true`		✕ No
Description is not available				
`accessToUnderlyingConnectionAllowed`	`boolean`	`true`		✕ No
Description is not available				
`maxAge`	`long`	`0`		✕ No
Description is not available				
`minEvictableIdleTimeMillis`	`int`	`60000`		✕ No
Description is not available				
`timeBetweenEvictionRunsMillis`	`int`	`5000`		✕ No

Property	Type	Default	Encrypted[1]	Required[2]
Description is not available				
`testOnReturn`	`boolean`	`false`		✕ No
Description is not available				
`useLock`	`boolean`	`false`		✕ No
Description is not available				
`maxActive`	`int`	`100`		✕ No
Description is not available				
`username`	`String`	`null`		✕ No
The connection username to be passed to our JDBC driver to establish a connection				
`password`	`String`	`null`	🔒 Yes	✓ Yes
SAP HANA Database login password to authenticate the user				
`pageSize`	`Integer`	`50`		✕ No
Page size of search				
`ignoreUsers`	`String[]`	`['SYS', 'SYSTEM']`		✕ No
List of database users to be ignored by the connector				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

SAP S/4HANA connector

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

> **Note**
>
> The SAP S/4HANA connector only supports SAP HANA Cloud. To connect to HANA DB use the SAP HANA Database connector.

Before you start

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

Username

Inbound Communication user of SAP S/4HANA.

Password

Inbound Communication user password of SAP S/4HANA.

Tenant ID

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

Install the SAP S/4HANA connector

> **Tip**
>
> To check for an Advanced Identity Cloud application for this connector, refer to:
>
> - Application management
> - App catalog

You can download any connector from Backstage, but some are included in the default deployment for Advanced Identity Cloud, IDM, or RCS. When using an included connector, you can skip installing it and move directly to configuration.

Connector included in default deployment

Connector	IDM	RCS
SAP S/4HANA	✕ No	✕ No

Download the connector .jar file from Backstage.

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

```
mv ~/Downloads/saphana-connector-1.5.20.26.jar /path/to/openidm/connectors/
```

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

Configure the SAP S/4HANA connector

Create a connector configuration using the IDM admin UI:

1. From the navigation bar, click **Configure > Connectors**.

2. On the **Connectors** page, click **New Connector**.
3. On the **New Connector** page, type a **Connector Name**.
4. From the **Connector Type** drop-down list, select **SAP S/4HANA Connector - 1.5.20.26**.
5. Complete the **Base Connector Details**.

> **Tip**
>
> For a list of all configuration properties, refer to SAP S/4HANA Connector Configuration

6. Click **Save**.

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

Refer to this procedure to create a connector configuration over REST.

Test the SAP S/4HANA connector

Test that the configuration is correct by running the following command:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system/saphana?_action=test"
{
  "name": "saphana",
  "enabled": true,
  "config": "config/provisioner.openicf/saphana",
  "connectorRef": {
    "bundleVersion": "[1.5.0.0,1.6.0.0)",
    "bundleName": "org.forgerock.openicf.connectors.saphana-connector",
    "connectorName": "org.forgerock.openicf.connectors.saphana.SapHanaConnector"
  },
  "displayName": "SAP HANA Connector",
  "objectTypes": [
    "__ACCOUNT__",
    "__ALL__"
  ],
  "ok": true
}
```

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

SAP S/4HANA remote connector

If you want to run this connector outside of PingOne Advanced Identity Cloud or IDM, you can configure the SAP S/4HANA connector as a remote connector. Java Connectors installed remotely on a Java Connector Server function identically to those bundled locally within PingOne Advanced Identity Cloud or installed locally on IDM.

You can download the SAP S/4HANA connector from here.

Refer to Remote connectors for configuring the SAP S/4HANA remote connector.

Configure connection pooling

The SAP S/4HANA connector uses a non-poolable mechanism to manage connections. Learn more about the different pooling mechanisms in Connectors by pooling mechanism.

Use the SAP S/4HANA connector

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

Attribute	Description
`PersonUUID`	Unique identifier for a user.
`PersonExternalID`	The external ID of the user. This can only include uppercase letters, numbers, `.`, `-`, and `_`. Required.
`UserId`	Auto-generated user id.
`PersonID`	Auto-generated user id.
`__USER__`	Login username for the user. This can only include uppercase letters, numbers, `.`, `-`, and `_`. Required.
`FirstName`	First name of the user. Required.
`LastName`	Last name of the user. Required.
`MiddleName`	Middle name of the user.
`LockedIndicator`	Status of the user. Either `active` or `inactive`.
`GenderCode`	Gender of the user. Permitted values are `1` (Male), or `2` (Female). Required; use `0` to leave it unspecified. If specified, this must match the values listed in `FormOfAddress`.
`FormOfAddress`	Salutation of the user. Permitted values are `00001` or `1` (Mr), or `00002` or `2` (Mrs). Values must match `GenderCode`.
`StartDate`	Start date for the created user, in YYYY-MM-DD format. Required.
`EndDate`	End date for the created user, in YYYY-MM-DD format.
`PersonFullName`	Full name of the user.
`AcademicTitle`	Academic title of the user. Permitted values are `0001` (DR.), `0002` (PROF.), `0003` (PROF. DR.), `0004` (B.A.), `0005` (MBA), or `0006` (PH.D.).

Attribute	Description
`CorrespondenceLanguage`	The correspondence language for the user. For example, `DE`, or `EN`.
`AdditionalLastName`	Additional last name of the user.
`BirthName`	Birth name of the user.
`NickName`	Nickname of the user.
`Initials`	Initials of the user.
`AcademicSecondTitle`	Academic secondary title of the user. Permitted values are `0001` (DR.), `0002` (PROF.), `0003` (PROF. DR.), `0004` (B.A.), `0005` (MBA), or `0006` (PH.D.).
`NameSupplement`	Supplemental titles of the user. Permitted values are `0001` or `1` (Earl), `0002` or `2` (Freifrau), `0003` or `3` (Freiherr), `0004` or `4` (Fürst), `0005` or `5` (Fürstin), `0006` or `6` (Graf), `0007` or `7` (Gräfin), and `0008` or `8` (Sir).
`PhoneInformation`	Object with the following sub-attributes: • `PhoneNumberType`: Type of phone number. Permitted values are `ECPC` (Cell phone), or `ECPB` (Landline). • `CountryDialingCode`: Country dialing code, such as `1` (USA) or `33` (France). Numbers only, `+` is not allowed. • `PhoneNumberSubscriberID`: Phone number of the user.
`EmailAddress`	Email address of the user.
`CompanyCode`	Predefined code of the company of the user. Required.
`WorkAgreementStatus`	Status of the work agreement for the user. Permitted values are `0` (Inactive), or `1` (Active).
`LogonLanguageCode`	Language code for the user.

Attribute	Description
`DateFormatCode`	What format dates should be displayed in. For example, `6` formats dates according to ISO 8601 (YYYY-MM-DD). Available codes: • `1` - DD.MM.YYYY (Gregorian Date) • `2` - MM/DD/YYYY (Gregorian Date) • `3` - MM-DD-YYYY (Gregorian Date) • `4` - YYYY.MM.DD (Gregorian Date) • `5` - YYYY/MM/DD (Gregorian Date) • `6` - YYYY-MM-DD (Gregorian Date, ISO 8601) • `7` - GYY.MM.DD (Japanese Date) • `8` - GYY/MM/DD (Japanese Date) • `9` - GYY-MM-DD (Japanese Date) • `A` - YYYY/MM/DD (Islamic Date 1) • `B` - YYYY/MM/DD (Islamic Date 2) • `C` - YYYY/MM/DD (Iranian Date)
`TimeFormatCode`	What format times should be displayed in. For example, `0` formats times in a 24 hour format (14:35:59 PM). Available codes: • `0` - 24 Hour Format (Example: 12:05:10) • `1` - 12 Hour Format (Example: 12:05:10 PM) • `2` - 12 Hour Format (Example: 12:05:10 pm) • `3` - Hours from 0 to 11 (Example: 00:05:10 PM) • `4` - Hours from 0 to 11 (Example: 00:05:10 pm)
`TimeZoneCode`	Time zone code of the user.
`DecimalFormatCode`	What decimal notation numbers should be displayed in. Available codes: • `1.234.567,89` • `X` - 1,234,567.89 • `Y` - 1 234 567,89
`Role`	Role assignment of the user.
`PersonWorkAgreementUUID`	Unique ID of the work agreement associated with the user.
`PersonWorkAgreementExternalID`	External ID of the work agreement associated with the user.
`PersonWorkAgreementType`	Role of the work agreement associated with the user. Permitted values are `1` (User), or `3` (Service performer).

Note

The following attributes are mapped in the connector automatically:

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

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

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

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request POST \
--data '{
  "__NAME__": "BJENSEN",
  "FirstName": "Barbara",
  "LastName": "Jensen",
  "StartDate": "2022-06-02",
  "PersonExternalID": "BJENSEN",
  "CompanyCode": 1010,
  "GenderCode": 2
}' \
"http://localhost:8080/openidm/system/saphana/__ACCOUNT__?_action=create"
{
  "_id": "9980004320",
  "TimeFormatCode": "0",
  "PersonFullName": "Barbara Jensen",
  "__UID__": "9980004320",
  "FirstName": "Barbara",
  "UserID": "CB9980004320",
  "__NAME__": "BJENSEN",
  "DecimalFormatCode": "X",
  "StartDate": "2022-06-02",
  "LogonLanguageCode": "EN",
  "LastName": "Jensen",
  "PersonExternalID": "BJENSEN",
  "PersonUUID": "FA163EA9-3617-1EEC-B8DA-AD865EF3B625",
  "__ENABLE__": true,
  "TimeZoneCode": "CET",
  "EndDate": "9999-12-31",
  "DateFormatCode": "1"
}
```

> **Note**
>
> When you create a new user, you must specify *at least*:
>
> - `__NAME__`
> - `PersonExternalID`
> - `FirstName`
> - `LastName`
> - `GenderCode`
> - `StartDate`
> - `CompanyCode`
>
> See the list of available attributes for more information.

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

- `__USER__`
- `PersonExternalID`
- `FirstName`
- `LastName`
- `GenderCode`
- `EmailAddress`
- `PhoneInformation`
- `PersonFullName`
- `AcademicTitle`
- `CorrespondenceLanguage`
- `MiddleName`
- `AdditionalLastName`
- `BirthName`
- `NickName`
- `Initials`
- `AcademicSecondTitle`
- `NameSupplement`
- `WorkAgreementStatus`
- `CompanyCode`
- `StartDate`
- `EndDate`

- `LockedIndicator`
- `DateFormatCode`
- `DecimalFormatCode`
- `TimeFormatCode`
- `TimeZoneCode`
- `LogonLanguageCode`
- `Role`

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

For example, to add an email address to a user:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "If-Match:*" \
--request PUT \
--data '{
  "__NAME__": "BJENSEN",
  "FirstName": "Barbara",
  "LastName": "Jensen",
  "StartDate": "2022-06-02",
  "PersonExternalID": "BJENSEN",
  "CompanyCode": 1010,
  "GenderCode": 2,
  "EmailAddress": "bjensen@example.com"
}' \
"http://localhost:8080/openidm/system/saphana/__ACCOUNT__/9980004320"
{
  "_id": "9980004320",
  "TimeFormatCode": "0",
  "PersonFullName": "Barbara Jensen",
  "__UID__": "9980004320",
  "EmailAddress": "bjensen@example.com",
  "FirstName": "Barbara",
  "UserID": "CB9980004320",
  "__NAME__": "BJENSEN",
  "DecimalFormatCode": "X",
  "StartDate": "2022-06-02",
  "LogonLanguageCode": "EN",
  "LastName": "Jensen",
  "PersonExternalID": "BJENSEN",
  "PersonUUID": "FA163EA9-3617-1EEC-B8DA-AD865EF3B625",
  "__ENABLE__": true,
  "TimeZoneCode": "CET",
  "EndDate": "9999-12-31",
  "DateFormatCode": "1"
}
```

The following example queries all SAP S/4HANA users:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/saphana/__ACCOUNT__?_queryId=query-all-ids"
{
  "result": [
    {
      "_id": "9980000000"
    },
    {
      "_id": "9980000002"
    },
    {
      "_id": "9980000004"
    },
    {
      "_id": "9980000006"
    },
    [ ... ]
    {
      "_id": "9980004314"
    },
    {
      "_id": "9980004316"
    },
    {
      "_id": "9980004318"
    },
    {
      "_id": "9980004320"
    }
  ],
  "resultCount": 2139,
  "pagedResultsCookie": null,
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

The following command queries a specific user by their ID:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"http://localhost:8080/openidm/system/saphana/__ACCOUNT__/9980004320"
{
  "_id": "9980004320",
  "TimeFormatCode": "0",
  "PersonFullName": "Barbara Jensen",
  "__UID__": "9980004320",
  "EmailAddress": "bjensen@example.com",
  "FirstName": "Barbara",
  "UserID": "CB9980004320",
  "__NAME__": "BJENSEN",
  "DecimalFormatCode": "X",
  "StartDate": "2022-06-02",
  "LogonLanguageCode": "EN",
  "LastName": "Jensen",
  "PersonExternalID": "BJENSEN",
  "PersonUUID": "FA163EA9-3617-1EEC-B8DA-AD865EF3B625",
  "__ENABLE__": true,
  "TimeZoneCode": "CET",
  "EndDate": "9999-12-31",
  "DateFormatCode": "1"
}
```

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "If-Match:*" \
--request PUT \
--data '{
  "__NAME__": "BJENSEN",
  "__ENABLE__": true
}' \
"http://localhost:8080/openidm/system/saphana/__ACCOUNT__/9980004320"
{
  "_id": "9980004320",
  "TimeFormatCode": "0",
  "PersonFullName": "Barbara Jensen",
  "__UID__": "9980004320",
  "EmailAddress": "bjensen@example.com",
  "FirstName": "Barbara",
  "UserID": "CB9980004320",
  "__NAME__": "BJENSEN",
  "DecimalFormatCode": "X",
  "StartDate": "2022-06-02",
  "LogonLanguageCode": "EN",
  "LastName": "Jensen",
  "PersonExternalID": "BJENSEN",
  "PersonUUID": "FA163EA9-3617-1EEC-B8DA-AD865EF3B625",
  "__ENABLE__": true,
  "TimeZoneCode": "CET",
  "EndDate": "9999-12-31",
  "DateFormatCode": "1"
}
```

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "If-Match:*" \
--request PUT \
--data '{
  "__NAME__": "BJENSEN",
  "__ENABLE__": false
}' \
"http://localhost:8080/openidm/system/saphana/__ACCOUNT__/9980004320"
{
  "_id": "9980004320",
  "TimeFormatCode": "0",
  "PersonFullName": "Barbara Jensen",
  "__UID__": "9980004320",
  "EmailAddress": "bjensen@example.com",
  "FirstName": "Barbara",
  "UserID": "CB9980004320",
  "__NAME__": "BJENSEN",
  "DecimalFormatCode": "X",
  "StartDate": "2022-06-02",
  "LogonLanguageCode": "EN",
  "LastName": "Jensen",
  "PersonExternalID": "BJENSEN",
  "PersonUUID": "FA163EA9-3617-1EEC-B8DA-AD865EF3B625",
  "__ENABLE__": false,
  "TimeZoneCode": "CET",
  "EndDate": "9999-12-31",
  "DateFormatCode": "1"
}
```

OpenICF Interfaces Implemented by the SAP HANA Connector

The SAP HANA Connector implements the following OpenICF interfaces. For additional details, see ICF interfaces:

Create

Creates an object and its `uid`.

Schema

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

Script on Connector

Enables an application to run a script in the context of the connector.

Any script that runs on the connector has the following characteristics:

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

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

Search

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

Test

Tests the connector configuration.

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

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

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

Update

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

SAP HANA Connector Configuration

The SAP HANA Connector has the following configurable properties:

Basic Configuration Properties

Property	Type	Default	Encrypted(1)	Required(2)
`userName`	`String`	`null`		✓ Yes
Provides the Username to authorize the SAP HANA APIs.				
`password`	`GuardedString`	`null`	🔒 Yes	✓ Yes
Provides the Password to authorize the SAP HANA APIs.				
`instanceUrl`	`String`	`null`		✓ Yes
Instance URL of the SAP HANA instance.				
`tenantId`	`String`	`null`		✕ No
Provides the Tenant ID to identify your custom SAP HANA APIs.				

Property	Type	Default	Encrypted[1]	Required[2]
`pageSize`	`Integer`	`1000`		✕ No
Page Size for search operation.				
`httpProxyHost`	`String`	`null`		✕ No
Provides the HTTP Proxy Host.				
`httpProxyPort`	`Integer`	`null`		✕ No
Provides the HTTP Proxy Port.				
`httpProxyUsername`	`String`	`null`		✕ No
Provides the HTTP Proxy Username.				
`httpProxyPassword`	`GuardedString`	`null`	🔒 Yes	✕ No
Provides the HTTP Proxy Password.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

SCIM connector

The SCIM connector is based on the Simple Cloud Identity Management (SCIM) protocol and lets you manage user and group accounts on any SCIM-compliant resource provider, such as Slack or Facebook. The SCIM connector implements both 1.1 and 2.0 endpoints.

> **Important**
>
> Ping strongly recommends that you do *not* use the SCIM connector to connect to Salesforce systems. Use the Salesforce connector.

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

Install the SCIM connector

Tip

To check for an Advanced Identity Cloud application for this connector, refer to:

- Application management
- App catalog

You can download any connector from Backstage, but some are included in the default deployment for Advanced Identity Cloud, IDM, or RCS. When using an included connector, you can skip installing it and move directly to configuration.

Connector included in default deployment

Connector	IDM	RCS
SCIM	✓ Yes	✓ Yes

Download the connector .jar file from Backstage.

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

```
mv ~/Downloads/scim-connector-1.5.20.30.jar /path/to/openidm/connectors/
```

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

Configure the SCIM connector

Tip

SCIM V2 supports a "path" attribute value describing the target of the operation. The attribute value is optional for patch "add" and "replace" operations. If the SCIM resource provider requires this configuration, do one of the following:

- Using the IDM admin UI, enable **Use Path for Patch v2** in the **Additional Options** area of the connector configuration. Refer to the following procedure for additional details on configuring the connector using the UI.
- Add the following to the connector configuration `configurationProperties` :

```
"usePathForPatchV2" : true
```

Create a connector configuration using the IDM admin UI:

1. From the navigation bar, click **Configure > Connectors**.
2. On the **Connectors** page, click **New Connector**.
3. On the **New Connector** page, type a **Connector Name**.
4. From the **Connector Type** drop-down list, select **SCIM Connector - 1.5.20.30**.

5. Complete the **Base Connector Details**.

> **Tip**
>
> For a list of all configuration properties, refer to SCIM Connector Configuration

6. Click **Save**.

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

Refer to this procedure to create a connector configuration over REST.

Configure the SCIM connector using the filesystem

Alternatively, create a connector configuration file in your project's `conf` directory:

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

`OAUTH`	Minimum required properties: `grantType`, `SCIMEndpoint`, `tokenEndpoint`, `clientId`, and `clientSecret`. To change the OAuth2 scope, use the property `scope`.
`BASIC`	Minimum required properties: `user` and `password`.
`TOKEN`	Minimum required property: `authToken`. To change the token prefix, use the property `authorizationTokenPrefix` (default value `Bearer`).

> **Tip**
>
> For a list of all configuration properties, refer to SCIM Connector Configuration.

```
"configurationProperties" : {
    "SCIMEndpoint" : "https://example.com/scim",
    "SCIMVersion" : 1,
    "authenticationMethod" : "OAUTH",
    "user" : null,
    "password" : null,
    "tokenEndpoint" : "https://example.com/oauth2/token",
    "clientId" : "Kdvl...j3fka",
    "clientSecret" : "xxxxxxxxxxxxxxxxxx",
    "acceptSelfSignedCertificates" : true,
    "grantType" : "client_credentials",
    "disableHostNameVerifier" : true,
    "connectionTimeout" : 30,
    "maximumConnections" : 10,
    "httpProxyHost" : null,
    "httpProxyPort" : null
}
```

Note

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

Test the SCIM connector

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

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system?_action=test"
[
  {
    "name": "SCIM",
    "enabled": true,
    "config": "config/provisioner.openicf/SCIM",
    "connectorRef": {
      "bundleName": "org.forgerock.openicf.connectors.scim-connector",
      "connectorName": "org.forgerock.openicf.connectors.scim.ScimConnector",
      "bundleVersion": "[1.5.0.0,1.6.0.0)"
    },
    "displayName": "Scim Connector",
    "objectTypes": [
      "__ACCOUNT__",
      "__ALL__",
      "__GROUP__"
    ],
    "ok": true
  }
]
```

A status of `"ok": true` indicates that the SCIM connector can reach the configured resource provider.

SCIM remote connector

If you want to run this connector outside of PingOne Advanced Identity Cloud or IDM, you can configure the SCIM connector as a remote connector. Java Connectors installed remotely on a Java Connector Server function identically to those bundled locally within PingOne Advanced Identity Cloud or installed locally on IDM.

You can download the SCIM connector from here.

Refer to Remote connectors for configuring the SCIM remote connector.

Configure connection pooling

The SCIM connector supports HTTP pooling, which can substantially improve the performance of the connector. Learn more about the basic connection pooling configuration and different pooling mechanisms described in Connection pooling configuration.

Implementation specifics

For PATCH requests, a connector can potentially add, remove, or replace an attribute value. The SCIM connector doesn’t implement the add or remove operations, so a PATCH request always replaces the entire attribute value with the new value.

Using the SCIM connector with a proxy server

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

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

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

OpenICF Interfaces Implemented by the Scim Connector

The Scim Connector implements the following OpenICF interfaces. For additional details, see ICF interfaces:

Create

Creates an object and its `uid`.

Delete

Deletes an object, referenced by its `uid`.

Schema

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

Script on Connector

Enables an application to run a script in the context of the connector.

Any script that runs on the connector has the following characteristics:

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

Search

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

Sync

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

Test

Tests the connector configuration.

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

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

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

Update

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

Scim Connector Configuration

The Scim Connector has the following configurable properties:

Basic Configuration Properties

Property	Type	Default	Encrypted[1]	Required[2]
`SCIMEndpoint`	`String`	`null`		✓ Yes
The HTTP URL defining the root for the SCIM endpoint (https://myserver.com/service/scim).				

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

Property	Type	Default	Encrypted[1]	Required[2]
To be used for debug/test purposes. To be avoided in production. Defaults to false.				
`disableHostNameVerifier`	`boolean`	`false`		✓ Yes
To be used for debug/test purposes. To be avoided in production. Defaults to false.				
`disableHttpCompression`	`boolean`	`false`		✓ Yes
Content compression is enabled by default. Set this property to true to disable it. Defaults to false.				
`clientCertAlias`	`String`	`null`		✓ Yes
If TLS Mutual Auth is needed, set this to the certificate alias from the keystore.				
`clientCertPassword`	`GuardedString`	`null`	🔒 Yes	✓ Yes
If TLS Mutual Auth is needed and the client certificate (private key) password is different than the keystore password, set this to the client private key password.				
`maximumConnections`	`Integer`	`10`		✓ Yes
Defines the max size of the http connection pool used. Defaults to 10.				
`httpProxyHost`	`String`	`null`		✓ Yes
Defines the Hostname if an HTTP proxy is used between the connector and the SCIM service provider. Defaults to null.				
`httpProxyPort`	`Integer`	`null`		✓ Yes
Defines the Port if an HTTP proxy is used between the connector and the SCIM service provider. Defaults to null.				
`httpProxyUsername`	`String`	`null`		✓ Yes
Defines Proxy Username if an HTTP proxy is used between the connector and the SCIM service provider. Defaults to null.				
`httpProxyPassword`	`GuardedString`	`null`	🔒 Yes	✓ Yes
Defines Proxy Password if an HTTP proxy is used between the connector and the SCIM service provider. Defaults to null.				
`connectionTimeout`	`int`	`30`		✕ No
Defines a timeout for the underlying http connection in seconds. Defaults to 30.				
`authorizationTokenPrefix`	`String`	`Bearer`		✕ No
The prefix to be used in the Authorization HTTP header for Token authentication. Defaults to "Bearer."				
`useBasicAuthForOauthTokenNeg`	`boolean`	`true`		✓ Yes

Property	Type	Default	Encrypted[1]	Required[2]
Client Id and Client Secret are sent in the Header when this is unchecked. Defaults to true.				
`readRateLimit`	`String`	`null`		✕ No
Defines throttling for read operations either per seconds ("30/sec") or per minute ("100/min").				
`acceptHeader`	`String`	`null`		✕ No
The connector is using "application/json" by default. SCIM V2 Service Provider may require "application/scim+json". It can be overwritten with this property.				
`contentTypeHeader`	`String`	`null`		✕ No
The connector is using "application/json" by default. SCIM V2 Service Provider may require "application/scim+json". It can be overwritten with this property.				
`writeRateLimit`	`String`	`null`		✕ No
Defines throttling for write operations (create/update/delete) either per second ("30/sec") or per minute ("100/min").				
`usePathForPatchV2`	`boolean`	`false`		✕ No
If true, then "path" will always be included in Patch add or replace operations. If false, it will only be used for remove operations. Defaults to false.				
`readSchema`	`boolean`	`false`		✕ 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. Defaults to false.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

Scripted REST connector

The Scripted REST connector is an implementation of the Scripted Groovy connector toolkit. It can interact with any REST API, using Groovy scripts for the ICF operations. This connector type lets you develop a fully functional REST-based connector for in-house applications or any cloud-based application not yet supported with the standard set of connectors.

To use this connector, you must write a Groovy script for each operation that you want the connector to perform (create, read, update, delete, authenticate, and so on). No sample scripts are bundled with the connector, but IDM customers have access to the Scripted REST connector source code in the connectors-customers-ga repo. This repository includes sample scripts for all the ICF operations.

You cannot configure the Scripted REST connector through the UI. Configure the connector over REST, as described in Configure Connectors Over REST.

Alternatively, a sample connector configuration and scripts are provided in the `/path/to/openidm/samples/scripted-rest-with-dj/` directory and described in Connect to DS with ScriptedREST. The scripts provided with this sample demonstrate how the connector can be used, but most likely cannot be used as is in your deployment. They are a good starting point from which to base your customization. For information about writing your own scripts, refer to Scripted connectors with Groovy.

Script custom behavior

The Scripted REST connector uses the Apache HTTP client library. Unlike the Scripted SQL connector, which uses JDBC drivers and a Tomcat JDBC connection pool, the Scripted REST connector includes a special script to customize the Apache HTTP client.

This *customizer script* lets you customize the Apache HTTP client connection pool, proxy, default headers, timeouts, and so on.

The customizer script is referenced in the connector configuration, in the `CustomizerScriptFileName` property:

```
{
...
   "configurationProperties": {
       ...
       "customizerScriptFileName": "CustomizerScript.groovy",
       ...
   }
}
```

The script can implement two predefined Groovy closures — `init {}` and `decorate {}`.

`init {}`

The Apache HTTP client provides an HTTPClientBuilder class, to build an instance of the HTTPClient. The Scripted REST connector injects this builder into the `init` closure when the connector is first instantiated. The `init` closure is the ideal place to customize the HTTP client with the builder.

You can customize the following elements of the client:

- Connection pool
- Connection timeouts
- Proxy
- Default HTTP headers
- Certificate handling

```
/**
 * A customizer script defines the custom closures to interact with the default implementation and customize
it.
 * Here, the {@link HttpClientBuilder} is passed to the customize closure. This is where the pooling, the
headers,
 * the timeouts etc... should be defined.
 */
customize {
    init { HttpClientBuilder builder ->

        //SETUP: org.apache.http
        def c = delegate as ScriptedRESTConfiguration
        def httpHost = new HttpHost(c.serviceAddress?.host, c.serviceAddress?.port, c.serviceAddress?.scheme)

        PoolingHttpClientConnectionManager cm = new PoolingHttpClientConnectionManager()
        // Increase max total connection to 200
        cm.setMaxTotal(200)
        // Increase default max connection per route to 20
        cm.setDefaultMaxPerRoute(20)
        // Increase max connections for httpHost to 50
        cm.setMaxPerRoute(new HttpRoute(httpHost), 50)

        builder.setConnectionManager(cm)

        // configure timeout on the entire client
        RequestConfig requestConfig = RequestConfig.custom()/*
                                                             * .
                                                             * setConnectionRequestTimeout
                                                             * ( 50).
                                                             * setConnectTimeout
                                                             * (50)
                                                             * .setSocketTimeout
                                                             * (50)
                                                             */.build();
        builder.setDefaultRequestConfig(requestConfig)

        ...
    }
}
```

Call the builder methods to fit your requirements. The `init{}` closure does not need to return anything.

`decorate {}`

The `init` closure configures a Java instance of the HTTP client, which is injected into every CRUD script. In addition to the libraries provided by the Apache HTTP client, Groovy provides a number of libraries to deal with requests and responses.

The `decorate` closure lets you inject a "decorated" instance of the HTTP client into your scripts. For example, the sample scripts use the `groovyx.net.http.RESTClient` library.

This excerpt of a sample `delete` script shows the injection of the `httpClient` and `connection` variables into the script. The `connection` variable is the output of the `decorate` closure.

```
def operation = operation as OperationType
def configuration = configuration as ScriptedRESTConfiguration
def httpClient = connection as HttpClient
def connection = customizedConnection as RESTClient
def log = log as Log
def objectClass = objectClass as ObjectClass
def options = options as OperationOptions
def uid = uid as Uid

log.info("Entering " + operation + " Script");

switch (objectClass) {
    case ObjectClass.ACCOUNT:
        connection.delete(path: '/api/users/' + uid.uidValue);
        break
    case ObjectClass.GROUP:
        connection.delete(path: '/api/groups/' + uid.uidValue);
}
```

Important

When you use the `defaultRequestHeaders` configuration property to set HTTP request headers, the syntax requires an `=` sign rather than a `:`. For example, to generate a request header such as `"Authorization: Bearer rg1cwAeQJxEf"`, you must set the following value for `defaultRequestHeaders` in the connector configuration:

```
"defaultRequestHeaders" : [ "Authorization = Bearer rg1cwAeQJxEf" ]
```

This example shows how to use the customizer script to implement OAuth2 authentication in the Scripted REST connector.

Although grant types are largely standardized across OAuth2 authentication providers, the way in which different providers handle flows, headers, attribute names, and so on, often differs. This makes it difficult to include a single implementation of OAuth2 authentication in the Scripted REST connector. To make sure that OAuth2 authentication works in your specific use case, you use the customizer script, which can be adapted without requiring a new version of the connector itself.

The Scripted REST connector includes a simple implementation of the OAuth2 Client Credentials grant type. The connector needs to get an access token, using the Client ID and the Client Secret, cache it, and renew it when it expires or when the server revokes it. The Apache client provides *interceptors* for requests and responses. These interceptors can be used in the customizer script to manage the access token:

- *In the request*: If the access token is absent or expired, renew the token and cache it in the Scripted REST connector *property bag*.
- *In the response*: If the server returns a 401 error, delete the Access Token from the connector property bag. This will ensure that the next connector request gets a new access token. The HTTP POST query to get the access token is also handled by the customizer script.

This example shows a complete customizer script for the OAuth2 implementation:

```
init { HttpClientBuilder builder ->

        switch (ScriptedRESTConfiguration.AuthMethod.valueOf(c.defaultAuthMethod)) {
 // ......
            case ScriptedRESTConfiguration.AuthMethod.OAUTH:
                // define a request interceptor to set the Authorization header if absent or expired
                HttpRequestInterceptor requestInterceptor = { HttpRequest request, HttpContext context ->
                    if (null == context.getAttribute("oauth-request")) {
                        def exp = c.propertyBag.tokenExpiration as Long
                        if (c.propertyBag.accessToken == null || exp < System.currentTimeMillis() / 1000) {
                            new NewAccessToken(c).clientCredentials()
                        }
                        request.addHeader(new BasicHeader(HttpHeaders.AUTHORIZATION, "Bearer " +
c.propertyBag.accessToken))
                    }
                }

                // define a response interceptor to catch a 401 response code and delete access token from cache
                HttpResponseInterceptor responseInterceptor = { HttpResponse response, HttpContext context ->
                    if (HttpStatus.SC_UNAUTHORIZED == response.statusLine.statusCode) {
                        if (c.propertyBag.accessToken != null) {
                            c.propertyBag.remove("accessToken")
                            Log.getLog(ScriptedRESTConnector.class).info("Code 401 - accessToken removed")
                        }
                    }
                }

                builder.addInterceptorLast(requestInterceptor)
                builder.addInterceptorLast(responseInterceptor)
                break

            default:
                throw new IllegalArgumentException()
        }
    }

class NewAccessToken {

    static final String  GRANT_TYPE = "grant_type"
    static final String  REFRESH_TOKEN = "refresh_token"
    static final String  CLIENT_CREDENTIALS = "client_credentials"
    static final String  CLIENT_ID = "client_id"
    static final String  CLIENT_SECRET = "client_secret"
    static final String  OAUTH_REQUEST = "oauth-request"

    Log logger = Log.getLog(NewAccessToken.class)
    ScriptedRESTConfiguration c = null
    final CloseableHttpClient client = null
    final HttpPost post = null

    NewAccessToken(ScriptedRESTConfiguration conf) {
        this.c = conf
        this.client = c.getHttpClient()
        this.post = new HttpPost(c.getOAuthTokenEndpoint())
        post.setHeader(HttpHeaders.CONTENT_TYPE, "application/x-www-form-urlencoded")
        post.setHeader(HttpHeaders.ACCEPT, "application/json")
    }

    @Synchronized
    void clientCredentials() {
        boolean expired = (c.propertyBag.tokenExpiration as Long) < System.currentTimeMillis() / 1000
        if (c.propertyBag.accessToken == null || expired ) {
            if (c.propertyBag.tokenExpiration != null && expired) {
                logger.info("Token expired!")
            }
            logger.info("Getting new access token...")

            final List<NameValuePair> pairs = new ArrayList<>()
            pairs.add(new BasicNameValuePair(GRANT_TYPE, CLIENT_CREDENTIALS))
            pairs.add(new BasicNameValuePair(CLIENT_ID, c.getOAuthClientId()))
            pairs.add(new BasicNameValuePair(CLIENT_SECRET, SecurityUtil.decrypt(c.getOAuthClientSecret())))
            post.setEntity(new UrlEncodedFormEntity(pairs))

            CloseableHttpResponse response = null
            try {
                HttpClientContext ctx = HttpClientContext.create()
                ctx.setAttribute(OAUTH_REQUEST, true)
                response = client.execute(post, ctx)
                int statusCode = response.getStatusLine().getStatusCode()
                if (HttpStatus.SC_OK == statusCode) {
                    def jsonSlurper = new JsonSlurper()
                    def oauthResponse = jsonSlurper.parseText(EntityUtils.toString(response.getEntity()))
                    c.propertyBag.accessToken = oauthResponse.access_token
                    c.propertyBag.tokenExpiration = System.currentTimeMillis() / 1000 + oauthResponse.expires_in as
Long
                } else {
                    throw new InvalidCredentialException("Retrieve Access Token failed with code: " + statusCode)
                }
            } catch (ClientProtocolException ex) {
                logger.info("Trace: {0}", ex.getMessage())
                throw new ConnectorException(ex)
            } catch (IOException ex) {
                logger.info("Trace: {0}", ex.getMessage())
                throw new ConnectionFailedException(ex)
            } finally {
                try {
                    if (response != null) {
                        response.close()
                    }
                } catch (IOException e) {
                    logger.info("Can't close HttpResponse")
                }
            }
        }
    }
}
```

Configure connection pooling

The Scripted REST connector supports HTTP pooling, which can substantially improve the performance of the connector. Learn more about the basic connection pooling configuration and different pooling mechanisms described in Connection pooling configuration.

Using the Scripted REST connector with a proxy server

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

To specify the proxy server details, set the `proxyAddress` property in the connector configuration. For example:

```
"configurationProperties": {
    ...
    "proxyAddress": "http://myproxy:8080",
    ...
}
```

Run scripts through the connector

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

runScriptOnConnector

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

configuration

A handler to the connector's configuration object.

options

A handler to the Operation Options.

operation

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

log

A handler to the connector's log.

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

```
"systemActions" : [
    {
        "scriptId" : "MyScript",
        "actions" : [
            {
                "systemType" : ".*ScriptedConnector",
                "actionType" : "groovy",
                "actionFile" : "path/to/<script-name>.groovy"
            }
        ]
    }
]
```

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

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

Note

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

Running `MyScript` will return:

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

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

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

You can also call it through the IDM script engine. Note that the system can accept arbitrary parameters, as demonstrated here:

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

`runScriptOnResource`

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

Define these properties in your provisioner file as follows:

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

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

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

Implemented interfaces

This table lists the ICF interfaces that are implemented for the Scripted REST connector:

OpenICF Interfaces Implemented by the Scripted REST Connector

The Scripted REST Connector implements the following OpenICF interfaces. For additional details, see ICF 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.

Scripted REST Connector Configuration

The Scripted REST Connector has the following configurable properties:

Basic Configuration Properties

Property	Type	Default	Encrypted[1]	Required[2]
`username`	`String`	`null`		✕ No
The Remote user to authenticate with.				
`password`	`GuardedString`	`null`	🔒 Yes	✕ No
The Password to authenticate with.				
`serviceAddress`	`URI`	`null`		✓ Yes
The service URI (example: http://myservice.com/api).				
`proxyAddress`	`URI`	`null`		✕ No
The optional Proxy server URI (example: http://myproxy:8080).				
`proxyUsername`	`String`	`null`		✕ No
The username to authenticate with the proxy server.				
`proxyPassword`	`GuardedString`	`null`	🔒 Yes	✕ No
The password to authenticate with the proxy server.				
`defaultAuthMethod`	`String`	`BASIC`		✕ No
Authentication method used. Can be: BASIC, BASIC_PREEMPTIVE, OAUTH or NONE.				
`defaultContentType`	`String`	`application/json`		✕ No
Default HTTP request content type. Can be: JSON, TEXT, XML, HTML, URLENC, BINARY.				
`defaultRequestHeaders`	`String[]`	`null`		✕ No
Placeholder for default HTTP request headers.				
`OAuthTokenEndpoint`	`URI`	`null`		✕ No
When using OAUTH, this property defines the endpoint where a new access token should be queried for (https://myserver.com/oauth2/token).				
`OAuthClientId`	`String`	`null`		✕ No
The client identifier.				

Property	Type	Default	Encrypted[1]	Required[2]
`OAuthClientSecret`	`GuardedString`	`null`	🔒 Yes	✕ No
Secure client secret for OAUTH.				
`OAuthRefreshToken`	`GuardedString`	`null`	🔒 Yes	✕ No
The refresh token used to renew the access token for the refresh_token grant type.				
`OAuthScope`	`String`	`null`		✕ No
The optional scope.				
`OAuthGrantType`	`String`	`client_credentials`		✕ No
The grant type to use. Can be: `client_credentials` or any grant type supported by the customizer script.				
`readRateLimit`	`String`	`null`		✕ No
Defines throttling for read operations either per seconds ("30/sec") or per minute ("100/min").				
`writeRateLimit`	`String`	`null`		✕ No
Defines throttling for write operations (create/update/delete) either per second ("30/sec") or per minute ("100/min").				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

Groovy Engine configuration

Property	Type	Default	Encrypted[1]	Required[2]
`scriptRoots`	`String[]`	`null`		✓ Yes
The root folder to load the scripts from. If the value is null or empty the classpath value is used.				
`classpath`	`String[]`	`[]`		✕ No
Classpath for use during compilation.				
`debug`	`boolean`	`false`		✕ No
If true, debugging code should be activated.				
`disabledGlobalASTTransformations`	`String[]`	`null`		✕ No

Property	Type	Default	Encrypted[1]	Required[2]
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.				
`minimumRecompilationInterval`	`int`	`100`		✕ No
Sets the minimum of time after a script can be recompiled.				
`recompileGroovySource`	`boolean`	`false`		✕ No
If set to true recompilation is enabled.				
`scriptBaseClass`	`String`	`null`		✕ No
Base class name for scripts (must derive from Script).				
`scriptExtensions`	`String[]`	`['groovy']`		✕ No
Gets the extensions used to find groovy files.				
`sourceEncoding`	`String`	`UTF-8`		✕ No
Encoding for source files.				
`targetDirectory`	`File`	`null`		✕ No
Directory into which to write classes.				
`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.				
`verbose`	`boolean`	`false`		✕ No
If true, the compiler should produce action information.				
`warningLevel`	`int`	`1`		✕ No
Warning Level of the compiler.				
`customConfiguration`	`String`	`null`		✕ No
Custom Configuration script for Groovy ConfigSlurper.				
`customSensitiveConfiguration`	`GuardedString`	`null`	🔒 Yes	✕ No
Custom Sensitive Configuration script for Groovy ConfigSlurper.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

Operation Script Files

Property	Type	Default	Encrypted[1]	Required[2]
`authenticateScriptFileName`	`String`	`null`		• Authenticate
The name of the file used to perform the AUTHENTICATE operation.				
`createScriptFileName`	`String`	`null`		• Create
The name of the file used to perform the CREATE operation.				
`customizerScriptFileName`	`String`	`null`		✕ No
The script used to customize some function of the connector. Read the documentation for more details.				
`deleteScriptFileName`	`String`	`null`		• Delete
The name of the file used to perform the DELETE operation.				
`resolveUsernameScriptFileName`	`String`	`null`		• Resolve Username
The name of the file used to perform the RESOLVE_USERNAME operation.				
`schemaScriptFileName`	`String`	`null`		• Schema
The name of the file used to perform the SCHEMA operation.				
`scriptOnResourceScriptFileName`	`String`	`null`		• Script on Resource
The name of the file used to perform the RUNSCRIPTONRESOURCE operation.				
`searchScriptFileName`	`String`	`null`		• Read • Search
The name of the file used to perform the SEARCH operation.				

Property	Type	Default	Encrypted[1]	Required[2]
`syncScriptFileName`	`String`	`null`		• Sync
The name of the file used to perform the SYNC 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.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

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

Scripted SQL connector

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

To use this connector, you must write a Groovy script for each operation that you want the connector to perform (create, read, update, delete, authenticate, and so on). No sample scripts are bundled with the connector, but IDM customers have access to the Scripted SQL connector source code in the connectors-customers-ga repo. This repository includes sample scripts for all the ICF operations.

Configure the Scripted SQL connector

You cannot configure the Scripted SQL connector through the UI. Configure the connector over REST, as described in Configure Connectors Over REST.

Alternatively, a sample connector configuration and scripts are provided in the `/path/to/openidm/samples/scripted-sql-with-mysql/` directory and described in Connect to a MySQL database with ScriptedSQL. The scripts provided with this sample demonstrate how the connector can be used, but most likely cannot be used as is in your deployment. They are a good starting point from which to base your customization. For information about writing your own scripts, refer to Scripted connectors with Groovy.

Configure connection pooling

The Scripted SQL connector embeds the Apache Tomcat 9 JDBC Connection Pool. Learn more about the different pooling mechanisms in Connectors by pooling mechanism.

Validate pooled connections

Occasionally, a JDBC resource accessed by the scripted SQL connector might become unavailable for a period. When the resource comes back online, IDM is able to recover automatically and resume operations. However, the connector might not be able to refresh its connection pool and might then pass a closed connection to its scripts. This can affect operations until IDM is restarted.

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

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

`testOnBorrow`

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

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

`validationQuery`

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

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

HyperSQL DataBase (HSQLDB)

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

Oracle DB

```
select 1 from dual
```

DB2

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

MySQL

```
select 1
```

Microsoft SQL

```
select 1
```

PostgreSQL

```
select 1
```

Ingres Database

```
select 1
```

Apache Derby

```
values 1
```

H2 Database

```
select 1
```

Firebird SQL

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

`validationInterval`

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

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

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

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

Run scripts through the connector

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

`runScriptOnConnector`

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

`configuration`

A handler to the connector's configuration object.

`options`

A handler to the Operation Options.

`operation`

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

`log`

A handler to the connector's log.

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

```
"systemActions" : [
    {
        "scriptId" : "MyScript",
        "actions" : [
            {
                "systemType" : ".*ScriptedConnector",
                "actionType" : "groovy",
                "actionFile" : "path/to/<script-name>.groovy"
            }
        ]
    }
]
```

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

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

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

Running `MyScript` will return:

```
{
  "actions" : [
    {
      "result": 4
    }
  ]
}
```

If your script accepts parameters, you may supply them in the request body or the query string. For example:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
--data-raw '{"param1":"value1"}'
"http://localhost:8080/openidm/system/groovy?_action=script&scriptId=MyScript&param2=value2"**
```

You can also call it through the IDM script engine. Note that the system can accept arbitrary parameters, as demonstrated here:

```
openidm.action("/system/groovy", "script", {"contentParameter": "value"}, {"scriptId": "MyScript",
"additionalParameter1": "value1", "additionalParameter2": "value2"})
```

runScriptOnResource

To run an arbitrary script using `runScriptOnResource`, you must add some configuration details to your provisioner file. These details include a `scriptOnResourceScriptFileName` which references a script file located in a path contained in the `scriptRoots` array.

Define these properties in your provisioner file as follows:

```
"configurationProperties": {
  "scriptRoots": [
    "path/to/scripts"
  ],
  "scriptOnResourceScriptFileName": "ScriptOnResourceScript.groovy"
},
"systemActions" : [
    {
        "scriptId" : "script-1",
        "actions" : [
            {
                "systemType" : ".*ScriptedConnector",
                "actionType" : "groovy",
                "actionFile" : "path/to/<script-name>.groovy"
            }
        ]
    }
]
```

When you have defined the script, you can call it over REST on the system endpoint, as follows:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system/groovy?
_action=script&scriptId=scriptOnResourceScript&scriptExecuteMode=resource"
```

Implemented interfaces

This table lists the ICF interfaces that are implemented for the scripted SQL connector:

OpenICF Interfaces Implemented by the Scripted SQL Connector

The Scripted SQL Connector implements the following OpenICF interfaces. For additional details, see ICF 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.

Scripted SQL Connector Configuration

The Scripted SQL Connector has the following configurable properties:

Configuration properties

Property	Type	Default	Encrypted[(1)]	Required[(2)]
`password`	`String`	`null`	🔒 Yes	✕ No
The connection password to be passed to our JDBC driver to establish a connection. Note that method DataSource.getConnection(username,password) by default will not use credentials passed into the method, but will use the ones configured here. See alternateUsernameAllowed property for more details.				
`connectionProperties`	`String`	`null`		✕ No
The connection properties that will be sent to our JDBC driver when establishing new connections. Format of the string must be [propertyName=property;]* NOTE - The "user" and "password" properties will be passed explicitly, so they do not need to be included here.				
`propagateInterruptState`	`boolean`	`false`		✕ No
Set this to true to propagate the interrupt state for a thread that has been interrupted (not clearing the interrupt state). Set the value as false for backwards compatibility.				
`useDisposableConnectionFacade`	`boolean`	`true`		✕ No

Property	Type	Default	Encrypted[1]	Required[2]
Set this to true if you wish to put a facade on your connection so that it cannot be reused after it has been closed. This prevents a thread holding on to a reference of a connection it has already called closed on, to execute queries on it.				
`defaultCatalog`	`String`	`null`		✕ No
The default catalog of connections created by this pool.				
`validationInterval`	`long`	`3000`		✕ No
avoid excess validation, only run validation at most at this frequency - time in milliseconds. If a connection is due for validation, but has been validated previously within this interval, it will not be validated again.				
`ignoreExceptionOnPreLoad`	`boolean`	`false`		✕ No
Flag whether ignore error of connection creation while initializing the pool. Set to true if you want to ignore error of connection creation while initializing the pool. Set to false if you want to fail the initialization of the pool by throwing exception.				
`jmxEnabled`	`boolean`	`true`		✕ No
Register the pool with JMX or not.				
`commitOnReturn`	`boolean`	`false`		✕ No
If autoCommit==false then the pool can complete the transaction by calling commit on the connection as it is returned to the pool If rollbackOnReturn==true then this attribute is ignored.				
`logAbandoned`	`boolean`	`false`		✕ No
Flag to log stack traces for application code which abandoned a Connection. Logging of abandoned Connections adds overhead for every Connection borrow because a stack trace has to be generated.				
`maxIdle`	`int`	`100`		✕ No
The maximum number of connections that should be kept in the pool at all times. Idle connections are checked periodically (if enabled) and connections that been idle for longer than minEvictableIdleTimeMillis will be released. (also see testWhileIdle).				
`testWhileIdle`	`boolean`	`false`		✕ No
The indication of whether objects will be validated by the idle object evictor (if any). If an object fails to validate, it will be dropped from the pool. NOTE - for a true value to have any effect, the validationQuery parameter must be set to a non-null string. This property has to be set in order for the pool cleaner/test thread is to run (also see timeBetweenEvictionRunsMillis).				
`removeAbandoned`	`boolean`	`false`		✕ No
Flag to remove abandoned connections if they exceed the removeAbandonedTimeout. If set to true a connection is considered abandoned and eligible for removal if it has been in use longer than the removeAbandonedTimeout Setting this to true can recover db connections from applications that fail to close a connection. See also logAbandoned.				

Property	Type	Default	Encrypted[1]	Required[2]
`abandonWhenPercentageFull`	`int`	`0`		✕ No
Connections that have been abandoned (timed out) wont get closed and reported up unless the number of connections in use are above the percentage defined by abandonWhenPercentageFull. The value should be between 0-100. The value 0 implies that connections are eligible for closure as soon as removeAbandonedTimeout has been reached.				
`minIdle`	`int`	`10`		✕ No
The minimum number of established connections that should be kept in the pool at all times. The connection pool can shrink below this number if validation queries fail (also see testWhileIdle).				
`defaultReadOnly`	`Boolean`	`null`		✕ No
The default read-only state of connections created by this pool. If not set then the setReadOnly method will not be called (Some drivers dont support read only mode, ex: Informix).				
`maxWait`	`int`	`30000`		✕ No
The maximum number of milliseconds that the pool will wait (when there are no available connections) for a connection to be returned before throwing an exception.				
`logValidationErrors`	`boolean`	`false`		✕ No
Set this to true to log errors during the validation phase to the log file. If set to true, errors will be logged as SEVERE. Set the value as false for backwards compatibility.				
`driverClassName`	`String`	`null`		✕ No
The fully qualified Java class name of the JDBC driver to be used. The driver has to be accessible from the same classloader as tomcat-jdbc.jar.				
`name`	`String`	`Tomcat Connection Pool[7-984729287]`		✕ No
Returns the name of the connection pool. By default a JVM unique random name is assigned.				
`useStatementFacade`	`boolean`	`true`		✕ No
If a statement proxy is set, wrap statements so that equals() and hashCode() methods can be called on closed statements.				
`initSQL`	`String`	`null`		✕ No
A custom query to be run when a connection is first created.				
`validationQueryTimeout`	`int`	`-1`		✕ No

Property	Type	Default	Encrypted[1]	Required[2]
The timeout in seconds before a connection validation queries fail. This works by calling java.test_sample.Statement.setQueryTimeout(seconds) on the statement that executes the validationQuery. The pool itself doesnt timeout the query, it is still up to the JDBC driver to enforce query timeouts. A value less than or equal to zero will disable this feature.				
`validationQuery`	`String`	`null`		✕ No
The SQL query that will be used to validate connections from this pool before returning them to the caller. If specified, this query does not have to return any data, it just cant throw a SQLException. Example values are SELECT 1(mysql), select 1 from dual(oracle), SELECT 1(MS Sql Server).				
`rollbackOnReturn`	`boolean`	`false`		✕ No
If autoCommit==false then the pool can terminate the transaction by calling rollback on the connection as it is returned to the pool.				
`alternateUsernameAllowed`	`boolean`	`false`		✕ No
By default, the jdbc-pool will ignore the DataSource.getConnection(username,password) call, and simply return a previously pooled connection under the globally configured properties username and password, for performance reasons. The pool can however be configured to allow use of different credentials each time a connection is requested. To enable the functionality described in the DataSource.getConnection(username,password) call, simply set the property alternateUsernameAllowed to true. Should you request a connection with the credentials user1/password1 and the connection was previously connected using different user2/password2, the connection will be closed, and reopened with the requested credentials. This way, the pool size is still managed on a global level, and not on a per schema level.				
`dataSourceJNDI`	`String`	`null`		✕ No
The JNDI name for a data source to be looked up in JNDI and then used to establish connections to the database. See the dataSource attribute.				
`validatorClassName`	`String`	`null`		✕ No
The name of a class which implements the org.apache.tomcat.jdbc.pool.Validator interface and provides a no-arg constructor (may be implicit). If specified, the class will be used to create a Validator instance which is then used instead of any validation query to validate connections. An example value is com.mycompany.project.SimpleValidator.				
`suspectTimeout`	`int`	`0`		✕ No
Timeout value in seconds. Similar to to the removeAbandonedTimeout value but instead of treating the connection as abandoned, and potentially closing the connection, this simply logs the warning if logAbandoned is set to true. If this value is equal or less than 0, no suspect checking will be performed. Suspect checking only takes place if the timeout value is larger than 0 and the connection was not abandoned or if abandon check is disabled. If a connection is suspect a WARN message gets logged and a JMX notification gets sent once.				
`useEquals`	`boolean`	`true`		✕ No

Property	Type	Default	Encrypted[1]	Required[2]
Set to true if you wish the ProxyConnection class to use String.equals and set to false when you wish to use == when comparing method names. This property does not apply to added interceptors as those are configured individually.				
`removeAbandonedTimeout`	`int`	`60`		✕ No
Timeout in seconds before an abandoned(in use) connection can be removed. The value should be set to the longest running query your applications might have.				
`defaultAutoCommit`	`Boolean`	`null`		✕ No
The default auto-commit state of connections created by this pool. If not set, default is JDBC driver default (If not set then the setAutoCommit method will not be called).				
`testOnConnect`	`boolean`	`false`		✕ No
Validate the connection when connecting to the database for the first time. Set to true if you want to use the validationQuery as an init query.				
`jdbcInterceptors`	`String`	`null`		✕ No
A semicolon separated list of classnames extending org.apache.tomcat.jdbc.pool.JdbcInterceptor class. See Configuring JDBC interceptors below for more detailed description of syntaz and examples. These interceptors will be inserted as an interceptor into the chain of operations on a java.test_sample.Connection object.				
`initialSize`	`int`	`10`		✕ No
The initial number of connections that are created when the pool is started.				
`defaultTransactionIsolation`	`int`	`-1`		✕ No
The default TransactionIsolation state of connections created by this pool. One of the following: NONE, READ_COMMITTED, READ_UNCOMMITTED, REPEATABLE_READ, SERIALIZABLE If not set, the method will not be called and it defaults to the JDBC driver.				
`numTestsPerEvictionRun`	`int`	`0`		✕ No
Property not used in tomcat-jdbc-pool.				
`url`	`String`	`null`		✕ No
The URL used to connect to the database.				
`testOnBorrow`	`boolean`	`false`		✕ No
The indication of whether objects will be validated before being borrowed from the pool. If the object fails to validate, it will be dropped from the pool, and we will attempt to borrow another. NOTE - for a true value to have any effect, the validationQuery parameter must be set to a non-null string. In order to have a more efficient validation, see validationInterval.				

Property	Type	Default	Encrypted[1]	Required[2]
`fairQueue`	`boolean`	`true`		✕ No
Set to true if you wish that calls to getConnection should be treated fairly in a true FIFO fashion. This uses the org.apache.tomcat.jdbc.pool.FairBlockingQueue implementation for the list of the idle connections. This flag is required when you want to use asynchronous connection retrieval. Setting this flag ensures that threads receive connections in the order they arrive. During performance tests, there is a very large difference in how locks and lock waiting is implemented. When fairQueue=true there is a decision making process based on what operating system the system is running. If the system is running on Linux (property os.name=Linux. To disable this Linux specific behavior and still use the fair queue, simply add the property org.apache.tomcat.jdbc.pool.FairBlockingQueue.ignoreOS=true to your system properties before the connection pool classes are loaded.				
`accessToUnderlyingConnectionAllowed`	`boolean`	`true`		✕ No
Property not used. Access can be achieved by calling unwrap on the pooled connection. see javax.test_sample.DataSource interface, or call getConnection through reflection or cast the object as javax.test_sample.PooledConnection.				
`maxAge`	`long`	`0`		✕ No
Time in milliseconds to keep this connection. When a connection is returned to the pool, the pool will check to see if the now - time-when-connected > maxAge has been reached, and if so, it closes the connection rather than returning it to the pool. The value 0 implies that connections will be left open and no age check will be done upon returning the connection to the pool.				
`minEvictableIdleTimeMillis`	`int`	`60000`		✕ No
The minimum amount of time an object may sit idle in the pool before it is eligible for eviction.				
`timeBetweenEvictionRunsMillis`	`int`	`5000`		✕ No
The number of milliseconds to sleep between runs of the idle connection validation/cleaner thread. This value should not be set under 1 second. It dictates how often we check for idle, abandoned connections, and how often we validate idle connections.				
`testOnReturn`	`boolean`	`false`		✕ No
The indication of whether objects will be validated before being returned to the pool. NOTE - for a true value to have any effect, the validationQuery parameter must be set to a non-null string.				
`useLock`	`boolean`	`false`		✕ No
Use a lock when performing operations on the connection object. Set to true if you will use a separate background thread for idle and abandon checking (e.g. JMX clients). If the pool sweeper is enabled, a lock is used, regardless of this setting.				
`maxActive`	`int`	`100`		✕ No
The maximum number of active connections that can be allocated from this pool at the same time.				
`username`	`String`	`null`		✕ No

Property	Type	Default	Encrypted[1]	Required[2]
The connection username to be passed to our JDBC driver to establish a connection. Note that method DataSource.getConnection(username,password) by default will not use credentials passed into the method, but will use the ones configured here. See alternateUsernameAllowed property for more details.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

[2] A list of operations in this column indicates that the property is required for those operations.

Groovy Engine configuration

Property	Type	Default	Encrypted[1]	Required[2]
`scriptRoots`	`String[]`	`null`		✓ Yes
The root folder to load the scripts from. If the value is null or empty the classpath value is used.				
`classpath`	`String[]`	`[]`		✕ No
Classpath for use during compilation.				
`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.				
`minimumRecompilationInterval`	`int`	`100`		✕ No
Sets the minimum of time after a script can be recompiled.				
`recompileGroovySource`	`boolean`	`false`		✕ No
If set to true recompilation is enabled.				
`scriptBaseClass`	`String`	`null`		✕ No
Base class name for scripts (must derive from Script).				
`scriptExtensions`	`String[]`	`['groovy']`		✕ No
Gets the extensions used to find groovy files.				
`sourceEncoding`	`String`	`UTF-8`		✕ No

Property	Type	Default	Encrypted[1]	Required[2]
Encoding for source files.				
`targetDirectory`	`File`	`null`		✕ No
Directory into which to write classes.				
`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.				
`verbose`	`boolean`	`false`		✕ No
If true, the compiler should produce action information.				
`warningLevel`	`int`	`1`		✕ No
Warning Level of the compiler.				
`customConfiguration`	`String`	`null`		✕ No
Custom Configuration script for Groovy ConfigSlurper.				
`customSensitiveConfiguration`	`GuardedString`	`null`	🔒 Yes	✕ No
Custom Sensitive Configuration script for Groovy ConfigSlurper.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

[2] A list of operations in this column indicates that the property is required for those operations.

Operation Script Files

Property	Type	Default	Encrypted[1]	Required[2]
`authenticateScriptFileName`	`String`	`null`		• Authenticate
The name of the file used to perform the AUTHENTICATE operation.				
`createScriptFileName`	`String`	`null`		• Create
The name of the file used to perform the CREATE operation.				
`customizerScriptFileName`	`String`	`null`		✕ No
The script used to customize some function of the connector. Read the documentation for more details.				

Property	Type	Default	Encrypted[1]	Required[2]
`deleteScriptFileName`	`String`	`null`		• Delete
The name of the file used to perform the DELETE operation.				
`resolveUsernameScriptFileName`	`String`	`null`		• Resolve Username
The name of the file used to perform the RESOLVE_USERNAME operation.				
`schemaScriptFileName`	`String`	`null`		• Schema
The name of the file used to perform the SCHEMA operation.				
`scriptOnResourceScriptFileName`	`String`	`null`		• Script on Resource
The name of the file used to perform the RUNSCRIPTONRESOURCE operation.				
`searchScriptFileName`	`String`	`null`		• Read • Search
The name of the file used to perform the SEARCH operation.				
`syncScriptFileName`	`String`	`null`		• Sync
The name of the file used to perform the SYNC 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.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

[2] A list of operations in this column indicates that the property is required for those operations.

ServiceNow connector

This connector lets you manage objects in the ServiceNow platform, integrating with ServiceNow's REST API.

Before you start

The connector requires a ServiceNow instance with OAuth enabled. You might need to activate the OAuth plugin and set the OAuth activation property if OAuth is not yet enabled on your ServiceNow instance. For more information, refer to the ServiceNow documentation that corresponds to your ServiceNow version.

When Oauth is enabled, register an OAuth client application for the connection to IDM. Take note of the `client_id` and `client_secret` of the application, as you need these values when you configure the connector.

The connector configuration must include a ServiceNow user who has the following roles:

- `admin`
- `rest_api_explorer`

If you don't want to give complete `admin` rights to this user, you can create a new role that provides access to the following tables:

- `sys_audit`
- `sys_group_has_role`
- `sys_user`
- `sys_user_has_role`
- `sys_user_grmember`
- `sys_user_delegate`
- `sys_user_role`
- `sys_user_group`
- `core_company`
- `cmn_department`
- `cmn_cost_center`
- `cmn_location`

Install the ServiceNow connector

> **Tip**
>
> To check for an Advanced Identity Cloud application for this connector, refer to:
>
> - Application management
> - App catalog

You can download any connector from Backstage, but some are included in the default deployment for Advanced Identity Cloud, IDM, or RCS. When using an included connector, you can skip installing it and move directly to configuration.

Connector included in default deployment

Connector	IDM	RCS
ServiceNow	✓ Yes	✕ No

Download the connector .jar file from Backstage.

- If you are running the connector locally, place it in the `/path/to/openidm/connectors` directory, for example:

```
mv ~/Downloads/servicenow-connector-1.5.20.30.jar /path/to/openidm/connectors/
```

- If you are using a remote connector server (RCS), place it in the `/path/to/openicf/connectors` directory on the RCS.

Configure the ServiceNow connector

Create a connector configuration using the IDM admin UI:

1. From the navigation bar, click **Configure > Connectors**.
2. On the **Connectors** page, click **New Connector**.
3. On the **New Connector** page, type a **Connector Name**.
4. From the **Connector Type** drop-down list, select **ServiceNow Connector - 1.5.20.30**.
5. Complete the **Base Connector Details**.

   > **Tip**
   >
   > For a list of all configuration properties, refer to ServiceNow Connector Configuration

6. Click **Save**.

When your connector is configured correctly, the connector displays as **Active** in the admin UI.

Refer to this procedure to create a connector configuration over REST.

The following excerpt of connector configuration shows the required `configurationProperties`:

```
"configurationProperties" : {
    "instance" : "example.service-now.com/",
    "username" : "admin",
    "password" : {encrypted-password},
    "clientID" : "4xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxee",
    "clientSecret" : {encrypted-client-secret},
    "readSchema" : false
}
```

`instance` *(string)*

The ServiceNow instance URL; for example, `example.service-now.com/`.

`username` *(string)*

The name of a ServiceNow user with the `admin` and `rest_api_explorer` roles.

`password` *(string)*

The password of the ServiceNow user.

`clientID` *(string)*

The ID of your OAuth application.

`clientSecret` *(string)*

The client secret of your OAuth application.

IDM encrypts the value of the `password` and `clientSecret` on startup.

Test the ServiceNow connector

When your connector is configured correctly, test its status by running the following command:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system?_action=test"
[
  {
    "name": "serviceNow",
    "enabled": true,
    "config": "config/provisioner.openicf/serviceNow",
    "connectorRef": {
      "bundleVersion": "[1.5.0.0,1.6.0.0)",
      "bundleName": "org.forgerock.openicf.connectors.servicenow-connector",
      "connectorName": "org.forgerock.openicf.connectors.servicenow.ServiceNowConnector"
    },
    "displayName": "ServiceNow Connector",
    "objectTypes": [
      "delegate",
      "role",
      "__ALL__",
      "costCenter",
      "location",
      "company",
      "userHasGroup",
      "department",
      "__ACCOUNT__",
      "userHasRole",
      "group"
    ],
    "ok": true
  }
]
```

A status of `"ok": true` indicates that the ServiceNow connector can reach the configured resource provider.

ServiceNow remote connector

If you want to run this connector outside of PingOne Advanced Identity Cloud or IDM, you can configure the ServiceNow connector as a remote connector. Java Connectors installed remotely on a Java Connector Server function identically to those bundled locally within PingOne Advanced Identity Cloud or installed locally on IDM.

You can download the ServiceNow connector from here.

Refer to Remote connectors for configuring the ServiceNow remote connector.

Configure connection pooling

The ServiceNow connector uses ICF pooling to manage connections. Learn more about the different pooling mechanisms in Connectors by pooling mechanism.

Manage users with the ServiceNow connector

These examples show the basic CRUD operations using the ServiceNow connector.

Query all ServiceNow users

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
"http://localhost:8080/openidm/system/serviceNow/__ACCOUNT__?_queryId=query-all-ids"
{
  "result": [
    {
      "_id": "02826bf03710200044e0bfc8bcbe5d3f",
      "__NAME__": "lucius.bagnoli@example.com"
    },
    {
      "_id": "02826bf03710200044e0bfc8bcbe5d55",
      "__NAME__": "jimmie.barninger@example.com"
    },
    {
      "_id": "02826bf03710200044e0bfc8bcbe5d5e",
      "__NAME__": "melinda.carleton@example.com"
    },
    ...
  ],
  "resultCount": 578,
  "pagedResultsCookie": null,
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

Query a single ServiceNow user

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
"http://localhost:8080/openidm/system/serviceNow/__ACCOUNT__/02826bf03710200044e0bfc8bcbe5d3f"
{
  "_id": "02826bf03710200044e0bfc8bcbe5d3f",
  "internal_integration_user": false,
  "department": "5d7f17f03710200044e0bfc8bcbe5d43",
  "sys_mod_count": "5",
  "location": "0002c0a93790200044e0bfc8bcbe5df5",
  "web_service_access_only": false,
  "sys_updated_on": "2018-02-25 16:42:47",
  "sys_domain": "global",
  "notification": "2",
  "sys_created_by": "admin",
  "locked_out": "false",
  "__NAME__": "lucius.bagnoli@example.com",
  "company": "81fd65ecac1d55eb42a426568fc87a63",
  "sys_domain_path": "/",
  "password_needs_reset": "false",
  "active": "true",
  "gender": "Male",
  "sys_created_on": "2012-02-18 03:04:49",
  "sys_class_name": "sys_user",
  "calendar_integration": "1",
  "email": "lucius.bagnoli@example.com",
  "sys_id": "02826bf03710200044e0bfc8bcbe5d3f",
  "user_password": "md5230ls7L",
  "user_name": "lucius.bagnoli",
  "sys_updated_by": "developer.program@snc",
  "vip": "false",
  "last_name": "Bagnoli",
  "first_name": "Lucius"
}
```

Create a ServiceNow user

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request POST \
--data '{
  "__NAME__": "bjensen@example.com",
  "first_name": "Barbara",
  "last_name": "Jensen",
  "email": "bjensen@example.com",
  "phone": "555-123-1234"
}' \
"http://localhost:8080/openidm/system/serviceNow/__ACCOUNT__?_action=create"
{
  "_id": "4116e0690fa01300f6af65ba32050e7a",
  "sys_mod_count": "0",
  "password_needs_reset": "false",
  "notification": "2",
  "locked_out": "false",
  "phone": "555-123-1234",
  "sys_created_on": "2018-02-27 13:33:38",
  "first_name": "Barbara",
  "email": "bjensen@example.com",
  "active": "true",
  "sys_domain": "global",
  "calendar_integration": "1",
  "web_service_access_only": false,
  "vip": "false",
  "sys_id": "4116e0690fa01300f6af65ba32050e7a",
  "sys_updated_on": "2018-02-27 13:33:38",
  "sys_domain_path": "/",
  "sys_created_by": "admin",
  "sys_class_name": "sys_user",
  "last_name": "Jensen",
  "__NAME__": "bjensen@example.com",
  "sys_updated_by": "admin",
  "internal_integration_user": false
}
```

Update a ServiceNow user

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--header "If-Match:*" \
--request PUT \
--data '{
  "__NAME__": "bjensen@example.com",
  "first_name": "Barbara",
  "last_name": "Jensen",
  "email": "bjensen@example.com",
  "phone": "555-000-0000"
}' \
"http://localhost:8080/openidm/system/serviceNow/__ACCOUNT__/4116e0690fa01300f6af65ba32050e7a"
{
  "_id": "4116e0690fa01300f6af65ba32050e7a",
  "sys_mod_count": "1",
  "password_needs_reset": "false",
  "notification": "2",
  "locked_out": "false",
  "phone": "555-000-0000",
  "sys_created_on": "2018-02-27 13:33:38",
  "first_name": "Barbara",
  "email": "bjensen@example.com",
  "active": "true",
  "sys_domain": "global",
  "calendar_integration": "1",
  "web_service_access_only": false,
  "vip": "false",
  "sys_id": "4116e0690fa01300f6af65ba32050e7a",
  "sys_updated_on": "2018-02-27 13:35:32",
  "sys_domain_path": "/",
  "sys_created_by": "admin",
  "sys_class_name": "sys_user",
  "last_name": "Jensen",
  "__NAME__": "bjensen@example.com",
  "sys_updated_by": "admin",
  "internal_integration_user": false
}
```

Delete a ServiceNow user

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "If-Match:*" \
--request DELETE \
"http://localhost:8080/openidm/system/serviceNow/__ACCOUNT__/4116e0690fa01300f6af65ba32050e7a"
{
  "_id": "4116e0690fa01300f6af65ba32050e7a",
  "sys_mod_count": "1",
  "password_needs_reset": "false",
  "notification": "2",
  "locked_out": "false",
  "phone": "555-000-0000",
  "sys_created_on": "2018-02-27 13:33:38",
  "first_name": "Barbara",
  "email": "bjensen@example.com",
  "active": "true",
  "sys_domain": "global",
  "calendar_integration": "1",
  "web_service_access_only": false,
  "vip": "false",
  "sys_id": "4116e0690fa01300f6af65ba32050e7a",
  "sys_updated_on": "2018-02-27 13:35:32",
  "sys_domain_path": "/",
  "sys_created_by": "admin",
  "sys_class_name": "sys_user",
  "last_name": "Jensen",
  "__NAME__": "bjensen@example.com",
  "sys_updated_by": "admin",
  "internal_integration_user": false
}
```

Synchronize ServiceNow users

The ServiceNow connector supports bidirectional reconciliation and liveSync. To set up user synchronization, create a mapping between managed users and ServiceNow users.

This example assumes that you have configured a mapping. The example runs a reconciliation operation from ServiceNow to the managed user repository:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/recon?_action=recon&mapping=systemServicenowUser_managedUser"
{
  "_id": "19755e51-5c3b-4362-b316-601856cb282c-13624",
  "state": "ACTIVE"
}
```

The following example runs a liveSync operation from ServiceNow to the managed user repository:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system/serviceNow/__ACCOUNT__?_action=liveSync"
{
  "connectorData": {
    "nativeType": "string",
    "syncToken": "2018-02-275 11:29:15"
  },
  "_rev": "0000000031285d9b",
  "_id": "SYSTEMSERVICENOWUSER"
}
```

Note

The ServiceNow connector does not support the `__ALL__` object type, so you must specify the object type (for example, `__ACCOUNT__`) in your liveSync operation.

Implementation specifics

For PATCH requests, a connector can potentially add, remove, or replace an attribute value. The ServiceNow connector does not implement the add or remove operations, so a PATCH request always replaces the entire attribute value with the new value.

OpenICF Interfaces Implemented by the ServiceNow Connector

The ServiceNow Connector implements the following OpenICF interfaces. For additional details, see ICF interfaces:

Create

Creates an object and its `uid`.

Delete

Deletes an object, referenced by its `uid`.

Schema

Describes the object types, operations, and options that the connector supports.

Script on Connector

Enables an application to run a script in the context of the connector.

Any script that runs on the connector has the following characteristics:

- The script runs in the same execution environment as the connector and has access to all the classes to which the connector has access.
- The script has access to a `connector` variable that is equivalent to an initialized instance of the connector. At a minimum, the script can access the connector configuration.
- The script has access to any script arguments passed in by the application.

Search

Searches the target resource for all objects that match the specified object class and filter.

Sync

Polls the target resource for synchronization events, that is, native changes to objects on the target resource.

Test

Tests the connector configuration.

Testing a configuration checks all elements of the environment that are referred to by the configuration are available. For example, the connector might make a physical connection to a host that is specified in the configuration to verify that it exists and that the credentials that are specified in the configuration are valid.

This operation might need to connect to a resource, and, as such, might take some time. Do not invoke this operation too often, such as before every provisioning operation. The test operation is not intended to check that the connector is alive (that is, that its physical connection to the resource has not timed out).

You can invoke the test operation before a connector configuration has been validated.

Update

Updates (modifies or replaces) objects on a target resource.

ServiceNow Connector Configuration

The ServiceNow Connector has the following configurable properties:

Basic Configuration Properties

Property	Type	Default	Encrypted[1]	Required[2]
`instance`	`String`	`null`		✓ Yes
URL of the ServiceNow instance, for example: dev00000.service-now.com.				
`username`	`String`	`null`		✓ Yes
An API user in ServiceNow that can consume the REST API.				
`password`	`GuardedString`	`null`	🔒 Yes	✓ Yes
Password for the user.				
`clientID`	`String`	`null`		✓ Yes
Client ID of the OAuth application in ServiceNow.				
`clientSecret`	`GuardedString`	`null`	🔒 Yes	✓ Yes
Client Secret for the preceding Client ID.				
`pageSize`	`int`	`100`		✗ No
Default page size.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

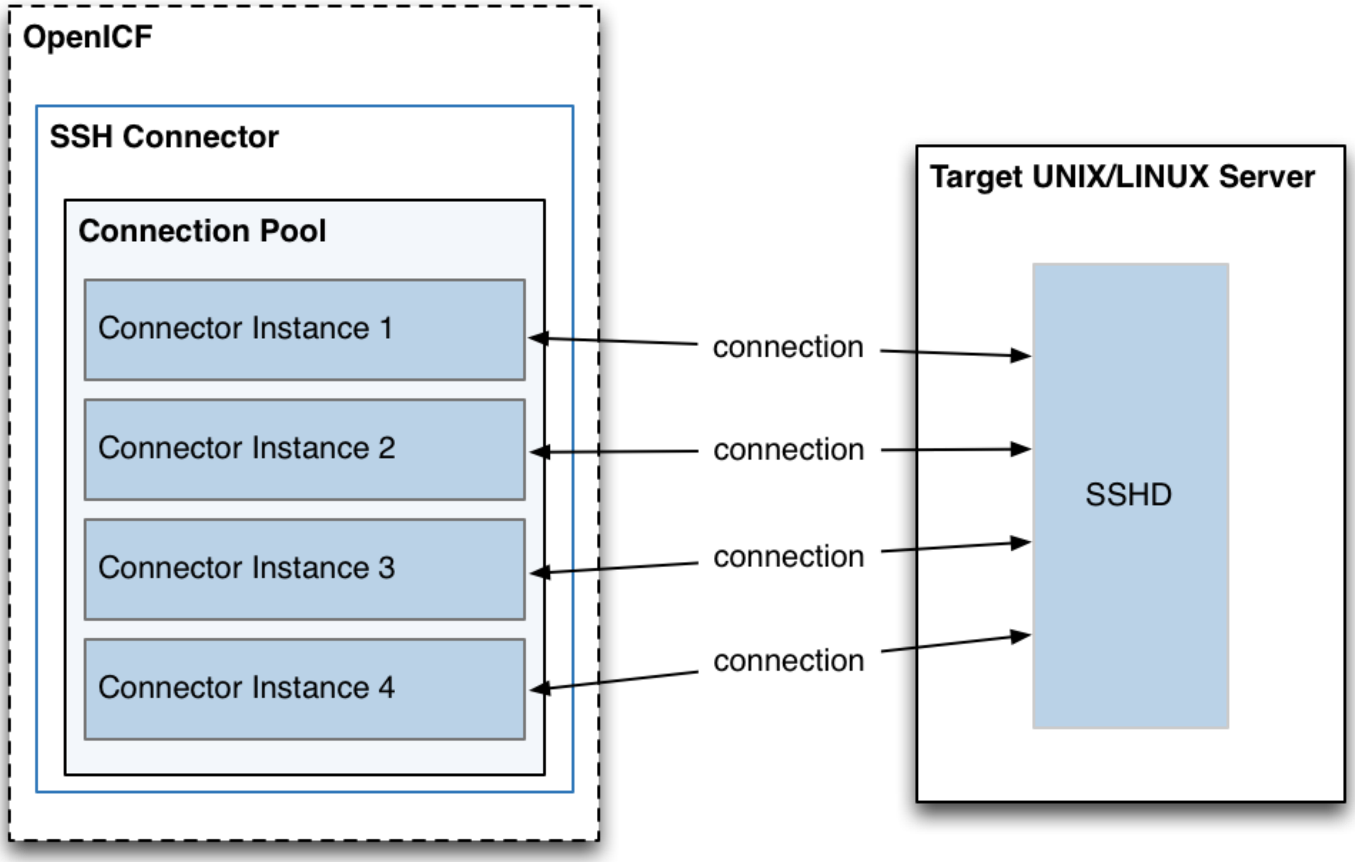
[2] A list of operations in this column indicates that the property is required for those operations.

SSH connector

The SSH connector is an implementation of the Scripted Groovy connector toolkit based on Java Secure Channel (JSch) and the Java implementation of the Expect library (Expect4j). This connector lets you interact with any SSH server, using Groovy scripts for the ICF operations.

The SSH connector is a *poolable connector*. This means that each connector instance is placed into a connection pool every time an action is completed. Subsequent actions can re-use connector instances from the connector pool. When a new connector instance is created, a new SSH client connection is created against the target SSH server. This SSH connection remains open as long as the connector instance is in the connection pool. Note that when a new action is performed, it finds the SSH connection in the exact state that it was left by the previous action.

The following image shows the relationship between SSH connector instances and SSH connections to the target server:



Configure 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.

Authenticate with a login and password

To authenticate with a login and password, set the `authenticationType` to `PASSWORD` in the connector configuration file, and set a `user` and `password`. For example:

```
"configurationProperties" : {
    ...
    "authenticationType" : "PASSWORD",
    "user" : "<USERNAME>",
    "password" : "<PASSWORD>",
    ...
```

The password is encrypted when IDM loads the provisioner file.

Authenticate with a passphrase and private key

To authenticate with a secure certificate, generate a pair of public/private keys. Install the public key on the server side and the private key on the IDM host (where the connector is located). Set the `authenticationType` to `PUBKEY` in the connector configuration file and set the `user`, `password`, `passphrase` and `privateKey` properties. For example:

```
"configurationProperties" : {
    ...
    "authenticationType" : "PUBKEY",
    "user" : "<USERNAME>",
    "password" : "<PASSWORD>",
    "passphrase" : "secret",
    "privateKey" : ["-----BEGIN DSA PRIVATE KEY-----",
              "MIIBugIBAAKBgQDcB0ztVMCFptpJhqlLNZSdN/5cDL3S7aOVy52Ae7vwwCqQPCQr",
              "6NyUk+wtkDr07NlYd3sg7a9hbsEnlYChsuX+/WUIvbOKdMfeqcQ+jKK26YdkTCGj",
              "g86dBj9JYhobSHDoQ9ov31pYN/cfW5BAZwkm9TdpEjHPvMIaOxx7GPGKWwIVALbD",
              "CEuf1yJk9UB7v0dmJS7bKkbxAoGARcbAuDP4rB6MsgAAkVwf+1sHXEiGPShYWrVV",
              "qBgCZ/S45ELqUuiaN/1N/nip/Cc/0SBPKqwl7o50CUg9GH9kTAjmXiwmbkwvtUv+",
              "Xjn5vCHS0w18yc3rGwyr2wj+D9KtDLFJ8+T5HmsbPoDQ3mIZ9xPmRQuRFfVMd9wr",
              "DY0Rs7cCgYAxjGjWDSKThowsvOUCiE0ySz6tWggHH3LTrS4Mfh2t0tnbUfrXq2cw",
              "3CN+T6brgnpYbyX5XI17p859C+cw90MD8N6vvBxaN8QMDRFk+hHNUeSy8gXeem9x",
              "O0vdIxCgKvA4dh5nSVb5VGKENEGNEHRlYxEPzbqlPa/C/ZvzIvdKXQIUQMoidPFC",
              "n9z+mE2dAADnPf2m9vk=",
              "-----END DSA PRIVATE KEY-----"
             ],
    ...
```

The default value for the `passphrase` property is `null`. If you do not set a passphrase for the private key, the passphrase value must be equal to an empty string.

You *must* set a value for the `password` property, because the connector uses sudo to perform actions on the SSH server.

The private key (PEM certificate) must be defined as a JSON String array.

The values of the `passphrase`, `password` and `privateKey` are encrypted when IDM loads the provisioner file.

Install the SSH connector

Tip

To check for an Advanced Identity Cloud application for this connector, refer to:

- Application management
- App catalog

You can download any connector from Backstage, but some are included in the default deployment for Advanced Identity Cloud, IDM, or RCS. When using an included connector, you can skip installing it and move directly to configuration.

Connector included in default deployment

Connector	IDM	RCS
SSH	✓ Yes	✓ Yes

Download the connector .jar file from Backstage.

- If you are running the connector locally, place it in the `/path/to/openidm/connectors` directory, for example:

```
mv ~/Downloads/ssh-connector-1.5.20.28.jar /path/to/openidm/connectors/
```

- If you are using a remote connector server (RCS), place it in the `/path/to/openicf/connectors` directory on the RCS.

Configure the SSH connector

You cannot configure the SSH connector through the UI. Configure the connector over REST, as described in Configure Connectors Over REST.

Alternatively, copy the sample connector configuration file (`/path/to/openidm/samples/example-configurations/provisioners/provisioner.openicf-ssh.json`) to your project's `conf/` directory, and edit it to match your environment.

Set the authentication properties, as described in Configure 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.

This list describes the required configuration properties of the SSH connector. Typically, you can use the default values. For a list of all the configuration properties, refer to SSH Connector Configuration

`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, refer to 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, refer to Configure 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 perform the ICF operations, relative to your IDM installation directory. The sample connector configuration expects the scripts in `project-dir/tools`, so this parameter is set to `&{idm.instance.dir}/tools` in the sample configuration.

`classpath`

The directory in which the compiler should look for compiled classes. The default classpath, if not is specified, is `install-dir/lib`.

`*ScriptFileName`

The name of the Groovy script that is used for each ICF operation.

SSH remote connector

If you want to run this connector outside of PingOne Advanced Identity Cloud or IDM, you can configure the SSH connector as a remote connector. Java Connectors installed remotely on a Java Connector Server function identically to those bundled locally within PingOne Advanced Identity Cloud or installed locally on IDM.

You can download the SSH connector from here.

Refer to Remote connectors for configuring the SSH remote connector.

Configure connection pooling

The SSH connector uses ICF pooling to manage connections. Learn more about the different pooling mechanisms in Connectors by pooling mechanism.

OpenICF Interfaces Implemented by the SSH Connector

The SSH Connector implements the following OpenICF interfaces. For additional details, see ICF interfaces:

Authenticate

Provides simple authentication with two parameters, presumed to be a user name and password.

Create

Creates an object and its `uid`.

Delete

Deletes an object, referenced by its `uid`.

Resolve Username

Resolves an object by its username and returns the `uid` of the object.

Schema

Describes the object types, operations, and options that the connector supports.

Script on Connector

Enables an application to run a script in the context of the connector.

Any script that runs on the connector has the following characteristics:

- The script runs in the same execution environment as the connector and has access to all the classes to which the connector has access.
- The script has access to a `connector` variable that is equivalent to an initialized instance of the connector. At a minimum, the script can access the connector configuration.
- The script has access to any script arguments passed in by the application.

Script on Resource

Runs a script on the target resource that is managed by this connector.

Search

Searches the target resource for all objects that match the specified object class and filter.

Sync

Polls the target resource for synchronization events, that is, native changes to objects on the target resource.

Test

Tests the connector configuration.

Testing a configuration checks all elements of the environment that are referred to by the configuration are available. For example, the connector might make a physical connection to a host that is specified in the configuration to verify that it exists and that the credentials that are specified in the configuration are valid.

This operation might need to connect to a resource, and, as such, might take some time. Do not invoke this operation too often, such as before every provisioning operation. The test operation is not intended to check that the connector is alive (that is, that its physical connection to the resource has not timed out).

You can invoke the test operation before a connector configuration has been validated.

Update

Updates (modifies or replaces) objects on a target resource.

SSH Connector Configuration

The SSH Connector has the following configurable properties:

Basic Configuration Properties

Property	Type	Default	Encrypted(1)	Required(2)
`host`	`String`	`null`		✓ Yes
The hostname to connect to.				
`port`	`int`	`22`		✓ Yes
TCP port to use.				
`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.				
`prompt`	`String`	`root@localhost:# `		✓ Yes
A string representing the remote SSH session prompt.				
`sudoCommand`	`String`	`/usr/bin/sudo`		✓ Yes
A string representing the sudo command.				
`echoOff`	`boolean`	`true`		✓ Yes
Disable the input command echo.				
`terminalType`	`String`	`vt102`		✓ Yes
Defines the terminal type to use for the session.				
`locale`	`String`	`en_US.utf8`		✓ Yes

Property	Type	Default	Encrypted[1]	Required[2]
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.				
`connectionTimeout`	`int`	`5000`		✓ Yes
Defines the connection timeout to the remote server in milliseconds.				
`expectTimeout`	`long`	`5000`		✓ Yes
Defines the timeout used by the expect() calls in the scripts in milliseconds.				
`throwOperationTimeoutException`	`boolean`	`true`		✓ Yes
Defines if an OperationTimeoutException should be thrown if any call to expect times out.				
`promptReadyTimeout`	`long`	`20`		✕ No
Defines the "prompt ready" timeout for the promptReady() command in milliseconds.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

[2] A list of operations in this column indicates that the property is required for those operations.

Groovy Engine configuration

Property	Type	Default	Encrypted[1]	Required[2]
`scriptRoots`	`String[]`	`null`		✓ Yes
The root folder to load the scripts from. If the value is null or empty the classpath value is used.				
`classpath`	`String[]`	`[]`		✕ No
Classpath for use during compilation.				
`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.				

Property	Type	Default	Encrypted[1]	Required[2]
`minimumRecompilationInterval`	`int`	`100`		✕ No
Sets the minimum of time after a script can be recompiled.				
`recompileGroovySource`	`boolean`	`false`		✕ No
If set to true recompilation is enabled.				
`scriptBaseClass`	`String`	`null`		✕ No
Base class name for scripts (must derive from Script).				
`scriptExtensions`	`String[]`	`['groovy']`		✕ No
Gets the extensions used to find groovy files.				
`sourceEncoding`	`String`	`UTF-8`		✕ No
Encoding for source files.				
`targetDirectory`	`File`	`null`		✕ No
Directory into which to write classes.				
`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.				
`verbose`	`boolean`	`false`		✕ No
If true, the compiler should produce action information.				
`warningLevel`	`int`	`1`		✕ No
Warning Level of the compiler.				
`customConfiguration`	`String`	`null`		✕ No
Custom Configuration script for Groovy ConfigSlurper.				
`customSensitiveConfiguration`	`GuardedString`	`null`	🔒 Yes	✕ No
Custom Sensitive Configuration script for Groovy ConfigSlurper.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

[2] A list of operations in this column indicates that the property is required for those operations.

Operation Script Files

Property	Type	Default	Encrypted[1]	Required[2]
`authenticateScriptFileName`	`String`	`null`		• Authenticate
The name of the file used to perform the AUTHENTICATE operation.				
`createScriptFileName`	`String`	`null`		• Create
The name of the file used to perform the CREATE operation.				
`customizerScriptFileName`	`String`	`null`		✕ No
The script used to customize some function of the connector. Read the documentation for more details.				
`deleteScriptFileName`	`String`	`null`		• Delete
The name of the file used to perform the DELETE operation.				
`resolveUsernameScriptFileName`	`String`	`null`		• Resolve Username
The name of the file used to perform the RESOLVE_USERNAME operation.				
`schemaScriptFileName`	`String`	`null`		• Schema
The name of the file used to perform the SCHEMA operation.				
`scriptOnResourceScriptFileName`	`String`	`null`		• Script on Resource
The name of the file used to perform the RUNSCRIPTONRESOURCE operation.				
`searchScriptFileName`	`String`	`null`		• Read • Search
The name of the file used to perform the SEARCH operation.				

Property	Type	Default	Encrypted[1]	Required[2]
`syncScriptFileName`	`String`	`null`		- Sync
The name of the file used to perform the SYNC 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.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

[2] A list of operations in this column indicates that the property is required for those operations.

SAP SuccessFactors connector

The SAP SuccessFactors connector enables you to manage and synchronize objects between SuccessFactors and the IDM managed user repository. These instructions assume you have a SuccessFactors administrator account, and you have created an OAuth2 Client Application.

Before you start

Before you configure the connector, gather the following details:

Host

The SuccessFactors API hostname. For example, `apisalesdemo2.successfactors.eu`.

Client ID

The SuccessFactors API Key or client ID. To locate this value:

1. Log in to your SuccessFactors administrator account.
2. Click **Manage OAuth2 Client Applications**.
3. Select your registered **OAuth2 Client Application**.
4. Click **View**.
5. Copy the **API key**.

User ID

The API User ID of the SuccessFactors user who authenticates to the REST server.

Private Key

A private key. To configure this, generate a key pair from the X.509 certificate and copy the value of the private key.

Company ID

The API Company ID of the admin user. This is specified in the SuccessFactors login URL.

Person Segments (optional)

SuccessFactors person segments; for example, `EmpJob` , `EmpEmployment` , `PerPersonal` . You can add segments and their attributes that are not provided by the connector by default.

- If the person segment field is empty, the following default segments are added to the schema:
  - `EmpEmployment`
  - `EmpJob`
  - `PerPerson`
  - `PerPersonal`
- If the person segment field contains `EmpJob_All` , `PerPersonal_All` , `PerPhone_All` , all attributes of the applicable segment(s) `EmpJob` , `PerPersonal` , and `PerPhone` are added to the schema, respectively.
- If the person segment field contains `PerPhone_phoneNumber` , `PerPhone_secondaryPhoneNumber` , `PerEmail_emailAddress` , `PerEmail_secondaryEmailAddress` , the applicable attributes of the segment(s) `EmpJob` , `PerPersonal` , and `PerPhone` are added to the schema.

> **Note**
>
> Add segment nav-specific attributes in `EmpJob_locationNav` and `PerEmail_emailTypeNav`.

Install the SuccessFactors connector

> **Tip**
>
> To check for an Advanced Identity Cloud application for this connector, refer to:
>
> - Application management
> - App catalog

You can download any connector from Backstage, but some are included in the default deployment for Advanced Identity Cloud, IDM, or RCS. When using an included connector, you can skip installing it and move directly to configuration.

Connector included in default deployment

Connector	IDM	RCS
SAP SuccessFactors	✕ No	✕ No

Download the connector .jar file from Backstage.

- If you are running the connector locally, place it in the `/path/to/openidm/connectors` directory, for example:

```
mv ~/Downloads/successfactors-connector-1.5.20.29.jar /path/to/openidm/connectors/
```

- If you are using a remote connector server (RCS), place it in the `/path/to/openicf/connectors` directory on the RCS.

Configure the SuccessFactors connector

Create a connector configuration using the IDM admin UI:

1. From the navigation bar, click **Configure > Connectors**.
2. On the **Connectors** page, click **New Connector**.
3. On the **New Connector** page, type a **Connector Name**.
4. From the **Connector Type** drop-down list, select **SuccessFactors Connector - 1.5.20.29**.
5. Complete the **Base Connector Details**.

> **Tip**
>
> For a list of all configuration properties, refer to SuccessFactors Connector Configuration

6. Click **Save**.

When your connector is configured correctly, the connector displays as **Active** in the admin UI.

Refer to this procedure to create a connector configuration over REST.

Sample Configuration

```
"configurationProperties" : {
    "host" : null,
    "clientId" : null,
    "userId" : null,
    "privateKey" : null,
    "companyId" : null,
    "personSegments" : "EmpJob,PerPersonal,EmpEmployment",
    "pageSize" : 0,
    "maximumConnections" : 10,
    "connectionTimeout" : 600,
    "httpProxyHost" : null,
    "httpProxyPort" : null,
    "httpProxyUsername" : null,
    "httpProxyPassword" : null
}
```

> **Tip**
>
> For more information, refer to:
>
> - Before you start
> - Configuration properties

Configure connection pooling

The SuccessFactors connector supports HTTP pooling, which can substantially improve the performance of the connector. Learn more about the basic connection pooling configuration and different pooling mechanisms described in Connection pooling configuration.

Test the SuccessFactors connector

Test that the configuration is correct by running the following command:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system/Successfactors?_action=test"
{
  "name" : "Successfactors",
  "enabled" : true,
  "config" : "config/provisioner.openicf/Successfactors",
  "connectorRef" : {
    "bundleVersion" : "[1.5.0.0,1.6.0.0)",
    "bundleName" : "org.forgerock.openicf.connectors.successfactors-connector",
    "connectorName" : "org.forgerock.openicf.connectors.successfactors.SuccessFactorsConnector"
  },
  "displayName" : "SuccessFactors Connector",
  "objectTypes" : [
    "__GROUP__",
    "__PERSON__",
    "__ACCOUNT__",
    "__ALL__"
  ],
  "ok" : true
}
```

If the command returns `"ok": true`, your connector has been configured correctly and can authenticate to the SuccessFactors system.

SuccessFactors remote connector

If you want to run this connector outside of PingOne Advanced Identity Cloud or IDM, you can configure the SuccessFactors connector as a remote connector. Java Connectors installed remotely on a Java Connector Server function identically to those bundled locally within PingOne Advanced Identity Cloud or installed locally on IDM.

You can download the SuccessFactors connector from here.

Refer to Remote connectors for configuring the SuccessFactors remote connector.

Supported resource types

The SuccessFactors connector supports the following resource types:

SuccessFactors connector supported resource types

ICF Native Type	SuccessFactors Resource Type	Naming Attribute
__ACCOUNT__	User profile	`userId`, `username`
__GROUP__	Dynamic group [1]	`groupID`

ICF Native Type	SuccessFactors Resource Type	Naming Attribute

__PERSON__	Segments in Person object:
	`PerPerson`
	`PerPersonal`
	`EmpEmployment`
	`EmpJob`
	`PerPhone`
	`PerEmail`
	`EmpCompensation`
	`PerAddressDEFLT`
		`PerPerson_personIdExternal`
		`PerPersonal_personIdExternal`
		`EmpEmployment_userId`
		`EmpJob_userId`
		`PerPhone_personIdExternal`
		`PerEmail_personIdExternal`
		`EmpCompensation_userId`
		`PerAddressDEFLT_personIdExternal`

[1] Permission is the only supported sub-attribute for dynamic group resources.

Supported search filters

The SuccessFactors connector supports search operations with the following filter operators and attributes:

Supported Operators and Filter Attributes With SuccessFactors Searches

Object Type	Operators	Attributes
__ACCOUNT__	Equals, And, Or, StartsWith, EndsWith, Less than, Less than or equal, Greater than, Greater than or equal, Contains	• `username` • `userId` • `country` • `department` • `division` • `location` • `lastModifiedWithTZ` • `dateOfBirth` • `lastModified` • `lastModifiedDateTime` • `jobCode` • `firstName` • `lastName` • `manager`
__GROUP__	Equals, And, Or, StartsWith, EndsWith, Less than, Less than or equal, Greater than, Greater than or equal, Contains	• `groupID` • `groupType` • `staticGroup` • `createdBy` • `lastModifiedDate`

__PERSON__	Equals, And, Or, StartsWith, EndsWith, Less than, Less than or equal, Greater than, Greater than or equal, Contains	**Person Segments**
		`PerPerson`
		• `personIdExternal`
		• `dateOfBirth`
		• `lastModifiedOn`
		• `lastModifiedDateTime`
		• `createdOn`
		• `countryOfBirth`
		• `createdBy`
		• `createdDateTime`
		• `lastModifiedBy`
		• `perPersonUuid`
		• `personId`
		• `username`
		`EmpEmployment`
		• `personIdExternal`
		• `userId`
		• `assignmentIdExternal`
		• `serviceDate`
		• `professionalServiceDate`
		• `originalStartDate`
		• `seniorityDate`
		• `endDate`
		• `lastModifiedDateTime`
		`EmpJob`
		• `userId`
		• `payGrade`
		• `seqNumber`
		• `managerId`
		• `lastModifiedDateTime`
		• `eventReason`
		• `company`
		• `timezone`
		• `startDate`
		• `jobCode`
		• `jobTitle`
		• `position`
		• `location`
		• `payScaleType`
		• `payScaleArea`
		• `businessUnit`
		`PerPersonal`
		• `startDate`
		• `personIdExternal`
		• `nativePreferredLang`

Object Type	Operators	Attributes
		• `nationality` • `lastName` • `firstName` • `endDate` • `gender` • `maritalStatus` • `middleName` • `lastModifiedDateTime` `PerPhone` • `personIdExternal` • `areaCode` • `countryCode` • `createdBy` • `createdDateTime` • `createdOn` • `extension` • `includeAllRecords` • `isPrimary` • `lastModifiedBy` • `lastModifiedDateTime` • `lastModifiedOn` • `phoneNumber` • `phoneType` • `PerPhone_secondaryPhoneNumber`

Attributes

Account attributes

The SuccessFactors connector supports the following account attributes:

Attribute	Description
`userId`	The user’s User ID.
`userName`	The user’s username.
`status`	The user’s status.
`firstName`	The user’s first name.
`lastName`	The user’s last name.
`mi`	The user’s middle name.

Attribute	Description
`email`	The user’s email address.
`dateOfBirth`	The user’s birthdate.
`defaultFullName`	The default full name for the user.
`password`	The user’s password.
`lastModifiedDateTime`	The last modified date and time without time zone information.
`country`	The user’s country of residence.
`citizenship`	The user’s country of citizenship.
`married`	The user’s marital status.
`state`	The state where the user lives.
`city`	The city where the user lives.
`division`	The division the user works in.
`department`	The department the user works in.
`jobCode`	The Job code of the user.
`jobLevel`	The Job level of the user.
`timeZone`	The user’s time zone.
`location`	The user’s location.
`manager`	The user’s manager.
`hireDate`	The date the user was hired.
`lastModifiedWithTZ`	The last modified date and time with time zone information.
`lastModified`	The last modified date.
`__GROUP__`	The user’s group list.
`empId`	The user’s `empId`.
`custom09`	The user’s Custom 09 attribute.
`custom10`	The user’s Custom 10 attribute.

Group attributes

The SuccessFactors connector supports the following group attributes:

Attribute	Description
`groupId`	The unique ID of the group.
`groupName`	The name of the group.
`groupType`	The type of the group.
`activeMembershipCount`	The number of active members.
`totalMemberCount`	The number of total members.
`dgExcludePools`	Users excluded from the group.
`dgIncludePools`	Users included in the group.
`createdBy`	The user who created the group.
`lastModifiedDate`	The last modified date.

Person attributes

The SuccessFactors connector supports the following person segment attributes:

PerPerson attributes

The SuccessFactors connector supports the following PerPerson attributes:

Attribute	Description
`personIdExternal`	The person’s external ID.
`personId`	The person’s internal ID.
`userId`	The person’s user ID.
`dateOfBirth`	The person’s date of birth.
`lastModifiedOn`	The person’s last modified date.
`lastModifiedDateTime`	The person’s last modified date and time.
`createdOn`	The person’s date of creation.
`countryOfBirth`	The person’s country of birth.

Attribute	Description
`createdBy`	The ID of the user who created the person.
`createdDateTime`	The person’s date and time of creation.
`lastModifiedBy`	The ID of the last user to modify the person.
`perPersonUuid`	The person’s UUID.
`regionOfBirth`	The person’s birth region.
`username`	The person’s username.

PerPersonal attributes

The SuccessFactors connector supports the following PerPersonal attributes:

Attribute	Description
`personIdExternal`	The employee’s external ID.
`endDate`	The employment end date.
`startDate`	The employment start date.
`firstName`	The employee’s first name.
`lastName`	The employee’s last name.
`gender`	The employee’s gender.
`nativePreferredLang`	The employee’s preferred native language code.
`salutation`	The employee’s salutation.
`maritalStatus`	The employee’s marital status.
`nationality`	The employee’s nationality.
`middleName`	The employee’s middle name.
`preferredName`	The employee’s preferred name.
`lastModifiedDateTime`	The time the PerPersonal was last updated.

EmpEmployment attributes

The SuccessFactors connector supports the following EmpEmployment attributes:

Attribute	Description
`personIdExternal`	An ID used to represent the employee externally.
`userId`	The employee’s user ID.
`assignmentIdExternal`	An assignment ID used to identify users across the suite.
`firstDateWorked`	The first date the employee worked.
`endDate`	The end date of the employment.
`startDate`	The start date of the employment.
`eligibleForStock`	Whether or not the user is eligible for stock.
`initialOptionGrant`	The initial grant value of the employment.
`serviceDate`	The service date of employment.
`professionalServiceDate`	The professional service date of employment.
`originalStartDate`	The original start date of the employment.
`initialStockGrant`	The initial stock grant of the employment.
`seniorityDate`	The date of seniority.
`lastModifiedDateTime`	The time the EmpEmployment object was last updated.
`lastDateWorked`	The date of the last day the employee worked.

EmpJob attributes

The SuccessFactors connector supports the following EmpJob attributes:

Attribute	Description
`seqNumber`	The sequence number associated with the job.
`userId`	The employee’s user ID.
`eventReason`	The reason for action.
`company`	The company the job is for.
`managerId`	The ID of the job manager.
`timezone`	The time zone the job is in.

Attribute	Description
`startDate`	The date the job begins.
`endDate`	The date the job ends.
`payGrade`	The job’s pay grade.
`jobCode`	The job’s code.
`jobTitle`	The job’s title.
`position`	The position of the job.
`location`	The job’s location.
`payScaleType`	The payscale type for the job.
`payScaleArea`	The payscale area for the job.
`businessUnit`	The business unit the job belongs to.
`lastModifiedDateTime`	The date the job was last modified.

PerPhone attributes

To add all PerPhone attributes to the schema, you must define `PerPhone_ALL` in the person segment during configuration or define the specific attribute(s) with the segment name, such as `PerPhone_phoneNumber`.

The SuccessFactors connector supports the following PerPhone attributes:

Attribute	Description
`personIdExternal`	The employee’s external ID.
`areaCode`	The person’s phone number area code.
`countryCode`	The person’s phone number country code.
`createdBy`	The user who created the person’s phone number.
`createdDateTime`	The date and time of creation for a person’s phone number.
`createdOn`	The date of creation for a person’s phone number.
`includeAllRecords`	Include all records for the person’s phone number.
`extension`	The person’s phone number extension.

Attribute	Description
`isPrimary`	Whether the person's phone number is primary.
`lastModifiedBy`	The ID of the last user to modify the person's phone number.
`lastModifiedDateTime`	The time and date the person's phone number was last updated.
`lastModifiedOn`	The date the person's phone number was last updated.
`phoneNumber`	The person's phone number.
`phoneType`	The person's phone type.
`PerPhone_secondaryPhoneNumber`	The person's secondary phone number.

Use the SuccessFactors connector

Accounts

You can perform the following actions on a SAP SuccessFactors account:

The following example creates a user with every available attribute:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request  POST \
--data '{
  "userId": "BJENSEN",
  "username": "bjensen",
  "__ENABLE__": true,
  "email": "bjensen@example.com",
  "firstName": "Barbara",
  "lastName": "Jensen",
  "country": "USA",
  "married": false,
  "timeZone": "US/Eastern",
  "department": "Cloud",
  "state": "New York",
  "city": "New York City",
  "jobLevel": "2",
  "location": "40.6635°N 73.9387°W",
  "__PASSWORD__": "Test@123",
  "division": "Manufacturing",
  "hireDate": "2021-07-26 00:00:00",
  "dateOfBirth": "2012-08-22 00:00:00",
  "__GROUP__": [
    {"groupId": "6895"},
    {"groupId": "6095"}
  ]
}' \
"https://localhost:8443/openidm/system/successfactors/__ACCOUNT__?_action=create"
{
  "_id" : "BJENSEN",
  "userId" : "BJENSEN",
  "jobLevel" : "2",
  "__GROUP__" : [
    {
      "groupId" : "1586",
      "groupName" : "EVERYONE"
    }, {
      "groupId" : "6895",
      "groupName" : "SAP_Managers"
    }, {
      "groupId" : "6095",
      "groupName" : "SAP_ONB2_ErrorFlowAdmins"
    }
  ],
  "department" : "Cloud",
  "dateOfBirth" : "2012-08-22 00:00:00",
  "lastModifiedDateTime" : "2022-11-02 09:13:49",
  "__ENABLE__" : true,
  "email" : "bjensen@example.com",
  "country" : "USA",
  "lastModified" : "2022-11-02 10:13:49",
  "location" : "40.6635°N 73.9387°W",
  "lastName" : "Jensen",
  "lastModifiedWithTZ" : "2022-11-02 09:13:49",
  "username" : "bjensen",
  "timeZone" : "US/Eastern",
  "city" : "New York City",
  "state" : "New York",
  "__NAME__" : "bjensen",
  "hireDate" : "2021-07-26 00:00:00",
  "married" : false,
  "division" : "Manufacturing",
  "firstName" : "Barbara"
}
```

Note

New users must have at least the `username`, `userId`, and `status` properties.

The following example queries all SuccessFactors users:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"https://localhost:8443/openidm/system/successfactors/__ACCOUNT__?_queryId=query-all-ids"
{
  "result":[
    {"_id":"1007373"},
    {"_id":"1007371"},
    {"_id":"1007376"},
    {"_id":"1007370"},
    {"_id":"1007377"}
  ],
  "resultCount":5,
  "pagedResultsCookie":null,
  "totalPagedResultsPolicy":"NONE",
  "totalPagedResults":-1,
  "remainingPagedResults":-1
}
```

The following example queries a single user by their ID:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"https://localhost:8443/openidm/system/successfactors/__ACCOUNT__?_queryFilter=_id%20eq%20%22BJENSEN%22"
{
  "_id" : "BJENSEN",
  "userId" : "BJENSEN",
  "jobLevel" : "2",
  "__GROUP__" : [
    {
      "groupId" : "1586",
      "groupName" : "EVERYONE"
    }, {
      "groupId" : "6895",
      "groupName" : "SAP_Managers"
    }, {
      "groupId" : "6095",
      "groupName" : "SAP_ONB2_ErrorFlowAdmins"
    }
  ],
  "department" : "Cloud",
  "dateOfBirth" : "2012-08-22 00:00:00",
  "lastModifiedDateTime" : "2022-11-02 09:13:49",
  "__ENABLE__" : true,
  "email" : "bjensen@example.com",
  "country" : "USA",
  "lastModified" : "2022-11-02 10:13:49",
  "location" : "40.6635�N 73.9387�W",
  "lastName" : "Jensen",
  "lastModifiedWithTZ" : "2022-11-02 09:13:49",
  "username" : "bjensen",
  "timeZone" : "US/Eastern",
  "city" : "New York City",
  "state" : "New York",
  "__NAME__" : "bjensen",
  "hireDate" : "2021-07-26 00:00:00",
  "married" : false,
  "division" : "Manufacturing",
  "firstName" : "Barbara"
}
```

You can use the SuccessFactors connector to modify the following attributes of a user entry:

- `username`
- `email`
- `status`
- `country`
- `department`

- `timeZone`
- `jobLevel`
- `married`
- `city`
- `state`
- `division`
- `citizenship`
- `location`
- `firstName`
- `lastName`
- `gender`
- `dateOfBirth`
- `jobCode`

The following example updates the `division` property on a user:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "If-Match:*" \
--request PUT \
--data '{
  "division": "Engineering"
}' \
"https://localhost:8443/openidm/system/successfactors/__ACCOUNT__/BJENSEN"
{
  "_id" : "BJENSEN",
  "userId" : "BJENSEN",
  ...
  "division" : "Engineering",
  "firstName" : "Barbara"
}
```

The following example resets the password for a SuccessFactors user account:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request PATCH \
--data '[{
  "operation": "replace",
  "field": "__PASSWORD__",
  "value": "__CHANGEME__"
}]' \
"https://localhost:8443/openidm/system/successfactors/__ACCOUNT__/BJENSEN"
{
  "_id" : "BJENSEN",
  "userId" : "BJENSEN",
  ...
}
```

Note

The updated password is not included in the response object; however, the value is updated in the system.

The following example activates a user with the minimum required attributes:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
--data '{
  "username": "bjensen",
  "__ENABLE__": true,
  "firstName": "Barbara",
  "userId": "BJENSEN"
}' \
"https://localhost:8443/openidm/system/successfactors/__ACCOUNT__/BJENSEN"
{
  "_id": "BJENSEN",
  "userId": "BJENSEN",
  ...
  "__ENABLE__": true
}
```

The SuccessFactors connector does not support deleting accounts. To deactivate an unwanted account, set the account's `__ENABLE__` attribute value to `false`. A deactivated account remains in the SuccessFactors system and can still be queried by its ID, but cannot be accessed.

The following example deactivates a SuccessFactors user account:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
--data '{
  "username": "bjensen",
  "__ENABLE__": false,
  "firstName": "Barbara",
  "userId": "BJENSEN"
}' \
"https://localhost:8443/openidm/system/successfactors/__ACCOUNT__/BJENSEN"
{
  _id: "BJENSEN"
}
```

Groups

The following example assigns a user to a group:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "if-Match:*" \
--request PUT \
--data '{
  "__ENABLE__":true,
  "__GROUP__":[
    {
      "groupId":1001
    }
  ]
}' \
"https://localhost:8443/openidm/system/successfactors/__ACCOUNT__/BJENSEN"
{
  "_id" : "BJENSEN",
  "userId" : "BJENSEN",
  "jobLevel" : "2",
  "__GROUP__" : [
    {
      "groupId" : "1001",
      "groupName" : "Example Working Group"
    },
    ...
}
```

The following example queries all groups in the system:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--header "if-Match:*" \
--request GET \
"https://localhost:8443/openidm/system/successfactors/__GROUP__?_queryId=query-all-ids"
{
  "result": [
    {"_id":"6637"},
    {"_id":"2202"},
    {"_id":"1588"},
    {"_id":"6877"},
    {"_id":"2203"}
  ],
  "resultCount":5,
  "pagedResultsCookie": null,
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

The following example queries a single group:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"https://localhost:8443/openidm/system/successfactors/__GROUP__?/1001"
{
  "_id": "1001",
  "__NAME__": "1001",
  "groupName": "Example Working Group",
  "lastModifiedDate" : "2015-01-04 23:29:38",
  "createdBy" : "v4admin",
  "totalMemberCount" : "33590",
  "activeMembershipCount" : "2294",
  "groupID" : "1001",
  "groupType" : "permission"
}
```

Persons

The following example queries all persons in the system:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"https://localhost:8443/openidm/system/successfactors/__PERSON__?_queryId=query-all-ids"
{
  "result":[
    {"_id":"69119"},
    {"_id":"69120"},
    {"_id":"69121"},
    {"_id":"80279"},
    {"_id":"80280"}
  ],
  "resultCount":5,
  "pagedResultsCookie":null,
  "totalPagedResultsPolicy":"NONE",
  "totalPagedResults":-1,
  "remainingPagedResults":-1
}
```

The following example queries a single group:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request GET \
"https://localhost:8443/openidm/system/successfactors/__PERSON__?_queryFilter=_id%20%22scarter%22"
{
  "result":[{
    "_id":"scarter",
    "EmpJob_payGrade":"GR-08",
    "EmpEmployment_firstDateWorked":"2002-03-17 00:00:00",
    "PerPersonal_maritalStatus":"10819",
    "PerPersonal_nationality":"USA",
    "EmpEmployment_lastDateWorked":null,
    "EmpEmployment_userId":"scarter",
    "PerPersonal_personIdExternal":"scarter",
    "EmpEmployment_initialStockGrant":null,
    "PerPerson_countryOfBirth":"USA",
    "PerPersonal_endDate":"9999-12-31 00:00:00",
    "PerPersonal_firstName":"Sam",
    "EmpEmployment_eligibleForStock":null,
    "PerPersonal_lastName":"Carter",
    "EmpJob_payScaleArea":"USA/US2",
    "EmpJob_jobCode":"50070968",
    "PerPerson_regionOfBirth":null,
    "PerPersonal_startDate":"2002-03-17 00:00:00",
    "PerPerson_personIdExternal":"scarter",
    "PerPerson_lastModifiedDateTime":"2015-10-30 10:05:06",
    "EmpEmployment_lastModifiedDateTime":"2018-07-15 23:12:06",
    "PerPersonal_lastModifiedDateTime":"2018-10-25 23:51:29",
    "EmpJob_timezone":"US/Eastern",
    "PerPersonal_gender":"M",
    "PerPerson_dateOfBirth":"1983-02-15 00:00:00",
    "PerPersonal_nativePreferredLang":"10223",
    "EmpEmployment_serviceDate":null,
    "EmpEmployment_assignmentIdExternal":"scarter",
    "EmpJob_lastModifiedDateTime":"2020-06-23 10:50:43",
    "PerPerson_createdOn":"2015-01-05 23:34:22",
    "EmpJob_company":"1710",
    "EmpEmployment_originalStartDate":"2002-03-17 00:00:00",
    "EmpEmployment_endDate":null,
    "EmpJob_position":"3000325",
    "EmpJob_jobTitle":"Administrative Support",
    "PerPersonal_salutation":"10810",
    "EmpEmployment_seniorityDate":"2002-03-17 00:00:00",
    "PerPerson_createdDateTime":"2015-01-05 22:34:22",
    "EmpEmployment_professionalServiceDate":null,
    "EmpJob_startDate":"2017-01-01 00:00:00",
    "PerPersonal_middleName":null,
    "PerPerson_createdBy":"v4admin",
    "PerPersonal_preferredName":null,
    "PerPerson_lastModifiedBy":"scarter",
    "EmpJob_businessUnit":"CORP",
    "EmpJob_seqNumber":"1",
    "PerPerson_perPersonUuid":"87AF10389BCC4F29BC3F3A225B321E14",
    "EmpJob_location":"1710-2001",
    "EmpJob_managerId":"108743",
    "EmpJob_eventReason":"PAYOTH",
    "PerPerson_lastModifiedOn":"2015-10-30 11:05:06",
    "EmpJob_payScaleType":"USA/US2",
    "EmpJob_userId":"scarter",
    "EmpEmployment_initialOptionGrant":null,
    "EmpEmployment_personIdExternal":"scarter",
    "PerPerson_personId":"8",
    "__NAME__":"scarter"}],
  "resultCount":1,
  "pagedResultsCookie":null,
  "totalPagedResultsPolicy":"NONE",
  "totalPagedResults":-1,
  "remainingPagedResults":-1
}
```

You can use the SuccessFactors connector to modify the following attributes of a user entry:

- `PerPerson_dateOfBirth`
- `PerPerson_countryOfBirth`
- `PerPerson_regionOfBirth`
- `EmpEmployment_firstDateWorked`
- `EmpEmployment_eligibleForStock`
- `EmpEmployment_initialOptionGrant`
- `EmpEmployment_serviceDate`
- `EmpEmployment_professionalServiceDate`
- `EmpEmployment_originalStartDate`
- `EmpEmployment_initialStockGrant`
- `EmpEmployment_seniorityDate`
- `EmpEmployment_startDate`
- `EmpJob_jobCode`
- `EmpJob_startDate`
- `EmpJob_eventReason`
- `EmpJob_businessUnit`
- `EmpJob_managerId`
- `EmpJob_seqNumber`
- `EmpJob_payGrade`

- `EmpJob_jobTitle`
- `EmpJob_company`
- `EmpJob_location`
- `EmpJob_timezone`
- `EmpJob_position`
- `EmpJob_payScaleArea`
- `EmpJob_payScaleType`
- `PerPersonal_gender`
- `PerPersonal_endDate`
- `PerPersonal_startDate`
- `PerPersonal_firstName`
- `PerPersonal_lastName`
- `PerPersonal_nativePreferredLang`
- `PerPersonal_salutation`
- `PerPersonal_maritalStatus`
- `PerPersonal_nationality`
- `PerPersonal_preferredName`
- `PerPersonal_middleName`
- `PerPhone_phoneType`
- `PerPhone_phoneNumber`
- `PerPhone_isPrimary`
- `PerPhone_secondaryPhoneNumber`
- `PerEmail_emailType`
- `PerEmail_emailAddress`
- `PerEmail_isPrimary`
- `PerEmail_secondaryEmailAddress`
- `EmpCompensation_startDate`
- `EmpCompensation_endDate`
- `EmpCompensation_bonusTarget`
- `EmpCompensation_payGrade`

- `EmpCompensation_payrollSystemId`
- `EmpCompensation_benefitsRate`
- `EmpCompensation_isHighlyCompensatedEmployee`
- `EmpCompensation_eventReason`
- `EmpCompensation_payGroup`
- `EmpCompensation_isEligibleForCar`
- `EmpCompensation_isInsider`
- `EmpCompensation_isEligibleForBenefits`
- `PerAddressDEFLT_addressType`
- `PerAddressDEFLT_startDate`
- `PerAddressDEFLT_country`
- `PerAddressDEFLT_zipCode`
- `PerAddressDEFLT_notes`
- `PerAddressDEFLT_city`
- `PerAddressDEFLT_endDate`
- `PerAddressDEFLT_county`
- `PerAddressDEFLT_address2`
- `PerAddressDEFLT_address1`

To modify an existing person entry, use a PUT request and include all attributes of the person in the request:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "If-Match: *" \
--header "Content-Type: application/json" \
--request PUT \
--data '{
  "PerPerson_dateOfBirth":"1960-05-04 00:00:00",
  "PerPerson_countryOfBirth":"USA",
  "PerPerson_regionOfBirth":"null",
  "EmpEmployment_firstDateWorked":"2017-01-01 00:00:00",
  "EmpEmployment_eligibleForStock":true,
  "EmpEmployment_initialOptionGrant":"118.8",
  "EmpEmployment_serviceDate":"2018-01-01 00:00:00",
  "EmpEmployment_professionalServiceDate":"2019-02-01 00:00:00",
  "EmpEmployment_originalStartDate":"2018-03-01 00:00:00",
  "EmpEmployment_initialStockGrant":"3000",
  "EmpEmployment_seniorityDate":"2022-04-01 00:00:00",
  "EmpEmployment_startDate":"2016-05-01 00:00:00",
  "EmpJob_jobCode":"50070999",
  "EmpJob_startDate":"2019-01-01 00:00:00",
  "EmpJob_eventReason":"HIRNEW",
  "EmpJob_businessUnit":"PRODS",
  "EmpJob_managerId":"sMeias",
  "EmpJob_seqNumber":"2",
  "EmpJob_payGrade":"GR-08",
  "EmpJob_jobTitle":"Planning & Scheduling Manager",
  "EmpJob_company":"2100",
  "EmpJob_location":"2100-0001",
  "EmpJob_timezone":"Europe/Lisbon",
  "EmpJob_position":"3000909",
  "EmpJob_payScaleArea":"USA/US2",
  "EmpJob_payScaleType":"USA/US2",
  "PerPersonal_gender":"F",
  "PerPersonal_endDate":"9999-12-31 00:00:00",
  "PerPersonal_startDate":"2019-07-26 00:00:00",
  "PerPersonal_firstName":"sam",
  "PerPersonal_lastName":"carter",
  "PerPersonal_nativePreferredLang":"10223",
  "PerPersonal_salutation":"30085",
  "PerPersonal_maritalStatus":"10820",
  "PerPersonal_nationality":"USA",
  "PerPersonal_preferredName":"sammy",
  "PerPersonal_middleName":" M",
  "PerPhone_phoneType":"10605",
  "PerPhone_phoneNumber":"55555556",
  "PerPhone_isPrimary":true,
  "PerPhone_secondaryPhoneNumber":[
    {
      "phoneNumber":"555555555",
      "phoneType":"10606",
      "isPrimary":false,
      "areaCode":"312",
      "countryCode":"3",
      "extension":"76858"
    }
  ],
  "PerEmail_emailType":"8448",
  "PerEmail_emailAddress":"scarter@example.com",
  "PerEmail_isPrimary":true,
  "PerEmail_secondaryEmailAddress":[
    {
      "emailAddress":"scarter.21@example.com",
      "emailType":"8447",
      "isPrimary":false
    },
    {
      "emailAddress":"scarter.31@example.com",
      "emailType":"8446",
      "isPrimary":false
    }
  ],
  "EmpCompensation_startDate":"2021-12-31 00:00:00",
  "EmpCompensation_endDate":"9999-12-31 00:00:00",
  "EmpCompensation_bonusTarget":"47",
  "EmpCompensation_payGrade":"GR-12",
  "EmpCompensation_payrollSystemId":"ajukuio",
  "EmpCompensation_benefitsRate":"1",
  "EmpCompensation_isHighlyCompensatedEmployee":true,
  "EmpCompensation_eventReason":"PAYOTH",
  "EmpCompensation_payGroup":"D2",
  "EmpCompensation_isEligibleForCar":false,
  "EmpCompensation_isInsider":false,
  "EmpCompensation_isEligibleForBenefits":true,
  "PerAddressDEFLT_addressType":"home",
  "PerAddressDEFLT_startDate":"2012-01-01 00:00:00",
  "PerAddressDEFLT_country":"SGP",
  "PerAddressDEFLT_zipCode":"345653",
  "PerAddressDEFLT_notes":"notes",
  "PerAddressDEFLT_city":"Singapore",
  "PerAddressDEFLT_endDate":"9999-12-31 00:00:00",
  "PerAddressDEFLT_county":"11224",
  "PerAddressDEFLT_address2":"4815",
  "PerAddressDEFLT_address1":"Oceanic Avenue"
}' \
"https://localhost:8443/openidm/system/successfactors/__PERSON__/scarter"
{
  "_id":"scarter",
  "PerPersonal_maritalStatus":"10820",
  ...
  "PerPersonal_lastName":"carter",
  "__NAME__":"scarter"
}
```

OpenICF Interfaces Implemented by the SuccessFactors Connector

The SuccessFactors Connector implements the following OpenICF interfaces. For additional details, see ICF interfaces:

Create

Creates an object and its `uid`.

Delete

Deletes an object, referenced by its `uid`.

Schema

Describes the object types, operations, and options that the connector supports.

Script on Connector

Enables an application to run a script in the context of the connector.

Any script that runs on the connector has the following characteristics:

- The script runs in the same execution environment as the connector and has access to all the classes to which the connector has access.
- The script has access to a `connector` variable that is equivalent to an initialized instance of the connector. At a minimum, the script can access the connector configuration.
- The script has access to any script arguments passed in by the application.

Search

Searches the target resource for all objects that match the specified object class and filter.

Sync

Polls the target resource for synchronization events, that is, native changes to objects on the target resource.

Test

Tests the connector configuration.

Testing a configuration checks all elements of the environment that are referred to by the configuration are available. For example, the connector might make a physical connection to a host that is specified in the configuration to verify that it exists and that the credentials that are specified in the configuration are valid.

This operation might need to connect to a resource, and, as such, might take some time. Do not invoke this operation too often, such as before every provisioning operation. The test operation is not intended to check that the connector is alive (that is, that its physical connection to the resource has not timed out).

You can invoke the test operation before a connector configuration has been validated.

Update

Updates (modifies or replaces) objects on a target resource.

SuccessFactors Connector Configuration

The SuccessFactors Connector has the following configurable properties:

Configuration properties

Property	Type	Default	Encrypted[1]	Required[2]
`host`	`String`	`null`		✓ Yes
Hostname of the target.				
`clientId`	`String`	`null`		✓ Yes
The client identifier.				
`userId`	`String`	`null`		✓ Yes
User id for authentication.				
`privateKey`	`GuardedString`	`null`		✓ Yes
The private key which is used for signing JWT.				
`companyId`	`String`	`null`		✓ Yes
Company id as present in target application.				
`personSegments`	`String`	`null`		✕ No
The person segments to retrieve, specified as individual attributes or all (For example: EmpJob_All, EmpEmployment_startDate).				
`pageSize`	`int`	`0`		✕ No
Page size for search operation.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

[2] A list of operations in this column indicates that the property is required for those operations.

Basic Configuration Properties

Property	Type	Default	Encrypted[1]	Required[2]
`maximumConnections`	`Integer`	`10`		✕ No
Provides the maximum connections.				

Property	Type	Default	Encrypted[1]	Required[2]
`connectionTimeout`	`int`	`600`		✕ No
Provides the maximum connection timeout in seconds.				
`httpProxyHost`	`String`	`null`		✕ No
Provides the HTTP proxy host.				
`httpProxyPort`	`Integer`	`null`		✕ No
Provides the HTTP proxy port.				
`httpProxyUsername`	`String`	`null`		✕ No
Provides the HTTP proxy username.				
`httpProxyPassword`	`GuardedString`	`null`	🔒 Yes	✕ No
Provides the HTTP proxy password.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

[2] A list of operations in this column indicates that the property is required for those operations.

Webex Connector

Tip

This is a SaaS common connector.

The Webex connector allows you to manage users and groups between Webex Control Hub and IDM. A Webex administrator account is required for the connector to work.

Before you start

1. Create a Webex developer account.
2. Create an integration application. Add the required scopes to manage users, groups, licenses, and roles. Minimum scope required:
   - `spark-admin:people_write`
   - `spark-admin:people_read`
   - `spark-admin:licenses_read`

   - `spark-admin:roles_read`
   - `identity:groups_rw`
   - `identity:groups_read`

3. Remember to save the client secret and client id and get a refresh token.

> **Note**
>
> A refresh_token is not required when configuring the Connector via the UI.

Install the Webex connector

> **Tip**
>
> To check for an Advanced Identity Cloud application for this connector, refer to:
>
> - Application management
> - App catalog

You can download any connector from Backstage, but some are included in the default deployment for Advanced Identity Cloud, IDM, or RCS. When using an included connector, you can skip installing it and move directly to configuration.

Connector included in default deployment

Connector	IDM	RCS
Webex	✕ No	✕ No

Download the connector .jar file from Backstage.

- If you are running the connector locally, place it in the `/path/to/openidm/connectors` directory, for example:

```
mv ~/Downloads/webex-connector-1.5.20.30.jar /path/to/openidm/connectors/
```

- If you are using a remote connector server (RCS), place it in the `/path/to/openicf/connectors` directory on the RCS.

Configure the Webex connector

Create a connector configuration using the IDM admin UI:

1. From the navigation bar, click **Configure > Connectors**.
2. On the **Connectors** page, click **New Connector**.
3. On the **New Connector** page, type a **Connector Name**.
4. From the **Connector Type** drop-down list, select **Webex Connector - 1.5.20.30**.
5. Complete the **Base Connector Details**.

> **Tip**
>
> For a list of all configuration properties, refer to Webex Connector Configuration

6. Click **Save**.

When your connector is configured correctly, the connector displays as **Active** in the admin UI.

Refer to this procedure to create a connector configuration over REST.

- `Webex Endpoint` : https://webexapis.com/v1
- `Use Basic Auth For OAuth Token Neg` : `true` | `false`
- `Max connections` : Max size of the http connection pool used. Defaults to `10`.
- `Connection Timeout (seconds)` : Defines a timeout for the underlying http connection in seconds. Defaults to `30`.
- `Token Endpoint` : https://webexapis.com/v1/access_token
- `Client ID` : Your Client ID.
- `Client Secret` : Your Client Secret.
- `Refresh Token` : Your Refresh Token.
- `Read rate limit` : 2.5
- `Write read limi` : 2.5

If necessary, add or edit your object types to have these three objects with their properties:

PROPERTY NAME	TYPE	NATIVE TYPE	REQUIRED
`_id`	String	String	NO
`firstName`	String	String	NO
`lastName`	String	String	NO
`displayName`	String	String	YES
`emails`	Array	String	YES
`avatar`	String	String	NO
`licenses`	Array	String	YES
`groups`	Array	String	NO
`roles`	Array	String	NO
`extension`	String	String	NO

PROPERTY NAME	TYPE	NATIVE TYPE	REQUIRED
`locationId`	String	String	NO
`orgId`	String	String	NO
`department`	String	String	NO
`manager`	String	String	NO
`managerId`	String	String	NO
`title`	String	String	NO
`siteUrls`	Array	String	NO
`phoneNumbers`	Array	Object	NO
`addresses`	Array	Object	NO

PROPERTY NAME	TYPE	NATIVE TYPE	REQUIRED
`_id`	String	String	NO
`displayName`	String	String	YES
`orgId`	String	String	NO
`description`	String	String	NO
`members`	Array	Object	NO

PROPERTY NAME	TYPE	NATIVE TYPE	REQUIRED
`id`	String	String	NO
`name`	String	String	NO

PROPERTY NAME	TYPE	NATIVE TYPE	REQUIRED
`id`	String	String	NO
`name`	String	String	NO
`totalUnits`	Integer	Integer	NO
`consumedUnits`	Integer	Integer	NO

PROPERTY NAME	TYPE	NATIVE TYPE	REQUIRED
`subscriptionId`	String	String	NO
`siteUrl`	String	String	NO
`siteType`	String	String	NO

If configuring the connector over REST or through the filesystem, specify the connection details to the Webex resource provider in the `configurationProperties` for the connector. If you are using OAuth for your connection, the minimum required properties are `serviceUri`, `tokenEndpoint`, `refreshToken`, `clientId`, and `clientSecret`. The `readRateLimit` and `writeRateLimit` fields are for limiting the rate of requests. The recommended rate is 2.5.

```
{
    "configurationProperties" : {
        "tokenExpiration" : null,
        "accessToken" : null,
        "serviceUri" : "https://webexapis.com/v1",
        "readRateLimit" : "2.5",
        "login" : null,
        "password" : null,
        "writeRateLimit" : "2.5",
        "authenticationMethod" : "OAUTH",
        "tokenEndpoint" : "https://webexapis.com/v1/access_token",
        "clientId" : "k3..........5g",
        "clientSecret" : "xxxxxxxxxxxxxxxxxx",
        "refreshToken" : "xxxxxxxxxxxxxxxxxx",
        "authToken" : null,
        "acceptSelfSignedCertificates" : false,
        "disableHostNameVerifier" : false,
        "disableHttpCompression" : false,
        "clientCertAlias" : null,
        "clientCertPassword" : null,
        "maximumConnections" : "10",
        "httpProxyHost" : null,
        "httpProxyPort" : null,
        "httpProxyUsername" : null,
        "httpProxyPassword" : null,
        "connectionTimeout" : "30",
        "grantType" : "refresh_token",
        "scope" : null,
        "authorizationTokenPrefix" : "Bearer",
        "useBasicAuthForOauthTokenNeg" : true
    }
}
```

> **Note**
>
> On startup, IDM encrypts the value of the `clientSecret` and `refreshToken`.

Configure connection pooling

The Webex connector supports HTTP pooling, which can substantially improve the performance of the connector. Learn more about the basic connection pooling configuration and different pooling mechanisms described in Connection pooling configuration.

Mapping

Attributes Grid: Where the columns represent the attribute name mapped from source to target and the necessary data transformation to synchronize successfully.

SOURCE	TARGET	TRANSFORMATION SCRIPT
`id`	`_id`	N/A
`email`	`emails`	N/A
`lastName`	`sn`	N/A
`firstname`	`givenName`	N/A
`displayName`	`displayName`	N/A
`extension`	`extension`	N/A
`locationId`	`locationId`	N/A
`avatar`	`avatar`	N/A
`title`	`title`	N/A
`orgId`	`orgId`	N/A
`department`	`department`	N/A
`roles`	`roles`	N/A
`licenses`	`licenses`	N/A
`addresses`	`addresses`	N/A
`siteUrls`	`siteUrls`	N/A
`phoneNumbers`	`phoneNumbers`	N/A
`manager`	`manager`	N/A
`managerId`	`managerId`	N/A

Attributes Grid: Where the columns represent the attribute name mapped from source to target and the necessary data transformation to synchronize successfully.

This mapping depends on the previous mapping.

SOURCE	TARGET	TRANSFORMATION SCRIPT
`_id`	`id`	N/A
`sn`	`firstName`	N/A
`givenName`	`lastName`	N/A
`displayName`	`displayName`	N/A
`emails`	`emails`	N/A
`roles`	`roles`	N/A
`licenses`	`licenses`	N/A
`addresses`	`addresses`	N/A
`siteUrls`	`siteUrls`	N/A
`phoneNumbers`	`phoneNumbers`	N/A
`department`	`department`	N/A
`extension`	`extension`	N/A
`avatar`	`avatar`	N/A
`locationId`	`locationId`	N/A
`manager`	`manager`	N/A
`managerId`	`managerId`	N/A
`title`	`title`	N/A
`orgId`	`orgId`	N/A

Attributes Grid: Where the columns represent the attribute name mapped from source to target and the necessary data transformation to synchronize successfully.

SOURCE	TARGET	TRANSFORMATION SCRIPT
`id`	`_id`	N/A

SOURCE	TARGET	TRANSFORMATION SCRIPT
`displayName`	`displayName`	N/A
`orgId`	`orgId`	N/A
`members`	`members`	N/A
`description`	`description`	N/A

Attributes Grid: Where the columns represent the attribute name mapped from source to target and the necessary data transformation to synchronize successfully.

This mapping depends on the previous mapping.

SOURCE	TARGET	TRANSFORMATION SCRIPT
`displayName`	`displayName`	N/A
`groupManagementType`	`group_management_type`	N/A
`groupExternalId`	`group_external_id`	N/A
`members`	`members`	N/A

Test the Webex connector

Test that the connector was configured correctly:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Accept-API-Version: resource=1.0' \
--request POST \
'http://localhost:8080/system/webex?_action=test'
{
    "name": "Webex",
    "enabled": true,
    "config": "config/provisioner.openicf/Webex",
    "connectorRef": {
        "bundleVersion": "1.5.20.30",
        "bundleName": "org.forgerock.openicf.connectors.webex-connector",
        "connectorName": "org.forgerock.openicf.connectors.webex.WebexConnector"
    },
    "displayName": "Webex Connector",
    "objectTypes": [
        "__GROUP__",
        "__ACCOUNT__",
        "ROLES",
        "__ALL__",
        "LICENSES"
    ],
    "ok": true
}
```

Use the Webex connector

User

Create user

To create a user, it is necessary to *at least* provide the `email` and `displayName` fields. Some fields, such as `extension`, `locationId`, or `siteUrls` require their respective licenses. Field `locationId` requires the `extension` field, and `locationId` and `phoneNumbers` fields are mutually exclusive.

It is possible that licenses will be added by default, so it is advisable to perform a reconciliation to be up to date.

If you create a user with a role or license (bad format) that is not allowed to be assigned or with an incorrect avatar URL, the user is still created with the exception of the fields that failed:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request POST \
--data '{
    "displayName" : "John Doe",
    "firstName" : "John",
    "LastName" : "Doe",
    "groups" : [
            "groupId",
            "groupId"
    ],
    "emails" : [
        "john.doe@example.com"
    ],
    "licenses" : [
        "licenseId"
    ],
    "roles" : [
        "roleId"
    ],
    "extension" : "123",
    "avatar" : "urlAvatar",
    "title" : "Title",
    "department" : "Sales",
    "manager" : "Manager Name",
    "managerId" : "managerId",
    "phoneNumbers": [
        {
          "type": "work",
          "value": "+1 010 110 1101"
        }
    ],
    "addresses": [
        {
          "type": "work",
          "country": "US",
          "locality": "Milpitas",
          "region": "California",
          "streetAddress": "1111 Bird Ave.",
          "postalCode": "010101"
        }
    ],
    "siteUrls": [
        "mysite.webex.com#attendee"
    ],
    "orgId" : "orgId",
    "locationId" : "locationId"
}'\
'http://localhost:8080/openidm/system/webex/__ACCOUNT__?_action=create'
{
    "_id" : "userId",
    "displayName" : "John Doe",
    "firstName" : "John",
    "LastName" : "Doe",
    "groups" : [
            "groupId",
            "groupId"
    ],
    "emails" : [
        "john.doe@example.com"
    ],
    "licenses" : [
        "licenseId"
    ],
    "roles" : [
        "roleId"
    ],
    "extension" : "123",
    "avatar" : "urlAvatar",
    "title" : "Title",
    "department" : "Sales",
    "manager" , "Manager Name",
    "managerId" : "managerId",
    "phoneNumbers": [
        {
          "type": "work",
          "value": "+1 010 110 1101"
        }
    ],
    "addresses": [
        {
          "type": "work",
          "country": "US",
          "locality": "Milpitas",
          "region": "California",
          "streetAddress": "1111 Bird Ave.",
          "postalCode": "010101"
        }
    ],
    "siteUrls": [
        "mysite.webex.com#attendee"
    ],
    "orgId" : "orgId",
    "locationId" : "locationId"
}
```

Get Users

Retrieve a list of user ids from Webex. By default, retrieves all users:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request GET \
'http://localhost:8080/openidm/system/webex/__ACCOUNT__?_queryId=query-all-ids'
{
    "result": [
        {
            "_id" : "001"
        },
        {
            "_id" : "002"
        },
        {
            "_id" : "003"
        },
    ],
    "resultCount": 999,
    "pagedResultsCookie": null,
    "totalPagedResultsPolicy": "NONE",
    "totalPagedResults": -1,
    "remainingPagedResults": -1
}
```

Get user

Retrieve a user from Webex. The user id must be provided in the URI path.

The locationId and extension fields will only be displayed if they have their corresponding license:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request GET \
'http://localhost:8080/openidm/system/webex/__ACCOUNT__/ID'
{
    "_id" : "userId",
    "displayName" : "active",
    "firstName" : "active",
    "LastName" : "active",
    "groups" : [
            "groupId",
            "groupId",
    ],
    "emails" : [
        "test@email.com"
    ],
    "licenses" : [
        "licenseId"
    ],
    "roles" : [
        "roleId"
    ],
    "extension" : "gs",
    "avatar" : "avatarURL",
    "title" : "surname",
    "department" : "Sales",
    "manager" , "John Doe",
    "managerId" : "managerId",
    "phoneNumbers": [
        {
          "type": "work",
          "value": "+1 010 110 1101"
        }
    ],
    "addresses": [
        {
          "type": "work",
          "country": "US",
          "locality": "Milpitas",
          "region": "California",
          "streetAddress": "1099 Bird Ave.",
          "postalCode": "99212"
        }
    ],
    "siteUrls": [
        "mysite.webex.com#attendee"
    ],
    "orgId" : true,
    "locationId" : "locationId"
}
```

Get user emails

Retrieve a user in Webex filtering by the `emails` field:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request GET \
'http://localhost:8080/openidm/system/webex/__ACCOUNT__?_queryFilter=true&_fields=emails'
{
    "result": [
        {
            "_id" : "001",
            "emails": [
            "email001@example.wbx.ai"
            ],
        },
        {
            "_id" : "003",
            "emails": [
            "email003@example.wbx.ai"
            ],
        },
        {
            "_id" : "003",
            "emails": [
            "email003@example.wbx.ai"
            ],
        }
    ],
    "resultCount": 999,
    "pagedResultsCookie": null,
    "totalPagedResultsPolicy": "NONE",
    "totalPagedResults": -1,
    "remainingPagedResults": -1
}
```

Get user filter by email

Retrieve a user in Webex filtering by the `emails` field.

The locationId and extension fields will only be displayed if they have their corresponding license:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request GET \
'http://localhost:8080/openidm/system/webex/__ACCOUNT__?_queryFilter=emails%20eq%20"email@example.wbx.ai"'
{
    "result": [
        {
            "_id": "userId",
            "firstName": "John",
            "displayName": "John Doe",
            "lastName": "Doe",
            "orgId": "orgId",
            "emails": [
                "john@example.wbx.ai"
            ],
            "roles": [
                "roleId"
            ],
            "groups": [],
            "licenses": [
                "licenseId"
            ],
            "__NAME__": "John"
        }
    ]
}
```

Get user filter starts with

Retrieve a user in Webex filtering by the `displayName` field. Minimum characters to search for displayName.

The locationId and extension fields will only be displayed if they have their corresponding license:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request GET \
'http://localhost:8080/openidm/system/webex/__ACCOUNT__?_queryFilter=displayName%20sw%20"john"'
{
    "result": [
        {
            "_id": "userId",
            "firstName": "John",
            "displayName": "John Doe",
            "lastName": "Doe",
            "orgId": "orgId",
            "emails": [
                "john@example.wbx.ai"
            ],
            "roles": [
                "roleId"
            ],
            "groups": [
                "groupId_1",
                "groupId_2"
            ],
            "licenses": [
                "licenseId"
            ],
            "__NAME__": "John"
        }
    ],
    "resultCount": 1,
    "pagedResultsCookie": null,
    "totalPagedResultsPolicy": "NONE",
    "totalPagedResults": -1,
    "remainingPagedResults": -1
}
```

Update user

To update a user, it is necessary to *at least* provide the `email`, `displayName` and `licenses` fields. To update the `locationId`, `extension`, and `siteUrls` fields, the user must have the corresponding licenses.

`locationId` requires the extension field. `locationId` and `phoneNumbers` are mutually exclusive:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request PUT \
--data '{
    "displayName" : "John Doe",
    "firstName" : "John",
    "LastName" : "Doe",
    "groups" : [
            "groupId",
            "groupId"
    ],
    "emails" : [
        "john.doe@example.com"
    ],
    "licenses" : [
        "licenseId"
    ],
    "roles" : [
        "roleId"
    ],
    "extension" : "123",
    "avatar" : "urlAvatar",
    "title" : "Title",
    "department" : "Sales",
    "manager" , "Manager Name",
    "managerId" : "managerId",
    "phoneNumbers" : [
        {
          "type": "work",
          "value": "+1 010 110 1101"
        }
    ],
    "addresses" : [
        {
          "type": "work",
          "country": "US",
          "locality": "Milpitas",
          "region": "California",
          "streetAddress": "1111 Bird Ave.",
          "postalCode": "010101"
        }
    ],
    "siteUrls": [
        "mysite.webex.com#attendee"
    ],
    "orgId" : "orgId",
    "locationId" : "locationId"
}' \
'http://localhost:8080/system/webex/__ACCOUNT__/USER_ID'
{
    "_id" : "userId",
    "displayName" : "John Doe",
    "firstName" : "John",
    "LastName" : "Doe",
    "groups" : [
            "groupId",
            "groupId",
    ],
    "emails" : [
        "john.doe@example.com"
    ],
    "licenses" : [
        "licenseId"
    ],
    "roles" : [
        "roleId"
    ],
    "extension" : "123",
    "avatar" : "avatarURL",
    "title" : "Title",
    "department" : "Developer",
    "manager" , "Manager Name",
    "managerId" : "managerId",
    "phoneNumbers": [
        {
          "type": "work",
          "value": "+1 010 110 1101"
        }
    ],
    "addresses": [
        {
          "type": "work",
          "country": "US",
          "locality": "Milpitas",
          "region": "California",
          "streetAddress": "1111 Bird Ave.",
          "postalCode": "010101"
        }
    ],
    "siteUrls": [
        "mysite.webex.com#attendee"
    ],
    "orgId" : "orgId",
    "locationId" : "locationId"
}
```

Note

To update the licenses correctly, it is necessary to deactivate the default licenses and to have the option "Preserve licenses for users joining another group" activated in Webex Control Hub in the user licenses tab.

Delete user

Delete a user from the Webex organization. The user id must be provided in the URI path:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request DELETE \
'http://localhost:8080/openidm/system/webex/__ACCOUNT__/USER_ID'
{
    "_id" : "USER_ID",
    "email" : "deleted.member@email.com",
    ...
}
```

GROUPS

Create group

To create a group, it is necessary to *at least* provided `displayName` field:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request POST \
--data '{
    "displayName" : "Group name",
    "description" : "Group Description",
    "orgId" : "orgId"
}' \
'http://localhost:8080/openidm/system/webex/__GROUP__?_action=create'
{
    "_id" : "groupId",
    "displayName" : "Group name",
    "description" : "Group Description",
    "orgId" : "orgId",
    "members": []
}
```

Get groups

Retrieve a list of groups showing only the ids. By default returns all groups:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request GET \
'http://localhost:8080/openidm/system/webex/__GROUP__?_queryId=query-all-ids'
{
    "result": [
        {
            "_id" : "groupid1"
        },
        {
            "_id" : "groupId2"
        },
        {
            "_id" : "groupId3",
        },
        ...
    ],
    "resultCount": 999,
    "pagedResultsCookie": null,
    "totalPagedResultsPolicy": "NONE",
    "totalPagedResults": -1,
    "remainingPagedResults": -1
}
```

Update a group

The fields that can be updated for a group are `description` and `displayName`. The group id must be provided in the URI path:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request PUT \
--header 'If-Match: *' \
--data '{
    "displayName" : "New Group Name",
    "description" : "New Description"
}' \
'http://localhost:8080/openidm/system/webex/__GROUP__/GROUP_ID'
{
    "_id" : "groupId",
    "displayName" : "New Group Name",
    "description" : "New Description",
    "orgId" : "orgId",
    "members": []
}
```

Delete a group

The group id must be provided in the URI path:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request DELETE \
'http://localhost:8080/openidm/system/webex/__GROUP__/GROUP_ID'
{
    "_id" : "groupId",
    "displayName" : "New Group Name",
    "description" : "New Description",
    "orgId" : "orgId",
    "members": []
}
```

Get Roles

Retrieve a list of roles for Webex users.

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request GET \
'http://localhost:8080/openidm/system/webex/ROLES/?_queryFilter=true'
{
    "result" : [
        {
            "_id" : "roleId",
            "__NAME__" : "Role name",
        },
        ...
    ],
    "resultCount": 999,
    "pagedResultsCookie": null,
    "totalPagedResultsPolicy": "NONE",
    "totalPagedResults": -1,
    "remainingPagedResults": -1
}
```

Get Role

Retrieve a role from the Webex organization. The user id must be provided in the URI path:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request GET \
'http://localhost:8080/openidm/system/webex/ROLES/ROLE_ID'
{
    "_id": "roleId",
    "__NAME__": "Role Name"
}
```

Get Licenses

Retrieve a list of licenses for Webex users.

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request GET \
'http://localhost:8080/openidm/system/webex/LICENSES/?_queryFilter=true'
{
    "result" : [
        {
            "_id" : "LicenseId",
            "name" : "Role Name",
            "totalUnits" : 50,
            "consumedUnits": 1,
            "subscriptionId": "subscriptionId",
            "siteUrl": "site1-example.webex.com",
            "siteType": "Control Hub managed site",
        },
        ...
    ],
    "resultCount": 999,
    "pagedResultsCookie": null,
    "totalPagedResultsPolicy": "NONE",
    "totalPagedResults": -1,
    "remainingPagedResults": -1
}
```

Get License

Retrieve a user license. The user id must be provided in the URI path:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header 'Content-Type: application/json' \
--request GET \
'http://localhost:8080/openidm/system/webex/LICENSES/LICENSE_ID'
{
    "_id": "LicenseId",
    "name": "License Name",
    "totalUnits": "10",
    "subscriptionId": "subscriptionId",
    "siteType": "Control Hub managed site",
    "__NAME__": "License Name",
    "siteUrl": "sityUrl.webex.com"
}
```

OpenICF Interfaces Implemented by the Webex Connector

The Webex Connector implements the following OpenICF interfaces. For additional details, see ICF 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.

Webex Connector Configuration

The Webex Connector has the following configurable properties:

Basic Configuration Properties

Property	Type	Default	Encrypted[1]	Required[2]
`serviceUri`	`String`	`null`		✓ Yes
The service endpoint URI.				
`readRateLimit`	`String`	`null`		✓ Yes
Defines throttling for read operations either per seconds ("30/sec") or per minute ("100/min").				
`login`	`String`	`null`		✓ Yes
The service login name.				
`writeRateLimit`	`String`	`null`		✓ Yes
Defines throttling for write operations (create/update/delete) either per second ("30/sec") or per minute ("100/min").				
`password`	`GuardedString`	`null`	🔒 Yes	✕ No
The service user password.				
`authenticationMethod`	`String`	`OAUTH`		✓ Yes
Defines which method is to be used to authenticate on the remote server. Options are BASIC (username/password), OAUTH (Client id/secret) or TOKEN (static token).				
`tokenEndpoint`	`String`	`null`		✕ No

Property	Type	Default	Encrypted[1]	Required[2]
When using OAUTH as authentication method, this property defines the endpoint where a new access token should be queried for (https://myserver.com/oauth2/token).				
`clientId`	`String`	`null`		✓ Yes
The client identifier for OAuth2.				
`clientSecret`	`GuardedString`	`null`	🔒 Yes	✗ No
Secure client secret for OAuth2.				
`authToken`	`GuardedString`	`null`	🔒 Yes	✗ No
Static authentication token.				
`acceptSelfSignedCertificates`	`boolean`	`false`		✓ Yes
To be used for debug/test purposes. To be avoided in production.				
`disableHostNameVerifier`	`boolean`	`false`		✓ Yes
To be used for debug/test purposes. To be avoided in production.				
`disableHttpCompression`	`boolean`	`false`		✓ Yes
Content compression is enabled by default. Set this property to true to disable it.				
`clientCertAlias`	`String`	`null`		✓ Yes
If TLS Mutual Auth is needed, set this to the certificate alias from the keystore.				
`clientCertPassword`	`GuardedString`	`null`	🔒 Yes	✓ Yes
If TLS Mutual Auth is needed and the client certificate (private key) password is different from the keystore password, set this to the client private key password.				
`maximumConnections`	`Integer`	`10`		✓ Yes
Defines the max size of the HTTP connection pool used.				
`httpProxyHost`	`String`	`null`		✓ Yes
Defines the Hostname if an HTTP proxy is used between the connector and the service.				
`httpProxyPort`	`Integer`	`null`		✓ Yes
Defines the Port if an HTTP proxy is used between the connector and the service.				

Property	Type	Default	Encrypted[1]	Required[2]
`httpProxyUsername`	`String`	`null`		✓ Yes
Defines Proxy Username if an HTTP proxy is used between the connector and the service.				
`httpProxyPassword`	`GuardedString`	`null`	🔒 Yes	✓ Yes
Defines Proxy Password if an HTTP proxy is used between the connector and the service.				
`connectionTimeout`	`int`	`30`		✕ No
Defines a timeout for the underlying HTTP connection in seconds.				
`refreshToken`	`GuardedString`	`null`		✕ No
Used by the refresh_token grant type.				
`grantType`	`String`	`null`		✕ No
The OAuth2 grant type to use (client_credentials or refresh_token).				
`scope`	`String`	`null`		✕ No
The OAuth2 scope to use.				
`authorizationTokenPrefix`	`String`	`Bearer`		✕ No
The prefix to be used in the Authorization HTTP header for Token authentication.				
`useBasicAuthForOauthTokenNeg`	`boolean`	`true`		✓ Yes
The Authentication method for refresh token (Basic Authentication or Sending the ClientId and Client Secret in the Header).				
`jwtKey`	`String`	`null`		✕ No
Description is not available				
`jwtExpiration`	`Integer`	`null`		✕ No
Description is not available				
`jwtAlgorithm`	`String`	`null`		✕ No
Description is not available				
`jwtClaims`	`Map`	`null`		✕ No
Description is not available				

Property	Type	Default	Encrypted[1]	Required[2]
`jwtPem`	`String`	`null`		✕ No
Description is not available				
`jwtCert`	`String`	`null`		✕ No
Description is not available				
`keyAlgorithm`	`String`	`null`		✕ No
Description is not available				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

[2] A list of operations in this column indicates that the property is required for those operations.

Workday connector

> **Important**
>
> You can only use Workday connector version 1.5.20.29 and later with:
>
> - Connector framework 1.5.20.24 or later
> - RCS 1.5.20.24 or later
>
> Learn more in Changed functionality.

Workday is a multi-tenant Software-as-a-Service (SaaS) application. The Workday connector lets you synchronize user accounts between IDM and Workday's cloud-based HR system.

The connector supports reconciliation of users and organizations from Workday to an IDM repository, liveSync of users from Workday to IDM, and updating users in a Workday system.

To use the connector, you need a Workday instance with the required permissions and a set of credentials to access the instance, including the username, password, tenant name, and host name.

Install the Workday connector

> **Tip**
>
> To check for an Advanced Identity Cloud application for this connector, refer to:
>
> - Application management
> - App catalog

You can download any connector from Backstage, but some are included in the default deployment for Advanced Identity Cloud, IDM, or RCS. When using an included connector, you can skip installing it and move directly to configuration.

Connector included in default deployment

Connector	IDM	RCS
Workday	✕ No	✕ No

Download the connector .jar file from Backstage.

- If you are running the connector locally, place it in the `/path/to/openidm/connectors` directory, for example:

```
mv ~/Downloads/workday-connector-1.5.20.28.jar /path/to/openidm/connectors/
```

- If you are using a remote connector server (RCS), place it in the `/path/to/openicf/connectors` directory on the RCS.

Configure the Workday connector

Create a connector configuration using the IDM admin UI:

1. From the navigation bar, click **Configure > Connectors**.
2. On the **Connectors** page, click **New Connector**.
3. On the **New Connector** page, type a **Connector Name**.
4. From the **Connector Type** drop-down list, select **Workday Connector - 1.5.20.28**.
5. Complete the **Base Connector Details**.

> **Tip**
>
> For a list of all configuration properties, refer to Workday Connector Configuration

6. Click **Save**.

When your connector is configured correctly, the connector displays as **Active** in the admin UI.

Refer to this procedure to create a connector configuration over REST.

Alternatively, copy the sample configuration file `/path/to/openidm/samples/example-configurations/provisioners/provisioner.openicf-workday.json` to your project's `conf/` directory, and set `enabled` to `true`. Edit the `configurationProperties` to specify the connection to the Workday instance, for example:

```
"configurationProperties" : {
    "hostname" : "example.workday.net",
    "tenant" : "example-tenant",
    "username" : "admin",
    "password" : "Passw0rd",
    ...
}
```

Set at least the following properties:

`hostname`

The fully qualified name of the Workday instance. The connector uses the `hostname` to construct the endpoint URL.

`tenant`

The tenant to which you are connecting. The connector uses the `tenant` name to construct the endpoint URL, and the complete username (in the form `username@tenant`).

`username`

The `username` used to log in to the Workday instance. Do not specify the complete username including the `tenant`. The connector constructs the complete username.

`password`

The `password` used to log in to the Workday instance.

`connectionTimeout`

The timeout (in milliseconds) that the connector should wait for a request to be sent to the Workday instance. The default timeout is 60000ms or one minute. Requests that take longer than a minute throw an exception.

`receiveTimeout`

The timeout (in milliseconds) that the connector waits to receive a response. The default timeout is 60000ms or one minute. Because the Workday can be slow, and the amount of information returned can be very large, you should set this parameter carefully to avoid unnecessary timeouts.

Check that the connector is retrieving the exact data that you need.

The `configurationProperties` also specify the data that the connector should retrieve with a number of boolean `include...` and `exclude...` properties. These properties can be divided as follows:

Worker types

By default, all worker types are retrieved. Use any the following settings to exclude specific worker types:

- `excludeContingentWorkers` - exclude contingent workers from query results, `false` by default.
- `excludeEmployees` - exclude regular employees from query results, `false` by default.
- `excludeInactiveWorkers` - exclude inactive workers from query results, `false` by default.

Specific worker data

These parameters specify the properties to return for every included worker type. For performance reasons, set all of these to false initially, and then include *only* the necessary properties.

- `includeWorkerDocuments`
- `includeDevelopmentItems`
- `includeRoles`
- `includeQualifications`
- `includeTransactionLogData`
- `includeCareer`
- `includeContingentWorkerTaxAuthorityFormInformation`
- `includeUserAccount`
- `includeFeedbackReceived`
- `includeEmployeeContractData`
- `includeSkills`
- `includeAccountProvisioning`
- `includeGoals`
- `includeSuccessionProfile`
- `includeBackgroundCheckData`
- `includeEmployeeReview`
- `includeManagementChainData`
- `includeOrganizations`
- `includePhoto`
- `includeRelatedPersons`
- `includeBenefitEligibility`
- `includeTalentAssessment`
- `includeBenefitEnrollments`
- `includeCompensation`

Specific organizational data

Included in the data of each worker is the organization to which the user belongs. If you have set `includeOrganizations` to `true`, you can specify the organizational data that should be *excluded* from the query response. By default, all organizational data is included. To exclude data from a response, set its corresponding property to `true`. For performance reasons, set all of these to `true` initially, and then include *only* the necessary properties.

- `excludeCompanies`
- `excludeBusinessUnits`
- `excludeCustomOrganizations`
- `excludeMatrixOrganizations`
- `excludeGiftHierarchies`
- `excludeCostCenterHierarchies`
- `excludeGrants`
- `excludeProgramHierarchies`
- `excludeFunds`
- `excludeOrganizationSupportRoleData`
- `excludeGifts`
- `excludeBusinessUnitHierarchies`
- `excludeCostCenters`
- `excludePrograms`
- `excludeSupervisoryOrganizations`
- `excludeRegionHierarchies`
- `excludeTeams`
- `excludeLocationHierarchies`
- `excludeRegions`
- `excludePayGroups`
- `excludeFundHierarchies`
- `excludeGrantHierarchies`

You can find all configuration properties in the Workday Connector Configuration.

XPath transformations

> **Note**
>
> XPath transformations require ICF Framework and RCS version 1.5.20.24 or later and Workday connector version 1.5.20.26 or later.

You can use XPath transformations to simplify and map Workday attributes directly to read-only connector object type properties. The `xpathTransformations` configuration property is an array of nested object configuration properties and must contain `objectType`, `attribute`, and `transformation` properties.

The XPath transformation implementation uses the `javax.xml.xpath` package, which supports the XPath 1.0 specification for transformation expressions.

An example XPath transformation:

```
{
  "xpathTransformations": [
    {
      "objectType": "__ACCOUNT__", (1)
      "attribute": "WORKER_NAME", (2)
      "transformation": "ns1:Worker_Data/ns1:Personal_Data/ns1:Name_Data/ns1:Preferred_Name_Data/
ns1:Name_Detail_Data/@ns1:Formatted_Name" (3)
    },
    {
      "objectType": "__ACCOUNT__", (1)
      "attribute": "USER_ID", (2)
      "transformation": "ns1:Worker_Data/ns1:User_ID" (3)
    }
  ]
}
```

1 The ICF native `objectType` that the XPath-transformed property belongs to.

2 The target resource name (`attribute`) in the Workday object type.

3 The XPath `transformation` applied to extract or flatten the attribute value from Workday.

Workday remote connector

If you want to run this connector outside of PingOne Advanced Identity Cloud or IDM, you can configure the Workday connector as a remote connector. Java Connectors installed remotely on a Java Connector Server function identically to those bundled locally within PingOne Advanced Identity Cloud or installed locally on IDM.

You can download the Workday connector from here.

Refer to Remote connectors for configuring the Workday remote connector.

Configure connection pooling

The Workday connector uses ICF pooling to manage connections. Learn more about the different pooling mechanisms in Connectors by pooling mechanism.

Test the Workday connector

When your connector is configured correctly, test its status by running the following command:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system?_action=test"
[
  {
    "name": "workday",
    "enabled": true,
    "config": "config/provisioner.openicf/workday",
    "connectorRef": {
      "bundleVersion": "[1.5.0.0,1.6.0.0)",
      "bundleName": "org.forgerock.openicf.connectors.workday-connector",
      "connectorName": "org.forgerock.openicf.connectors.workday.WorkdayConnector"
    },
    "displayName": "Workday Connector",
    "objectTypes": [
      "employee",
      "__ALL__"
    ],
    "ok": true
  }
]
```

A status of `"ok": true` indicates that the connector can contact the Workday instance.

To retrieve the workers in the Workday system, run the following command:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
"http://localhost:8080/openidm/system/workday/employee?_queryId=query-all-ids"
{
  "result": [
    {
      "_id": "3aa5550b7fe348b98d7b5741afc65534",
      "employeeID": "21001"
    },
    {
      "_id": "0e44c92412d34b01ace61e80a47aaf6d",
      "employeeID": "21002"
    },
    {
      "_id": "3895af7993ff4c509cbea2e1817172e0",
      "employeeID": "21003"
    },
    ...
  ]
}
```

The first time the connector retrieves the employees from the Workday system, the following warning, which you can safely ignore, might display in the console:

```
WARNING: Default key managers cannot be initialized: Invalid keystore format
java.io.IOException: Invalid keystore format
```

To retrieve a specific user, include the user's ID in the URL. For example:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
"http://localhost:8080/openidm/system/workday/employee/3aa5550b7fe348b98d7b5741afc65534"
```

Reconcile users from Workday to IDM

To reconcile users from Workday to IDM, set up a mapping between Workday and IDM managed users.

When you have created a mapping, run reconciliation using the admin UI or with a REST call similar to the following:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/recon?
_action=recon&mapping=systemWorkdayEmployee_managedUser&waitForCompletion=true"
{
  "_id": "db2bc7f4-e9a8-4315-9dd1-e2cdcd85ae6e-33099",
  "state": "SUCCESS"
}
```

Update users in the Workday system

The connector supports updates to system users only for the following properties:

- Account credentials (`username` and `password`)
- `email`
- `mobile` (telephone number)

The following command updates a user's mobile number:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-type: application/json" \
--request PATCH \
--data '[
  {
    "operation": "replace",
    "field": "mobile",
    "value": "+1 (415) 859-4366"
  }
]' \
"http://localhost:8080/openidm/system/workday/employee/3aa5550b7fe348b98d7b5741afc65534"
```

Implementation specifics

For PATCH requests, a connector can potentially add, remove, or replace an attribute value. The Workday connector doesn’t implement the add or remove operations, so a PATCH request always replaces the entire attribute value with the new value.

OpenICF Interfaces Implemented by the Workday Connector

The Workday Connector implements the following OpenICF interfaces. For additional details, see ICF interfaces:

Schema

Describes the object types, operations, and options that the connector supports.

Script on Connector

Enables an application to run a script in the context of the connector.

Any script that runs on the connector has the following characteristics:

- The script runs in the same execution environment as the connector and has access to all the classes to which the connector has access.
- The script has access to a `connector` variable that is equivalent to an initialized instance of the connector. At a minimum, the script can access the connector configuration.
- The script has access to any script arguments passed in by the application.

Search

Searches the target resource for all objects that match the specified object class and filter.

Sync

Polls the target resource for synchronization events, that is, native changes to objects on the target resource.

Test

Tests the connector configuration.

Testing a configuration checks all elements of the environment that are referred to by the configuration are available. For example, the connector might make a physical connection to a host that is specified in the configuration to verify that it exists and that the credentials that are specified in the configuration are valid.

This operation might need to connect to a resource, and, as such, might take some time. Do not invoke this operation too often, such as before every provisioning operation. The test operation is not intended to check that the connector is alive (that is, that its physical connection to the resource has not timed out).

You can invoke the test operation before a connector configuration has been validated.

Update

Updates (modifies or replaces) objects on a target resource.

Workday Connector Configuration

The Workday Connector has the following configurable properties:

Worker Response Configuration Properties

Property	Type	Default	Encrypted[1]	Required[2]
`includeManagementChainDataForWorkers`	`Boolean`	`true`		✕ No
Indicates if the Management Chain Data element is included in the response.				
`includeOrganizationsForWorkers`	`Boolean`	`true`		✕ No
Indicates if the Organization Data element is included in the response.				
`includePersonalInformationForWorkers`	`Boolean`	`true`		✕ No
Indicates if the Personal Data element is included in the response.				
`excludeCostCentersForWorkers`	`Boolean`	`false`		✕ No
Excludes the cost centers from the Organization Data element response. This can only be selected when the Include Organization Data boolean is also selected.				
`excludeCustomOrganizationsForWorkers`	`Boolean`	`true`		✕ No
Excludes the custom organizations from the Organization Data element response. This can only be selected when the Include Organization Data boolean is also selected.				

Property	Type	Default	Encrypted[1]	Required[2]
`includeRolesForWorkers`	`Boolean`	`false`		✕ No
Indicates if the Role Data element is included in the response.				
`excludeMatrixOrganizationsForWorkers`	`Boolean`	`true`		✕ No
Excludes the matrix organizations from the Organization Data element response. This can only be selected when the Include Organization Data boolean is also selected.				
`includeEmploymentInformationForWorkers`	`Boolean`	`true`		✕ No
Indicates if the Employment Data element is included in the response.				
`includeAccountProvisioningForWorkers`	`Boolean`	`false`		✕ No
Indicates that Account Provisioning Data will be included in the web service response.				
`excludeBusinessUnitHierarchiesForWorkers`	`Boolean`	`true`		✕ No
Excludes the business unit hierarchies from the Organization Data element response. This can only be selected when the Include Organization Data boolean is also selected.				
`includeRelatedPersonsForWorkers`	`Boolean`	`false`		✕ No
Indicates if the Related Person Data element is included in the response.				
`includePhotoForWorkers`	`Boolean`	`false`		✕ No
Indicates if the Photo Data element is included in the response.				
`excludeSupervisoryOrganizationsForWorkers`	`Boolean`	`true`		✕ No
Excludes the supervisory organizations from the Organization Data element response. This can only be selected when the Include Organization Data boolean is also selected.				
`excludeTeamsForWorkers`	`Boolean`	`false`		✕ No
Excludes the teams from the Organization Data element response. This can only be selected when the Include Organization Data boolean is also selected.				
`includeTransactionLogDataForWorkers`	`Boolean`	`true`		✕ No

Property	Type	Default	Encrypted[1]	Required[2]
Indicates if the Transaction Log Data element is included in the response.				
`excludeCompaniesForWorkers`	`Boolean`	`false`		✕ No
Excludes the company organizations from the Organization Data element response. This can only be selected when the Include Organization Data boolean is also selected.				
`excludeBusinessUnitsForWorkers`	`Boolean`	`false`		✕ No
Excludes the Business Units from the Organization Data element response. This can only be selected when the Include Organization Data boolean is also selected.				
`includeEmployeeContractDataForWorkers`	`Boolean`	`false`		✕ No
Indicates if the Employee Contract element is included in the response.				
`includeUserAccountForWorkers`	`Boolean`	`true`		✕ No
Indicates that User Account Data will be included in the web service response.				
`excludeRegionsForWorkers`	`Boolean`	`false`		✕ No
Excludes the regions from the Organization Data element response. This can only be selected when the Include Organization Data boolean is also selected.				
`includeMultipleManagersInManagementChainDataForWorkers`	`Boolean`	`false`		✕ No
If set to true, multiple managers in the management chain data are returned.				
`includeAdditionalJobsForWorkers`	`Boolean`	`false`		✕ No
If true, the data for Additional Jobs and International Assignments is included in the response.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

[2] A list of operations in this column indicates that the property is required for those operations.

Organization Response Configuration Properties

Property	Type	Default	Encrypted[1]	Required[2]
`includeStaffingRestrictionsDataForOrganizations`	`Boolean`	`false`		✕ No
Indicates whether the Staffing Restrictions Data element is included in the response.				

Property	Type	Default	Encrypted[1]	Required[2]
`includeSupervisoryDataForOrganizations`	`Boolean`	`false`		✕ No
Indicates whether the Supervisory Data element is included in the response.				
`includeHierarchyDataForOrganizations`	`Boolean`	`false`		✕ No
Indicates whether the Hierarchy Data element is included in the response.				
`includeRolesDataForOrganizations`	`Boolean`	`false`		✕ No
Indicates whether the Roles Data element is included in the response.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

[2] A list of operations in this column indicates that the property is required for those operations.

Basic Configuration Properties

Property	Type	Default	Encrypted[1]	Required[2]
`hostname`	`String`	`null`		✓ Yes
The hostname for the Workday service. Example: https://[workday hostname]/ccx/service/[workday tenant]/. You need to configure the bracketed Workday hostname and tenant name to successfully connect to the proper instance.				
`xpathTransformations`	`Map[]`	`[]`		✕ No
List of xpath transformations to connector attributes (Supported by framework versions greater than 1.5.20.23).				
`tenant`	`String`	`null`		✓ Yes
The tenant in URL for the Workday service. For example: https://[workday hostname]/ccx/service/[workday tenant]/. You need to configure the bracketed Workday hostname and tenant name to successfully connect to the proper instance.				
`username`	`String`	`null`		✓ Yes
The username for logging into the Workday service. It will be concatenated with the tenant name (user@tenant)				
`password`	`GuardedString`	`null`	🔒 Yes	✓ Yes
The user password for logging into the Workday service				
`excludeInactiveWorkers`	`boolean`	`false`		✕ No

Property	Type	Default	Encrypted[1]	Required[2]
Excludes terminated employees or contingent workers whose contracts have ended from the query response (defaults to false).				
`excludeContingentWorkers`	`boolean`	`false`		✕ No
Excludes contingent workers from the query response.				
`excludeEmployees`	`boolean`	`false`		✕ No
Excludes employees from the query response.				
`connectionTimeout`	`int`	`30`		✕ No
Specifies the amount of time, in seconds, that the client will attempt to establish a connection before it times out. The default is 30 seconds). Set to 0 for no timeout.				
`receiveTimeout`	`int`	`60`		✕ No
Specifies the amount of time, in seconds, that the client will wait for a response before it times out. The default is 60. Set to 0 for no timeout.				
`pageSize`	`long`	`100`		✕ No
Sets the page size used for search operations (defaults to 100).				
`proxyHost`	`String`	`null`		✕ No
If defined, the connection to Workday goes through this HTTP proxy server.				
`proxyPort`	`int`	`8080`		✕ No
The HTTP proxy server port number (defaults to 8080).				
`xslTransformer`	`File`	`null`		✕ No
The file path to the XSL File to get the custom attributes.				
`asOfEffectiveDate`	`String`	`null`		✕ No
Optional configuration of Response_Filter/As_Of_Effective_Date element. Valid values are: Date (http://www.w3.org/TR/xpath-functions/#date-time-values http://www.w3.org/TR/xmlschema-2/#dateTime-order) or Duration (http://www.w3.org/TR/xpath-functions/#dt-dayTimeDuration). If set to Duration, the effective date is calculated as current date + duration.				
`effectiveFrom`	`String`	`null`		✕ No

Property	Type	Default	Encrypted[1]	Required[2]
Sets the Get_Workers_Request/Request_Criteria/Transaction_Log_Criteria_Data/Transaction_Date_Range_Data/Effective_From for every outbound query request. Valid value could be Date (http://www.w3.org/TR/xpath-functions/#date-time-values http://www.w3.org/TR/xmlschema-2/#dateTime-order) or string Today representing the current time of the request.				
`effectiveThrough`	`String`	`null`		✕ No
Sets the Get_Workers_Request/Request_Criteria/Transaction_Log_Criteria_Data/Transaction_Date_Range_Data/Effective_Through for every outbound query request. Valid value could be Date (http://www.w3.org/TR/xpath-functions/#date-time-values http://www.w3.org/TR/xmlschema-2/#dateTime-order) or Duration (http://www.w3.org/TR/xpath-functions/#dt-dayTimeDuration).				
`externalFieldAndParameterCriteria`	`String[]`	`[]`		✕ No
A list of external fields to add to the search/query criteria.				

[1] Whether the property value is considered confidential, and is therefore encrypted in IDM.

[2] A list of operations in this column indicates that the property is required for those operations.

Remote connectors

In many cases, IDM bundles the connectors required to connect to remote resources and assumes the connector runs on the same host as IDM. Sometimes, a connector can't run on the same host as IDM. This might be for security or network reasons or because IDM runs in the cloud while the resource is on-prem. Connectors that do not run on the same host as IDM are *remote connectors*. To run remotely, a connector needs a *Remote Connector Server* (RCS) that runs on the same host as the connector. IDM accesses the connector through the RCS.

Running connectors remotely requires the following high-level steps:

1. Install an RCS and any dependencies required.
   - Install Java RCS (Recommended)

     > **Tip**
     >
     > You can also deploy Java RCS in a Docker container.

   - Install a .NET-based RCS
2. RCS contains many bundled connectors, but you can download additional connectors from the BackStage download site, and place the .jar file or .dll file on your remote server in the `/path/to/openicf/connectors/` directory.
3. Configure IDM to connect to the RCS.

For a list of supported RCS versions, and compatibility between versions, refer to IDM / ICF Compatibility Matrix.

Install a Remote Connector Server (RCS)

There are two types of RCS:

- **Java**: Use the Java RCS if your Java connector needs to run in a different JVM to IDM. Unless the remote resource you are connecting to needs the .NET PowerShell connector, this is the recommended RCS to use.

  The system on which you install the Java RCS must run JRE version 17. Disk space and memory requirements depend on the number of connectors you are using and the volume of traffic through the RCS.

  > **Tip**
  >
  > You can also deploy Java RCS in a Docker container.

- **.NET**: Use the .NET RCS if you are using the PowerShell connector to connect to an identity store. IDM communicates with the .NET RCS over the network and the RCS runs the PowerShell connector.

Install connector dependencies

In most cases, ICF connectors come bundled with all third party libraries needed to run. In some cases, however, you'll need to download certain libraries (for example, the Database Table connector needs the appropriate JDBC driver for the database you are targeting). For local connectors, place these libraries in the `/path/to/openidm/lib/` directory. For remote connectors, place them in the `/path/to/openicf/lib/` directory on the RCS.

The following table lists the connector dependencies and indicates which ones must be downloaded:

Dependencies for bundled connectors	
Database Table connector	No external dependencies. However, you must include the JDBC driver for the database that you are targeting in the `/path/to/openidm/lib/` directory.
PeopleSoft connector	`lib/psft.jar` `lib/psjoa.jar`
SAP connector	`lib/sapjco3.jar` `lib/libsapjco3.so`

Install Java RCS

The Java RCS is supported on any platform on which Java runs and requires the following Java version(s):

- For Java RCS version 1.5.20.22 and earlier, Java 11 or 17.
- For Java RCS version 1.5.20.23 and later, Java 17.

Disk space, memory, and CPU requirements depend on the number of connectors you are using and the volume of traffic through the RCS.

> **Tip**
>
> You can also deploy Java RCS in a Docker container.

Install a Java RCS on Unix/Linux

1. Download and extract the Java RCS from the BackStage download site.
2. Change to the `openicf` directory:

```
cd /path/to/openicf
```

3. Review the `ConnectorServer.properties` file in the `/path/to/openicf/conf` directory, and adjust it to suit your deployment. For a complete list of properties in that file, refer to RCS Properties.
   - In server mode, the RCS uses a `connectorserver.key` property to authenticate the connection. The default value of the key is a hashed value of the string `changeit`. You cannot set this property directly in the configuration file. To change its value, use the command `ConnectorServer.sh /setKey`. This example sets the key value to `Passw0rd`:

```
/path/to/openicf/bin/ConnectorServer.sh /setKey Passw0rd
Key has been successfully updated.
```

In client mode, this isn't necessary and can be skipped. Learn more about the differences between client mode and server mode in Configure a remote connector server (RCS).

4. Start the Java RCS:

```
/path/to/openicf/bin/ConnectorServer.sh /run
```

By default, the RCS is now running and listening on port `8759`.

Log files are available in the `/path/to/openicf/logs` directory.

```
ls logs/
Connector.log  ConnectorServer.log  ConnectorServerTrace.log
```

5. To stop the Java RCS, press CTRL + C or q in the terminal where you started the server.

Install a Java RCS on Windows

1. Download and extract the Java RCS from the BackStage download site.

2. In a Command Prompt window, change to the `openicf` directory:

```
C:\> cd C:\path\to\openicf
```

3. Review the `ConnectorServer.properties` file in the `\path\to\openicf\conf` directory, and adjust it to suit your deployment. For a complete list of properties in that file, refer to RCS Properties.

   ◦ In server mode, the RCS uses a `connectorserver.key` property to authenticate the connection. The default value of the key is a hashed value of the string `changeit`. You cannot set this property directly in the configuration file. To change its value, use the `ConnectorServer.bat /setKey` command. This example sets the key value to `Passw0rd`:

```
c:\path\to\openicf> bin\ConnectorServer.bat /setKey Passw0rd
Key has been successfully updated.
```

Important

If you use an exclamation mark (`!`) with the `ConnectorServer.bat /setKey` command, you must escape the character according to your Windows Server version:

- For Windows Server 2012 R2, escape the character with double quotes:

```
ConnectorServer.bat /setKey "changeit"!""
```

- For Windows Server 2016 and later, escape the character with a caret (`^`):

```
ConnectorServer.bat /setKey changeit^!
```

In client mode, this isn't necessary and can be skipped. Learn more about the differences between client mode and server mode in Configure a remote connector server (RCS).

4. You can either run the Java RCS as a Windows service or start and stop it from the command line.

    1. To install the Java RCS as a Windows service, run the following command:

        ```
        c:\path\to\openicf> bin\ConnectorServer.bat /install
        ```

        If you install the RCS as a Windows service, you can use the Microsoft Services Console to start, stop, and restart the service. The Java Connector Service is named `OpenICFConnectorServerJava`.

        To uninstall the Java RCS as a Windows service, run the following command:

        ```
        c:\path\to\openicf> bin\ConnectorServer.bat /uninstall
        ```

    2. To start the Java RCS from the command line, enter the following command:

        ```
        c:\path\to\openicf> bin\ConnectorServer.bat /run
        ```

5. The RCS is now running, and listening on port `8759`, by default.

    Log files are available in the `\path\to\openicf\logs` directory.

6. To stop the Java RCS, press `CTRL`+`C`.

Run Java RCS as a service

The Java RCS can run as a service on a standard `systemd`-based Linux distribution. Once you have configured the RCS as a service, you can stop and start the RCS using `systemd`.

Configure RCS as a service

1. Create a service file using your preferred text editor:

    ```
    sudo vim /etc/systemd/system/rcs.service
    ```

2. Add the following content to this file, update the paths as needed, and save:

```
[Unit]
SourcePath=/path/to/openicf/bin
Description=Remote Connector Server (systemd init)
After=network.target
Conflicts=shutdown.target

[Service]
Type=simple
Restart=always
RestartSec=5sec
IgnoreSIGPIPE=no
KillMode=process
Environment="OPENICF_OPTS=-Xmx1024m"
ExecStart=/path/to/openicf/bin/ConnectorServer.sh /start

[Install]
WantedBy=multi-user.target
```

3. Make the new service launch on startup:

```
sudo systemctl enable rcs.service
```

4. Check the service is enabled:

```
systemctl is-enabled rcs.service
```

This command returns `enabled` or `disabled` as appropriate.

Use systemctl commands to manage the RCS service

Once you've configured RCS as a service and checked it's enabled, use `systemctl` commands to manage the RCS service:

- Start the service:

```
sudo systemctl start rcs.service
```

- Stop the service:

```
sudo systemctl stop rcs.service
```

- Restart the service:

```
sudo systemctl restart rcs.service
```

- Check the service status:

```
sudo systemctl status rcs.service
```

This command returns the service state (whether the service has started or stopped as expected) and the first few entries of the RCS log file.

Deploy Java RCS in a Docker container

You can deploy the Java Remote Connector Server (RCS) in a Docker container. This topic outlines the process to obtain, customize, configure, and run RCS using a Docker environment. The RCS download includes a sample dockerfile at `path/to/openicf/docker/Dockerfile`.

Getting started

You can pull RCS images from the `gcr.io/forgerock-io/rcs` image repository in the public Container Registry (browser access requires a Google account).

A public RCS image provides the default installation of the OpenICF (ICF) framework.

Customize the parent RCS image

To extend the parent image functionality, customize it by copying additional files to the ICF installation folder (`/opt/openicf`) in the final image. Additional files can include connector packages, scripts, drivers, and so on.

Capture RCS image customizations by saving them in a custom Docker image using a custom `Dockerfile`.

The following procedure includes the five example sections from the custom `Dockerfile` included with RCS 1.5.20.30 and later:

1. Accept build arguments and select the parent image.

   Dockerfile excerpt

   ```
   # 1. Accept build arguments in the FROM instruction.
   # @example
   ARG FROM_TAG=1.5.20.30
   FROM gcr.io/forgerock-io/rcs:$FROM_TAG
   ```

   During the `docker build` phase, use the `--build-arg` flag to specify the `gcr.io/forgerock-io/rcs` repository parent image tag for building the RCS image. For example:

   ```
   docker build --build-arg FROM_TAG=1.5.20.30 -t rcs .
   ```

2. Add additional files to the RCS installation.

Dockerfile excerpt

```
# 2. Merge custom files with the RCS installation.
# @example
COPY openicf/conf conf
COPY openicf/connectors connectors
COPY openicf/lib lib
COPY openicf/scripts scripts
```

Copy local files into the `/opt/openicf` location in the parent RCS image to extend or modify its functionality. The `openicf` installation folder structure can be replicated locally to copy everything at once, or separate `COPY` instructions can be used for individual folders and files, which can help cache respective layers in the final image.

Note

The RCS parent image `Dockerfile` specifies a working directory: `WORKDIR /opt/openicf`. In the `COPY` instruction destination, use an absolute path or a path relative to the `WORKDIR` location.

Files and directories

- `openicf/conf`

  Provide a custom version of the `openicf/conf/ConnectorServer.properties` file. Specify all or some RCS configuration properties in this file, as described in Configure a remote connector server (RCS).

- `openicf/connectors`

  Add custom or downloaded connectors packaged in JAR files into this sub-directory.

- `openicf/lib`

  Add any additional dependencies used by connectors into this sub-directory. For example, required database drivers.

  Provide a customized version of the `openicf/conf/logback.xml` file to overwrite the default one. This adjusts the detail of logs produced by the RCS, as described in Connector logs.

  You can find the resulting log files in a running container within the `/opt/openicf/logs` folder. Logs are printed to standard output by the container in the default attached mode or can be requested explicitly from the container with the `docker logs` command.

- `openicf/scripts`

  Add scripted connector Groovy scripts to this directory.

- `openicf/security`

  Contains the default RCS keystore and truststore.

> **Tip**
>
> You can copy any other content the connectors need to any other accessible location within the connector server image.

3. Use root user for operations requiring escalated privileges.

   Dockerfile excerpt

   ```
   # 3. Use root user for operations that require escalated permissions.
   # @example
   # USER root
   ```

   By default, a Docker container runs as the root user. It's recommended to run a Docker container as a non-root user, which is addressed in the RCS parent image `Dockerfile` that typically specifies `USER 11111`.

   However, some instructions require elevated permissions. For example, adding an untrusted certificate to the Java truststore requires root user privileges.

4. Add certificates to RCS truststore.

Dockerfile excerpt

```
# 4. Add untrusted public TLS certificate(s) to RCS `truststore` to allow for communications with
unrecognized hosts.
# Run `docker build` with the `--progress plain` option to check the outcome of the RUN instruction.
# @example
# COPY openicf/security/BadSSL-Untrusted-Root-Certificate-Authority.cer /opt/openicf/security
# RUN keytool -keystore $JAVA_HOME/lib/security/cacerts -storepass changeit -trustcacerts -import -
file /opt/openicf/security/BadSSL-Untrusted-Root-Certificate-Authority.cer -alias badssl-com-
untrusted-root-ca -noprompt
```

In the previous `Dockerfile` excerpt example, a public certificate is added to the Java `truststore` to allow RCS communications with an otherwise untrusted host, for instance, a third-party API. The destination of the `COPY` instruction, in this case, can be any accessible location within the image, because the change is applied using a command provided in the `RUN` instruction, not directly by ICF.

> **Note**
>
> PingOne Advanced Identity Cloud certificates are signed by a trusted CA and don't need to be added to the `truststore`.

5. Switch to a non-root user.

Dockerfile excerpt

```
 # 5. Switch back to a non-root user.
 # @example
 # USER 11111
```

After performing any specific instructions that require elevated permissions, switch back to the non-root user using the `USER` instruction.

During development, you might want to continue as a root user. For example, root privileges are required to enable file sync when using a Docker image in Skaffold development mode. In this case, specify the user in the `docker run` command with the `--user` or `-u` flag.

Pushing the RCS image to a Docker repository

To share the RCS image and make it available for orchestration tools, push it to a Docker repository.

If the Docker repository is served from a private Docker registry, ensure Docker is configured with appropriate privileges to push to this repository. The Docker registry provider typically provides specific instructions on setting up authentication for Docker.

For example, documentation on setting up authentication for Docker describes how to use the `gcloud` credential helper for configuring Docker with Google Cloud CLI session credentials. This can be used while developing RCS images and pushing them to a Google Cloud Artifact Registry.

The image must be tagged before pushing it to a repository. Tag the image with the reference to the repository in the `docker build` command using the `-t` or `--tag` flag. For example:

```
# Build and tag the image with the repository reference
$ docker build --build-arg FROM_TAG=1.5.20.30 -t gcr.io/<organization>/<username>/rcs .

# Push the tagged image
$ docker push gcr.io/<organization>/<username>/rcs
```

Registering RCS

Configure the RCS in client mode to connect it to a managed environment, such as Advanced Identity Cloud.

In client mode, the `connectorserver.connectorServerName` property must be provided in the RCS configuration. This property must match the registered name in IDM or Advanced Identity Cloud:

- In Advanced Identity Cloud, a connector server can be registered using the Advanced Identity Cloud admin UI (refer to the documentation on syncing identities and registering a remote server). In the UI, the **Name** input for a connector server accepts a value containing lower-case alphanumeric characters, hyphens (`-`), and underscores (`_`). Advanced Identity Cloud saves the **Name** value as the `connectorServerName` property in the connector server configuration.
- IDM saves the **Name** value as the `name` property of a `remoteConnectorClients` entry in the `provisioner.openicf.connectorinfoprovider.json` file.

Configuring RCS

To configure RCS in client mode, define at least the following RCS configuration properties:

- `connectorserver.url`

  > **Important**
  >
  > To specify more than one `connectorserver.url`, separate each value using a comma (`,`). Java RCS versions earlier than 1.5.20.26 don't support multiple values for `connectorserver.url` when using a Docker container.

- `connectorserver.connectorServerName`

To connect to an Advanced Identity Cloud tenant, the following must also be specified:

- `connectorserver.tokenEndpoint`
- `connectorserver.clientId`
- `connectorserver.clientSecret`
- `connectorserver.scope`

To enable more controlled debug output, specify the RCS logger class:

- `connectorserver.loggerClass`

Static connector server properties can be provided using instructions included in the `openicf/conf/ConnectorServer.properties` file. In addition, the following sample configurations are available in `openicf/conf/samples`:

- `ConnectorServer.properties.onprem-server`

- `ConnectorServer.properties.onprem-client`
- `ConnectorServer.properties.default-parameters`
- `ConnectorServer.properties.cloud-client`

For example, a `ConnectorServer.properties` file could contain the following instructions for setting properties that are unlikely to change and carry no sensitive data and could, therefore, be embedded into the image:

conf/ConnectorServer.properties

```
# Set connectorserver properties.
# See OpenICF documentation for details:
# https://docs.pingidentity.com/openicf/connector-reference/configure-server.html#rcs-properties

# Set static properties.
connectorserver.scope=fr:idm:*
connectorserver.loggerClass=org.forgerock.openicf.common.logging.slf4j.SLF4JLog
```

To provide dynamic values in the connector server configuration, which is a suitable approach for secrets and environment specifics, define JVM system properties using the `OPENICF_OPTS` environment variable. This variable needs to be exported into the JVM's environment before the connector server starts. Provide this information when the container runs so it's not embedded into the image.

Running RCS

Create and run an RCS container with the `docker run` command.

In a standalone Docker container, environment variables can be set with one of the `-e`, `--env`, or `--env-file` flags. For example:

1. Set environment variables with an `.env` file:

   .env file content

   ```
   OPENICF_OPTS=-Dconnectorserver.url=wss://<your-tenant-url>/openicf/0 -Dconnectorserver.tokenEndpoint=https://
   <your-tenant-url>/am/oauth2/realms/root/realms/alpha/access_token -Dconnectorserver.connectorServerName=rcs-
   docker-1 -Dconnectorserver.clientId=RCSClient -Dconnectorserver.clientSecret=YA...H?
   ```

2. User `docker run` and supply the environment file:

   ```
   $ docker run --env-file .env rcs
   ```

A Docker environment file is limited to one-line variable definitions and can be difficult to read. An alternative approach is defining the environment variable on the host machine and then using the `-e` or `--env` flag. For example:

1. Define `OPENICF_OPTS` in a file (`.env` in this example):

```
# .env file content
OPENICF_OPTS="-Dconnectorserver.url=wss://openam-dx-kl04.forgeblocks.com/openicf/0 \
-Dconnectorserver.tokenEndpoint=https://openam-dx-kl04.forgeblocks.com/am/oauth2/realms/root/realms/alpha/
access_token \
-Dconnectorserver.connectorServerName=rcs-docker-1 \
-Dconnectorserver.clientId=RCSClient \
-Dconnectorserver.clientSecret=YA...H?"
```

2. Source the file and run the container:

    ```
    # Export variables from .env file into the current shell
    $ set -a; source .env; set +a;

    # Run the container using the exported variable
    $ docker run -e OPENICF_OPTS rcs
    ```

    This lets you set sensitive and dynamic properties without embedding them into the image.

    You can substitute values using `OPENICF_OPTS` to address different environments or different connector server names registered in IDM or Advanced Identity Cloud. For example, you could run multiple containers referring to different connector server names registered in a server cluster.

Developing scripted connectors with Docker

The parent RCS image doesn't ship with any Groovy scripts. While developing scripted connectors, use the `docker run` flag `-v` or `--volume` as a convenient way of providing script content to the running RCS container. For example:

```
docker run --rm --env-file .env -v ./openicf/scripts:/opt/openicf/scripts rcs
```

Now, any updates to the scripts made in the `./openicf/scripts` folder on the host machine become available to the RCS running in the container.

Upgrade Java RCS

Before you upgrade the Java RCS and any connectors running on it, you must update the connector configuration (specifically the `bundleVersion`) to include the versions you're upgrading to. The `bundleVersion` defines a range of versions or a specific version.

If the `bundleVersion` doesn't include the RCS version or connector version you're upgrading to, the new RCS or connector won't be usable after the upgrade.

> **Tip**
>
> Use a range for the `bundleVersion` to make future RCS and connector upgrades easier. The following example includes all versions starting from `1.5.0.0` up to, but not including, `1.6.0.0`:
>
> ```
> "bundleVersion": "[1.5.0.0,1.6.0.0)"
> ```

Update connector configuration

For IDM, you can update the connector over REST or in the provisioner file. For Advanced Identity Cloud, you can only update the configuration over REST.

Update connector configuration over REST

1. Get the connector configuration:

   ```
   curl \
   --header "X-OpenIDM-Username: openidm-admin" \
   --header "X-OpenIDM-Password: openidm-admin" \
   --header "Content-Type: application/json" \
   --request GET \
   "http://localhost:8080/openidm/config/provisioner.openicf/connectorName"
   ```

   > **Tip**
   >
   > Keep a copy of this configuration before making changes in case you need to re-create the connector.

2. Check the `bundleVersion` in the response includes the new RCS and connector versions.

   If the `bundleVersion` in the response includes the new RCS and connector versions, move ahead to Upgrade the RCS and connectors; otherwise, continue to the next step of this procedure.

3. Update the `bundleVersion` to include the new RCS and connector versions. This example uses non-matching brackets to specify a range:

   ```
   "bundleVersion": "[1.5.0.0,1.6.0.0)"
   ```

   > **Note**
   >
   > For more information about version range options, refer to Connector reference properties.

4. Use a PUT request to update the connector configuration previously returned with the updated `bundleVersion`:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request PUT \
--data-raw '{complete-configuration}' \
"http://localhost:8080/openidm/config/provisioner.openicf/connectorName"
```

5. If you are using Advanced Identity Cloud, promote your connector configuration changes to staging and production.

Update connector configuration in provisioner file (IDM only)

1. Open the `project-dir/conf/provisioner.openicf-connectorName` file.

   If the `bundleVersion` in the file includes the new RCS and connector versions, move ahead to Upgrade the RCS and connectors; otherwise, continue to the next step of this procedure.

2. Update the `bundleVersion` to include the new RCS and connector versions. This example uses non-matching brackets to specify a range:

   ```
   "bundleVersion": "[1.5.0.0,1.6.0.0)"
   ```

   > **Note**
   >
   > For more information about version range options, refer to Connector reference properties.

Upgrade the RCS and connectors

Upgrade the RCS and any connectors running on the RCS

> **Tip**
>
> You can upgrade a connector running on the RCS without upgrading the RCS if needed.

1. Download the new Java RCS and any required connectors from BackStage download site.

2. Stop the RCS:

   - If the RCS is running in a terminal, press CTRL + C or q.
   - If the RCS is running as a service, stop the service:

     ```
     sudo systemctl stop rcs.service
     ```

3. Rename the existing RCS directory to create a backup:

```
mv /path/to/openicf /path/to/openicf_old
```

4. Extract the downloaded Java RCS to the original directory (`/path/to/openicf`).

5. Copy the downloaded connector .jar file(s) to the `/path/to/openicf/connectors` directory.

6. Copy any custom connectors, such as scripted connectors from your backup to the `/path/to/openicf/connectors` directory to retain your customizations.

7. Copy the following files from your backup to the new RCS directory to retain your previous settings:

   ◦ `conf/logback.xml` (if the RCS is set up for debug logging)

   ◦ Any truststores and keystores. These are typically located in the `/path/to/openicf/security` directory.

8. Copy the applicable lines from your backup `conf/ConnectorServer.properties` file to the new RCS version of the file. Copying specific lines from the file ensures you get the latest updates to new and important configuration properties. In a typical deployment, you should copy:

ConnectorServer.properties excerpt

```
connectorserver.url=wss://my-tenant.forgeblocks.com:8443/openicf (1)
connectorserver.connectorServerName=myConnectorServer (1)
connectorserver.hostId=MY_UNIQUE_RCS_HOST_ID (1)
connectorserver.tokenEndpoint=https://my-tenant.forgeblocks.com/am/oauth2/realms/root/realms/alpha/
access_token (1)
connectorserver.clientId=my-client-id (1)
connectorserver.clientSecret=my-client-secret (1)
connectorserver.trustStoreFile=security/truststore.pkcs12 (2)
connectorserver.trustStoreType=PKCS12 (2)
connectorserver.trustStorePass=changeit (2)
connectorserver.keyStoreFile=security/keyStore.pkcs12 (3)
connectorserver.keyStoreType=PKCS12 (3)
connectorserver.keyStorePass=changeit (3)
connectorserver.keyPass=changeit (3)
connectorserver.proxyHost=my-proxy-host (4)
connectorserver.proxyPort=my-proxy-port (4)
connectorserver.proxyPrincipal=my-proxy-principal (4)
connectorserver.proxyPassword=my-proxy-password (4)
```

1 Connection details.
2 Copy these lines if you have configured a truststore.
3 Copy these lines if you have configured a keystore.
4 Copy these lines if you have confiured a proxy.

9. Start the RCS:

Terminal

```
/path/to/openicf/bin/ConnectorServer.sh /run
```

Service

```
sudo systemctl start rcs.service
```

Note

In Advanced Identity Cloud, you must:

1. Make these changes to the RCS in all three Advanced Identity Cloud environments (development, staging, and production).
2. Restart your staging and production environments using the Restart API endpoint:

```
curl \
--header "authorization: Bearer <access-token>" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json"
--request POST \
"https://<tenant-env-fqdn>/environment/startup?_action=restart"
```

Upgrade a connector running on the RCS

You can upgrade a connector running on the RCS without upgrading the RCS itself.

1. Download the new connector version from BackStage download site.

2. Stop the RCS:

   - If the RCS is running in a terminal, press CTRL + C or q.

   - If the RCS is running as a service, stop the service:

```
sudo systemctl stop rcs.service
```

3. Copy the downloaded connector .jar file to the `/path/to/openicf/connectors` directory and remove the old .jar file version.

4. Start the RCS:

Terminal

```
/path/to/openicf/bin/ConnectorServer.sh /run
```

Service

```
sudo systemctl start rcs.service
```

Note

In Advanced Identity Cloud, you must:

1. Make these changes to the RCS in all three Advanced Identity Cloud environments (development, staging, and production).
2. Restart your staging and production environments using the Restart API endpoint:

```
curl \
--header "authorization: Bearer <access-token>" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json"
--request POST \
"https://<tenant-env-fqdn>/environment/startup?_action=restart"
```

Install .NET RCS

The .NET remote connector server (RCS) requires .NET 4.6.2 or later, and Windows Server version 2012 R2, 2016, 2019, or 2022. Exact memory, disk space, and CPU requirements will vary on the connectors used and how many connectors are run.

The .NET RCS comes bundled with a PowerShell connector. Refer to PowerShell connector toolkit for more information on how to configure and run this connector.

Installation

The .NET RCS is distributed in two file formats:

- `openicf-version-dotnet.msi` is a wizard that installs the RCS as a Windows service.
- `openicf-version-dotnet.zip` is just a bundle of files required to run the RCS.

Depending on how you want to install the RCS, download the corresponding file from the BackStage download site.

1. Double-click the `openicf-version-dotnet.msi` installation file and complete the wizard.

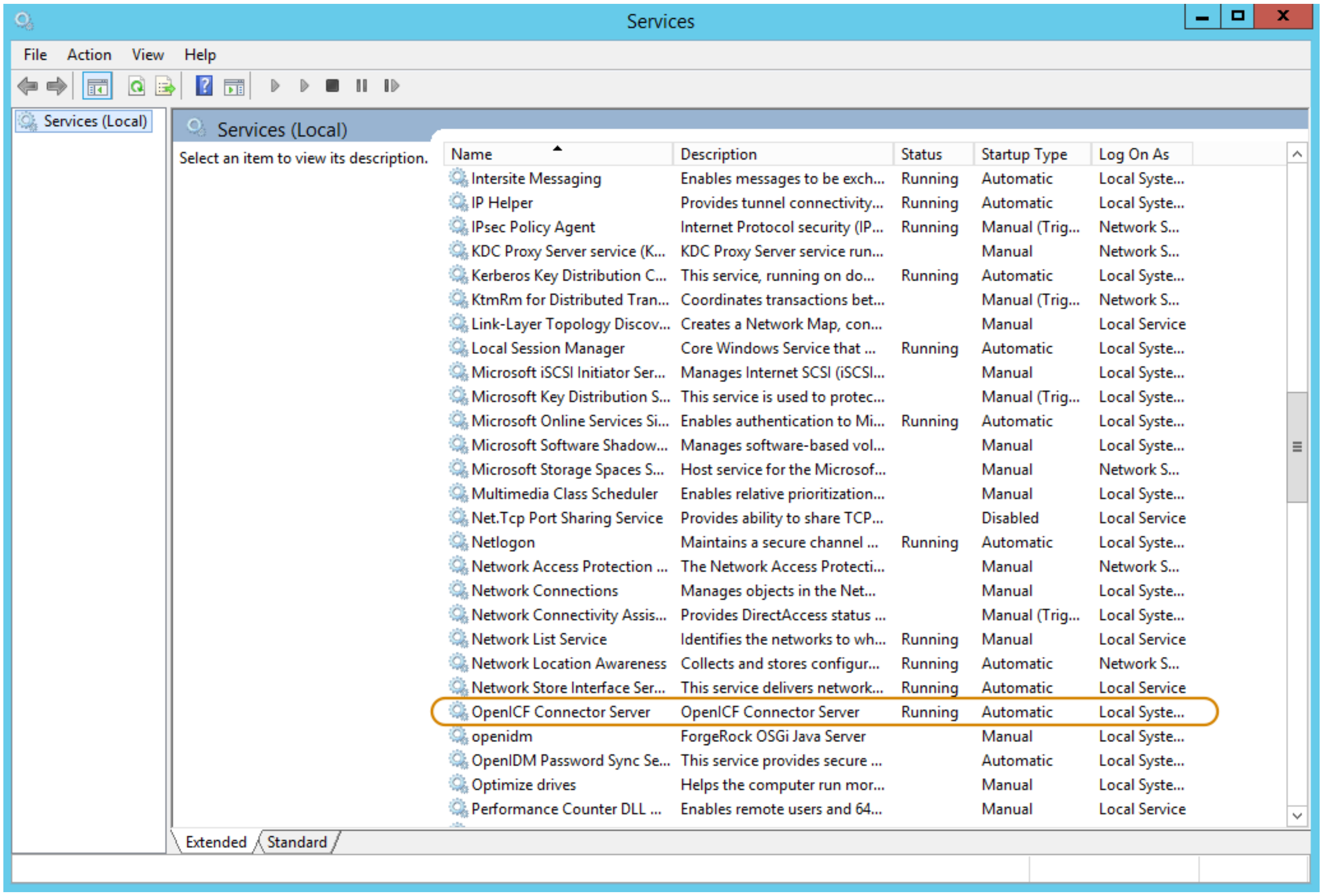
You must run the wizard as a user who has permission to start and stop a Windows service; otherwise, the service will not start.

Select **Typical** as the **Setup Type**.

When the wizard has completed, the RCS is installed as a Windows service.

2. Open the Microsoft Services Console and make sure that the RCS is listed there.

By default, the name of the service is `OpenICF Connector Server`.



Name	Description	Status	Startup Type	Log On As
Intersite Messaging	Enables messages to be exch...	Running	Automatic	Local Syste...
IP Helper	Provides tunnel connectivity...	Running	Automatic	Local Syste...
IPsec Policy Agent	Internet Protocol security (IP...	Running	Manual (Trig...	Network S...
KDC Proxy Server service (K...	KDC Proxy Server service run...		Manual	Network S...
Kerberos Key Distribution C...	This service, running on do...	Running	Automatic	Local Syste...
KtmRm for Distributed Tran...	Coordinates transactions bet...		Manual (Trig...	Network S...
Link-Layer Topology Discov...	Creates a Network Map, con...		Manual	Local Service
Local Session Manager	Core Windows Service that ...	Running	Automatic	Local Syste...
Microsoft iSCSI Initiator Ser...	Manages Internet SCSI (iSCSI...		Manual	Local Syste...
Microsoft Key Distribution S...	This service is used to protec...		Manual (Trig...	Local Syste...
Microsoft Online Services Si...	Enables authentication to Mi...	Running	Automatic	Local Syste...
Microsoft Software Shadow...	Manages software-based vol...		Manual	Local Syste...
Microsoft Storage Spaces S...	Host service for the Microsof...		Manual	Network S...
Multimedia Class Scheduler	Enables relative prioritization...		Manual	Local Syste...
Net.Tcp Port Sharing Service	Provides ability to share TCP...		Disabled	Local Service
Netlogon	Maintains a secure channel ...	Running	Automatic	Local Syste...
Network Access Protection ...	The Network Access Protecti...		Manual	Network S...
Network Connections	Manages objects in the Net...		Manual	Local Syste...
Network Connectivity Assis...	Provides DirectAccess status ...		Manual (Trig...	Local Syste...
Network List Service	Identifies the networks to wh...	Running	Manual	Local Service
Network Location Awareness	Collects and stores configur...	Running	Automatic	Network S...
Network Store Interface Ser...	This service delivers network...	Running	Automatic	Local Service
OpenICF Connector Server	OpenICF Connector Server	Running	Automatic	Local Syste...
openidm	ForgeRock OSGi Java Server		Manual	Local Syste...
OpenIDM Password Sync Se...	This service provides secure ...		Automatic	Local Syste...
Optimize drives	Helps the computer run mor...		Manual	Local Syste...
Performance Counter DLL ...	Enables remote users and 64...		Manual	Local Service

> **Note**
>
> Before continuing with the setup, make sure the RCS is not currently running. If it is running, use the Microsoft Services Console to stop it.

1. If you do *not* want to run the RCS as a Windows service, download and extract the `openicf-version-dotnet.zip` file.

   - If you have already extracted the .zip file and then decide to run the RCS as a service, install the service manually with the following command:

```
.\ConnectorServerService.exe /install /serviceName service-name
```

2. At the command prompt, change to the directory where the RCS was installed, for example:

```
cd "c:\Program Files (x86)\ForgeRock\OpenICF"
```

Server setup

1. Run the `ConnectorServerService /setKey` command to set a secret key for the RCS. The key can be any string value. This example sets the secret key to `Passw0rd`:

   ```
   ConnectorServerService /setKey Passw0rd
   Key has been successfully updated.
   ```

   This key is used by clients connecting to the RCS. The key that you set here must also be set in the IDM or Advanced Identity Cloud Configure a remote connector server (RCS).

2. Edit the RCS configuration.

   The RCS configuration is saved in a file named `ConnectorServerService.exe.Config` (in the directory where the RCS is installed).

   Check and edit this file, as necessary, to reflect your installation. Specifically, verify that the `baseAddress` reflects the host and port on which the RCS is installed:

   ```
   <system.serviceModel>
     <services>
       <service name="Org.ForgeRock.OpenICF.Framework.Service.WcfServiceLibrary.WcfWebsocket">
         <host>
           <baseAddresses>
             <add baseAddress="http://0.0.0.0:8759/openicf" />
           </baseAddresses>
         </host>
       </service>
     </services>
   </system.serviceModel>
   ```

   ### Note

   The `baseAddress` specifies the host and port on which the RCS listens and is set to `http://0.0.0.0:8759/openicf` by default. If you set a host value other than the default `0.0.0.0`, connections from all IP addresses other than the one specified are denied.

   ### Important

   If Windows Firewall is enabled, you must create an inbound port rule to open the TCP port for the RCS (8759 by default). If you do not open the TCP port, IDM won't be able to contact the RCS. For more information, refer to the corresponding Microsoft documentation.

Configure the RCS to use SSL

> **Note**
>
> The following section does not apply to Advanced Identity Cloud, as it requires filesystem access to your installation.

1. Open a PowerShell terminal as a user with administrator privileges, then change to the ICF installation directory:

   ```
   cd 'C:\Program Files (x86)\ForgeRock\OpenICF'
   ```

2. Use an existing CA certificate, or use the `New-SelfSignedCertificate` cmdlet to create a self-signed certificate:

   ```
   New-SelfSignedCertificate -DnsName "dotnet", "dotnet.example.com" -CertStoreLocation "cert:
   \LocalMachine\My"
   PSParentPath: Microsoft.PowerShell.Security\Certificate::LocalMachine\My

   Thumbprint                                Subject
   ----------                                -------
   770F531F14AF435E963E14AD82B70A47A4BFFBF2  CN=dotnet
   ```

3. Assign the certificate to the RCS:

   ```
   .\ConnectorServerService.exe /setCertificate
   Select certificate you want to use:
   Index  Issued To                Thumbprint
   -----  ---------                -------------------------

   0)  dotnet                    770F531F14AF435E963E14AD82B70A47A4BFFBF2 0
   Certificate Thumbprint has been successfully updated to 770F531F14AF435E963E14AD82B70A47A4BFFBF2.
   ```

4. Bind the certificate to the RCS port (`8759` by default). To bind the certificate:

   1. Use the `New-Guid` cmdlet to generate a new UUID:

      ```
      New-Guid
      Guid
      ----
      0352cf0f-2e7a-4aee-801d-7f27f8344c77
      ```

   2. Enter the `netsh http` console and add the certificate thumbprint generated in the previous step, and the UUID that you have just generated:

      ```
      netsh
      netsh> http
      netsh http> add sslcert ipport=0.0.0.0:8759 certhash=770F5...FFBF2 appid={0352c...4c77}
      SSL Certificate successfully added
      ```

5. Change the RCS configuration (in the `ConnectorServerService.exe.Config` file) to use HTTPS and not HTTP.

    - Change `baseAddress="http..."` to `baseAddress="https..."` :

    ```
    <host>
      <baseAddresses>
        ...
        <add baseAddress="https://0.0.0.0:8759/openicf"/>
      </baseAddresses>
    </host>
    ```

    - Change `httpTransport` to `httpsTransport` :

    ```
    <httpsTransport authenticationScheme="Basic" realm="OpenICF">
        <webSocketSettings transportUsage="Always" createNotificationOnConnection="true" .../>
    </httpsTransport>
    ```

6. Export the certificate:

    1. Launch the certificate management MMC (`certlm.msc`).

    2. Right-click the `dotnet` certificate, and select **All Tasks > Export to launch the Certificate Export Wizard**.

    3. Select **Next > No, do not export the private key > DER encoded binary X.509 (.CER) > Next**.

    4. Save the file in an accessible location (for example, `C:\Users\Administrator\Desktop\dotnet.cer`), and click **Finish**.

7. Import the certificate into the IDM truststore:

    - Transfer the certificate from the Windows machine to the machine that’s running IDM.

    - Change to the `openidm/security` directory and use the Java `keytool` command to import the certificate:

```
cd /path/to/openidm/security
keytool -import -alias dotnet -file ~/Downloads/dotnet.cer -keystore ./truststore
Enter keystore password: changeit
Owner: CN=dotnet
Issuer: CN=dotnet
Serial number: 1e3af7baed05ce834da5cd1bf1241835
Valid from: Tue Aug 08 15:58:32 SAST 2017 until: Wed Aug 08 16:18:32 SAST 2018
Certificate fingerprints:
MD5:  D1:B7:B7:46:C2:59:1A:3C:94:AA:65:99:B4:43:3B:E8
SHA1: 77:0F:53:1F:14:AF:43:5E:96:3E:14:AD:82:B7:0A:47:A4:BF:FB:F2
SHA256: C0:52:E2:E5:E5:72:9D:69:F8:11:4C:B8:4C:E4:E3:1C:
19:95:86:19:70:E5:31:FA:D8:81:4B:F2:AC:30:9C:73
Signature algorithm name: SHA256withRSA
Version: 3

...

Trust this certificate? [no]: yes
Certificate was added to keystore
```

  - When you Configure a remote connector server (RCS), remember to set `"usessl": true`.

Log tracing

- By default, the RCS outputs log messages to a file named `connectorserver.log` , in the `\path\to\openicf` directory. To change the location of the log file, set the `initializeData` parameter in the configuration file. The following example sets the log directory to `C:\openicf\logs\connectorserver.log`:

```
<add name="file" type="System.Diagnostics.TextWriterTraceListener" initializeData="C:
\openicf\logs\connectorserver.log" traceOutputOptions="DateTime">
    <filter type="System.Diagnostics.EventTypeFilter" initializeData="Information"/>
</add>
```

- Check the trace settings under `system.diagnostics` in the RCS configuration file:

```
<system.diagnostics>
  <trace autoflush="true" indentsize="4">
    <listeners>
      <remove name="Default" />
      <add name="console" />
      <add name="file" />
    </listeners>
  </trace>
  <sources>
    <source name="ConnectorServer" switchName="switch1">
      <listeners>
        <remove name="Default" />
        <add name="file" />
      </listeners>
    </source>
  </sources>
  <switches>
    <add name="switch1" value="Information" />
  </switches>
  <sharedListeners>
    <add name="console" type="System.Diagnostics.ConsoleTraceListener" />
    <add name="file" type="System.Diagnostics.TextWriterTraceListener"
         initializeData="logs\ConnectorServerService.log"
         traceOutputOptions="DateTime">
      <filter type="System.Diagnostics.EventTypeFilter" initializeData="Information" />
    </add>
  </sharedListeners>
</system.diagnostics>
```

The RCS uses the standard .NET trace mechanism. For more information about tracing options, refer to Microsoft's .NET documentation for `System.Diagnostics`.

The default trace settings are a good starting point. For less tracing, set the EventTypeFilter's `initializeData` to `Warning` or `Error`. For very verbose logging, set the value to `Verbose` or `All`. The logging level has a direct effect on the RCS performance, so take care when setting this level.

Running the server

Start the .NET RCS in one of the following ways:

- Start the server as a Windows service, by using the Microsoft Services Console.

  Locate the RCS service (`OpenICF connector server`), and click **Start the service** or **Restart the service**.

  The service runs with the credentials of the "run as" user (`System`, by default).

- Start the server as a Windows service, by using the command line.

  In the Windows Command Prompt, run the following command:

  ```
  net start ConnectorServerService
  ```

  To stop the service, run the following command:

```
net stop ConnectorServerService
```

- Start the server without using Windows services.

  In the Windows Command Prompt, change to the RCS installation directory. The default location is `c:\> cd "c:\Program Files (x86)\ForgeRock\OpenICF"`.

  Start the server with the following command:

```
ConnectorServerService.exe /run
```

> **Note**
>
> This command starts the RCS with the credentials of the current user. It does not start the server as a Windows service.

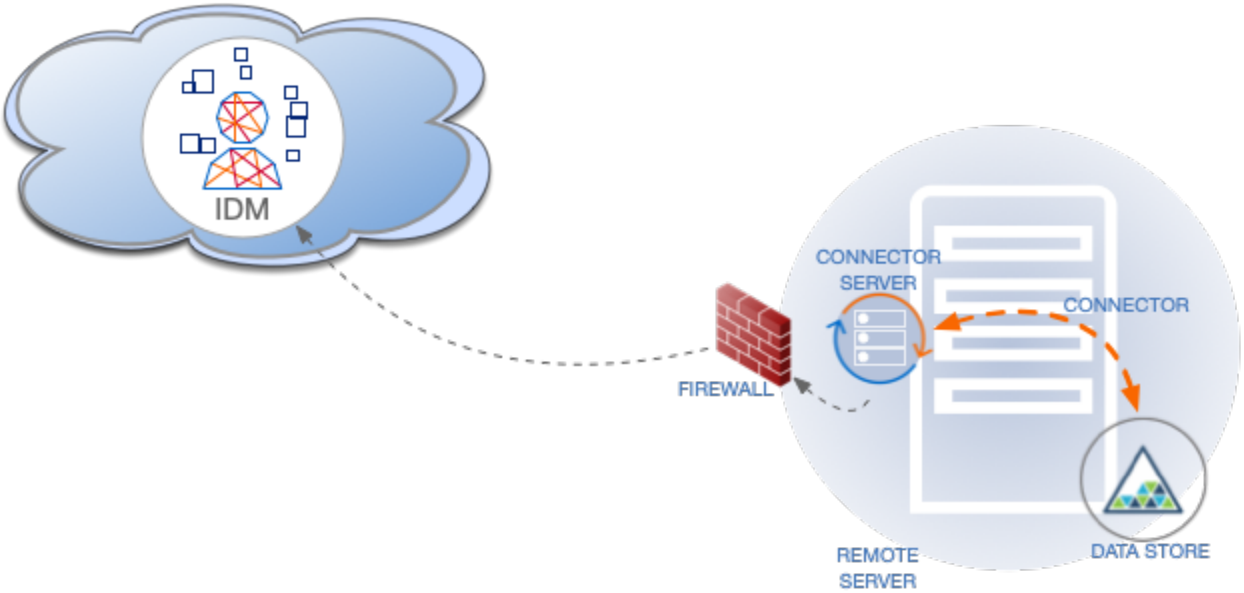
Configure a remote connector server (RCS)

RCS runs in one of two modes:

Client mode

In client mode, RCS initiates the connection with IDM. Run the RCS in client mode if you need to communicate with a system that is behind a firewall and IDM is outside that firewall (such as Advanced Identity Cloud).

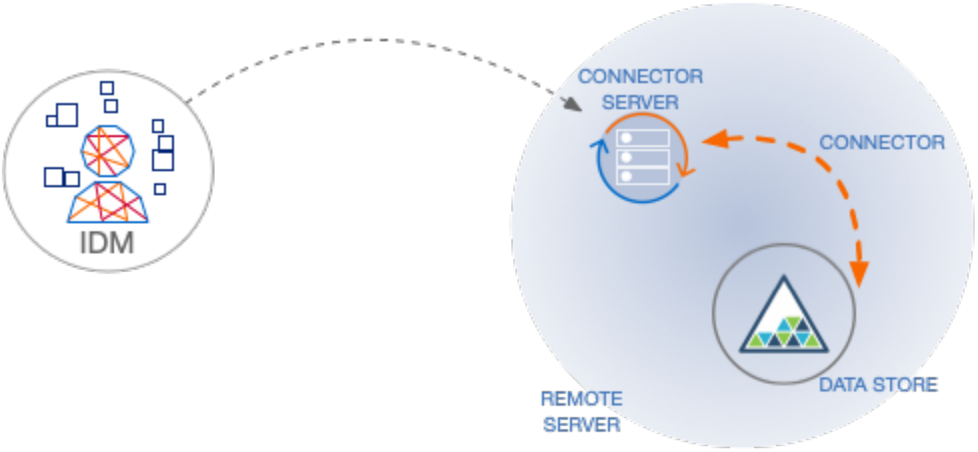
The following diagram shows an RCS in client mode:



Server mode

In server mode, RCS acts as the server, with IDM acting as a client. IDM initiates the connection with the RCS. Run the RCS in server mode if IDM can initiate the connection.

The following diagram shows an RCS in server mode:



This example shows how to retrieve the RCS types over REST:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system?_action=availableConnectorServers"
{
  "connectorServers": [
    {
      "displayName": "Remote Connector Server",
      "systemType": "provisioner.openicf",
      "type": "remoteConnectorServer"
    },
    {
      "displayName": "Remote Connector Servers Group",
      "systemType": "provisioner.openicf",
      "type": "remoteConnectorServersGroup"
    },
    {
      "displayName": "Remote Connector Server in Client mode",
      "systemType": "provisioner.openicf",
      "type": "remoteConnectorClient"
    },
    {
      "displayName": "Remote Connector Servers Group in Client mode",
      "systemType": "provisioner.openicf",
      "type": "remoteConnectorClientsGroup"
    }
  ]
}
```

RCS configuration properties

The following table displays the complete list of RCS configuration properties with truncated property names for readability. The full name for each property is prefixed with `connectorserver.` in the `conf/ConnectorServer.properties` configuration file included with RCS.

> ## Time interval properties
>
> The default values for the `nameInterval` and `webSocketConnections` properties are suitable for most RCS deployments. Do not adjust these property values without specific guidance from Ping.

RCS properties

Property	RCS Mode (Server or Client)	Description	Example
`connectorServerName`	Client	Name of the remote connector client. This name is used to identify the remote connector server in the list of connector reference objects. The name must be lower case alphanumeric characters (`^[a-z0-9]*$`), and must match the `name` property in the `provisioner.openicf.connectorinfoprovider.json` file on your IDM server.	rcs1
`url`	Client	The IDM or Advanced Identity Cloud server URL. To use multiple values, use the applicable delimiter: • For Java RCS *not* deployed in a Docker container, separate each value with a space. • For Java RCS deployed in a Docker container, separate each value with a comma (only supported for version 1.5.20.26 and later).	`wss://openidm.example.com:8443/openicf` [1] 1. Note the `wss` (WebSocket transport protocol) and the `openicf` endpoint.
`hostId`	Client	Unique identifier for the RCS.	`MY_UNIQUE_RCS_HOST_ID`
`proxyHost`	Client	Proxy server host.	
`proxyPort`	Client	Proxy server port number.	
`proxyPrincipal`	Client	Proxy server principal.	

Property	RCS Mode (Server or Client)	Description	Example
`proxyPassword`	Client	Proxy server password.	
`housekeepingInterval`	Client	Interval at which RCS checks WebSockets to determine if they should be closed and recycled according to the specified interval, in seconds.	20
`groupCheckInterval`	Client	Interval at which RCS checks WebSocket connection groups (group of WebSocket connections associated with the same IDM <-> RCS link) to see if they should be closed, in seconds. WebSocket connection groups are closed when they no longer contain any active WebSocket connections.	60
`webSocketConnections`	Client	Number of WebSocket connections to open.	2
`connectionTtl`	Client	Time to live of a WebSocket connection, in seconds.	300
`newConnectionsInterval`	Client	Interval at which RCS establishes new WebSocket connections, in seconds. Ensures that connection establishment is staggered.	10
`tokenEndpoint`	Client	Token endpoint from which to retrieve the access token if you are using OAuth2 to authenticate against AM.	`https://am.example.com/am/oauth2/realms/root/access_token`
`scope`	Client	OAuth2 token scope, if you are using OAuth2 to authenticate against AM.	fr:idm:*
`clientId`	Client	OAuth2 Client ID for which to request an access token.	connectorServer
Important If the RCS is authenticating against AM, you must update your IDM authentication configuration (in `conf/authentication.json`). Add a user mapping for this client ID in the `rsFilter` authentication module configuration. For more information, refer to Authenticate through AM.			
`clientSecret`	Client	OAuth2 Client Secret.	openidm

Property	RCS Mode (Server or Client)	Description	Example
`pingPongInterval`	Both	Interval at which RCS sends ping/pong messages between IDM <-> RCS, in seconds. Used to determine health/connectivity of the underlying WebSocket connection. The purpose of the *ping* is to keep connections alive (for firewalls or load balancers that honor connections in use). If your firewall or load balancer doesn't honor connections in use (that is, connections are timed out, regardless of their usage), the ping has no effect, and you should disable it. Set this property to `0` to disable the ping.	60
`trustStoreFile`	Both	The IDM truststore file. You do not need to set this property if the IDM certificate is a CA-signed certificate.	`security/truststore.pkcs12`
`trustStoreType`	Both	The IDM truststore type. You do not need to set this property if the IDM certificate is a CA-signed certificate.	`PKCS12`
`trustStorePass`	Both	The IDM truststore password. You do not need to set this property if the IDM certificate is a CA-signed certificate.	changeit
`keyStoreFile`	Both	The IDM keystore file. You do not need to set this property if the IDM certificate is a CA-signed certificate.	`security/keyStore.pkcs12`
`keyStoreType`	Both	The IDM keystore type. You do not need to set this property if the IDM certificate is a CA-signed certificate.	`PKCS12`
`keyStorePass`	Both	The IDM keystore password. You do not need to set this property if the IDM certificate is a CA-signed certificate.	changeit
`keyPass`	Both	The IDM certificate password. You do not need to set this property if the IDM certificate is a CA-signed certificate.	changeit
`libDir`	Both	Directory on the RCS host in which connector library file dependencies are located (relative to `/path/to/openicf/`).	`lib`

Property	RCS Mode (Server or Client)	Description	Example
`bundleDir`	Both	Directory on the RCS host in which connector .jar files are located (relative to `/path/to/openicf/`).	`connectors`
`loggerClass`	Both	The RCS logger class.	`org.forgerock.openicf.common.logging.slf4j.SLF4JLog`
`loggingConfigFile`	Both	The path to the RCS logging configuration file. The path can be absolute or relative to `/path/to/openicf/`. The default value is `conf/logback.xml`.	`/path/to/dir/filename.xml`
`principal`	Both	Principal to authenticate to the RCS. This property is not used if the RCS obtains its access token through PingAM (AM) (which is the case when using Advanced Identity Cloud).	anonymous
`password`	Both	Password to authenticate to the RCS. This property is not used if the RCS obtains its access token through AM (which is the case when using Advanced Identity Cloud.	changeit
`usessl`	Server	Whether the connection between IDM and the RCS should be over SSL.	false/true
`port`	Server	Port on which the RCS listens for the connection from IDM.	8759

Certain configuration properties are dependent on the RCS mode. For more information, refer to Configure a Remote Connector Server (RCS).

```
connectorserver.url=wss://my-tenant.forgeblocks.com:8443/openicf
connectorserver.connectorServerName=myConnectorServer
connectorserver.hostId=MY_UNIQUE_RCS_HOST_ID
connectorserver.pingPongInterval=60
connectorserver.housekeepingInterval=20
connectorserver.groupCheckInterval=60
connectorserver.webSocketConnections=2
connectorserver.connectionTtl=300
connectorserver.newConnectionsInterval=10
connectorserver.tokenEndpoint=https://my-tenant.forgeblocks.com/am/oauth2/realms/root/realms/alpha/
access_token
connectorserver.clientId=my-client-id
connectorserver.clientSecret=my-client-secret
connectorserver.trustStoreFile=security/truststore.pkcs12
connectorserver.trustStoreType=PKCS12
connectorserver.trustStorePass=changeit
connectorserver.keyStoreFile=security/keyStore.pkcs12
connectorserver.keyStoreType=PKCS12
connectorserver.keyStorePass=changeit
connectorserver.keyPass=changeit
connectorserver.scope=fr:idm:*
connectorserver.bundleDir=connectors
connectorserver.libDir=lib
connectorserver.loggerClass=org.forgerock.openicf.common.logging.slf4j.SLF4JLog
connectorserver.loggingConfigFile=conf/logback.xml
```

```
connectorserver.port=8759
connectorserver.pingPongInterval=60
connectorserver.principal=anonymous
connectorserver.password=changeit
connectorserver.usessl=true
connectorserver.trustStoreFile=security/truststore.pkcs12
connectorserver.trustStoreType=PKCS12
connectorserver.trustStorePass=changeit
connectorserver.keyStoreFile=security/keyStore.pkcs12
connectorserver.keyStoreType=PKCS12
connectorserver.keyStorePass=changeit
connectorserver.keyPass=changeit
connectorserver.bundleDir=connectors
connectorserver.libDir=lib
connectorserver.key=lmA6bMfENJGlIDbfrVtklXFK32s\=
connectorserver.loggerClass=org.forgerock.openicf.common.logging.slf4j.SLF4JLog
connectorserver.loggingConfigFile=conf/logback.xml
```

Mask clear text passwords in RCS

When you configure the RCS, you specify the settings for your server in the `ConnectorServer.properties` file. By default, you enter clear text passwords for multiple settings, such as `connectorserver.clientSecret` and `connectorserver.password`. Depending on your requirements, you could want to mask specific details for security reasons. You can do this using `OPENICF_OPTS` environment variables.

Use `OPENICF_OPTS` environment variables

If you don't want to save clear text information in the `ConnectorServer.properties` file, you can specify potentially sensitive settings at runtime through the `OPENICF_OPTS` environment variables.

For example, to set `connectorserver.clientSecret` and `connectorserver.password`, you can run the following command before starting the RCS:

```
cd /path/to/openicf/bin
export OPENICF_OPTS="-Dconnectorserver.clientSecret=Passw0rd! -Dconnectorserver.password=Passw0rd!"
./ConnectorServer.sh /start
```

You can use `OPENICF_OPTS` environment variables for as many settings as you require.

ICFServlet configuration options

You can configure the following optional ICFServlet settings in your `conf/provisioner.openicf.connectorinfoprovider.json` file:

`maxMessageSize`

Integer.

You can set a maximum message size in kilobytes. The default is 20MB.

`idleTimeout`

Integer.

The maximum time, in minutes, that a WebSocket connection can be idle before it's removed. The default is 15 minutes.

Example `provisioner.openicf.connectorinfoprovider.json`:

```
{
  "_id": "provisioner.openicf.connectorinfoprovider",
  "connectorsLocation": "connectors",
  "ICFServlet": {
    "maxMessageSize": 40960,
    "idleTimeout": 23
  },
  ...
}
```

Configure RCS in client mode

The RCS configuration will differ between server mode and client mode. Refer to RCS Properties for a list of properties and the mode to which they apply.

To generate the core configuration, use the `createConnectorServerCoreConfig` action on the `system` endpoint. Include at least the RCS `type` (`remoteConnectorClient`) and the `systemType` in the JSON payload. The `systemType` is always `provisioner.openicf` , regardless of the RCS type:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request POST \
--data '{
  "type": "remoteConnectorClient",
  "systemType": "provisioner.openicf"
}' \
"http://localhost:8080/openidm/system?_action=createConnectorServerCoreConfig"
{
  "displayName": "",
  "name": "",
  "enabled": true,
  "usessl": false
}
```

IDM returns the basic configuration properties for an RCS in client mode. The configuration that is returned is not functional. It does not contain the required configuration property values, such as the name of the RCS.

Use the output returned by the previous example to create your complete RCS configuration. Specify at least the `name` of the RCS, and use a PUT request on the `config` endpoint. Note that this step creates an RCS configuration on IDM. The values of these properties must match the RCS configuration, specified in the `ConnectorServer.properties` file on the RCS:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request PUT \
--data '{
  "_id": "provisioner.openicf.connectorinfoprovider",
  "connectorsLocation": "connectors",
  "enabled": true,
  "remoteConnectorClients": [
    {
      "displayName": "On premise 1",
      "name": "onprem",
      "enabled": true
    }
  ]
}' \
"http://localhost:8080/openidm/config/provisioner.openicf.connectorinfoprovider"
{
  "_id": "provisioner.openicf.connectorinfoprovider",
  "connectorsLocation": "connectors",
  "enabled": true,
  "remoteConnectorClients": [
    {
      "displayName": "On premise 1",
      "name": "onprem",
      "enabled": true
    }
  ]
}
```

Configure RCS in server mode

> **Note**
>
> Server mode is not compatible with PingOne Advanced Identity Cloud. If you are using Advanced Identity Cloud, configure RCS in client mode instead.

The RCS configuration will differ between server mode and client mode. Refer to RCS Properties for a list of properties and the mode to which they apply.

To generate the core configuration, use the `createConnectorServerCoreConfig` action on the `system` endpoint. Include at least the RCS `type` (`remoteConnectorServer`) and the `systemType` in the JSON payload. The `systemType` is always `provisioner.openicf`, regardless of the RCS type:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request POST \
--data '{
  "type": "remoteConnectorServer",
  "systemType": "provisioner.openicf"
}' \
"http://localhost:8080/openidm/system?_action=createConnectorServerCoreConfig"
{
  "displayName": "",
  "proxyPassword": null,
  "proxyHost": null,
  "enabled": true,
  "usessl": false,
  "proxyPort": 8080,
  "port": "",
  "name": "",
  "host": "",
  "proxyUser": null,
  "housekeepingInterval": 20,
  "connectionGroupCheckInterval": 60,
  "pingPongInterval": 60,
  "key": "password",
  "webSocketConnections": 2
}
```

IDM returns the required configuration properties for an RCS in server mode. The configuration that is returned is not functional. It does not contain the specific property values, such as the host name and port of the RCS.

Use the output returned by the previous example to create your complete RCS configuration. Specify at least the `host` and `port` of the RCS, and use a PUT request on the `config` endpoint. Note that this step creates an RCS configuration on IDM. The values of these properties must match the RCS configuration, specified in the `ConnectorServer.properties` file on the RCS:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request PUT \
--data '{
  "_id": "provisioner.openicf.connectorinfoprovider",
  "connectorsLocation": "connectors",
  "enabled": true,
  "remoteConnectorServers": [
    {
      "type": "remoteConnectorServer",
      "displayName": "Remote Connector Server 1",
      "proxyPassword": null,
      "proxyHost": null,
      "enabled": true,
      "usessl": false,
      "proxyPort": 8080,
      "port": 8759,
      "name": "rcs1",
      "host": "rcs.example.com",
      "proxyUser": null,
      "housekeepingInterval": 20,
      "connectionGroupCheckInterval": 60,
      "pingPongInterval": 60,
      "key": "Passw0rd",
      "webSocketConnections": 2
    }
  ]
}' \
"http://localhost:8080/openidm/config/provisioner.openicf.connectorinfoprovider"
{
  "_id": "provisioner.openicf.connectorinfoprovider",
  "connectorsLocation": "connectors",
  "enabled": true,
  "remoteConnectorServers": [
    {
      "type": "remoteConnectorServer",
      "displayName": "Remote Connector Server 1",
      "proxyPassword": null,
      "proxyHost": null,
      "enabled": true,
      "usessl": false,
      "proxyPort": 8080,
      "port": 8759,
      "name": "rcs1",
      "host": "rcs.example.com",
      "proxyUser": null,
      "housekeepingInterval": 20,
      "connectionGroupCheckInterval": 60,
      "pingPongInterval": 60,
      "key": {
        "$crypto": {
          "type": "x-simple-encryption",
          "value": {
            "cipher": "AES/CBC/PKCS5Padding",
            "stableId": "openidm-sym-default",
            "salt": "3Mq1UJuZXqANx2AzUtbFbg==",
            "data": "4WHBEI3nSVWJ2DfIs2dPZg==",
            "keySize": 16,
            "purpose": "idm.config.encryption",
            "iv": "BvFAQ4sjwJCNY2e7WZPkGw==",
            "mac": "ximBz/BlqC8SEsBTuYQX5Q=="
          }
        }
      },
      "webSocketConnections": 2
    }
  ]
}
```

Configure failover between RCS servers

For failover purposes, you can configure a *group* of RCSs, in either server or client mode. Failover is particularly important when you configure an RCS in client mode because IDM has no way of knowing whether the RCS is available.

To prevent the RCS from being a single point of failure, you can specify a list of RCS servers that the connector can target. To set up a failover configuration, you create either a `remoteConnectorServersGroup` or a `remoteConnectorClientsGroup` and list the RCS servers. The connector attempts to contact the first RCS in the list. If that RCS is down, it proceeds to the next RCS.

This example configures a `remoteConnectorClientsGroup` that lists two remote RCS servers, on hosts `remote-host-1` and `remote-host-2`. The RCS servers are listed by their `name` property. You can configure multiple groups and multiple servers per group.

First, generate the core configuration to obtain the required properties:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request POST \
--data '{
  "type" : "remoteConnectorClientsGroup",
  "systemType" : "provisioner.openicf"
}' \
"http://localhost:8080/openidm/system?_action=createConnectorServerCoreConfig"
{
   "displayName": "",
   "name": "",
   "serversList": [],
   "algorithm": "failover"
 }
```

Use the output returned by the previous example to create your RCS group configuration. Use a PUT request on the `config` endpoint:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request PUT \
--data '{
  "_id": "provisioner.openicf.connectorinfoprovider",
  "connectorsLocation": "connectors",
  "enabled": true,
  "remoteConnectorClients": [
    {
      "type": "remoteConnectorClientsGroup",
      "displayName": ".NET Failover Group",
      "name" : "dotnet-ha",
      "algorithm" : "failover",
      "serversList" : [
        {"name": "remote-host-1"},
        {"name": "remote-host-2"}
      ]
    }
  ]
}' \
"http://localhost:8080/openidm/config/provisioner.openicf.connectorinfoprovider"
{
  "_id": "provisioner.openicf.connectorinfoprovider",
  "connectorsLocation": "connectors",
  "enabled": true,
  "remoteConnectorClients": [
    {
      "type": "remoteConnectorClientsGroup",
      "displayName": ".NET Failover Group",
      "name": "dotnet-ha",
      "algorithm": "failover",
      "serversList": [
        {
          "name": "remote-host-1"
        },
        {
          "name": "remote-host-2"
        }
      ]
    }
  ]
}
```

The `algorithm` can be either `failover` or `roundrobin`. If the algorithm is `failover`, requests are always sent to the first RCS in the list, unless it is unavailable; in which case, requests are sent to the next RCS in the list. If the algorithm is `roundrobin`, requests are distributed equally between the RCS servers in the list, in the order in which they are received.

Your connector configuration (`provisioner.openicf-connectorName.json`) references the RCS group, rather than a single RCS. For example, the following excerpt of a PowerShell connector configuration file references the `dotnet-ha` RCS group created in the previous example:

```
{
   "connectorRef" : {
     "bundleName" : "MsPowerShell.Connector",
     "connectorName" : "Org.ForgeRock.OpenICF.Connectors.MsPowerShell.MsPowerShellConnector",
     "connectorHostRef" : "dotnet-ha",
     "bundleVersion" : "[1.4.3.0,1.5.0.0)"
   },
   ...
 }
```

This example configures a `remoteConnectorServersGroup` that lists two remote RCS servers, on hosts `remote-host-1` and `remote-host-2` . The RCS servers are listed by their `name` property. You can configure multiple groups and multiple servers per group.

First, generate the core configuration to obtain the required properties:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request POST \
--data '{
  "type" : "remoteConnectorServersGroup",
  "systemType" : "provisioner.openicf"
}' \
"http://localhost:8080/openidm/system?_action=createConnectorServerCoreConfig"
{
   "displayName": "",
   "name": "",
   "serversList": [],
   "algorithm": "failover"
 }
```

Use the output returned by the previous example to create your RCS group configuration. Use a PUT request on the `config` endpoint:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request PUT \
--data '{
  "_id": "provisioner.openicf.connectorinfoprovider",
  "connectorsLocation": "connectors",
  "enabled": true,
  "remoteConnectorServers": [
    {
      "type": "remoteConnectorServersGroup",
      "displayName": ".NET Failover Group",
      "name" : "dotnet-ha",
      "algorithm" : "failover",
      "serversList" : [
        {"name": "remote-host-1"},
        {"name": "remote-host-2"}
      ]
    }
  ]
}' \
"http://localhost:8080/openidm/config/provisioner.openicf.connectorinfoprovider"
{
  "_id": "provisioner.openicf.connectorinfoprovider",
  "connectorsLocation": "connectors",
  "enabled": true,
  "remoteConnectorServers": [
    {
      "type": "remoteConnectorServersGroup",
      "displayName": ".NET Failover Group",
      "name": "dotnet-ha",
      "algorithm": "failover",
      "serversList": [
        {
          "name": "remote-host-1"
        },
        {
          "name": "remote-host-2"
        }
      ]
    }
  ]
}
```

The `algorithm` can be either `failover` or `roundrobin`. If the algorithm is `failover`, requests are always sent to the first RCS in the list, unless it is unavailable; in which case, requests are sent to the next RCS in the list. If the algorithm is `roundrobin`, requests are distributed equally between the RCS servers in the list, in the order in which they are received.

Your connector configuration (`provisioner.openicf-connectorName.json`) references the RCS group, rather than a single RCS. For example, the following excerpt of a PowerShell connector configuration file references the `dotnet-ha` RCS group created in the previous example:

```
{
   "connectorRef" : {
     "bundleName" : "MsPowerShell.Connector",
     "connectorName" : "Org.ForgeRock.OpenICF.Connectors.MsPowerShell.MsPowerShellConnector",
     "connectorHostRef" : "dotnet-ha",
     "bundleVersion" : "[1.4.3.0,1.5.0.0)"
   },
   ...
 }
```

Secure the connection to the RCS with SSL

> **Note**
>
> The following section does not apply to Advanced Identity Cloud, as it requires filesystem access to your installation.

The SSL configuration for an RCS depends on whether you are running the RCS in server mode or in client mode:

- In **server mode**, IDM initiates the connection to the RCS.

  The RCS needs a public/private key pair and a certificate (either self-signed or CA-signed). The RCS sends its certificate to the client (IDM) during the SSL handshake.

  If you are using a CA-signed certificate, IDM will trace the certificate back to the root certificate. If you are using a self-signed certificate (or a certificate that depends on an unreachable issuer in the chain from the root certificate), you must import the certificate into the IDM truststore.

- In **client mode**, the RCS initiates the connection to IDM. IDM sends its certificate during the SSL handshake. If you are using the IDM self-signed certificate, you must import the certificate into the RCS truststore.

  If you are using TLS Mutual Authentication, the RCS needs a public/private key pair and a certificate. IDM requests the certificate from the RCS during the SSL handshake.

Configure the RCS for SSL

On the RCS, edit the `conf/ConnectorServer.properties` file to specify a secure connection between IDM and the RCS:

RCS in server mode

- Set `connectorserver.usessl=true`.
- Specify the RCS keystore and truststore. For example:

  ```
  connectorserver.trustStoreFile=security/truststore.pkcs12
  connectorserver.trustStoreType=PKCS12
  connectorserver.trustStorePass=changeit
  connectorserver.keyStoreFile=security/keyStore.pkcs12
  connectorserver.keyStoreType=PKCS12
  connectorserver.keyStorePass=changeit
  connectorserver.keyPass=changeit
  ```

RCS in client mode

- Connection security is determined by the value of the `connectorserver.url` property. Use the `wss` protocol to establish a WebSocket over an encrypted TLS connection; for example, `wss://my-tenant.forgeblocks.com/openicf`.

  The `connectorserver.usessl` property is not used in client mode.

- Specify the RCS keystore and truststore. For example:

```
connectorserver.trustStoreFile=security/truststore.pkcs12
connectorserver.trustStoreType=PKCS12
connectorserver.trustStorePass=changeit
connectorserver.keyStoreFile=security/keyStore.pkcs12
connectorserver.keyStoreType=PKCS12
connectorserver.keyStorePass=changeit
connectorserver.keyPass=changeit
```

Configure IDM for SSL

In your `conf/provisioner.openicf.connectorinfoprovider.json` file, set `"usessl" : true`.

Generate keys for an RCS in server mode

1. Generate the RCS private/public key pair and create a new PKCS12 keystore:

```
keytool \
-genkeypair \
-keyalg EC \
-alias icf-rcs \
-dname "CN=icf.example.com,O=Example Corp,C=FR" \
-keystore rcsKeystore \
-storetype PKCS12 \
-storepass changeit \
```

2. Verify the contents of the new keystore:

```
keytool \
-list \
-v \
-keystore rcsKeystore
Enter keystore password:  changeit
Keystore type: PKCS12
Keystore provider: SUN

Your keystore contains 1 entry

Alias name: icf-rcs
Creation date: Jul 13, 2020
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=icf.example.com, O=Example Corp, C=FR
Issuer: CN=icf.example.com, O=Example Corp, C=FR
Serial number: 611e093d
Valid from: Mon Jul 13 23:58:49 SAST 2020 until: Sun Oct 11 23:58:49 SAST 2020
Certificate fingerprints:
SHA1: Fingerprint
SHA256: Fingerprint
Signature algorithm name: SHA256withECDSA
Subject Public Key Algorithm: 256-bit EC key
...
```

3. Export the RCS certificate:

```
keytool \
-export \
-alias icf-rcs \
-file rcs.cert \
-keystore rcsKeystore.pkcs12
Enter keystore password: changeit
Certificate stored in file <rcs.cert>
```

4. If you are not using a self-signed certificate, have the certificate signed by a Certificate Authority (CA):

    1. Create a Certificate Signing Request (CSR):

    ```
    keytool \
    -keystore rcsKeystore.pkcs12 \
    -certreq \
    -alias icf-rcs \
    -file rcs.csr
    ```

```
more rcs.csr
-----BEGIN NEW CERTIFICATE REQUEST-----

MIIEKTCCA9QCAQAwVzELMAkGA1UEBhMCRlIxCzAJBgNVBAgTAkZSMQswCQYDVQQH
xZ47rzcY6OrElh8+/TYG50NRqcQYMzm4CefCrhxTm6dHW4XQEa24tHmHdUmEaVys
A1UdDgQWBBSivxV9AzgbrIo3gG6vCBlNaXf3wjANBglghkgBZQMEAwIFAANAADA9
...
AhxL791/ikf1hqxOD3uttV7qumg+TNednsgtk6uOAh0AlINk+1LBeyUkQA7iUHy/
3KLYWog/Npu5USdCeA==

-----END NEW CERTIFICATE REQUEST-----
```

2. Submit the CSR to your CA for signature.

5. Import the signed certificate into the RCS keystore:

```
keytool \
-importcert \
-trustcacerts \
-file rcs.cert \
-keystore rcsKeystore.pkcs12 \
-storetype pkcs12 \
-alias icf-rcs
Enter keystore password: changeit
Certificate reply was installed in keystore
```

> **Note**
>
> If your CA certificate is not trusted, you might need to import the CA certificate into the keystore too.

6. Import the RCS certificate into the IDM truststore:

```
keytool \
-import \
-alias icf-rcs \
-keystore /path/to/openidm/truststore \
-file rcs.cert
Enter keystore password: changeit
Owner: CN=icf.example.com, O=Example Corp, C=FR
Issuer: CN=icf.example.com, O=Example Corp, C=FR
Serial number: 611e093d
Valid from: Fri Apr 05 16:04:04 CEST 2019 until: Mon Aug 17 16:04:04 CEST 2020
Certificate fingerprints:
MD5:  Fingerprint
SHA1: Fingerprint
SHA256: Fingerprint
Signature algorithm name: SHA256withRSA
Subject Public Key Algorithm: 2048-bit DSA key
Version: 1
Trust this certificate? [no]:  yes
Certificate was added to keystore
```

Generate keys for an RCS in client mode

1. Generate the RCS private/public key pair and create a new PKCS12 keystore:

```
keytool \
-genkeypair \
-keyalg EC \
-alias icf-rcs \
-dname "CN=icf.example.com,O=Example Corp,C=FR" \
-keystore rcsKeystore \
-storetype PKCS12 \
-storepass changeit \
```

2. Verify the contents of the new keystore:

```
keytool \
-list \
-v \
-keystore rcsKeystore
Enter keystore password:  changeit
Keystore type: PKCS12
Keystore provider: SUN

Your keystore contains 1 entry

Alias name: icf-rcs
Creation date: Jul 13, 2020
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=icf.example.com, O=Example Corp, C=FR
Issuer: CN=icf.example.com, O=Example Corp, C=FR
Serial number: 611e093d
Valid from: Mon Jul 13 23:58:49 SAST 2020 until: Sun Oct 11 23:58:49 SAST 2020
Certificate fingerprints:
SHA1: Fingerprint
SHA256: Fingerprint
Signature algorithm name: SHA256withECDSA
Subject Public Key Algorithm: 256-bit EC key
...
```

3. Export the RCS certificate:

```
keytool \
-export \
-alias icf-rcs \
-file rcs.cert \
-keystore rcsKeystore.pkcs12
Enter keystore password: changeit
Certificate stored in file <rcs.cert>
```

4. If you are not using a self-signed certificate, have the certificate signed by a Certificate Authority (CA):

    1. Create a Certificate Signing Request (CSR):

    ```
    keytool \
    -keystore rcsKeystore.pkcs12 \
    -certreq \
    -alias icf-rcs \
    -file rcs.csr
    ```

    ```
    more rcs.csr
    -----BEGIN NEW CERTIFICATE REQUEST-----

    MIIEKTCCA9QCAQAwVzELMAkGA1UEBhMCRlIxCzAJBgNVBAgTAkZSMQswCQYDVQQH
    xZ47rzcY6OrElh8+/TYG50NRqcQYMzm4CefCrhxTm6dHW4XQEa24tHmHdUmEaVys
    A1UdDgQWBBSivxV9AzgbrIo3gG6vCBlNaXf3wjANBglghkgBZQMEAwIFAANAADA9
    ...
    AhxL791/ikf1hqxOD3uttV7qumg+TNednsgtk6uOAh0AlINk+1LBeyUkQA7iUHy/
    3KLYWog/Npu5USdCeA==

    -----END NEW CERTIFICATE REQUEST-----
    ```

    2. Submit the CSR to your CA for signature.

5. Import the signed certificate into the RCS keystore:

    ```
    keytool \
    -importcert \
    -trustcacerts \
    -file rcs.cert \
    -keystore rcsKeystore.pkcs12 \
    -storetype pkcs12 \
    -alias icf-rcs
    Enter keystore password: changeit
    Certificate reply was installed in keystore
    ```

    > **Note**
    >
    > If your CA certificate is not trusted, you might need to import the CA certificate into the keystore too.

6. Import the RCS certificate into the IDM truststore:

```
keytool \
-import \
-alias icf-rcs \
-keystore /path/to/openidm/truststore \
-file rcs.cert
Enter keystore password: changeit
Owner: CN=icf.example.com, O=Example Corp, C=FR
Issuer: CN=icf.example.com, O=Example Corp, C=FR
Serial number: 611e093d
Valid from: Fri Apr 05 16:04:04 CEST 2019 until: Mon Aug 17 16:04:04 CEST 2020
Certificate fingerprints:
MD5:  Fingerprint
SHA1: Fingerprint
SHA256: Fingerprint
Signature algorithm name: SHA256withRSA
Subject Public Key Algorithm: 2048-bit DSA key
Version: 1
Trust this certificate? [no]:  yes
Certificate was added to keystore
```

7. Export the IDM self-signed certificate:

```
keytool \
-export \
-alias openidm-localhost \
-keystore keystore.jceks \
-storetype jceks \
-file idm.cert \
Enter keystore password: changeit
Certificate stored in file <idm.cert>
```

8. Import the IDM self-signed certificate into the RCS truststore:

```
keytool \
-import \
-alias openidm-localhost \
-keystore /path/to/rcs/security/truststore.pkcs12 \
-storetype pkcs12 \
-file idm.cert
Enter keystore password: changeit

Owner: CN=openidm-localhost, O=OpenIDM Self-Signed Certificate, OU=None, L=None, ST=None, C=None
Issuer: CN=openidm-localhost, O=OpenIDM Self-Signed Certificate, OU=None, L=None, ST=None, C=None
Serial number: 16981c79d8d
Valid from: Wed Feb 13 15:35:36 CET 2019 until: Thu Mar 15 15:35:36 CET 2029
Certificate fingerprints:
MD5:  fingerprint
SHA1: fingerprint
SHA256: fingerprint
Signature algorithm name: SHA512withRSA
Subject Public Key Algorithm: 2048-bit RSA key
Version: 3
Trust this certificate? [no]:  yes

Certificate was added to keystore
```

Example connector using RCS

Use the CSV connector to reconcile users in a remote CSV data store

This example shows reconciliation of users stored in a CSV file on a remote machine. The remote Java RCS lets IDM synchronize its repository with the remote CSV file.

The example assumes that a remote Java RCS is installed and running on a host named `remote-host`.

The example uses the small CSV data set provided with the *Getting Started* sample (`hr.csv`). The CSV connector runs as a *remote connector* on the host where the Java RCS is running. Before you start, copy the CSV data file from the *Getting Started* sample (`/path/to/openidm/samples/getting-started/data/hr.csv`) to an accessible location on the machine that hosts the remote Java RCS. For example:

```
cd /path/to/openidm/samples/getting-started/data/
scp hr.csv testuser@remote-host:/home/testuser/csv-sample/data/
Password:**
hr.csv     100%   651      0.6KB/s    00:00
```

Configure IDM for the remote CSV connector example

1. Copy the following files to your `/path/to/openidm/conf` directory:

   sync.json

      A customized mapping file for this example.

`/openidm/samples/example-configurations/provisioners/provisioner.openicf.connectorinfoprovider.json`

A sample RCS configuration.

`/openidm/samples/example-configurations/provisioners/provisioner.openicf-csvfile.json`

A sample connector configuration file.

2. Edit the RCS configuration file (`provisioner.openicf.connectorinfoprovider.json`) to match your network setup.

   The following example indicates that the Java RCS is running on the host `remote-host` , listening on the default port, and configured with a secret key of `Passw0rd` :

   ```
   {
       "remoteConnectorServers" : [
           {
               "name" : "csv",
               "host" : "remote-host",
               "port" : 8759,
               "usessl" : false,
               "key" : "Passw0rd"
           }
       ]
   }
   ```

   The `name` you set in this file will be referenced in the `connectorHostRef` property of the connector configuration in the next step.

   The `key` you specify here must match the password you set when you installed the Java RCS.

3. Edit the CSV connector configuration file (`provisioner.openicf-csvfile.json`) as follows:

   ```
   {
       "connectorRef" : {
           "connectorHostRef" : "csv",
           "bundleName" : "org.forgerock.openicf.connectors.csvfile-connector",
           "bundleVersion" : "[1.5.0.0,1.6.0.0)",
           "connectorName" : "org.forgerock.openicf.csvfile.CSVFileConnector"
       },
       ...
       "configurationProperties" : {
           "csvFile" : "/home/testuser/csv-sample/data/hr.csv"
       }
   }
   ```

   - The `connectorHostRef` property sets the RCS to use, and refers to the `name` property you specified in the `provisioner.openicf.connectorinfoprovider.json` file.
   - The `bundleVersion : "[1.5.0.0,1.6.0.0)",` must either be exactly the same as the version of the CSV connector that you are using or, if you specify a range, the CSV connector version must be included in this range.

   - The `csvFile` property must specify the absolute path to the CSV data file that you copied to the remote host on which the Java RCS is running.

Run the remote CSV connector example

1. Start IDM:

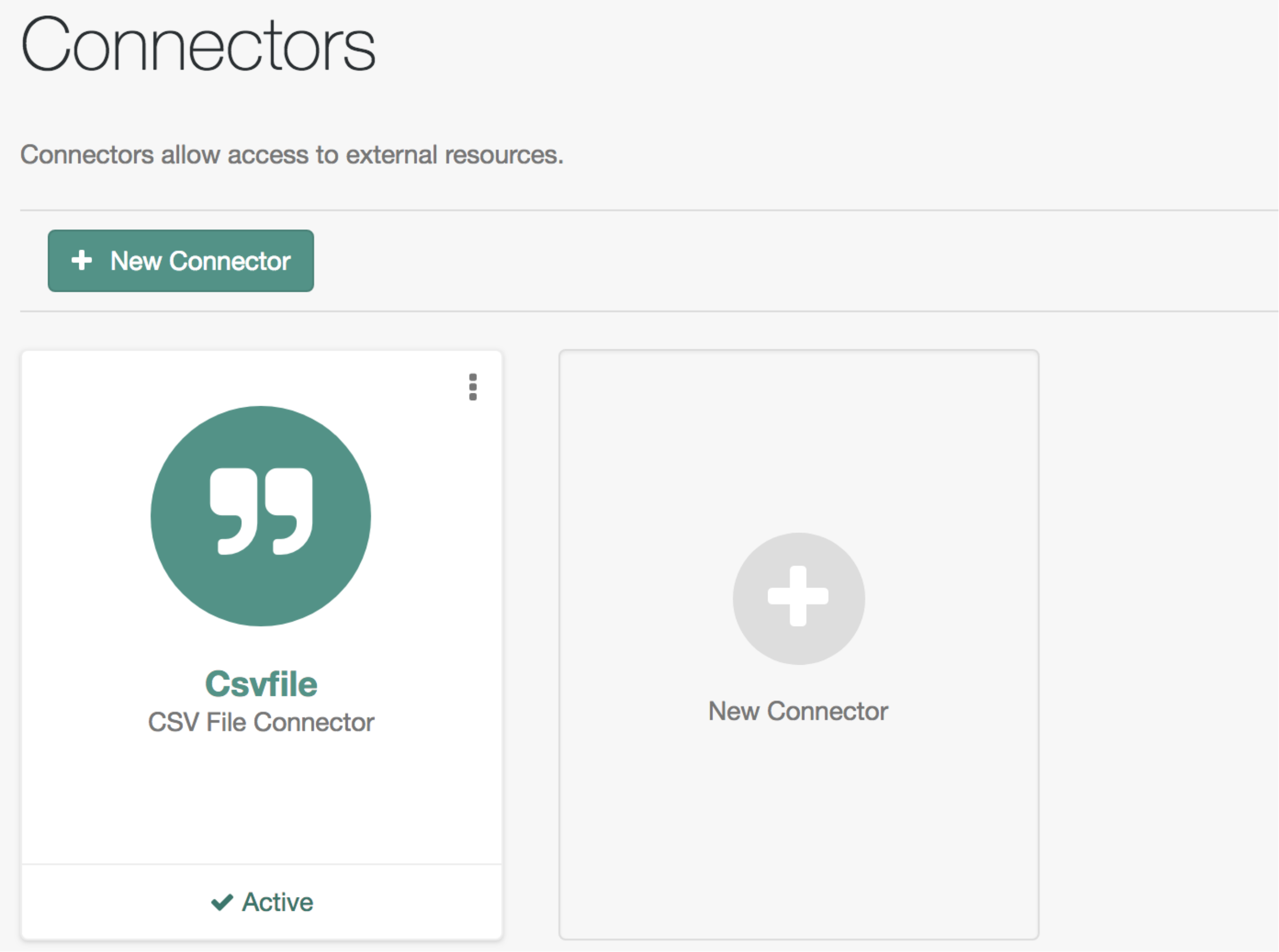
```
/path/to/openidm/startup.sh
```

2. Verify that IDM can reach the RCS, and that the CSV connector has been configured correctly:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/system?_action=test"
[
  {
    "name": "csv",
    "enabled": true,
    "config": "config/provisioner.openicf/csv",
    "objectTypes": [
      "__ALL__",
      "account"
    ],
    "connectorRef": {
      "bundleName": "org.forgerock.openicf.connectors.csvfile-connector",
      "connectorName": "org.forgerock.openicf.csvfile.CSVFileConnector",
      "bundleVersion": "[1.5.0.0,1.6.0.0)"
    },
    "displayName": "CSV File Connector",
    "ok": true
  }
]
```

The connector must return `"ok": true`.

Alternatively, use the admin UI to verify that IDM can reach the RCS and that the CSV connector is active. Log in to the admin UI (`https://localhost:8443/openidm/admin`), and select Configure > Connectors. The CSV connector should be listed on the Connectors page, and its status should be Active.



Connectors

Connectors allow access to external resources.

Figure 1. Connectors Tab Showing an Active CSV Connector

3. To test that the connector has been configured correctly, run a reconciliation operation as follows:

   1. Select Configure > Mappings, and click the systemCsvAccounts_managedUser mapping.

   2. Click Reconcile.

      If the reconciliation is successful, the three users from the remote CSV file should have been added to the managed user repository.

      To check this, click Manage > User.

Connector logs

> **Important**
>
> By default, logging is not enabled for RCS.

It can be difficult to determine if the root cause of a problem is at the ICF or connector level, or at the application level.

The ICF API sets the `LoggingProxy` at a very high level. You can consider the Logging Proxy as the *border* between the application (IDM) and the ICF framework.

Logging levels

Finer logging levels generate more noise but can be helpful when troubleshooting:

- `SEVERE` (highest value, least noise)
- `WARNING`
- `INFO`
- `CONFIG`
- `FINE`
- `FINER`
- `FINEST` (lowest value, most noise)

Enable IDM connector logging

If you are using ICF connectors bundled with IDM, you can adjust the log levels for specific parts of the system in the `path/to/openidm/conf/logging.properties` file. To start logging, enable the Logging Proxy and set the level for all or some operations:

Enable the LoggingProxy

```
org.identityconnectors.framework.impl.api.LoggingProxy.level=FINE
```

Log all operations

```
org.identityconnectors.framework.api.operations.level=FINE
```

Log specific operations

```
org.identityconnectors.framework.api.operations.CreateApiOp.level=FINE
org.identityconnectors.framework.api.operations.UpdateApiOp.level=FINE
org.identityconnectors.framework.api.operations.DeleteApiOp.level=FINE
```

You can log any of the following operations:

- `AuthenticationApiOp`
- `CreateApiOp`
- `DeleteApiOp`
- `GetApiOp`
- `ResolveUsernameApiOp`
- `SchemaApiOp`
- `ScriptOnConnectorApiOp`
- `ScriptOnResourceApiOp`
- `SearchApiOp`
- `SyncApiOp`
- `TestApiOp`
- `UpdateApiOp`
- `ValidateApiOp`

Java RCS logging

Logging configuration file

The default location for the logging configuration file is `/path/to/openicf/conf/logback.xml`. To use another location, edit the following property in your `conf/ConnectorServer.properties` configuration file:

```
connectorserver.loggingConfigFile=otherDirectory/filename.xml (1)
```

1 The path can be absolute or relative to **`/path/to/openicf/`**.

Enable Java RCS debug logging

> **Tip**
>
> For additional Java RCS debug logging information, refer to this Knowledge Base article.

To enable debug logging in the remote Java Connector Server, uncomment the following line in the logging configuration file:

```
<logger name="org.identityconnectors.framework.impl.api.LoggingProxy" level="DEBUG" additivity="false">
    <appender-ref ref="TRACE-FILE"/>
</logger>
<logger name="org.identityconnectors.framework.api.operations" level="DEBUG" additivity="false">
    <appender-ref ref="TRACE-FILE"/>
</logger>
```

Rolling log policy

To change the total size for all log files or maximum time length before a log rolls over to a new file, edit `conf/logback.xml` and update the applicable `maxHistory` and `totalSizeCap` properties. The default rolling log policy has the following configuration:

```
<maxHistory>30</maxHistory>
<totalSizeCap>1GB</totalSizeCap>
```

For more information, refer to maxHistory and totalSizeCap in the Logback documentation.

.NET RCS logging

To enable logging in the .NET RCS, edit the `ConnectorServer.exe.config` configuration file, and set the `logging.proxy` key to `true`:

```
<add key="logging.proxy" value="true"/>
```

Connector development

> *Hands-on guide to developing connectors using the OpenICF (ICF). ICF provides connectors for a consistent generic layer between applications and target resources.*

Quick Start

About ICF

Learn about the OpenICF and ICF connectors.

ICF API

Learn about the ICF API.

ICF SPI

Learn about the ICF Service Provider Interface (SPI).

Java Connectors

Write Java connectors.

Groovy Connectors

Write scripted Groovy connectors.

PowerShell Connectors

Write scripted PowerShell connectors.

?

Troubleshoot Connectors

Troubleshoot ICF and connector problems.

Name changes for ForgeRock products

Product names changed when ForgeRock became part of Ping Identity.

The following name changes have been in effect since early 2024:

Old name	New name
ForgeRock Identity Cloud	PingOne Advanced Identity Cloud
ForgeRock Access Management	PingAM
ForgeRock Directory Services	PingDS
ForgeRock Identity Management	PingIDM
ForgeRock Identity Gateway	PingGateway

Learn more about the name changes in New names for ForgeRock products in the Knowledge Base.

About ICF and ICF connectors

The OpenICF (ICF) provides interoperability between identity, compliance, and risk management solutions. An ICF connector enables provisioning software, such as PingIDM, to manage the identities that are maintained by a specific identity provider.

ICF connectors provide a consistent layer between identity applications and target resources, and expose a set of operations for the complete lifecycle of an identity. The connectors provide a way to decouple applications from the target resources to which data is provisioned.

ICF focuses on provisioning and identity management, but also provides general purpose capabilities, including authentication, create, read, update, delete, search, scripting, and synchronization operations. Connector bundles rely on the ICF Framework, but applications remain completely separate from the connector bundles. This lets you change and update connectors without changing your application or its dependencies.

Many connectors have been built within the ICF framework, and are maintained and supported by Ping and by the ICF community. However, you can also develop your own ICF connector, to address a requirement that is not covered by one of the existing connectors. In addition, ICF provides two *scripted connector toolkits*, that let you write your own connectors based on Groovy or PowerShell scripts.

The ICF framework can use IDM, Sun Identity Manager, and Oracle Waveset connectors (version 1.1), and can use ConnID connectors up to version 1.4.

This guide provides the following information:

- An overview of the ICF framework and its components
- Information on how to use the ICF existing connectors in your application (both locally and remotely)
- Information on how to write your own Java and .NET connectors, scripted Groovy connectors, or scripted PowerShell connectors

Overview of ICF functionality

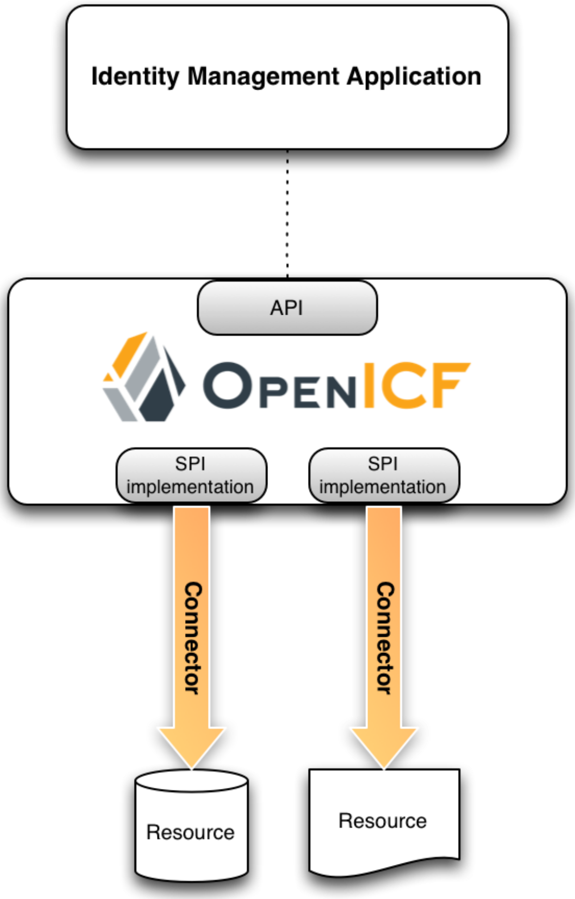
ICF provides many capabilities, including the following:

- Connector pooling
- Timeouts on all operations
- Search filtering
- Search and synchronization buffering and result streaming
- Scripting with Groovy, JavaScript, shell, and PowerShell
- Classloader isolation
- An independent logging API/SPI
- Java and .NET platform support
- Opt-in operations that support both simple and advanced implementations for the same CRUD operation
- A logging proxy that captures all API calls
- A Maven connector archetype to create connectors

ICF architecture overview

ICF is situated between the identity management application and the target resource. The framework provides a generic layer between the application and the connector bundle that accesses the resource. The framework implements an API, that includes a defined set of operations. When you are building a connector, you implement the Service Provider Interface (SPI), and include only those operations that are supported by your target resource. Each connector implements a set of SPI operations. The API operations call the SPI operations that you implement.

The following image shows a high-level overview of an ICF deployment.



Understanding the ICF framework components

When you are building, or modifying, an identity management application to use the ICF Framework and its connectors, you use the following interfaces of the API:

- Connector Info Manager Component

  The connector info manager maintains a set of connector info instances, each of which describes an available connector. The ICF Framework provides three different types of connector info manager:

  - Local

    A local connector info manager accesses the connector bundle or assembly directly.

  - Remote

    A remote connector info manager accesses the connector bundle or assembly through a remote connector server.

  - OSGi

    An OSGi connector info manager accesses the connector bundle within the OSGi context.

  For more information, refer to Connector info manager.

- Connector Info Component

The connector info component provides meta information (display name, category, messages, and so forth) for a given connector.

- Connector Key Component

  The connector key component uniquely identifies a specific connector implementation.

- API Configuration

  The API configuration holds the available configuration properties and values from both the API and the SPI, based on the connector type. For more information, refer to Configuration interface.

- Connector Facade Interface

  The connector facade is the main interface through which an application invokes connector operations. The connector facade represents a specific connector instance, that has been configured in a specific way. For more information, refer to ConnectorFacade interface.

When you are building a new connector, you implement the SPI, including the following interfaces:

- The `connector` interface.

  The connector interface handles initialization and disposal of the connector, and determines whether the connector is poolable. For more information, refer to Connector interface.

- The `configuration` interface.

  The configuration interface implementation includes all of the required information to enable the connector to connect to the target system, and to perform its operations. The configuration interface implements getters and setters for each of its defined properties. It also provides a validate method that determines whether all the required properties are available, and valid. For more information, refer to Configuration interface.

  The ICF framework uses the configuration interface implementation to build the *configuration properties* inside the API configuration.

  When the configuration interface is implemented, it becomes available to the default API configuration.

- Any `operations` that the target resource can support, such as `CreateOp`, `UpdateOp`, `DeleteOp` and so forth. For more information, refer to Operation interfaces.

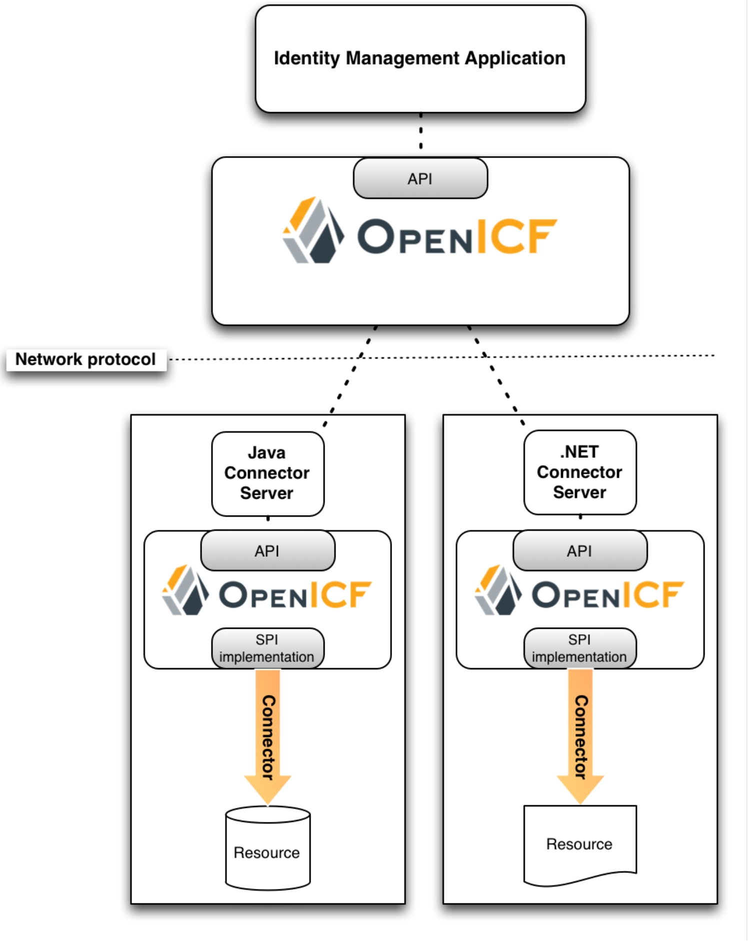
Remote connector overview

Connectors can run locally (on the same host as your application) or remotely (on a host that is remote to your application). Connectors that run remotely require a *connector server*, running on the same host as the connector. Applications access the connector implementation *through* the connector server.

> **Note**
>
> The ICF framework can support both local and remote connector implementations simultaneously.

Connector servers also let you run connector bundles that are written in C# on a .NET platform, and to access them over the network from a Java or .NET application.

The following image shows a high-level overview of an ICF deployment, including a remote connector server.



For more information about connector servers, and how to use them in your application, refer to Remote connectors.

OpenICF API

This chapter describes how to use the ICF API, which lets you call ICF connector implementations from your application. The chapter demonstrates creating a connector facade, required for applications to access connectors, and then how to call the various ICF operations from your application.

Before You Start

Before you can use an ICF connector in your application, you must download the ICF framework libraries, and the required connector bundles.

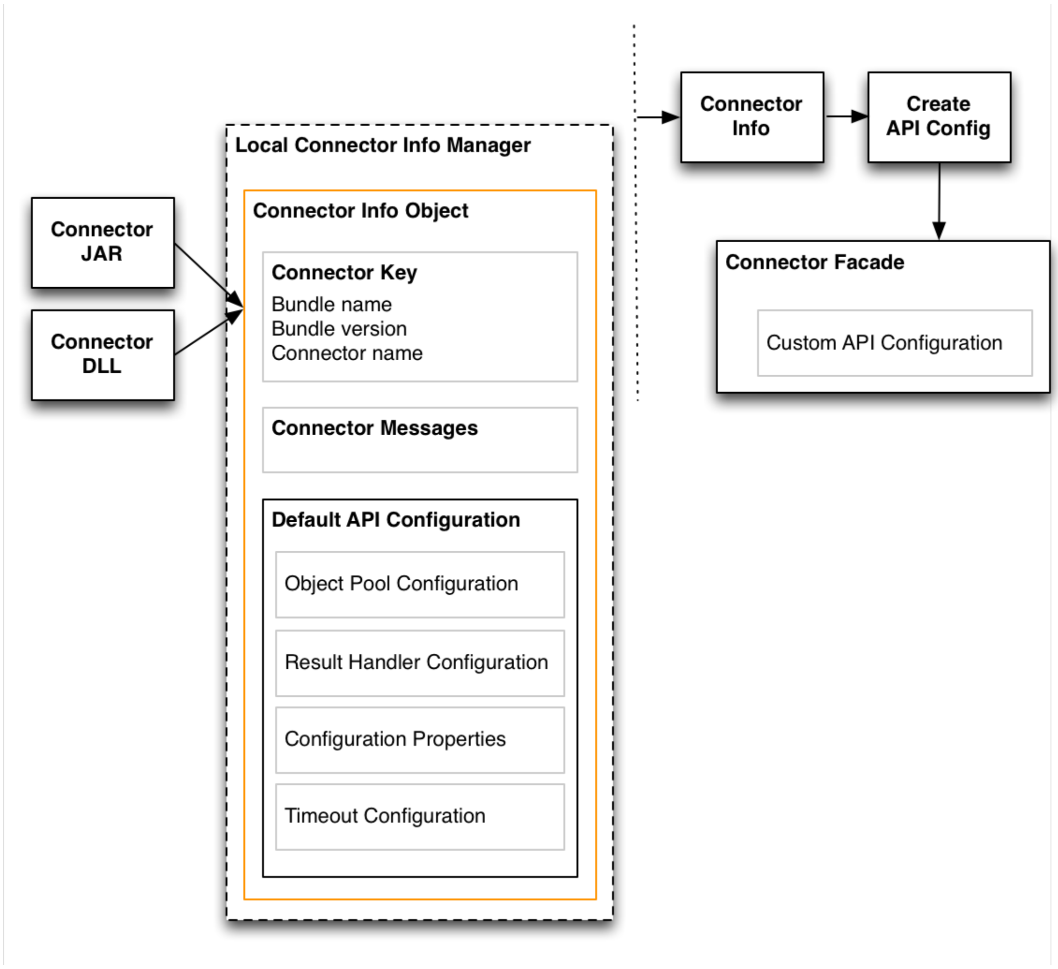
The easiest way to start using the ICF framework, from Java, is to use the sample Maven project file as a starting point. This sample project includes comprehensive comments about its use.

To use a .NET connector remotely, you must install the .NET remote connector server, as described in Install .NET RCS. You must also download and install the specific connectors that you want to use from the BackStage download site.

You can now start integrating the connector with your application.

ConnectorFacade interface

An application interacts with a connector through an instance of the `ConnectorFacade` class. The following diagram shows the creation and configuration of the connector facade. The components shown here are described in more detail in the sections that follow.



The connector facade is instantiated and configured in the following steps:

1. The application creates a `LocalConnectorInfoManager` instance (or instances) and adds the individual connector bundles (or assemblies).

   The `LocalConnectorInfoManager` processes these bundles or assemblies to instantiate a `ConnectorInfo` object.

   To be processed by the connector info manager, the connector bundle or assembly must have the following characteristics:

   ### *Java Connector Bundle*

   The `META-INF/MANIFEST.MF` file *must* include the following entries:

   - `ConnectorBundle-FrameworkVersion` - Minimum required ICF Framework version (either 1.1, 1.4, or 1.5)
   - `ConnectorBundle-Name` - Unique name of the connector bundle

- `ConnectorBundle-Version` - Version of the connector bundle

The combination of the `ConnectorBundle-Name` and the `ConnectorBundle-Version` must be unique.

The connector bundle JAR must contain at least one class, that has the `ConnectorClass` annotation and implements the `Connector` interface.

.NET Connector Assembly

The `AssemblyInfo.cs` is used to determine the bundle version, from the `AssemblyVersion` property.

The bundle name is derived from the `Name` property of the assembly. For more information, refer to the corresponding Microsoft documentation.

> ⚠ **Warning**
>
> If you change the name of your assembly, you must adjust the `bundleName` property in your connector configuration file, accordingly.

The connector assembly DLL must contain at least one class, that has the `ConnectorClassAttribute` attribute and implements the `Connector` interface.

2. For each connector, the `LocalConnectorInfoManager` processes the `MessageCatalog`, which contains the localized help and description messages for the configuration, and any log or error messages for the connector.

   Your application can use this information to provide additional help during the connector configuration process.

3. For each connector, the `LocalConnectorInfoManager` then processes the `ConfigurationClass`, to build the configuration properties for the connector.

4. Your application finds the connector info by its *connector key*. When the application has the connector info, it creates an API Configuration object that customizes the following components:

   - Object pool configuration
   - Result handler configuration
   - Configuration properties
   - Timeout configuration

   The API Configuration object is described in more detail in OpenICF API.

5. The `ConnectorFacade` takes this customized API configuration object, determines which connector to use and how to configure it, and implements all of the ICF API operations.

Creating a ConnectorFacade

Applications access the connector API through a `ConnectorFacade` class, and interact with the connector through a `ConnectorFacade` instance.

The following steps describe how to create a `ConnectorFacade` in your application.

1. Create a `ConnectorInfoManager` and acquire the `ConnectorInfo` object for your connector, as described in the previous section.

2. From the `ConnectorInfo` object, create the default `APIConfiguration`.

```
APIConfiguration apiConfig = info.createDefaultAPIConfiguration();
```

3. Use the default `APIConfiguration` to set the `ObjectPoolConfiguration`, `ResultsHandlerConfiguration`, `ConfigurationProperties`, and `TimeoutConfiguration`.

```
ConfigurationProperties properties = apiConfig.getConfigurationProperties();
```

4. Set all of the `ConfigurationProperties` that you need for the connector, using `setPropertyValue()`.

```
properties.setPropertyValue("host", SAMPLE_HOST);
properties.setPropertyValue("adminName", SAMPLE_ADMIN);
properties.setPropertyValue("adminPassword", SAMPLE_PASSWORD);
properties.setPropertyValue("usessl", false);
```

5. Use the `newInstance()` method of the `ConnectorFacadeFactory` to create a new instance of the connector.

```
ConnectorFacade conn = ConnectorFacadeFactory.getInstance()
        .newInstance(apiConfig);
```

6. Validate that you have set up the connector configuration correctly.

```
conn.validate();
```

7. Use the new connector with the supported operations (described in the following sections).

```
conn.[authenticate|create|update|delete|search|...]
```

Connector messages object

The Connector Messages interface sets the message catalog for each connector, and enables localization of messages. The interface has one method (`format()`), which formats a given message key in the current locale.

For more information, refer to the corresponding Javadoc.

API configuration object

The API configuration object holds the runtime configuration of the connector facade instance. The ICF framework creates a default API configuration object inside the Connector Info object. The application creates a copy of the API configuration object and customizes it according to its requirements. The API configuration object includes the following components:

Object Pool Configuration

The object pool configuration specifies the pool configuration for poolable connectors only. Non-poolable connectors ignore this parameter. The object pool configuration includes the following parameters:

maxObjects

The maximum number of idle and active instances of the connector.

maxIdle

The maximum number of idle instances of the connector.

maxWait

The maximum time, in milliseconds, that the pool waits for an object before timing out. A value of `0` means that there is no timeout.

minEvictableIdleTimeMillis

The maximum time, in milliseconds, that an object can be idle before it's removed. A value of `0` means there is no idle timeout.

minIdle

The minimum number of idle instances of the connector.

Results Handler Configuration

The results handler configuration defines how the ICF framework chains together the different results handlers to filter search results.

enableNormalizingResultsHandler

boolean

If the connector implements the attribute normalizer interface, you can enable this interface by setting this configuration property to `true`. If the connector does not implement the attribute normalizer interface, the value of this property has no effect.

enableFilteredResultsHandler

boolean

If the connector uses the filtering and search capabilities of the remote connected system, you can set this property to `false`. If the connector does not use the remote system's filtering and search capabilities (for example, the CSV file connector), you *must* set this property to `true`, otherwise the connector performs an additional, case-sensitive search, which can cause problems.

enableCaseInsensitiveFilter

boolean

By default, the filtered results handler (described previously) is case sensitive. If the filtered results handler is enabled this property lets you enable case-insensitive filtering. When case-insensitive filtering is not enabled, a search will not return results unless the case matches exactly. For example, a search for `lastName = "Jensen"` will not match a stored user with `lastName : jensen`.

enableAttributesToGetSearchResultsHandler

boolean

By default, IDM determines which attributes that should be retrieved in a search. If the `enableAttributesToGetSearchResultsHandler` property is set to `true`, the ICF framework removes all attributes from the READ/QUERY response, except for those that are specifically requested. For performance reasons, it is recommended that you set this property to `false` for local connectors, and to `true` for remote connectors.

Configuration Properties

The Configuration Properties object is built and populated by the framework as it parses the connectors configuration class.

Timeout Configuration

The timeout configuration enables you to configure timeout values per operation type. By default, there is no timeout configured for any operation type.

Connector info manager

You need to create a `ConnectorInfoManager` and a `ConnectorKey` for your connector.

The `ConnectorKey` uniquely identifies the connector instance. The `ConnectorKey` class takes a `bundleName` (the name of the Connector bundle), a `bundleVersion` (the version of the Connector bundle) and a connectorName (the name of the Connector).

The `ConnectorInfoManager` retrieves a `ConnectorInfo` object for the connector by its connector key.

You must initiate a specific connector info manager type, depending on whether your connector is local or remote. The following samples show how to create a local connector info manager and a remote connector info manager.

Acquiring a Local Connector Info Object (Java)

```
ConnectorInfoManagerFactory fact = ConnectorInfoManagerFactory.getInstance();
File bundleDirectory = new File("/connectorDir/bundles/myconnector");
URL url = IOUtil.makeURL(bundleDirectory,
     "/dist/org.identityconnectors.myconnector-1.0.jar");
ConnectorInfoManager manager = fact.getLocalManager(url);
ConnectorKey key = new ConnectorKey("org.identityconnectors.myconnector",
     "1.0", "MyConnector");
```

Acquiring a Remote Connector Info Object (Java)

```
ConnectorInfoManagerFactory fact = ConnectorInfoManagerFactory.getInstance();
File bundleDirectory = new File("/connectorDir/bundles/myconnector");
URL url = IOUtil.makeURL(bundleDirectory,
     "/dist/org.identityconnectors.myconnector-1.0.jar");
ConnectorInfoManager manager = fact.getLocalManager(url);
ConnectorKey key = new ConnectorKey("org.identityconnectors.myconnector",
     "1.0", "MyConnector");
```

Schema and supported operations

Different connectors support different subsets of the overall set of operations provided by OpenICF. When your connector is ready to use, you can use the `ConnectorFacade` to determine which operations your connector supports.

The quickest way to check whether an operation is supported is to determine whether that specific operation is part of the set of supported operations. The following sample test checks if the `CreateApiOp` is supported:

```
Set<Class< ? extends APIOperation>> ops = conn.getSupportedOperations();
return ops.contains(CreateApiOp.class);
```

Note that a connector might support a particular operation, only for specific object classes. For example, the connector might let you *create* a user, but not a group.

To be able to determine the list of supported operations for each object class, you need to check the schema. To determine whether the connector supports an operation for a specific object class, check the object class on which you plan to perform the operation, as shown in the following example.

```
Schema schema = conn.schema();
Set<ObjectClassInfo> objectClasses = schema.getObjectClassInfo();
Set<ObjectClassInfo> ocinfos = schema
        .getSupportedObjectClassesByOperation(CreateApiOp.class);

for(ObjectClassInfo oci : objectClasses) {
    // Check that the operation is supported for your object class.
    if (ocinfos.contains(ocinfo)) {
        // object class is supported
    }
}
```

In addition to determining the supported operations for an object class, your application can check which attributes are *required* and which attributes are *allowed* for a particular object class. The `ObjectClassInfo` class contains this information as a set of `AttributeInfo` objects.

The following example shows how to retrieve the attributes for an object class.

```
Schema schema = conn.schema();
Set<ObjectClassInfo> objectClasses = schema.getObjectClassInfo();
for(ObjectClassInfo oci : objectClasses) {
    Set<AttributeInfo> attributeInfos = oci.getAttributeInfo();
    String type = oci.getType();
    if(ObjectClass.ACCOUNT_NAME.equals(type)) {
        for(AttributeInfo info : attributeInfos) {
            System.out.println(info.toString());
        }
    }
}
```

Using the schema object, you can obtain the following information:

- Object classes and their attributes
- Operation options per operation

The following example shows how to retrieve the schema as a list of `ObjectClass` objects, from the `ObjectClassInfo` class.

```
ObjectClass objectClass = new ObjectClass(objectClassInfo.getType());
```

Operation options

Operation options provide an extension point to an operation, letting you request additional information from the application, for each operation. The connector framework includes a number of predefined operation options for the most common use cases. For example, the option `OP_ATTRIBUTES_TO_GET` enables you to specify a list of attributes that should be returned by an operation. When you write a connector, you must define the operation options that your connector supports in the schema, so that the application knows which operation options are supported.

For a list of the predefined operation options, refer to the corresponding Javadoc.

ICF special attributes

ICF includes a number of *special* attributes, that all begin and end with `__` (for example `__NAME__`, and `__UID__`). These special attributes are essentially functional aliases for specific attributes or object types. The purpose of the special attributes is to enable a connector developer to create a contract regarding how a property can be referenced, regardless of the application that is using the connector. In this way, the connector can map specific object information between an arbitrary application and the resource, without knowing how that information is referenced in the application.

The special attributes are used extensively in the generic LDAP connector, which can be used with PingDS (DS), Active Directory, OpenLDAP, and other LDAP directories. Each of these directories might use a different attribute name to represent the same type of information. For example, Active Directory uses `unicodePassword` and DS uses `userPassword` to represent the same thing, a user's password. The LDAP connector uses the special OpenICF `__PASSWORD__` attribute to abstract that difference.

For a list of the special attributes, refer to the corresponding Javadoc.

Connector instance management

The ICF framework supports multiple *connector types*, based on the implementation of the `connector` interface, and the `configuration` interface. These two interfaces determine the following:

- Whether the connector instance is obtained from a pool or whether a new instance is created *for each operation*.
- Whether the connector configuration instance is retained and reused for each operation (stateful configuration), or a new configuration instance is created for each operation (stateless).

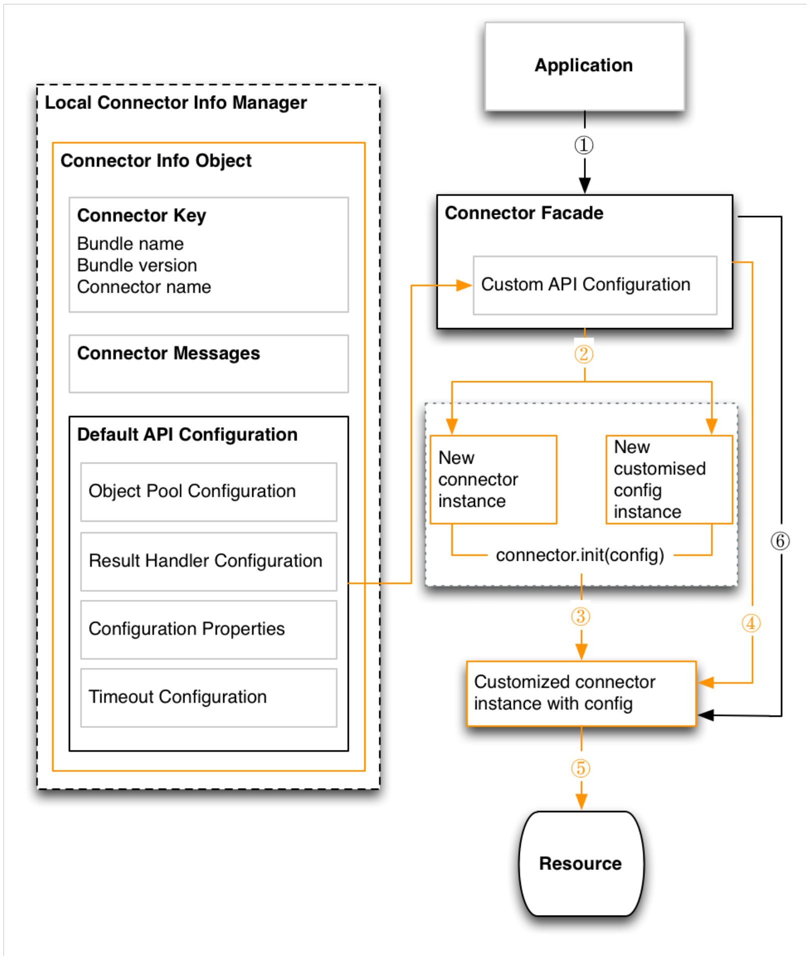
Connector developers determine which type of connector to implement, assessing the best match for the resource to which they are connecting. The interaction between the `connector` and `configuration` interface implementations is described in detail in Connector types. This section illustrates how the ICF framework manages connector instantiation, depending on the connector type.

Instantiate a stateless, non-poolable connector

The most basic connector has a stateless configuration, and is not pooled. A basic connector is initialized as follows:

1. The application calls an operation (for example, CREATE) on the connector facade.
2. The ICF framework creates a new *configuration instance*, and initializes it with its configuration properties.
3. When the framework has the configuration instance, with all the attributes in the configuration set, the framework creates a new *connector instance*, and initializes it, with the configuration that has been set.
4. The framework executes the operation (for example, CREATE) on the connector instance.
5. The connector instance executes the operation on the resource.
6. The framework calls the `dispose()` method to release all resources that the connector instance was using.

The following illustration shows the initialization process for a basic connector, and references the numbered steps in the preceding list.



Instantiate a stateless, poolable connector

The second connector type has a stateless configuration, but can be pooled. A stateless, poolable connector is instantiated as follows:

1. The application calls an operation (for example, CREATE) on the connector facade.
2. The ICF framework calls on the object pool, to borrow a *live* connector instance to execute the operation.

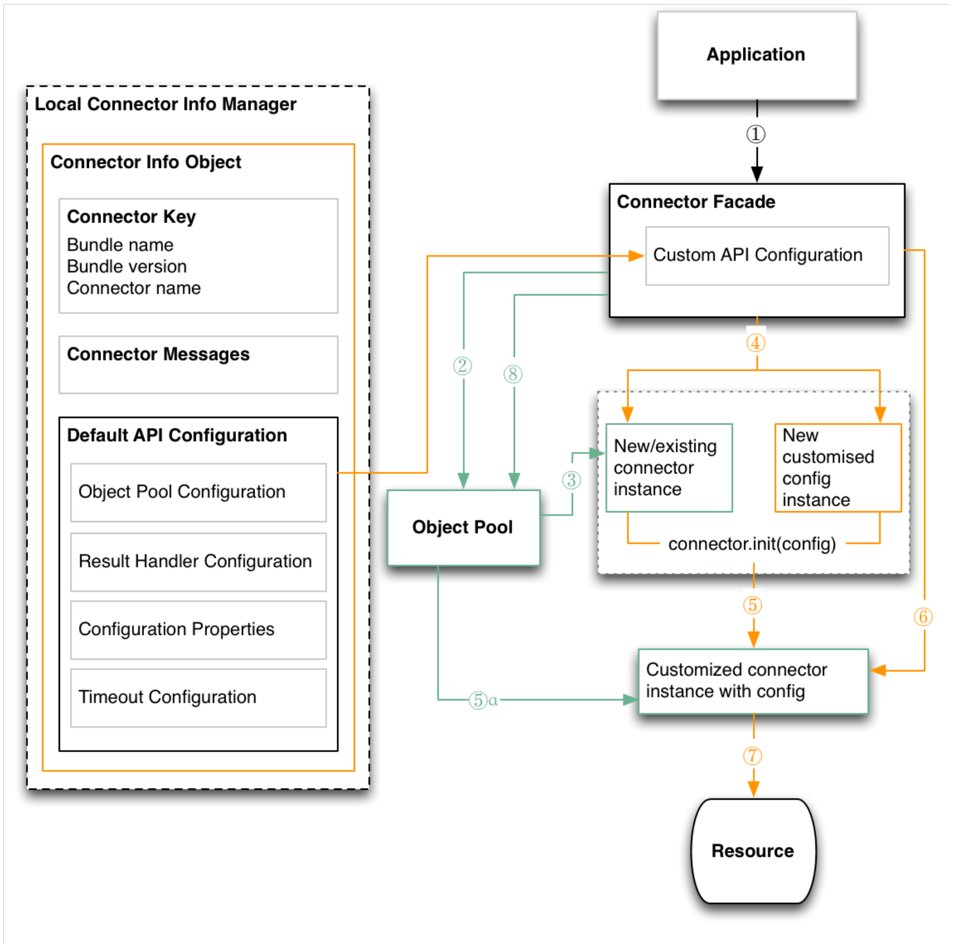
   If the object pool has an idle connector instance available, the framework *borrows* that one instance (step 5a in the illustration that follows).

   The framework calls the `checkAlive` method on the customized connector instance with its configuration, to check if the instance that was borrowed from the pool is still alive, and ready to execute the operation. If the instance is no longer alive and ready, the framework disposes of the instance and borrows another one.

   The thread that borrows the object has exclusive access to that connector instance; that is, it is thread-safe.
3. If the object pool has no idle connector instances, the pool creates a new connector instance.
4. The framework creates a new *configuration instance*, and initializes it with its configuration properties.
5. The framework initializes the borrowed connector instance, with the configuration that has been set.
6. The framework executes the operation (for example, CREATE) on the connector instance.

7. The connector instance executes the operation on the resource.
8. When the operation is complete, the framework releases the connector instance back into the pool. No `dispose()` method is called.

The following illustration shows the initialization process for a stateless, poolable connector, and references the numbered steps in the preceding list.



Instantiate a stateful, non-poolable connector

The third connector type has a stateful configuration, and cannot be pooled. A stateful, non-poolable connector is instantiated as follows:

1. The ICF framework creates a new *configuration instance*, initializes it with its configuration properties, and stores it in the connector facade, before any operations are called.

   This single configuration instance is shared between multiple threads. The framework does not guarantee isolation, so connector developers must ensure that their implementation is thread-safe.

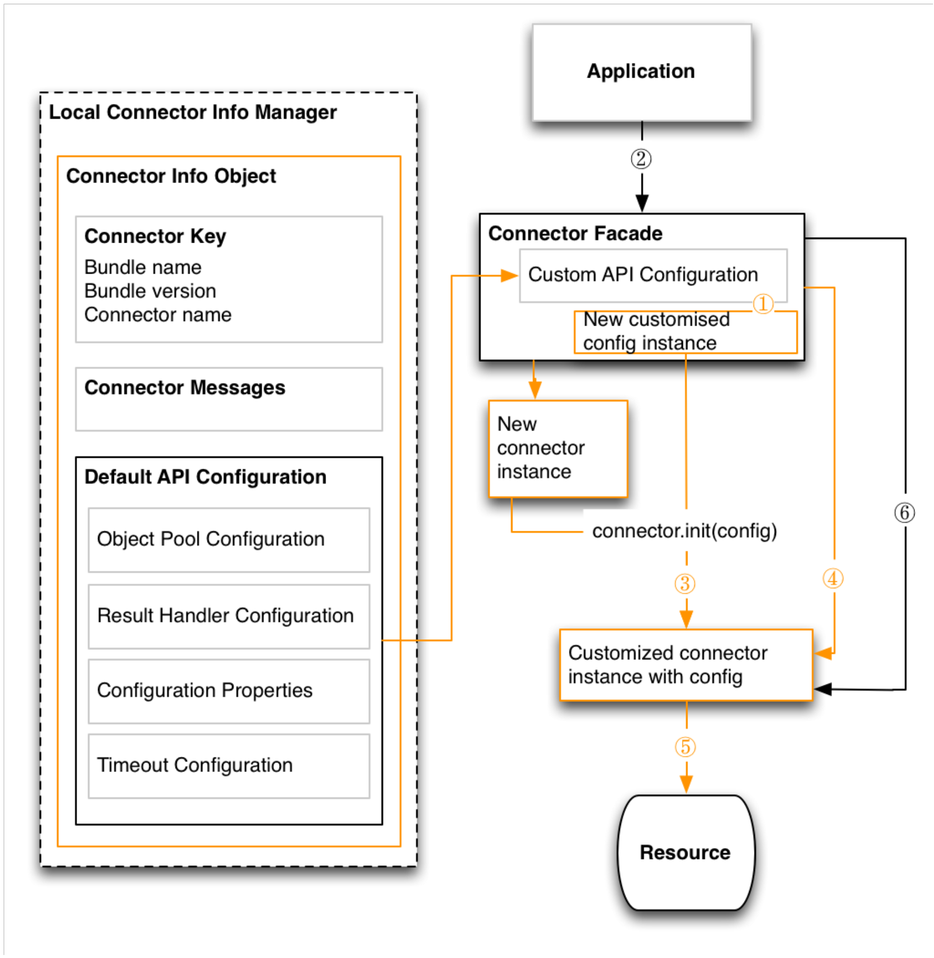
2. The application calls an operation (for example, CREATE) on the connector facade.
3. The ICF framework creates a new connector instance, and calls the `init()` method on that connector instance, with the stored configuration. The framework initializes the connector with the single configuration instance stored within the connector facade.
4. The framework executes the operation (for example, CREATE) on the connector instance.

5. The connector instance executes the operation on the resource.

6. The framework calls the `dispose()` method to release all resources that the connector instance was using.

   Note that the customized config instance remains in the connector facade, and is reused for the next operation.

The following illustration shows the initialization process for a non-poolable connector, with a stateful configuration. The illustration references the numbered steps in the preceding list.



Instantiate a stateful, poolable connector

The fourth connector type has a stateful configuration, and can be pooled. A stateful, poolable connector is instantiated as follows:

1. The ICF framework creates a new *configuration instance*, initializes it with its configuration properties, and stores it in the connector facade, before any operations are called.

   This single configuration instance is shared between multiple threads. The framework does not guarantee isolation, so connector developers must ensure that their implementation is thread-safe.

2. The application calls an operation (for example, CREATE) on the connector facade.

3. The framework calls on the object pool, to borrow a *live* connector instance to execute the operation.

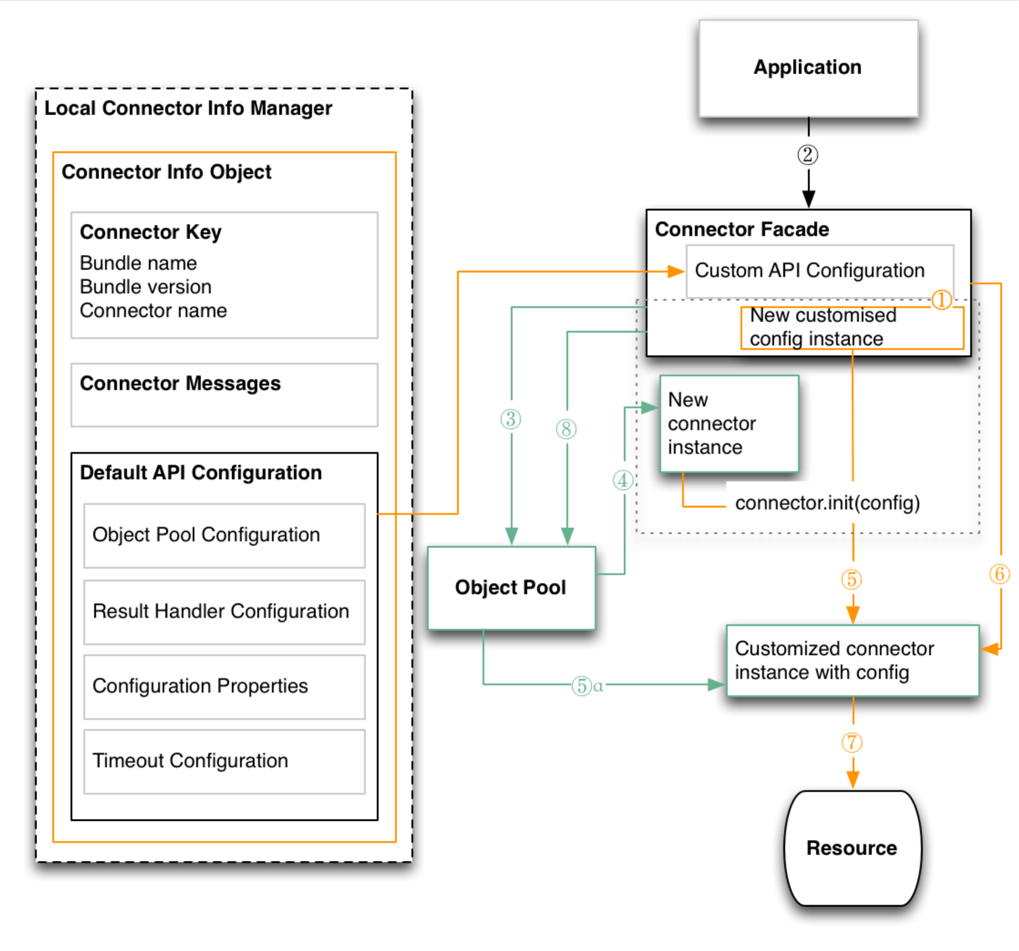
   If the object pool has an idle connector instance available, the framework *borrows* that one instance (step 5a in the illustration that follows).

The framework calls the `checkAlive` method on the customized connector instance with its configuration, to check if the instance that was borrowed from the pool is still alive, and ready to execute the operation. If the instance is no longer alive and ready, the framework disposes of the instance and borrows another one.

The thread that borrows the object has exclusive access to that connector instance; that is, it is thread-safe.

4. If the object pool has no idle connector instances, the pool creates a new connector instance.
5. The framework initializes the borrowed connector instance, with the stored configuration.
6. The framework executes the operation (for example, CREATE) on the connector instance.
7. The connector instance executes the operation on the resource.
8. When the operation is complete, the framework releases the connector instance back into the pool. No `dispose()` method is called.

The following illustration shows the initialization process for a stateful, poolable connector, and references the numbered steps in the preceding list.



OpenICF SPI

This chapter describes the ICF SPI, which lets you create connectors that are compatible with the ICF framework.

The SPI includes a number of interfaces, but you need only implement those that are supported by the target resource to which you are connecting. For information about how to get started with writing connectors, refer to Java connectors and Scripted connectors with Groovy.

The order in which you implement your connector is as follows:

1. Decide on the connector type (refer to Connector types).
2. Implement the configuration interface (refer to Configuration interface).
3. Implement the connector interface (refer to Connector interface).
4. Implement the operation interfaces (refer to Operation interfaces).

Connector types

OpenICF supports multiple connector types based on the implementation of the `connector` interface and the `configuration` interface. These two interfaces determine whether the connector can be pooled and whether its configuration is stateful. Before you begin developing your connector, decide on the *connector type* based on the system to which you are connecting. Learn more about how the OpenICF framework manages each connector type in Connector instance management.

This section outlines the different connector types.

Connector

The basic connector is a *non-poolable* connector. Each operation is executed on a new instance of the connector. OpenICF creates a new instance of the connector class and uses a new or existing instance of the connector configuration to initialize the instance before the operation is started. After the operation completes, OpenICF disposes of the connector instance.

Poolable Connector

Before an operation is started, an existing connector instance is pulled from the connector pool. If there’s no existing instance, a new instance is created. After the operation runs, the connector instance is released and placed back into the pool.

The OpenICF framework pools *instances* of a poolable connector, rather than pooling connections within the connector. Learn more about pooling in Connection pooling configuration.

Configuration

For a basic non-stateful configuration, a new configuration instance is created and configured with the configuration properties each time the configuration is used when an operation is validated or when a new connector instance is initialized.

Stateful Configuration

With a stateful configuration, the configuration instance is created only once and is used until the facade or connector pool associated with the configuration is disposed of.

The following table illustrates how these elements combine to determine the connector type.

Connector Types

	Connector	Poolable Connector
Configuration	Entirely stateless combination. A new configuration and connector instance are created for each operation.	It's preferable to keep connector instances in a pool. A new configuration is required only when a new connector instance is added to the pool.
Stateful Configuration	The configuration can be used to make the heavy resource initialization. The less intensive connector instance can then run the operation.	The configuration must be shared between the instances in the same pool and the connector initialization is expensive.

Learn how the OpenICF framework manages each connector type in Connector instance management.

Configuration interface

The ICF connector framework uses the configuration interface implementation to build the *configuration properties* inside the API configuration.

The configuration interface implementation includes the required information to enable the connector to connect to the target system, and to perform its operations. The configuration interface implements getters and setters for each of its defined properties. It also provides a validate method that your application can use to check whether all the required properties are available, and valid, before passing them to the connector.

The configuration interface has three methods:

- `setConnectorMessages(ConnectorMessages messages)` sets the message catalog instance, and lets the connector provide localized messages.

  The message catalog is defined in the file `Messages.properties`, and can be localized as required by appending the locale to the file name, for example, `Messages_fr.properties`. For more information on the message catalog, refer to Connector messages object.

- `getConnectorMessages()` returns the message catalog that is set by setConnectorMessages(ConnectorMessages).

- `validate()` checks that all the required properties have been set and that their values are valid.

  The purpose of this method is to test that the configuration that the application provides to your connector is valid.

Each property that is declared is not necessarily required. If a property is required, it must be included in the `ConfigurationProperty` annotation.

The `ConfigurationProperty` annotation (Java) or attribute (.NET) lets you add custom meta information to properties. The ICF framework scans the meta information and collects this information to build the `ConfigurationProperties` object inside the `APIConfiguration`. The following meta information can be provided:

Element	Description	Implementation in Java	Implementation in C#
order	The order in which this property is displayed		
helpMessageKey	Lets you change the default help message key	`propertyName.help`	`help_propertyName`
displayMessageKey	Lets you change the default display message key	`propertyName.display`	`display_propertyName`
groupMessageKey	Lets you change the default group message key	`propertyName.group`	`group_propertyName`
confidential	Indicates that this is a confidential property and that its value should be encrypted by the application when persisted		
required	Boolean, indicates whether the property is required		
operations	The array of operations that require this property		

The following examples show how the meta information is provided, in both Java and C#.

Java

Stateless Configuration Implementation (Java)

```
public class SampleConfiguration extends AbstractConfiguration  {

    /**
     * {@inheritDoc}
     */
    public void validate() {
    }

    @ConfigurationProperty(
        order = 1,
        helpMessageKey = "passwordFieldName.help",
        displayMessageKey = "passwordFieldName.display",
        groupMessageKey = "authenticateOp.group",
        confidential = false,
        required = false,
        operations = {AuthenticateOp.class,CreateOp.class}
    )
    public String getPasswordFieldName() {
        return passwordFieldName;
    }

    public void setPasswordFieldName(String value) {
        passwordFieldName = value;
    }
}
```

C#

Stateless Configuration Implementation (C#)

```
public class ActiveDirectoryConfiguration : AbstractConfiguration
    {

        [ConfigurationProperty(
            Order = 1,
            HelpMessageKey = "help_PasswordFieldName",
            DisplayMessageKey = "display_PasswordFieldName",
            GroupMessageKey = "group_PasswordFieldName",
            Confidential = false,
            Required = false,
            OperationTypes = new[] { typeof(AuthenticateOp) })
        ]
        public String PasswordFieldName
        { get; set; }

        public override void Validate()
        {
            throw new NotImplementedException();
        }
    }
```

Java

Stateful Configuration Implementation (Java)

```
public class SampleConfiguration extends AbstractConfiguration
    implements StatefulConfiguration {

    /**
     * {@inheritDoc}
     */
    public void release() {
    }

    /**
     * {@inheritDoc}
     */
    public void validate() {
    }
}
```

C#

Stateful Configuration Implementation (C#)

```
public class ActiveDirectoryConfiguration : AbstractConfiguration,
    StatefulConfiguration
    {

        public override void Validate()
        {
            throw new NotImplementedException();
        }

        public void Release()
        {
            throw new NotImplementedException();
        }
    }
```

Validate operation

The validate operation validates the connector configuration. A valid configuration is one that is *ready to be used* by the connector.

A configuration that is *ready*, has the following characteristics:

- It is complete, that is, all required properties are present and have values.
- All property values are well-formed, that is, they are in the expected range and have the expected format.

ValidateApiOp

The validate operation returns a `ConfigurationException` in the following situations:

- The Framework version is not compatible with the connector.
- The connector does not have the required attributes in `MANIFEST.MF`.
- The `ConfigurationProperties` cannot be merged into the configuration.

Implementation of the valid operation, at the API Level

```
@Test
public void ValidateTest() {
    logger.info("Running Validate Test");
    final ConnectorFacade facade = createConnectorFacade(BasicConnector.class, null);
    facade.validate();
}
```

Validate SPI implementation

The `validate()` method of the configuration operation must return one of the following:

- `RuntimeException` if the configuration is not valid.
- `NullPointerException` if a required configuration property is null.
- `IllegalArgumentException` if a required configuration property is blank.

Implementation of the validate method

```
public void validate() {
    if (StringUtil.isBlank(host)) {
        throw new IllegalArgumentException("Host User cannot be null or empty.");
    }

    Assertions.blankCheck(remoteUser, "remoteUser");

    Assertions.nullCheck(password, "password");
}
```

Supported configuration types

The ICF framework supports a limited number of configuration property types. This limitation is necessary, because ICF must serialise and deserialize the configuration property values when sending them over the network.

You can use any of the following types, or an array of these types. Lists of types are not supported.

```
String.class
long.class
Long.class
char.class
Character.class
double.class
Double.class
float.class
Float.class
int.class
Integer.class
boolean.class
Boolean.class
URI.class
File.class
GuardedByteArray.class
GuardedString.class
Script.class
```

```
typeof(string),
typeof(long),
typeof(long?),
typeof(char),
typeof(char?),
typeof(double),
typeof(double?),
typeof(float),
typeof(float?),
typeof(int),
typeof(int?),
typeof(bool),
typeof(bool?),
typeof(Uri),
typeof(FileName),
typeof(GuardedByteArray),
typeof(GuardedString),
typeof(Script)
```

The framework introspects the implemented configuration class and adds all properties that have a `set/get` method to the `ConfigurationProperties` object.

The `ConfigurationClass` annotation (Java) or attribute (.NET) provides additional information to the ICF framework about the configuration class. The following information is provided:

Element	Description
privateProperty	If this is set, the property is hidden from the application, and the application cannot set the property through the `APIConfiguration`.
skipUnsupported	If the type of an added property is not supported, the framework throws an exception. To avoid the exception, set the value of `skipUnsupported` to `true`.

Java

ConfigurationClass Annotation (Java)

```
@ConfigurationClass(ignore = { "privateProperty", "internalProperty" }, skipUnsupported = true)
```

C#

ConfigurationClass Attribute (C#)

```
[ConfigurationClass(Ignore = { "privateProperty", "internalProperty" }, SkipUnsupported = true)]
```

Connector interface

The connector interface declares a connector, and manages its life cycle. You *must* implement the `connector` interface. A typical connector lifecycle is as follows:

- The connector creates a connection to the target system.
- Any operations implemented in the connector are called.
- The connector discards the connection and disposes of any resources it has used.

The `connector` interface has only three methods:

- `init(Configuration)` initializes the connector with its configuration
- `getConfiguration()` returns the configuration that was passed to `init(Configuration)`
- `dispose()` disposes of any resources that the connector uses.

The `ConnectorClass`, which is the implementation of the connector interface, must have the `ConnectorClass` annotation (Java) or attribute (.NET) so that the ICF framework can find the connector class. The following table shows the elements within the connector class.

Element	Description
configurationClass	The configuration class for the connector.
displayNameKey	A key in the message catalog that holds a human readable name for the connector.
categoryKey	The category to which the connector belongs, such as LDAP, or DB.
messageCatalogPaths	The resource path(s) to the message catalog. If multiple paths are provided, the message catalogs are collated. By default, if no path is specified, the `connector-package.Messages.properties` is used.

The following examples show the connector interface implementation, in Java and C#.

Java

Connector Interface Implementation in Java

```
@ConnectorClass(
    displayNameKey = "Sample.connector.display",
    configurationClass = SampleConfiguration.class)
public class SampleConnector implements Connector...
```

C#

Connector Interface Implementation in C#

```
[ConnectorClass(
    "connector_displayName",
    typeof (SampleConfiguration)
    ]
public class SampleConnector : Connector ...
```

Implement a poolable connector interface

Certain connectors support the ability to be pooled. For a pooled connector, ICF maintains a pool of connector instances and reuses these instances for multiple provisioning and reconciliation operations. When an operation must be executed, an existing connector instance is taken from the connector pool. If no connector instance exists, a new instance is initialized. When the operation has been executed, the connector instance is released back into the connector pool, ready to be used for a subsequent operation.

For an unpooled connector, a new connector instance is initialized for every operation. When the operation has been executed, ICF disposes of the connector instance. Because the initialization of a connector is an expensive operation, reducing the number of connector initializations can substantially improve performance.

The following connection pooling configuration parameters can be set:

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 2 minutes.

A connection pool cleaner thread runs every minute and closes connections whose `lastUsed` time is larger than the `minEvictableIdleTimeMillis`.

minIdle

The minimum number of idle connector instances in the pool. The default value is 1 instance.

A `PoolableConnector` extends the connector interface with the `checkAlive()` method. You should use a `PoolableConnector` when the `init(Configuration)` method is so expensive that it is worth keeping the connector instance in a pool and reusing it between operations. When an existing connector instance is pooled, the framework calls the `checkAlive()` method. If this method throws an error, the framework discards it from the pool and obtains another instance, or creates a new connector instance and calls the `init()` method. The `checkAlive()` method is used to make sure that the instance in the pool is still operational.

Operation interfaces

The SPI provides several operations. The subset of operations that you implement will depend on the target resource to which you are connecting. Each operation interface defines an action that the connector can perform on the target resource.

The following sections describe the operation interfaces that are provided by the SPI, and provide examples of how they can be implemented in your connector. The sections include the API- and SPI-level rules for each operation.

- Authenticate operation
- Create operation
- Delete operation
- Resolve username operation
- Schema operation
- Script on connector operation
- Script on resource operation
- Search operation
- Sync operation
- Test operation
- Update operation
- Update attribute values operation

Authenticate operation

The authenticate operation authenticates an object on the target system, based on two parameters, usually a unique identifier (username) and a password. If possible, your connector should try to authenticate these credentials natively.

If authentication fails, the connector should throw a runtime exception. The exception must be an `IllegalArgumentException` or, if a native exception is available and is of type `RuntimeException`, that native runtime exception. If the native exception is not a `RuntimeException`, it should be wrapped in a `RuntimeException`, and then thrown.

The exception should provide as much detail as possible for logging problems and failed authentication attempts. Several exceptions are provided in the `exceptions` package, for this purpose. For example, one of the most common authentication exceptions is the `InvalidPasswordException`.

For more information about the common exceptions provided in the OpenICF framework, refer to Common exceptions.

Use the ICF authenticate operation

This section shows how your application can use the framework's `authentication` operation, and how to write a unit test for this operation, when you are developing your connector.

The `authentication` operation throws a `RuntimeException` if the credentials do not pass authentication, otherwise returns the `UID`.

Sample Unit Test for the Authentication Operation (Java)

```
@Test
public void authenticateTest() {
    logger.info("Running Authentication Test");
    final ConnectorFacade facade = createConnectorFacade(BasicConnector.class, null);
    final OperationOptionsBuilder builder = new OperationOptionsBuilder();
    Uid uid =
        facade.authenticate(ObjectClass.ACCOUNT, "username", new GuardedString("Passw0rd"
        .toCharArray()), builder.build());
    Assert.assertEquals(uid.getUidValue(), "username");
}
```

Implement the authenticate operation

To implement the `authenticate` operation in your connector, add the `AuthenticateOp` interface to your connector class, for example:

```
@ConnectorClass(
    displayNameKey = "Sample.connector.display",
    configurationClass = SampleConfiguration.class)
public class SampleConnector implements Connector, AuthenticateOp...
```

For more information, refer to the AuthenticateOp JavaDoc.

The SPI provides the following detailed exceptions:

- UnknownUidException - the UID does not exist on the resource.

`(org.identityconnectors.framework.common.exceptions.UnknownUidException)`

- ConnectorSecurityException - base exception for all security-related exceptions.

  `(org.identityconnectors.framework.common.exceptions.ConnectorSecurityException)`

- InvalidCredentialException - generic invalid credential exception that should be used if the specific error cannot be obtained.

  `(org.identityconnectors.framework.common.exceptions.UnknownUidException)`

- InvalidPasswordException - the password provided is incorrect.

  `(org.identityconnectors.framework.common.exceptions.InvalidPasswordException)`

- PasswordExpiredException - the password is correct, but has expired.

  `(org.identityconnectors.framework.common.exceptions.PasswordExpiredException)`

- PermissionDeniedException - the user can be identified but does not have permission to authenticate.

  `(org.identityconnectors.framework.common.exceptions.PermissionDeniedException)`

Implementation of the Authentication Operation, at the SPI Level

```
public Uid authenticate(final ObjectClass objectClass, final String userName,
        final GuardedString password, final OperationOptions options) {
    if (ObjectClass.ACCOUNT.equals(objectClass)) {
        return new Uid(userName);
    } else {
        logger.warn("Authenticate of type {0} is not supported", configuration
                .getConnectorMessages().format(objectClass.getDisplayNameKey(),
                        objectClass.getObjectClassValue()));
        throw new UnsupportedOperationException("Authenticate of type"
                + objectClass.getObjectClassValue() + " is not supported");
    }
}
```

Create operation

The create operation interface enables the connector to create objects on the target system. The operation includes one method (`create()`). The method takes an `ObjectClass` , and any provided attributes, and creates the object and its UID. The connector must return the UID so that the caller can refer to the created object.

The connector should make a best effort to create the object, and should throw an informative `RuntimeException` , indicating to the caller why the operation could not be completed. Defaults can be used for any required attributes, as long as the defaults are documented.

The UID is never passed in with the attribute set for this method. If the resource supports a mutable UID, you can create a resource-specific attribute for the ID, such as `unix_uid` .

If the `create` operation is only partially successful, the connector should attempt to roll back the partial change. If the target system does not allow this, the connector should report the partial success of the create operation and throw a `RetryableException` . For example:

```
public static RetryableException wrap(final String message, final Uid uid) {
    return new RetryableException(message, new AlreadyExistsException().initUid(Assertions
    .nullChecked(uid, "Uid")));
}
```

Use the ICF create operation

The following exceptions are thrown by the Create API operation:

- `IllegalArgumentException` - if `ObjectClass` is missing, or if elements of the set produce duplicate values of `Attribute#getName()`
- `NullPointerException` - if the `createAttributes` parameter is `null`
- `RuntimeException` - if the `Connector` SPI throws a native exception

Consumption of the Create Operation, at the API Level

```
@Test
public void createTest() {
    logger.info("Running Create Test");
    final ConnectorFacade facade = createConnectorFacade(BasicConnector.class, null);
    final OperationOptionsBuilder builder = new OperationOptionsBuilder();
    Set<Attribute> createAttributes = new HashSet<Attribute>();
    createAttributes.add(new Name("Foo"));
    createAttributes.add(AttributeBuilder.buildPassword("Password".toCharArray()));
    createAttributes.add(AttributeBuilder.buildEnabled(true));
    Uid uid = facade.create(ObjectClass.ACCOUNT, createAttributes, builder.build());
    Assert.assertEquals(uid.getUidValue(), "foo");
}
```

Implement the create operation

The SPI provides the following detailed exceptions:

- `UnsupportedOperationException` - the create operation is not supported for the specified object class
- `InvalidAttributeValueException` - a required attribute is missing, an attribute is present that cannot be created, or a provided attribute has an invalid value
- `AlreadyExistsException` - an object with the specified `Name` already exits on the target system
- `PermissionDeniedException` - the target resource will not allow the connector to perform the specified operation
- `ConnectorIOException`, `ConnectionBrokenException`, `ConnectionFailedException` - a problem as occurred with the connection
- `RuntimeException` - thrown if anything else goes wrong. You should try to throw a native exception in this case.

Implementation of the Create Operation, at the SPI Level

```
public Uid create(final ObjectClass objectClass, final Set<Attribute> createAttributes,
        final OperationOptions options) {
    if (ObjectClass.ACCOUNT.equals(objectClass) || ObjectClass.GROUP.equals(objectClass)) {
        Name name = AttributeUtil.getNameFromAttributes(createAttributes);
        if (name != null) {
            // do real create here
            return new Uid(AttributeUtil.getStringValue(name).toLowerCase());
        } else {
            throw new InvalidAttributeValueException("Name attribute is required");
        }
    } else {
        logger.warn("Delete of type {0} is not supported", configuration.getConnectorMessages()
                .format(objectClass.getDisplayNameKey(), objectClass.getObjectClassValue()));
        throw new UnsupportedOperationException("Delete of type"
                + objectClass.getObjectClassValue() + " is not supported");
    }
}
```

Delete operation

The delete operation interface enables the connector to delete an object on the target system. The operation includes one method (`delete()`). The method takes an `ObjectClass` , a `Uid` , and any operation options.

The connector should call the native delete methods to remove the object, specified by its unique ID.

Use the ICF delete operation

The following exceptions are thrown by the Delete API operation:

- `UnknownUidException` - the UID does not exist on the resource

Consumption of the Delete Operation, at the API Level

```
@Test
public void deleteTest() {
    logger.info("Running Delete Test");
    final ConnectorFacade facade = createConnectorFacade(BasicConnector.class, null);
    final OperationOptionsBuilder builder = new OperationOptionsBuilder();
    facade.delete(ObjectClass.ACCOUNT, new Uid("username"), builder.build());
}
```

Implement the delete operation

Implementation of the Delete Operation, at the SPI Level

```
public void delete(final ObjectClass objectClass, final Uid uid, final OperationOptions options) {
    if (ObjectClass.ACCOUNT.equals(objectClass) || ObjectClass.GROUP.equals(objectClass)) {
        // do real delete here
    } else {
        logger.warn("Delete of type {0} is not supported", configuration.getConnectorMessages()
                .format(objectClass.getDisplayNameKey(), objectClass.getObjectClassValue()));
        throw new UnsupportedOperationException("Delete of type"
                + objectClass.getObjectClassValue() + " is not supported");
    }
}
```

Resolve username operation

The resolve username operation enables the connector to resolve an object to its UID, based on its username. This operation is similar to the simple authentication operation. However, the resolve username operation does not include a password parameter, and does not attempt to authenticate the credentials. Instead, it returns the UID that corresponds to the supplied username.

The implementation must, however, validate the username (that is, the connector must throw an exception if the username does not correspond to an existing object). If the username validation fails, the the connector should throw a runtime exception, either an `IllegalArgumentException` or, if a native exception is available and is of type `RuntimeException`, simply throw that exception. If the native exception is not a `RuntimeException`, it should be wrapped in a `RuntimeException`, and then thrown.

The exception should provide as much detail as possible for logging problems and failed attempts. Several exceptions are provided in the `exceptions` package, for this purpose. For example, one of the most common exceptions is the `UnknownUidException`.

Use the ICF resolve username operation

The operation throws a `RuntimeException` if the username validation fails, otherwise returns the `UID`.

Consumption of the ResolveUsername operation, at the API Level

```
@Test
public void resolveUsernameTest() {
    logger.info("Running ResolveUsername Test");
    final ConnectorFacade facade = createConnectorFacade(BasicConnector.class, null);
    final OperationOptionsBuilder builder = new OperationOptionsBuilder();
    Uid uid = facade.resolveUsername(ObjectClass.ACCOUNT, "username", builder.build());
    Assert.assertEquals(uid.getUidValue(), "username");
}
```

Implement the resolve username operation

The SPI provides the following detailed exceptions:

- UnknownUidException - the UID does not exist on the resource

Implementation of the ResolveUsername Operation, at the SPI Level

```
public Uid resolveUsername(final ObjectClass objectClass, final String userName,
        final OperationOptions options) {
    if (ObjectClass.ACCOUNT.equals(objectClass)) {
        return new Uid(userName);
    } else {
        logger.warn("ResolveUsername of type {0} is not supported", configuration
                .getConnectorMessages().format(objectClass.getDisplayNameKey(),
                        objectClass.getObjectClassValue()));
        throw new UnsupportedOperationException("ResolveUsername of type"
                + objectClass.getObjectClassValue() + " is not supported");
    }
}
```

Schema operation

The Schema Operation interface enables the connector to describe the types of objects that it can handle on the target system, and the operations and options that the connector supports for each object type.

The operation has one method, `schema()`, which returns the types of objects on the target system that the connector supports. The method should return the object class name, its description, and a set of attribute definitions.

The implementation of this operation includes a mapping between the native object class and the corresponding connector object. The special `Uid` attribute should not be returned, because it is not a true attribute of the object, but a reference to it. For more information about special attributes in ICF, refer to ICF Special Attributes.

If your resource object class has a writable unique ID attribute that is different to its `Name`, your schema should contain a resource-specific attribute that represents this unique ID. For example, a Unix account object might contain a `unix_uid`.

Use the ICF schema operation

Consumption of the Schema Operation, at the API Level

```
@Test
public void schemaTest() {
    logger.info("Running Schema Test");
    final ConnectorFacade facade = createConnectorFacade(BasicConnector.class, null);
    Schema schema = facade.schema();
    Assert.assertNotNull(schema.findObjectClassInfo(ObjectClass.ACCOUNT_NAME));
}
```

Implement the schema operation

Implementation of the SchemaOp operation, at the SPI Level

```
public Schema schema() {
    if (null == schema) {
        final SchemaBuilder builder = new SchemaBuilder(BasicConnector.class);
        // Account
        ObjectClassInfoBuilder accountInfoBuilder = new ObjectClassInfoBuilder();
        accountInfoBuilder.addAttributeInfo(Name.INFO);
        accountInfoBuilder.addAttributeInfo(OperationalAttributeInfos.PASSWORD);
        accountInfoBuilder.addAttributeInfo(PredefinedAttributeInfos.GROUPS);
        accountInfoBuilder.addAttributeInfo(AttributeInfoBuilder.build("firstName"));
        accountInfoBuilder.addAttributeInfo(AttributeInfoBuilder.define("lastName")
                .setRequired(true).build());
        builder.defineObjectClass(accountInfoBuilder.build());

        // Group
        ObjectClassInfoBuilder groupInfoBuilder = new ObjectClassInfoBuilder();
        groupInfoBuilder.setType(ObjectClass.GROUP_NAME);
        groupInfoBuilder.addAttributeInfo(Name.INFO);
        groupInfoBuilder.addAttributeInfo(PredefinedAttributeInfos.DESCRIPTION);
        groupInfoBuilder.addAttributeInfo(AttributeInfoBuilder.define("members").setCreatable(
                false).setUpdateable(false).setMultiValued(true).build());

        // Only the CRUD operations
        builder.defineObjectClass(groupInfoBuilder.build(), CreateOp.class, SearchOp.class,
                UpdateOp.class, DeleteOp.class);

        // Operation Options
        builder.defineOperationOption(OperationOptionInfoBuilder.buildAttributesToGet(),
                SearchOp.class);

        // Support paged Search
        builder.defineOperationOption(OperationOptionInfoBuilder.buildPageSize(),
                SearchOp.class);
        builder.defineOperationOption(OperationOptionInfoBuilder.buildPagedResultsCookie(),
                SearchOp.class);

        // Support to execute operation with provided credentials
        builder.defineOperationOption(OperationOptionInfoBuilder.buildRunWithUser());
        builder.defineOperationOption(OperationOptionInfoBuilder.buildRunWithPassword());

        schema = builder.build();
    }
    return schema;
}
```

Script on connector operation

The script on connector operation runs a script in the environment of the connector. This is different to the script on resource operation, which runs a script on the target resource that the connector manages.

The corresponding API operation (`scriptOnConnectorApiOp`) provides a minimum contract to which the connector must adhere. (Refer to the javadoc for more information). If you do not implement the `scriptOnConnector` interface in your connector, the framework provides a default implementation. If you intend your connector to provide more to the script than what is required by this minimum contract, you must implement the `scriptOnConnectorOp` interface.

Use the ICF script on connector operation

The API operation allows an application to run a script in the context of any connector.

This operation runs the script in the same JVM or .Net Runtime as the connector. That is, if you are using a local framework, the script runs in your JVM. If you are connected to a remote framework, the script runs in the remote JVM or .Net Runtime.

Consumption of the ScriptOnConnector operation, at the API Level

```
@Test
public void runScriptOnConnectorTest() {
    logger.info("Running RunScriptOnConnector Test");
    final ConnectorFacade facade = createConnectorFacade(BasicConnector.class, null);
    final OperationOptionsBuilder builder = new OperationOptionsBuilder();
    builder.setRunAsUser("admin");
    builder.setRunWithPassword(new GuardedString("Passw0rd".toCharArray()));

    final ScriptContextBuilder scriptBuilder =
            new ScriptContextBuilder("Groovy", "return argument");
    scriptBuilder.addScriptArgument("argument", "value");

    Object result = facade.runScriptOnConnector(scriptBuilder.build(), builder.build());
    Assert.assertEquals(result, "value");
}
```

Implement the script on connector operation

The `scriptOnConnector` SPI operation takes the following parameters:

- `request` - the script and the arguments to be run
- `options` - additional options that control how the script is run

The operation returns the result of the script. The return type must be a type that the framework supports for serialization. Refer to the ObjectSerializerFactory javadoc for a list of supported return types.

Implementation of the ScriptOnConnector operation, at the SPI Level

```
public Object runScriptOnConnector(ScriptContext request, OperationOptions options) {
    final ScriptExecutorFactory factory =
            ScriptExecutorFactory.newInstance(request.getScriptLanguage());
    final ScriptExecutor executor =
            factory.newScriptExecutor(getClass().getClassLoader(), request.getScriptText(),
                    true);

    if (StringUtil.isNotBlank(options.getRunAsUser())) {
        String password = SecurityUtil.decrypt(options.getRunWithPassword());
        // Use these to execute the script with these credentials
    }
    try {
        return executor.execute(request.getScriptArguments());
    } catch (Throwable e) {
        logger.warn(e, "Failed to execute Script");
        throw ConnectorException.wrap(e);
    }
}
```

Script on resource operation

The script on resource operation runs a script directly on the target resource (unlike the Script on connector operation, which runs a script in the context of a specific connector.)

Implement this interface if your connector intends to support the `ScriptOnResourceApiOp` API operation. If your connector implements this interface, you must document the script languages that the connector supports, as well as any supported `OperationOptions`.

Use the ICF script on resource operation

The contract at the API level is intentionally very loose. Each connector decides what script languages it supports, what running a script on a target resource actually means, and what script options (if any) the connector supports.

Consumption of the ScriptOnResource operation, at the API Level

```
@Test
public void runScriptOnResourceTest() {
    logger.info("Running RunScriptOnResource Test");
    final ConnectorFacade facade = createConnectorFacade(BasicConnector.class, null);
    final OperationOptionsBuilder builder = new OperationOptionsBuilder();
    builder.setRunAsUser("admin");
    builder.setRunWithPassword(new GuardedString("Passw0rd".toCharArray()));

    final ScriptContextBuilder scriptBuilder = new ScriptContextBuilder("bash", "whoami");

    Object result = facade.runScriptOnResource(scriptBuilder.build(), builder.build());
    Assert.assertEquals(result, "admin");
}
```

Implement the script on resource operation

The `scriptOnResource` SPI operation takes the following parameters:

- `request` - the script and the arguments to be run
- `options` - additional options that control how the script is run

The operation returns the result of the script. The return type must be a type that the framework supports for serialization. Refer to the ObjectSerializerFactory javadoc for a list of supported return types.

Implementation of the ScriptOnResource operation, at the SPI Level

```
public Object runScriptOnResource(ScriptContext request, OperationOptions options) {
    try {
        // Execute the script on remote resource
        if (StringUtil.isNotBlank(options.getRunAsUser())) {
            String password = SecurityUtil.decrypt(options.getRunWithPassword());
            // Use these to execute the script with these credentials
            return options.getRunAsUser();
        }
        throw new UnknownHostException("Failed to connect to remote SSH");
    } catch (Throwable e) {
        logger.warn(e, "Failed to execute Script");
        throw ConnectorException.wrap(e);
    }
}
```

Search operation

The search operation enables the connector to search for objects on the target system.

The ICF framework handles searches as follows:

1. The application sends a query, with a search filter, to the OpenICF framework.
2. The framework submits the query, with the filter, to the connector.
3. The connector implements the `createFilterTranslator()` method to obtain a `FilterTranslator` object.
4. The framework then uses this `FilterTranslator` object to transform the filter to a format that the `executeQuery()` method expects.

You can implement the `FilterTranslator` object in two ways:

- The `FilterTranslator` translates the original filter into one or more native queries.

  The framework then calls the `executeQuery()` method for each native query.

- The `FilterTranslator` does not modify the original filter.

  The framework then calls the `executeQuery()` method with the original ICF filter.

Using this second approach enables your connector to distinguish between a search and a get operation and to benefit from the visitor design pattern.

Based on the `resultsHandlerConfiguration`, the OpenICF framework can perform additional filtering on the returning results. For more information on the `resultsHandlerConfiguration`, refer to Configure How Results Are Handled.

The connector facade calls the `executeQuery` method once for each native query that the filter translator produces. If the filter translator produces more than one native query, the connector facade merges the results from each query and eliminates any duplicates.

Note that this implies an in-memory data structure that holds a set of UID values. Memory usage, in the event of multiple queries, will be O(N) where N is the number of results. It is therefore important that the filter translator for the connector implement `OR` operators, if possible.

Whether the application calls a `get` API operation, or a `search` API operation, the ICF framework translates that request to a `search` request on the connector.

Use the ICF get operation

The `GetApiOp` returns `null` when the UID does not exist on the resource.

Consumption of the Get operation, at the API Level

```
@Test
public void getObjectTest() {
    logger.info("Running GetObject Test");
    final ConnectorFacade facade = createConnectorFacade(BasicConnector.class, null);
    final OperationOptionsBuilder builder = new OperationOptionsBuilder();
    builder.setAttributesToGet(Name.NAME);
    ConnectorObject co =
            facade.getObject(ObjectClass.ACCOUNT, new Uid(
                    "3f50eca0-f5e9-11e3-a3ac-0800200c9a66"), builder.build());
    Assert.assertEquals(co.getName().getNameValue(), "Foo");
}
```

Use the ICF search operation

Consumption of the Search operation, at the API Level

```
@Test
public void searchTest() {
    logger.info("Running Search Test");
    final ConnectorFacade facade = createConnectorFacade(BasicConnector.class, null);
    final OperationOptionsBuilder builder = new OperationOptionsBuilder();
    builder.setPageSize(10);
    final ResultsHandler handler = new ToListResultsHandler();

    SearchResult result =
            facade.search(ObjectClass.ACCOUNT, FilterBuilder.equalTo(new Name("Foo")), handler,
                    builder.build());
    Assert.assertEquals(result.getPagedResultsCookie(), "0");
    Assert.assertEquals(((ToListResultsHandler) handler).getObjects().size(), 1);
}
```

Implement the search operation

Implementation of the Search operation, at the SPI Level

```
public FilterTranslator<String> createFilterTranslator(ObjectClass objectClass,
        OperationOptions options) {
    return new BasicFilterTranslator();
}

public void executeQuery(ObjectClass objectClass, String query, ResultsHandler handler,
        OperationOptions options) {
    final ConnectorObjectBuilder builder = new ConnectorObjectBuilder();
    builder.setUid("3f50eca0-f5e9-11e3-a3ac-0800200c9a66");
    builder.setName("Foo");
    builder.addAttribute(AttributeBuilder.buildEnabled(true));

    for (ConnectorObject connectorObject : CollectionUtil.newSet(builder.build())) {
        if (!handler.handle(connectorObject)) {
            // Stop iterating because the handler stopped processing
            break;
        }
    }
    if (options.getPageSize() != null && 0 < options.getPageSize()) {
        logger.info("Paged Search was requested");
        ((SearchResultsHandler) handler).handleResult(new SearchResult("0", 0));
    }
}
```

Sync operation

The sync operation polls the target system for synchronization events, that is, native changes to target objects.

The operation has two methods:

- `sync()` - request synchronization events from the target system

  This method calls the specified handler, once, to pass back each matching synchronization event. When the method returns, it will no longer invoke the specified handler.

- `getLatestSyncToken()` - returns the token corresponding to the most recent synchronization event

Use the ICF sync operation

Consumption of the Sync Operation(`getLatestSyncToken()` Method), at the API Level

```
@Test
public void getLatestSyncTokenTest() {
    logger.info("Running GetLatestSyncToken Test");
    final ConnectorFacade facade = createConnectorFacade(BasicConnector.class, null);
    SyncToken token = facade.getLatestSyncToken(ObjectClass.ACCOUNT);
    Assert.assertEquals(token.getValue(), 10);
}
```

The `getLatestSyncToken` method throws an `IllegalArgumentException` if the `objectClass` is null or invalid.

Consumption of the Sync Operation (`sync()`Method), at the API Level

```
@Test
public void syncTest() {
    logger.info("Running Sync Test");
    final ConnectorFacade facade = createConnectorFacade(BasicConnector.class, null);
    final OperationOptionsBuilder builder = new OperationOptionsBuilder();
    builder.setPageSize(10);
    final SyncResultsHandler handler = new SyncResultsHandler() {
        public boolean handle(SyncDelta delta) {
            return false;
        }
    };

    SyncToken token =
            facade.sync(ObjectClass.ACCOUNT, new SyncToken(10), handler, builder.build());
    Assert.assertEquals(token.getValue(), 10);
}
```

The `sync` method throws an `IllegalArgumentException` if the `objectClass` or `handler` is null, or if any argument is invalid.

Implement the sync operation

Implementation of the Sync Operation at the SPI Level

```
public void sync(ObjectClass objectClass, SyncToken token, SyncResultsHandler handler,
        final OperationOptions options) {
    if (ObjectClass.ALL.equals(objectClass)) {
        //
    } else if (ObjectClass.ACCOUNT.equals(objectClass)) {
        final ConnectorObjectBuilder builder = new ConnectorObjectBuilder();
        builder.setUid("3f50eca0-f5e9-11e3-a3ac-0800200c9a66");
        builder.setName("Foo");
        builder.addAttribute(AttributeBuilder.buildEnabled(true));

        final SyncDeltaBuilder deltaBuilder = new SyncDeltaBuilder();
        deltaBuilder.setObject(builder.build());
        deltaBuilder.setDeltaType(SyncDeltaType.CREATE);
        deltaBuilder.setToken(new SyncToken(10));

        for (SyncDelta connectorObject : CollectionUtil.newSet(deltaBuilder.build())) {
            if (!handler.handle(connectorObject)) {
                // Stop iterating because the handler stopped processing
                break;
            }
        }
    } else {
        logger.warn("Sync of type {0} is not supported", configuration.getConnectorMessages()
                .format(objectClass.getDisplayNameKey(), objectClass.getObjectClassValue()));
        throw new UnsupportedOperationException("Sync of type"
                + objectClass.getObjectClassValue() + " is not supported");
    }
    ((SyncTokenResultsHandler) handler).handleResult(new SyncToken(10));
}

public SyncToken getLatestSyncToken(ObjectClass objectClass) {
    if (ObjectClass.ACCOUNT.equals(objectClass)) {
        return new SyncToken(10);
    } else {
        logger.warn("Sync of type {0} is not supported", configuration.getConnectorMessages()
                .format(objectClass.getDisplayNameKey(), objectClass.getObjectClassValue()));
        throw new UnsupportedOperationException("Sync of type"
                + objectClass.getObjectClassValue() + " is not supported");
    }
}
```

Test operation

The test operation tests the connector configuration. Unlike validation, testing a configuration verifies that every part of the environment that is referred to by the configuration is available. The operation therefore validates that the connection details that are provided in the configuration are accurate, and that the backend is accessible when using them.

For example, the connector might make a physical connection to the host that is specified in the configuration, to check that it exists and that the credentials supplied in the configuration are valid.

The test operation can be invoked before the configuration has been validated, or can validate the configuration before testing it.

Use the ICF test operation

At the API level, the test operation throws a `RuntimeException` if the configuration is not valid, or if the test fails. Your connector implementation should throw the most specific exception available. When no specific exception is available, your connector implementation should throw a `ConnectorException`.

Consumption of the Test Operation at the API Level

```
@Test
public void testTest() {
    logger.info("Running Test Test");
    final ConnectorFacade facade = createConnectorFacade(BasicConnector.class, null);
    facade.test();
}
```

Implement the test operation

Implementation of the Test Operation at the SPI Level

```
public void test() {
    logger.ok("Test works well");
}
```

Update operation

If your connector will allow an authorized caller to update (modify or replace) objects on the target system, you must implement either the update operation, or the Update attribute values operation. At the API level, update operation calls either the `UpdateOp` or the `UpdateAttributeValuesOp`, depending on what you have implemented.

The update operation is somewhat simpler to implement than the Update attribute values operation, because the update attribute values operation must handle any type of update that the caller might specify. However a true implementation of the update attribute values operation offers better performance and atomicity semantics.

Use the ICF update operation

At the API level, the update operation returns an `UnknownUidException` if the UID does not exist on the target system resource and if the connector does not implement the Update attribute values operation interface.

Consumption of the Update Operation at the API Level

```
@Test
public void updateTest() {
    logger.info("Running Update Test");
    final ConnectorFacade facade = createConnectorFacade(BasicConnector.class, null);
    final OperationOptionsBuilder builder = new OperationOptionsBuilder();
    Set<Attribute> updateAttributes = new HashSet<Attribute>();
    updateAttributes.add(new Name("Foo"));

    Uid uid = facade.update(ObjectClass.ACCOUNT, new Uid("Foo"), updateAttributes, builder
                  .build());
    Assert.assertEquals(uid.getUidValue(), "foo");
}
```

Implement the update operation

At the SPI level, the update operation returns an `UnknownUidException` if the UID does not exist on the target system.

Implementation of the Update Operation at the SPI Level

```
public Uid update(ObjectClass objectClass, Uid uid, Set<Attribute> replaceAttributes,
        OperationOptions options) {
    AttributesAccessor attributesAccessor = new AttributesAccessor(replaceAttributes);
    Name newName = attributesAccessor.getName();
    Uid uidAfterUpdate = uid;
    if (newName != null) {
        logger.info("Rename the object {0}:{1} to {2}", objectClass.getObjectClassValue(), uid
                .getUidValue(), newName.getNameValue());
        uidAfterUpdate = new Uid(newName.getNameValue().toLowerCase());
    }

    if (ObjectClass.ACCOUNT.equals(objectClass)) {

    } else if (ObjectClass.GROUP.is(objectClass.getObjectClassValue())) {
        if (attributesAccessor.hasAttribute("members")) {
            throw new InvalidAttributeValueException(
                    "Requested to update a read only attribute");
        }
    } else {
        logger.warn("Update of type {0} is not supported", configuration.getConnectorMessages()
                .format(objectClass.getDisplayNameKey(), objectClass.getObjectClassValue()));
        throw new UnsupportedOperationException("Update of type"
                + objectClass.getObjectClassValue() + " is not supported");
    }
    return uidAfterUpdate;
}
```

Approaches for deleting attributes and attribute values

If the target resource to which you are connecting supports the removal of attributes, you can implement the removal in several ways. All the samples in this document assume the following syntax rules for deleting attributes or removing their values.

Update	Syntax rule	Query filter
Set an empty attribute value	`[""]` (application sends an attribute value that is a list containing one empty string)	`equal=""`
Set an attribute value to null	`[]` (application sends an attribute value that is an empty list)	`ispresent` search returns 1
Removing an attribute	`null` (application sends an attribute value that is `null`	`ispresent` search returns 1

Update attribute values operation

The update attribute values operation is an advanced implementation of the update operation. You should implement this operation if you want your connector to offer better performance and atomicity for the following methods:

- `UpdateApiOp.addAttributeValues(ObjectClass, Uid, Set, OperationOptions)`
- `UpdateApiOp.removeAttributeValues(ObjectClass, Uid, Set, OperationOptions)`

Consumption of the Add and Remove Attribute Values Methods at the API Level

```
@Test
public void addAttributeValuesTest() {
    logger.info("Running AddAttributeValues Test");
    final ConnectorFacade facade = createConnectorFacade(BasicConnector.class, null);
    final OperationOptionsBuilder builder = new OperationOptionsBuilder();
    Set<Attribute> updateAttributes = new HashSet<Attribute>();
    // add 'group2' to existing groups
    updateAttributes.add(AttributeBuilder.build(PredefinedAttributes.GROUPS_NAME, "group2"));

    Uid uid =
            facade.addAttributeValues(ObjectClass.ACCOUNT, new Uid("Foo"), updateAttributes,
                    builder.build());
    Assert.assertEquals(uid.getUidValue(), "foo");
}

@Test
public void removeAttributeValuesTest() {
    logger.info("Running RemoveAttributeValues Test");
    final ConnectorFacade facade = createConnectorFacade(BasicConnector.class, null);
    final OperationOptionsBuilder builder = new OperationOptionsBuilder();
    Set<Attribute> updateAttributes = new HashSet<Attribute>();
    // remove 'group2' from existing groups
    updateAttributes.add(AttributeBuilder.build(PredefinedAttributes.GROUPS_NAME, "group2"));

    Uid uid =
            facade.removeAttributeValues(ObjectClass.ACCOUNT, new Uid("Foo"), updateAttributes,
                    builder.build());
    Assert.assertEquals(uid.getUidValue(), "foo");
}
```

Implement the update attribute values operation

At the SPI level, the update attribute values operation returns an `UnknownUidException` when the UID does not exist on the resource.

Implementation of the update attribute values operation, at the SPILevel

```
public Uid addAttributeValues(ObjectClass objectClass, Uid uid, Set<Attribute> valuesToAdd,
        OperationOptions options) {
    return uid;
}

public Uid removeAttributeValues(ObjectClass objectClass, Uid uid,
        Set<Attribute> valuesToRemove, OperationOptions options) {
    return uid;
}
```

Common exceptions

The following sections describe the commonly used exceptions that can be thrown, depending on the operation.

AlreadyExistsException

The `AlreadyExistsException` is thrown if a create operation attempts to create an object that exists prior to the method execution, or if an update operation attempts to rename an object to that exists prior to the method execution.

ConfigurationException

A `ConfigurationException` is thrown if a configuration problem is encountered when the connector bundles are loaded. A `ConfigurationException` can also be thrown during validation operations in the SPI.

ConnectionBrokenException

A `ConnectionBrokenException` is thrown when a connection to a target resource instance fails during an operation. An instance of the `ConnectionBrokenException` generally wraps the native exception (or describes the native error) that is returned by the target resource.

ConnectionFailedException

A `ConnectionFailedException` is thrown when a connector cannot reach the target resource. An instance of the `ConnectionFailedException` generally wraps the native exception (or describes the native error) that is returned by the target resource.

ConnectorException

This is the base exception for the connector framework. The framework only throws exceptions that extend `ConnectorException`.

ConnectorIOException

This is the base exception for all Input-Output (I/O-related) exceptions, including instance connection failure, socket error, and so forth.

ConnectorSecurityException

This is the base exception for all security-related exceptions.

InvalidAttributeValueException

An `InvalidAttributeValueException` is thrown when an attempt is made to add to an attribute a value that conflicts with the attribute’s schema definition. This might happen, for example, in the following situations:

- The connector attempts to add an attribute with no value when the attribute is required to have at least one value.
- The connector attempts to add more than one value to a single valued-attribute.
- The connector attempts to add a value that conflicts with the attribute type.
- The connector attempts to add a value that conflicts with the attribute syntax.

InvalidCredentialException

An `InvalidCredentialException` indicates that user authentication has failed. This exception is thrown by the connector when authentication fails, and when the specific reason for the failure is not known. For example, the connector might throw this exception if a user has entered an incorrect password, or username.

InvalidPasswordException

An `InvalidPasswordException` is thrown when a password credential is invalid.

OperationTimeoutException

An `OperationTimeoutException` is thrown when an operation times out. The framework cancels an operation when the corresponding method has been executing for longer than the limit specified in `APIConfiguration`.

PasswordExpiredException

A `PasswordExpiredException` indicates that a user password has expired. This exception is thrown by the connector when it can determine that a password has expired. For example, after successfully authenticating a user, the connector might determine that the user's password has expired. The connector throws this exception to notify the application, which can then take the appropriate steps to notify the user.

PermissionDeniedException

A `PermissionDeniedException` is thrown when the target resource will not allow a connector to perform a particular operation. An instance of the `PermissionDeniedException` generally describes a native error (or wraps a native exception) that is returned by the target resource.

PreconditionFailedException

A `PreconditionFailedException` is thrown to indicate that a resource's current version does not match the version provided. This exception is equivalent to the HTTP status: `412 Precondition Failed`.

PreconditionRequiredException

A `PreconditionRequiredException` is thrown to indicate that a resource requires a version, but that no version was supplied in the request. This exception is equivalent to the HTTP status: `428 Precondition Required`.

RetryableException

A `RetryableException` indicates that the failure might be temporary, and that retrying the same request might succeed in the future.

UnknownUidException

An `UnknownUidException` is thrown when a UID that is specified as input to a connector operation identifies no object on the target resource. When you implement the `AuthenticateOp`, your connector can throw this exception if it is unable to locate the account necessary to perform authentication.

NullPointerException (c# NullReferenceException)

Generic native exception

UnsupportedOperationException (c# NotSupportedException)

Generic native exception

IllegalStateException (c# InvalidOperationException)

Generic native exception

IllegalArgumentException (c# ArgumentException)

Generic native exception

Mapping ICF Exceptions to Common REST Exceptions

The following table maps the errors that are thrown by the OpenICF framework to the errors that are returned by the Common REST implementation.

ICF Exception	Common REST Exception
AlreadyExistsException	ConflictException
ConfigurationException	InternalServerErrorException
ConnectionBrokenException	InternalServerErrorException
ConnectionFailedException	ConnectionFailedException
ConnectorException	InternalServerErrorException
ConnectorIOException	InternalServerErrorException
ConnectorSecurityException	ForbiddenException
InvalidAttributeValueException	BadRequestException
InvalidCredentialException	ForbiddenException
InvalidPasswordException	ForbiddenException
OperationTimeoutException	
PasswordExpiredException	ForbiddenException
PermissionDeniedException	ForbiddenException
PreconditionFailedException	PreconditionFailedException
PreconditionRequiredException	PreconditionRequiredException
RetryableException	RetryableException (ServiceUnavailableException)

ICF Exception	Common REST Exception
UnknownUidException	NotFoundException
UnsupportedOperationException	NotSupportedException
IllegalArgumentException	InternalServerErrorException
NullPointerException	InternalServerErrorException

Generic exception rules

The generic exception rules are common to all API- or SPI-level operations and are described in the following sections.

Framework (API-level) exception rules

IllegalArgumentException or NullPointerException

Thrown when the ObjectClass is null or the name is blank.

OperationTimeoutException

Thrown when the operation timed out.

ConnectionFailedException

Thrown if any problem occurs with the connector server connection.

UnsupportedOperationException

Thrown if the connector does not implement the required interface.

ConnectorIOException

Thrown if the connector failed to initialize a remote connection due to a SocketException.

ConnectorException

Thrown in the following situations:

- The connector failed to initiate the remote connection due to a `SocketException`.
- An unexpected request was sent to the remote connector server.
- An unexpected response was received from the remote connector server.

InvalidCredentialException

Thrown if the remote framework key is invalid.

The following exceptions are thrown specifically in the context of a poolable connector.

ConnectorException

Thrown if the pool has no available connectors after the `maxWait` time has elapsed.

IllegalStateException

Thrown if the object pool has already shut down.

Connector (SPI Level) Exception Rules

InvalidAttributeValueException

Thrown when single-valued attribute has multiple values.

IllegalArgumentException

Thrown when the value of the `__PASSWORD__` or the `__CURRENT_PASSWORD__` attribute is not a `GuardedString`.

IllegalStateException

Thrown when the `Attribute` name is blank.

PermissionDeniedException

Thrown when the target resource will not allow a specific operation to be performed. An instance of the `PermissionDeniedException` generally describes a native error that is returned by (or wraps a native exception that is thrown by) the target resource.

ConnectorIOException, ConnectionBrokenException, ConnectionFailedException

Thrown when any problem occurs with the connection to the target resource.

PreconditionFailedException

Thrown when the current version of the resource object does not match the version provided by the connector.

PreconditionRequiredException

Thrown when a resource object requires a version, but no version was supplied in the `getRevision` operation.

Java connectors

If none of the existing ICF connectors are suitable for your deployment, you can write your own connector. This chapter describes the steps to develop an OpenICF-compatible Java connector. Similar chapters exist to help you with writing scripted Groovy, and PowerShell connectors.

Types of connectors

In general, it takes longer to write a new Java connector than it does to use one of the scripted connector toolkits to write a scripted connector. Before you can write a Java connector, you must have a good understanding of the ICF SPI (refer to OpenICF SPI).

Scripted connectors do not require a complete understanding of the SPI, so connector development should be faster. The scripted connector implementations provided with IDM follow a general *pattern* and you can assess which implementation to use based on what the connector must be able to do.

For example, if you need to connect to a database, use the scriptedSQL implementation. To execute a remote command over SSH, use the scriptedSSH implementation. The details of these different scripted connector types are described in Selecting a Scripted Connector Implementation.

If the main purpose of your connector is to call a number of stored procedures or perform some SQL inserts, you can avoid learning the OpenICF SPI and focus on the required "actions" (create, delete, update, and so on). You can then implement these actions in a scripted connector. When you have stable scripts that do what they need to do, package them in a JAR, version them and your connector development is complete.

If you need to connect to *new* system with a client/server API in written in Java, you must write a new Java connector. This chapter helps you get started with that process.

Before you begin

Before you start developing your own connector, familiarize yourself with the structure of the SPI, by reading OpenICF SPI and the corresponding Javadoc for the ICF framework and its supported operations.

Connector archetype

ICF provides a Maven connector archetype that lets you get started with connector development.

The connector archetype assumes that you have Apache Maven installed on your system. Before you use the connector archetype, add the following to your Maven `settings.xml` file, replacing `backstage-username` and `backstage-password` with your Backstage credentials:

```
<servers>
 ...
 <server>
    <username>backstage-username</username>
    <password>backstage-password</password>
    <id>archetype</id>
  </server>
</servers>
...
<profiles>
  <profile>
    <id>test</id>
    <activation>
      <activeByDefault>true</activeByDefault>
    </activation>
    <repositories>
      <repository>
         <id>archetype</id>
         <url>https://maven.forgerock.org/artifactory/private-releases</url>
       </repository>
    </repositories>
  </profile>
</profiles>
```

To start building a connector by using the connector archetype, execute the following command, customizing these options to describe your new connector:

- `-DartifactId=sample-connector`
- `-Dversion=0.0-SNAPSHOT`
- `-Dpackage=org.forgerock.openicf.connectors.sample`
- `-DconnectorName=Sample`

This command imports the connector archetype and generates a new connector project:

```
mvn archetype:generate \
 -DarchetypeGroupId=org.forgerock.openicf \
 -DarchetypeArtifactId=connector-archetype \
 -DarchetypeVersion=1.4.0 \
 -DremoteRepositories=https://maven.forgerock.org/artifactory/private-releases \
 -DarchetypeRepository=https://maven.forgerock.org/artifactory/private-releases \
 -DgroupId=org.forgerock.openicf.connectors \
 -DartifactId=sample-connector \
 -Dversion=0.0-SNAPSHOT \
 -Dpackage=org.forgerock.openicf.connectors.sample \
 -DconnectorName=Sample
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building Maven Stub Project (No POM) 1
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] >>> maven-archetype-plugin:3.0.1:generate (default-cli) > generate-sources @ standalone-pom >>>
[INFO]
[INFO] <<< maven-archetype-plugin:3.0.1:generate (default-cli) < generate-sources @ standalone-pom <<<
[INFO]
[INFO] --- maven-archetype-plugin:3.0.1:generate (default-cli) @ standalone-pom ---
[INFO] Generating project in Interactive mode
...
ALL_OPERATIONS: n
OP_AUTHENTICATE: n
OP_CREATE: y
OP_DELETE: y
OP_RESOLVEUSERNAME: n
OP_SCHEMA: n
OP_SCRIPTONCONNECTOR: n
OP_SCRIPTONRESOURCE: n
OP_SEARCH: y
OP_SYNC: n
OP_TEST: y
OP_UPDATE: y
OP_UPDATEATTRIBUTEVALUES: n
attributeNormalizer: n
compatibility_version: 1.1
connectorName: Sample
framework_version: 1.0
jira_componentId: 10191
jira_fixVersionIds: 0
poolableConnector: n
 Y: :
```

At this point, you can enter Y (YES) to accept the default project, or N (NO) to customize the project for your connector.

You will notice in the preceding output that the default connector supports only the `create`, `delete`, `search`, `test`, and `update` operations, and is not a poolable connector. To add support for additional operations, or to change any of the connector parameters, enter N (NO). The archetype then prompts you to set values for each additional parameter.

After you have imported the archetype once, you can use the local version of the archetype, as follows:

```
mvn archetype:generate -DarchetypeCatalog=local
```

Implement ICF operations

When you have generated the archetype, implement the ICF operations that your connector will support.

For information about implementing operations, and examples for a Java connector, refer to OpenICF SPI.

Build the Java connector

To build the connector, run:

```
cd /path/to/sample-connector/
mvn install
```

Scripted connectors with Groovy

The Groovy connector toolkit lets you run Groovy scripts to interact with any external resource.

The Groovy connector toolkit is not a complete connector in the traditional sense. Instead, it is a framework where you must write your own Groovy scripts to address your deployment requirements. The toolkit is bundled with IDM in the JAR `openidm/connectors/groovy-connector-1.5.20.30.jar`.

IDM provides a number of deployment-specific scripts to help you get started with the Groovy connector toolkit. These scripts demonstrate how the toolkit can be used. The scripts cannot be used "as is" in your deployment but can be used as a starting point to base your customization.

The Groovy connector toolkit can be used with any ICF-enabled project (that is, any project where the OpenICF is installed).

About the Groovy scripting language

Groovy is a powerful, convenient scripting language for the Java platform. Groovy lets you take advantage of existing Java resources and generally makes development quicker. Syntactically, Groovy is similar to JavaScript. Extensive information about Groovy is available on the Groovy documentation site.

Selecting a scripted connector implementation

The Groovy connector toolkit provides five default connector implementations. The default implementations address the requirements of most target resources. If you use one of the default implementations, you only need to write the accompanying scripts and point your connector to their location. If the default implementations do not cover your target resource, you can use the Maven archetype to create a new connector project and write a custom Groovy-based connector from scratch.

The following list describes the default scripted connector implementations provided with the Groovy connector toolkit:

- `GROOVY` - a basic non-pooled Groovy connector, provided in the `org.forgerock.openicf.connectors.groovy.ScriptedConnector` class.

`POOLABLEGROOVY` - a poolable Groovy connector, provided in the `org.forgerock.openicf.connectors.groovy.ScriptedPoolableConnector` class.

When you have selected a scripted connector implementation, write the required scripts that correspond to that connector type. ICF operations with Groovy scripts provides information and examples on how to write scripts for the basic scripted connector implementation, and information on the extensions available for the other implementations.

ICF operations with Groovy scripts

The Groovy connector toolkit lets you run a Groovy script for any ICF operation, such as search, update, create, and so forth, on any external resource.

You must write a Groovy script that corresponds to each operation that your connector will support. For information about all the operations that are supported by the ICF framework, refer to OpenICF SPI.

Your scripted connector can implement the following ICF interfaces:

Authenticate operation

Provides simple authentication with two parameters, presumed to be a user name and password.

Create operation

Creates an object and its `uid`.

Delete operation

Deletes an object, referenced by its `uid`.

Resolve username operation

Resolves an object to its `uid` based on its `username`.

Schema operation

Describes the object types, operations, and options that the connector supports.

Script on connector operation

Enables IDM 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 IDM.

Script on resource operation

Runs a script directly on the target resource that is managed by the connector.

Search operation

Searches the target resource for all objects that match the specified object class and filter.

Sync operation

Polls the target resource for synchronization events; that is, native changes to objects on the target resource.

Test operation

Tests the connector configuration. Testing a configuration checks that 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 the 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 operation

Updates (modifies or replaces) objects on a target resource.

The following sections provide more information and pointers to sample scripts for all the operations implemented in the Groovy connector toolkit.

Variables available to all Groovy scripts

The following variables are available to all scripts used by the Groovy Connector. Additional variables are available to specific scripts, as described in the sections that follow:

`configuration`

A handle to the connector's configuration object is injected into all scripts.

`operation`

The connector injects the name of the action or operation into the script, to indicate which action is being called.

The sample scripts for the Groovy connector define one script file per action. You can use a single file, or amalgamate multiple actions into one file. For example, the CREATE and UPDATE operations often share the same code.

The operation type can be one of the following:

- `ADD_ATTRIBUTE_VALUES`
- `AUTHENTICATE`
- `CREATE`
- `DELETE`
- `GET_LATEST_SYNC_TOKEN`

- `REMOVE_ATTRIBUTE_VALUES`
- `RESOLVE_USERNAME`
- `RUNSCRIPTONCONNECTOR`
- `RUNSCRIPTONRESOURCE`
- `SCHEMA`
- `SEARCH`
- `SYNC`
- `TEST`
- `UPDATE`

`options`

The ICF framework passes an `OperationOptions` object to most of the operations. The Groovy connector injects this object, as is, into the scripts. For example, the search, query, and sync operations pass the attributes to get as an operation option.

The most common options are as follows:

- `AttributesToGet` (String[]) for search and sync operations
- `RunAsUser` (String) for any operation
- `RunWithPassword` (GuardedString) for any operation
- `PagedResultsCookie` (String) for search operations
- `PagedResultsOffset` (Int) for search operations
- `PageSize` (Int) for search operations
- `SortKeys` (Sortkey[]) for search operations

objectClass

The category or type of object that is managed by the connector, such as ACCOUNT and GROUP.

log

A handle to the default ICF logging facility.

connection

Available to the ScriptedREST, ScriptedCREST, and ScriptedSQL implementations, this variable initiates the HTTP or SQL connection to the resource.

Authenticate script

An authenticate script is *required* if you want to use pass-through authentication to the backend resource. If your connector does not need to authenticate to the resource, the authenticate script should allow the `authId` to pass through by default.

A sample authenticate script for an SQL database is provided in `openidm/samples/scripted-sql-with-mysql/tools/AuthenticateScript.groovy`.

The following connectors support pass-through authentication using the `AuthenticateOp` interface by default:

- LDAP connector
- CSV file connector
- Database Table connector
- Microsoft Graph API connector
- Scripted SQL connector

> **Note**
>
> All Scripted Groovy-based connectors are capable of pass-through authentication if the `AuthenticateScript.groovy` script is implemented, but the only default implementation is the ScriptedSQL connector.

Input variables

The following variables are available to the authenticate script:

configuration

A handler to the connector's configuration object.

options

A handler to the Operation Options.

operation

An OperationType that corresponds to the action (`AUTHENTICATE`).

objectClass

The object class being used to authenticate, such as `__ACCOUNT__` or `__GROUP__`.

username

A string that provides the username to authenticate.

password

A guarded string that provides the password with which to authenticate.

log

A logger instance for the connector.

Returns

The user unique ID (ICF `__UID__`). The `type` of the returned UID must be a `string` or a `Uid`. The script must throw an exception in the case of failure.

Authenticate Script

```
def operation = operation as OperationType
def configuration = configuration as ScriptedConfiguration
def username = username as String
def log = log as Log
def objectClass = objectClass as ObjectClass
def options = options as OperationOptions
def password = password as GuardedString;

if (username.equals("TEST")) {
    def clearPassword = SecurityUtil.decrypt(password)
    if ("Passw0rd".equals(clearPassword)) {
        return new Uid(username);
    }
}
```

Test script

A test script tests the connection to the external resource to ensure that the other operations that are provided by the connector can succeed.

A sample test script for an SQL database is provided in `openidm/samples/scripted-sql-with-mysql/tools/TestScript.groovy`.

Input variables

The following variables are available to the test script:

configuration

A handler to the connector's configuration object.

operation

An OperationType that corresponds to the action (`TEST`).

log

A logger instance for the connector.

Returns

Nothing, if the test is successful. The script can throw any exception if it fails.

Test Script

```
import org.identityconnectors.common.logging.Log
import org.forgerock.openicf.connectors.groovy.OperationType
import org.forgerock.openicf.misc.scriptedcommon.ScriptedConfiguration

def operation = operation as OperationType
def configuration = configuration as ScriptedConfiguration
def log = log as Log

log.info("This is a TestScript")
throw new MissingResourceException("Test Failed", operation.name(), "")
```

Create script

A create script creates a new object on the external resource. If your connector does not support creating an object, this script should throw an `UnsupportedOperationException`.

A sample create script for an SQL database is provided in `openidm/samples/scripted-sql-with-mysql/tools/CreateScript.groovy`.

Input variables

The following variables are available to a create script:

configuration

A handler to the connector’s configuration object.

options

A handler to the Operation Options.

operation

An OperationType that corresponds to the action (`CREATE`).

objectClass

The object class that is created, such as `__ACCOUNT__` or `__GROUP__`.

attributes

The set of attributes that describe the object to be created.

id

The UID of the object to be created, if specified. If the UID is `null`, the UID should be generated by the server. The UID corresponds to the ICF `__NAME__` attribute if it is provided as part of the attribute set.

log

A logger instance for the connector.

Returns

The user unique ID (ICF `__UID__`) of the newly created object. The `type` of the returned UID must be a `string` or a `Uid`. If a null value or an object type other than `string` or `Uid` is returned, the script must throw an exception.

Create Script

```
def operation = operation as OperationType
 def configuration = configuration as SapConfiguration
 def log = log as Log
 def objectClass = objectClass as ObjectClass
 def createAttributes = new AttributesAccessor(attributes as Set<Attribute>)
 def name = id as String
 def options = options as OperationOptions

 log.info("Entering {0} script",operation);


 assert operation == OperationType.CREATE, 'Operation must be a CREATE'
 // We only deal with users
 assert objectClass.getObjectClassValue() == ObjectClass.ACCOUNT_NAME


 def password = createAttributes.getPassword() as GuardedString;
 assert password != null, 'Password must be provided on create'


 //...
 def uid = createTheUser(createAttributes);
 return uid
```

Search or query script

A search script searches for one or more objects on the external resource. Connectors that do not support searches should throw an `UnsupportedOperationException`.

A sample search script for an SQL database is provided in `openidm/samples/scripted-sql-with-mysql/tools/SearchScript.groovy`.

Input variables

The following variables are available to the search script:

configuration

A handler to the connector's configuration object.

options

A handler to the Operation Options.

operation

An OperationType that corresponds to the action (`SEARCH`).

objectClass

The object class to search, such as `__ACCOUNT__` or `__GROUP__` .

filter

The ICF native Query filter for this operation.

query

A Map representation of the native Query filter that is easy to process.

Provides a convenient way to access the query filter parameter. For example:

```
query = [ operation: "CONTAINS", left: attribute, right: "value", not: true/false ]
 query = [ operation: "ENDSWITH", left: attribute, right: "value", not: true/false ]
 query = [ operation: "STARTSWITH", left: attribute, right: "value", not: true/false ]
 query = [ operation: "EQUALS", left: attribute, right: "value", not: true/false ]
 query = [ operation: "GREATERTHAN", left: attribute, right: "value", not: true/false ]
 query = [ operation: "GREATERTHANOREQUAL", left: attribute, right: "value", not: true/false ]
 query = [ operation: "LESSTHAN", left: attribute, right: "value", not: true/false ]
 query = [ operation: "LESSTHANOREQUAL", left: attribute, right: "value", not: true/false ]
 query = null : then we assume we fetch everything

 // AND and OR filter - embed these left/right queries:
 query = [ operation: "AND", left: query1, right: query2 ]
 query = [ operation: "OR", left: query1, right: query2 ]
```

For example, the equality query filter `"sn == Smith"` would be represented by the following query Map:

```
[ operation: "EQUALS", left: "sn", right: "Smith", not: false ]
```

handler

A Closure handler for processing the search results.

log

A logger instance for the connector.

Returns

Optionally, the script can return a search result. The result can be be returned as a `SearchResult` object or as a `String` that represents the `pagedResultsCookie` to be used for the next paged results.

Returning search results

In a search operation, a result handler (callback) is passed to the script to return the results one by one. The handler must be called for every query result. The handler variable that is passed to the script is a `Groovy Closure`. You can call the handler in the following ways:

- Using an ICF `ConnectorObject` object.

  You can use the `ConnectorObjectBuilder` to build this object. For example:

  ```
  def builder = new ConnectorObjectBuilder()
   builder.setUid(uidValue)
   builder.setName(nameValue)
   builder.setObjectClass(ObjectClass.ACCOUNT)
   builder.addAttribute("sn", snValue)

   // Call the handler with the ConnectorObject object
   handler builder.build()
  ```

- Using a Groovy Closure.

  In this case the Closure delegates calls to a specific Object that implements these calls. For example:

  ```
  handler {           // The handler parameter here is a Closure
       uid uidValue    // (mandatory), the method resolution for 'uid' is delegated to the Object
                       // handling the Closure. This is the ICF __UID__
       id nameValue    // (mandatory), the method resolution for 'id' is delegated to the Object
                       // handling the Closure. This is the ICF __NAME__
       attribute "sn", snValue // (optional), the method resolution for 'id' is delegated to the
                              //  Object handling the Closure
       // attribute <attribute2Name>, <attribute2Value>
       // etc...
   }
  ```

  In the following example, the handler is called within a loop to return all the results of a query:

  ```
   for (user in userList) {
       handler {
                   uid user.userName
                   id user.userName
                   user.attributes.each(){ key,value -> attribute key, value }
               }
   }
  ```

Update script

An update script updates an object in the external resource. Connectors that do not support update operations should throw an `UnsupportedOperationException`.

A sample update script for an SQL database is provided in `openidm/samples/scripted-sql-with-mysql/tools/UpdateScript.groovy`.

Input variables

The following variables are available to an update script:

configuration

A handler to the connector's configuration object.

options

A handler to the Operation Options.

operation

An OperationType that corresponds to the action (`UPDATE`).

objectClass

The object class that is updated, such as `__ACCOUNT__` or `__GROUP__` .

attributes

A collection of `ConnectorAttributes` that represent the entry attributes to update.

uid

The UID of the object to be updated. The UID corresponds to the OpenICF `UID` attribute.

id

The name of the object to be updated (optional). The id corresponds to the ICF `__NAME__` attribute. It will not be injected and set unless the update is a rename.

log

A logger instance for the connector.

Returns

The user unique ID (ICF `__UID__`) of the updated object. The `type` of the returned UID must be a `string` or a `Uid` . If the UID is not modified by the update operation, return the value of the uid injected into the script.

Update Script

```
def operation = operation as OperationType
def updateAttributes = attributes as Set<Attribute>
def configuration = configuration as ScriptedConfiguration
def id = id as String
def log = log as Log
def objectClass = objectClass as ObjectClass
def options = options as OperationOptions
def uid = uid as Uid

log.ok("Update...")
switch (operation) {
    case OperationType.UPDATE:
        switch (objectClass) {
            case ObjectClass.ACCOUNT:
// ...
                    for (Attribute a : updateAttributes) {
                        if (a.is(Name.NAME)) {
// ...
return uid
```

Delete script

A delete script deletes an object in the external resource. Connectors that do not support delete operations should throw an `UnsupportedOperationException`.

A sample delete script for an SQL database is provided in `openidm/samples/scripted-sql-with-mysql/tools/DeleteScript.groovy`.

Input variables

The following variables are available to an update script:

configuration

A handler to the connector’s configuration object.

options

A handler to the Operation Options.

operation

An OperationType that corresponds to the action (`DELETE`).

objectClass

The object class that is deleted, such as `__ACCOUNT__` or `__GROUP__`.

uid

The UID of the object to be deleted. The UID corresponds to the OpenICF `__UID__` attribute.

log

A logger instance for the connector.

Returns

This script has no return value but should throw an exception if the delete is unsuccessful.

Synchronization script

A synchronization script synchronizes objects between two resources. The script should retrieve all objects in the external resource that have been updated since some defined token.

A sample synchronization script for an SQL database is provided in `openidm/samples/scripted-sql-with-mysql/tools/SyncScript.groovy`.

Input variables

The following variables are available to a sync script:

configuration

A handler to the connector's configuration object.

options

A handler to the Operation Options.

operation

An OperationType that corresponds to the action (`GET_LATEST_SYNC_TOKEN` or `SYNC`).

objectClass

The object class that is synchronized, such as `__ACCOUNT__` or `__GROUP__` .

token

The value of the sync token.

handler

A `Closure` handler for processing the sync results.

log

A logger instance for the connector.

Returns

If the operation type is `GET_LATEST_SYNC_TOKEN` , the script must return an object that represents the last known `SyncToken` for the corresponding `ObjectClass` . For example:

```
def operation = operation as OperationType
def configuration = configuration as ScriptedConfiguration
def log = log as Log
def objectClass = objectClass as ObjectClass
def options = options as OperationOptions
def token = token as Object

case OperationType.GET_LATEST_SYNC_TOKEN:
        switch (objectClass) {
            case ObjectClass.ACCOUNT:
                return new SyncToken(17);
            case ObjectClass.GROUP:
                return new SyncToken(16);
            case ObjectClass.ALL:
                return new SyncToken(17);
// ....
```

If the operation type is `SYNC`, the script must return a new `SyncToken` for the corresponding `ObjectClass`. A Sync result handler (callback) is passed to the script to return the Sync results one by one. The handler must be called for each result.

The handler variable that is passed to the script is a Groovy Closure. It can be called in the following ways:

- With an ICF `SyncDelta` object.

  You can use a `SyncDeltaBuilder` to build this object. For example:

  ```
  def builder = new SyncDeltaBuilder()
  builder.setUid(uidValue)
  builder.setToken(new SyncToken(5))
  builder.setDeltaType(SyncDeltaType.CREATE)
  builder.setObject(connectorObject)  // Use the ConnectorObjectBuilder class to build the ConnectorObject
  object.

  // Call the handler with the SyncDelta object
  handler builder.build()
  ```

- Using a Groovy Closure.

  In this case, the Closure delegates calls to a specific Object that implements these calls. For example:

  ```
  handler {                       // The handler parameter here
                                                                            is a Closure
      syncToken tokenValue        // (mandatory), the method resolution for 'syncToken' is delegated to
                                     the Object handling the Closure
      <DELTA_TYPE>()              // (mandatory), DELTA_TYPE should be one of: CREATE, UPDATE, DELETE,
                                     CREATE_OR_UPDATE
      object connectorObject      // (optional if DELTA_TYPE is a DELETE), the method resolution for
                                     'object' is delegated to the Object handling the Closure
      previousUid prevUidValue    // (optional), use only if UID has changed
  }
  ```

  In the following example, the handler is called twice - first for a CREATE and then for a DELETE:

```
// CREATE
handler({
    syncToken 15
    CREATE()
    object {
        id nameValue
        uid uidValue as String
        objectClass ObjectClass.GROUP
        attribute 'gid', gidValue
        attribute 'description', descriptionValue
    }
})

// DELETE
handler({
    syncToken 16
    DELETE(uidValue)
}
```

Optionally, when the action is SYNC, you might want to return a `SyncToken` at the end of the script. This is a convenient way to update the sync token if no relevant sync events are found.

Schema script

A schema script builds the schema for the connector, either from a static, predefined schema, or by reading the schema from the external resource. The script should use the `builder` object to create the schema.

A sample schema script for an SQL database is provided in `openidm/samples/scripted-sql-with-mysql/tools/SchemaScript.groovy`.

Input variables

The following variables are available to a sync script:

configuration

A handler to the connector’s configuration object.

operation

An OperationType that corresponds to the action (`SCHEMA`).

builder

An instance of the `ICFObjectBuilder`. The `schema()` method should be called with a Closure parameter defining the schema objects.

For more information, refer to Using the `builder` Parameter.

log

A logger instance for the connector.

Returns

This script has no return value.

Using the `builder` parameter

The `builder.schema()` must call the delegates `objectClass` method and `operationOption` method to define the schema.

Call the `objectClass()` method for each object type (account, group, and so on) that must be defined. Include the call to the following delegates:

- `type()` - the name for this object type
- `attribute()` - define a single attribute for this object type
- `attributes()` - define multiple attribute for this object type
- `disable()` - list the operations for which this object type is forbidden

The following example defines a simple ACCOUNT object type:

```
builder.schema({
    objectClass {
        type ObjectClass.ACCOUNT_NAME
        attribute OperationalAttributeInfos.PASSWORD
        attribute PredefinedAttributeInfos.DESCRIPTION
        attribute 'groups', String.class, EnumSet.of(MULTIVALUED)
        attributes {
            userName String.class, REQUIRED
            email REQUIRED, MULTIVALUED
            __ENABLE__ Boolean.class
            createDate  NOT_CREATABLE, NOT_UPDATEABLE
            lastModified Long.class, NOT_CREATABLE, NOT_UPDATEABLE, NOT_RETURNED_BY_DEFAULT
            passwordHistory String.class, MULTIVALUED, NOT_UPDATEABLE, NOT_READABLE, NOT_RETURNED_BY_DEFAULT
            firstName()
            sn()
        }
    }
}
```

Resolve username script

A resolve username script resolves an object to its `uid` based on its `username`.

A sample resolve username script for an SQL database is provided in `openidm/samples/scripted-sql-with-mysql/tools/ResolveUsernameScript.groovy`.

Input variables

The following variables are available to a resolve username script:

configuration

A handler to the connector's configuration object.

options

A handler to the Operation Options.

operation

An OperationType that corresponds to the action (`RESOLVE_USERNAME`).

objectClass

The object class for which the username is resolved, such as `__ACCOUNT__` or `__GROUP__` .

username

A string that represents the username of the object.

log

A logger instance for the connector.

Returns

The user unique ID (ICF `__UID__`) of the object. The `type` of the returned UID must be a `string` or a `Uid` . If a null value or an object type other than `string` or `Uid` is returned, the script must throw an exception.

Resolve Username Script

```
def operation = operation as OperationType
def configuration = configuration as ScriptedConfiguration
def username = username as String
def log = log as Log
def objectClass = objectClass as ObjectClass
def options = options as OperationOptions
if (objectClass.is(ObjectClass.ACCOUNT_NAME)) {
    if (username.equals("TESTOK1")) {
        return new Uid("123")
    }
    throw new UnknownUidException();
}
```

Run on resource script

A run on resource script runs directly on the target resource that is managed by the connector.

A sample run on resource script for a connector that connects to DS over REST is provided in `openidm/samples/scripted-rest-with-dj/tools/ScriptOnResourceScript.groovy` .

Input variables

The following variables are available to a run on resource script:

configuration

A handler to the connector’s configuration object.

options

A handler to the Operation Options.

operation

An OperationType that corresponds to the action (`RUNSCRIPTONRESOURCE`).

arguments

The `arguments (Map)` of the script (can be `null`).

log

A logger instance for the connector.

Returns

Any object that is returned by the script.

Run on Resource Script

```
import groovyx.net.http.RESTClient
import org.apache.http.client.HttpClient
import org.forgerock.openicf.connectors.scriptedrest.ScriptedRESTConfiguration
import org.forgerock.openicf.connectors.groovy.OperationType
import org.identityconnectors.common.logging.Log
import org.identityconnectors.framework.common.objects.OperationOptions

def operation = operation as OperationType
def configuration = configuration as ScriptedRESTConfiguration
def httpClient = connection as HttpClient
def connection = customizedConnection as RESTClient
def log = log as Log
def options = options as OperationOptions
def scriptArguments = scriptArguments as Map
def scriptLanguage = scriptLanguage as String
def scriptText = scriptText as String
```

Run on connector script

A run on connector script enables IDM to run a script in the context of the connector.

Input variables

The following variables are available to a run on connector script:

configuration

A handler to the connector’s configuration object.

options

A handler to the Operation Options.

operation

An OperationType that corresponds to the action (`RUNSCRIPTONCONNECTOR`).

arguments

The `arguments (Map)` of the script (can be `null`).

log

A logger instance for the connector.

Returns

Any object that is returned by the script.

Custom configuration initialization

Connectors created with the Groovy connector toolkit are stateful connectors by default. This means the connector configuration instance is created only once.

The Groovy connector toolkit is precompiled code and connector configurations are initialized in a specific way. If you have specific initialization requirements, you can customize how the connector configuration instance is initialized before the first script is evaluated. The `CustomizerScript.groovy` file lets you define custom closures to interact with the default implementation.

The `CustomizerScript.groovy` file, provided with each compiled connector implementation, defines closures, such as `init {}` , `decorate {}` , and `release {}` . These closures are called during the lifecycle of the configuration.

When you unpack the Groovy connector toolkit JAR file, the `CustomizerScript.groovy` file is located at `org/forgerock/openicf/connectors/connector-implementation` .

Scripted connectors with PowerShell

You can use the PowerShell connector toolkit to create connectors that can provision any Microsoft system, including, but not limited to, Active Directory, Microsoft SQL, MS Exchange, Sharepoint, Office365, and Azure. Any task you can perform with PowerShell can be executed through connectors based on this toolkit.

The PowerShell connector toolkit is not a complete connector, in the traditional sense. Instead, it is a framework where you must write your own PowerShell scripts to address your Microsoft Windows ecosystem requirements.

Connectors created with the PowerShell connector toolkit run on the .NET platform and require the installation of a .NET connector server on the Windows system. To install the .NET connector server, follow the instructions in Install .NET RCS. These connectors also require PowerShell V2.

The PowerShell connector toolkit is available from the BackStage download site. To install the connector, download the archive (`mspowershell-connector-1.4.7.0.zip`) and extract the `MsPowerShell.Connector.dll` to the same folder where the Connector Server (`connectorserver.exe`) is located. IDM provides sample connector configurations and scripts that enable you to get started with this toolkit.

> **Tip**
>
> About PowerShell
> PowerShell combines a command-line shell and scripting language, built on the .NET Framework. For more information, refer to PowerShell Documentation.

Connector troubleshooting

Logs

It can be difficult to determine if the root cause of a problem is at the ICF or connector level, or at the application level.

The ICF API sets the `LoggingProxy` at a very high level. You can consider the Logging Proxy as the *border* between the application (IDM) and the ICF framework.

For more information about logging, refer to Connector logs.

Custom connector required version pattern

If your custom connector configuration fails to display in the admin UI, make sure that it’s using 4-digit versioning. The admin UI expects this type of version pattern. For example:

```
1.5.0.0
```

ICF release notes

These release notes cover the ICF releases that are supported in a deployment of PingIDM, Remote Connector Server (RCS), or PingOne Advanced Identity Cloud.

Subscribe for automatic updates: ICF release notes RSS Feed

Downloads are available on Backstage.

Recent updates

Version	Product	Date
1.5.20.30	• ICF Connector Framework • Remote Connector Server • ICF Connectors • Deprecation	2025/07/07
1.5.20.29	• ICF Connector Framework • Remote Connector Server • ICF Connectors • Changed functionality	2025/04/23
1.5.20.28	• ICF Connector Framework • Remote Connector Server • ICF Connectors	2025/02/27
1.5.20.27	• Remote Connector Server • ICF Connectors	2025/01/09
1.5.20.26	• ICF Connector Framework • Remote Connector Server • ICF Connectors	2024/12/06
1.5.20.25	• ICF Connector Framework	2024/10/18
1.5.20.24	• ICF Connector Framework	2024/10/02

Version	Product	Date
1.5.20.23	• ICF Connector Framework • Remote Connector Server • ICF Connectors • Changed functionality	2024/08/01
1.5.20.22	• ICF Connector Framework • Remote Connector Server • ICF Connectors • Changed functionality	2024/06/21
1.5.20.21	• ICF Connector Framework • Remote Connector Server • ICF Connectors • Changed functionality • Deprecation	2024/03/19
1.5.20.20	• ICF Connectors	2024/02/06
1.5.20.19	• ICF Connectors	2023/11/17
1.5.20.18	• ICF Connector Framework • Remote Connector Server • ICF Connectors	2023/11/17
1.5.20.17	• ICF Connectors	2023/09/07
1.5.20.16	• ICF Connectors	2023/08/02
1.5.20.15	• ICF Connector Framework • Remote Connector Server • ICF Connectors	2023/05/12
1.5.20.14	• Remote Connector Server • ICF Connectors	2023/03/20

Version	Product	Date
1.5.7.0	• .NET remote connector server	2023/02/02
1.5.20.12	• Remote Connector Server • ICF Connectors	2022/12/09
1.5.20.11	• ICF Connector Framework • Remote Connector Server • ICF Connectors	2022/11/01
1.5.20.9	• Remote Connector Server • ICF Connectors	2022/09/09
1.5.20.8	• ICF Connector Framework • Remote Connector Server • ICF Connectors	2022/08/08
1.5.20.7	• ICF Connector Framework • Remote Connector Server • ICF Connectors	2022/06/06
1.5.20.6	• ICF Connector Framework • Remote Connector Server • ICF Connectors	2022/05/05
1.5.20.5	• ICF Connector Framework • Remote Connector Server • ICF Connectors	2022/02/14
1.5.20.4	• ICF Connector Framework • Remote Connector Server • ICF Connectors	2021/12/08

Version	Product	Date
1.5.20.3	• ICF Connector Framework • Remote Connector Server • ICF Connectors	2021/11/22
1.5.20.2	• Remote Connector Server • ICF Connectors	2021/07/27
1.5.20.1	• Database Table Connector	2021/05/28
1.5.20.0	• ICF Connector Framework • ICF Connectors • Remote Connector Server	2021/04/16
1.5.19.6	• ICF Connector Framework • ICF Connectors • Remote Connector Server	2021/02/26
1.5.19.5	• ICF Connector Framework • ICF Connectors • Remote Connector Server	2021/02/12
1.5.19.4	• ICF Connector Framework • ICF Connectors • Remote Connector Server	2021/01/22
1.5.19.3	• ICF Connector Framework • ICF Connectors • Remote Connector Server	2020/12/13
1.5.19.2	• ICF Connector Framework • ICF Connectors • Remote Connector Server	2020/11/30

Version	Product	Date
1.5.19.1	• ICF Connector Framework • ICF Connectors • Remote Connector Server	2020/11/18
1.5.19.0	• ICF Connector Framework • ICF Connectors • Remote Connector Server	2020/10/20
1.5.18.0	• ICF Connector Framework • ICF Connectors • Remote Connector Server	2020/09/14

Note

IDM support
The releases listed in this document cover the connector releases since ICF 1.5.18.0. In most cases, these releases are backwards compatible with previous versions of IDM. Check the IDM / ICF Compatibility Matrix for your version of IDM for compatibility before installing a new version of a connector.

This document does not describe all ICF connectors. Additional connectors are available from the BackStage download site. If a connector release is listed here, but is not yet on the site, contact Ping Support for access.

Name changes for ForgeRock products

Product names changed when ForgeRock became part of Ping Identity.

The following name changes have been in effect since early 2024:

Old name	New name
ForgeRock Identity Cloud	PingOne Advanced Identity Cloud
ForgeRock Access Management	PingAM
ForgeRock Directory Services	PingDS
ForgeRock Identity Management	PingIDM
ForgeRock Identity Gateway	PingGateway

Learn more about the name changes in New names for ForgeRock products in the Knowledge Base.

Connector release notes

Subscribe for automatic updates: ICF release notes RSS Feed

Refer to Connector framework release notes for details regarding any changes to the ICF Connector Framework that can affect connector behavior.

Downloads are available on Backstage.

> **Important**
>
> All updated connectors can include security, formatting, and other internal-facing fixes.

1.5.20.30 Connectors

Updated connectors with change details

Database Table connector

- OPENICF-2250: Connector attributes with the configuration property `changeLogColumn` can pass to the connector object.

Google Apps connector

- OPENICF-3081: Use the `passwordHashAlgorithm` property to hash the connector `__PASSWORD__` attribute during transport.
- OPENICF-3088: Bug fix preventing update for `__ACCOUNT__` and `__GROUP__` secondary objects when the payload didn't include changes for the primary Google object.

Kerberos connector

- OPENICF-3170: The `scriptRoots` value returned by the `createCoreConfig` action was invalid.

LDAP connector

- OPENICF-3101: The LDAP connector can read the Novell eDirectory GUID attribute.

Microsoft Graph API connector

- OPENICF-1911: Support for the ability to use environment variables when authenticating with Azure.

SaaS Common

- OPENICF-3097: JWT auth flow now supports PEM-formatted private keys.

SaaS REST Connector

- OPENICF-3114: Ability to send a payload in a delete request.
- OPENICF-3167: Fixes a Gson serialization issue preventing the SaaS REST connector from working with an RCS.

Salesforce connector

- OPENICF-3122: Adds the `initialSyncTokenOffset` configuration property. Use this property to define a period, in hours, to subtract from the current time when generating the initial Salesforce sync token. Default value is `0` hours.

SCIM connector

- OPENICF-3091: When generating the ICF Schema, the ICF ObjectClass type was incorrectly set to the SCIM Schema name instead of the ResourceType name.

ServiceNow connector

- OPENICF-2422: The connector supports the ServiceNow user object property `cost_center`.

Updated connectors without change details

Note

Connectors without change details can include security, formatting, and other internal-facing fixes.

- Box connector
- CSV connector
- Scripted REST connector
- Hubspot connector
- Webex connector

1.5.20.29 Connectors

New connectors

SaaS REST Connector

Updated connectors with change details

Adobe Marketing Cloud connector

- OPENICF-2983: Connector invalidates access token on authentication failure.
- OPENICF-3044: Removed runtime configuration properties. Learn more in Changed functionality.

Box connector

- OPENICF-2978: Connector invalidates access token on authentication failure.

Epic connector

- OPENICF-2982: Connector invalidates access token on authentication failure.

- OPENICF-3046: Removed runtime configuration properties. Learn more in Changed functionality.

Google Cloud Platform connector

- OPENICF-2980: Connector invalidates access token on authentication failure.

HubSpot connector

- OPENICF-2979: Connector invalidates access token on authentication failure.

LDAP connector

- OPENICF-3018: For DS 8.0 or later LDAP servers, the connector pages the change log with a search control cookie instead of filtering against change numbers. Change numbers are no longer required to set up and use the sync action.

Marketo connector

- OPENICF-2984: Connector invalidates access token on authentication failure.
- OPENICF-3045: Removed runtime configuration properties. Learn more in Changed functionality.

SaaS Common

- OPENICF-2967: Support for JWT authentication flow with framework 1.5.20.24 or later and RCS 1.5.20.24 or later.
- OPENICF-2985: Connector invalidates access token on authentication failure.

Salesforce connector

- OPENICF-2977: Connector invalidates access token on authentication failure.

SAP SuccessFactors connector

- OPENICF-2981: Connector invalidates access token on authentication failure.

SCIM connector

- OPENICF-2975: Connector invalidates access token on authentication failure.

ServiceNow connector

- OPENICF-2976: Connector invalidates access token on authentication failure.

1.5.20.28 Connectors

New connectors

- Multiple CSV connector

Updated connectors with change details

Microsoft Graph API connector

- **OPENICF-2910: You can now read the following Contacts attributes:**
  - `directReports`
  - `memberOf`
  - `transitiveMemberOf`
  - `manager`

> **Note**
>
> These attributes are not returned by default.

- **OPENICF-3005: You can now read the following `servicePrincipal` attributes:**
  - `owners`
  - `memberOf`
  - `transitiveMemberOf`
  - `oauth2PermissionGrants`

> **Note**
>
> These attributes are not returned by default.

Updated connectors without change details

> **Note**
>
> Connectors without change details can include security, formatting, and other internal-facing fixes.

- **AWS IAM Identity Center connector**
- **LDAP connector**
- **SaaS Common connectors**
- **Scripted Groovy connectors**
- **Workday connector**

1.5.20.27 Connectors

Updated connectors with change details

Google Apps connector

- OPENICF-2996: Correctly maps License Assignment read operation parameters to Google API calls.

LDAP connector

- OPENICF-2992: Improved support for IBM directory changelog "changes" binary attribute.

1.5.20.26 Connectors

New connectors

- Duo connector

Updated connectors with change details

AWS IAM Identity Center connector

- OPENICF-2968: Error when renewing access token.

Epic connector

- OPENICF-2941: Querying Epic accounts could fail.

LDAP connector

- OPENICF-2931: PingDirectory is now a recognized LDAP directory.

Microsoft Graph API connector

- OPENICF-2900: Added a user resource attribute `authenticationMethods` that is a read-only list of objects containing the authentication methods associated with a user.
- OPENICF-2901: User email authentication methods can be added/updated/deleted using a new String attribute `__emailAuthenticationMethod__` that contains the email associated with the user’s authentication preference.
- OPENICF-2902: The connector can now manage phone authentication methods on a user using a new virtual multivalued String attribute `__phoneAuthenticationMethods__` that contains a definitive list of concatenated `"{phoneNumber}:{phoneType}"`.
- OPENICF-2903: Adds multivalued string attribute `__removeFido2Methods__` to the user schema. This attribute takes a list of String GUIDs to be deleted as Fido2 auth method IDs associated with a user.

- OPENICF-2912: Adds multivalued string attribute `__removeMicrosoftAuthenticatorMethods__` to the user schema. This attribute holds a list of GUIDs associated with MicrosoftAuthenticator authentication method IDs to be removed from a user.
- OPENICF-2913: Adds multivalued string attribute `__removeSoftwareOathMethods__` to the user schema. This attribute holds a list of GUIDs associated with Software Oath authentication method IDs to be removed from a user.

MongoDB connector (Scripted Groovy)

- OPENICF-2987: Update MongoDB driver to version 4.11.4.

SAP S/4HANA connector

- OPENICF-2915: You can specify the `instanceUrl` of the SAP Hana instance in the connector configuration properties.
- OPENICF-2934: Query paging fixes.

SCIM connector

- OPENICF-2880: Reduce logging noise when a schema extension overrides a core schema attribute.

Scripted Groovy

- OPENICF-2955: The Scripted Groovy `scriptRoots` configuration property can now reference Groovy scripts embedded within the connector JAR file using the `!` prefix.

Workday connector

- OPENICF-1148: Support for updating the primary work phone number using the `primaryWorkPhone` connector attribute.
- OPENICF-2622: You can use XPath transformations to simplify and map Workday attributes directly to read-only connector object type properties.

  > **Note**
  >
  > Requires connector framework version 1.5.20.24 or later.

- OPENICF-2891: Deprecate Workday connector schema attribute `mobile`.

Updated connectors without change details

> **Note**
>
> Connectors without change details can include security, formatting, and other internal-facing fixes.

- Adobe Admin Console connector
- Adobe Marketing Cloud connector
- AS400 connector
- AWS connector

- Box connector
- Cerner connector
- CSV connector
- Database table connector
- DocuSign connector
- Dropbox connector
- Google Apps connector
- Google Cloud Platform (GCP) connector
- HubSpot connector
- IBM RACF connector
- Kerberos connector
- Marketo connector (Scripted Groovy)
- Oracle EBS
- Peoplesoft connector
- PingOne connector
- Powershell connector toolkit
- Salesforce connector
- SAP connector (Scripted Groovy)
- SAP HANA Database connector
- SAP SuccessFactors connector
- ScriptedREST connector (Scripted Groovy)
- ScriptedSQL connector (Scripted Groovy)
- ServiceNow connector
- SSH connector (Scripted Groovy)
- Webex connector

1.5.20.23 Connectors

New connectors

- AWS IAM Identity Center connector

- Box connector (SaaS common)

Updated connectors with change details

Adobe Admin Console connector (SaaS common)

- OPENICF-2792: Set the type for the `orgSpecific` and `businessAccount` schema attributes to boolean.
- OPENICF-2845: Ability to update the user's email address.
- OPENICF-2851: Updating group memberships for an Adobe account may result in excessive email notifications.

Amazon Web Services (AWS) connector

- OPENICF-2755: Support for groups, roles, managed policy, inline policy, service control policy, and org unit object types.

LDAP connector

- OPENICF-2805: SASL-EXTERNAL (mTLS) is now available with the LDAP connector.

Microsoft Graph API connector

- OPENICF-2834: The virtual resource displaying service plans as their own object now includes the `skuPartNumber` as a readable attribute. Additionally, the display name of service plans is now a combination of the `skuPartNumber` and the `servicePlanName` .

SaaS Common

- OPENICF-2781: During token renewal, properly cache new refresh token in the connector configuration.

SAP SuccessFactors connector

- OPENICF-2847: Resolve `ArrayIndexOutOfBoundsException` when consuming older connector configuration.

Webex Connector (SaaS common)

- OPENICF-2619: Properly handle HTTP 400 error responses during Webex user creation.

Workday connector

- OPENICF-2524: Paged queries with no results throw an internal server error.

Updated connectors without change details

> **Note**
>
> Connectors without change details can include security, formatting, and other internal-facing fixes.

- Adobe Marketing Cloud connector
- AS400 connector

- Cerner connector
- CSV connector
- Database table connector
- DocuSign connector (SaaS common)
- Dropbox connector (SaaS common)
- Epic connector
- Google Apps connector
- Google Cloud Platform (GCP) connector
- Groovy connector toolkit
- HubSpot connector
- IBM RACF connector
- Kerberos connector
- Marketo connector
- MongoDB connector
- Oracle EBS
- Peoplesoft connector
- PingOne connector (SaaS common)
- Powershell connector toolkit
- Salesforce connector
- SAP connector
- SAP HANA Database connector
- SAP S/4HANA connector
- SAP SuccessFactors connector
- SCIM connector
- ScriptedREST connector
- ScriptedSQL connector
- ServiceNow connector
- SSH connector

1.5.20.22 Connectors

Updated connectors with change details

Adobe Admin Console connector

- OPENICF-2559: Initial release of the Adobe Admin Console connector. Refer to Adobe Admin Console connector for more information.

Database Table connector

- OPENICF-2679: Reduce log level of many operations

DocuSign connector

- OPENICF-2557: DocuSign connector v2
    - OPENICF-2583: Add ObjectClass UserGroups
    - OPENICF-2587: Add filter support for Users ObjectClass
    - OPENICF-2588: Add filter support for the UserGroups ObjectClass
    - OPENICF-2766: Wrong exception message when the connector is configured incorrectly

IBM RACF connector

- OPENICF-2757: Support for new object types, segments, and attributes

MongoDB connector

- OPENICF-2784: Update MongoDB driver to version 4.11.2

Oracle EBS connector

- OPENICF-1760: EBS Connector v2, support responsibilities

PingOne connector

- OPENICF-2740: Enhance user password to accept external password assignments

SAP SuccessFactors connector

- OPENICF-2428: Account Object: Group Name not required
- OPENICF-2528: Support schema discovery and writeback

SCIM connector

- OPENICF-1617: Ability to assign groups to users
- OPENICF-2669: The read rate limit may be exceeded during queries

- OPENICF-2672: Reconciliation after patch remove on managed user throws NPE and full updates omit remove operations
- OPENICF-2682: Using dynamic schema, multivalued attributes of schema extensions are improperly handled
- OPENICF-2710: Creating users/groups with multivalued extension attributes fails
- OPENICF-2726: Do not fail on unknown Enum values when deserializing schemas
- OPENICF-2735: The endpoint in use for a given ResourceType was incorrectly derived from the objectClass defined by the IDM provisioner configuration instead of from the endpoint specified by the ResourceTypes response from the SCIM Provider

Workday connector

- OPENICF-2524: liveSync on Workday custom fields now works

Updated connectors without change details

> **Note**
>
> Connectors without change details can include security, formatting, and other internal-facing fixes.

- Adobe Marketing Cloud connector
- AS400 connector
- AWS connector
- Cerner connector
- CSV connector
- Dropbox connector
- Epic connector
- Google Apps connector
- Google Cloud Platform (GCP) connector
- Groovy connector toolkit
- HubSpot connector
- Kerberos connector
- LDAP connector
- Marketo connector
- MS Graph API connector
- Peoplesoft connector
- Powershell connector toolkit

- Salesforce connector
- SAP connector
- SAP HANA Database connector
- SAP S/4HANA connector
- ScriptedREST connector
- ScriptedSQL connector
- ServiceNow connector
- SSH connector
- Webex connector

1.5.20.21 Connectors

Updated connectors with change details

Dropbox connector

- OPENICF-2664: SaaS Client Initializer should not automatically add default HTTP headers
- OPENICF-2655: Logging levels in use by generated connector class are too verbose

Epic connector

- OPENICF-2233: Add support for managing SER resources
- OPENICF-2492: EMP Enhancements

Google Apps connector

- OPENICF-2617: Deprecate __SECONDARY_EMAIL__ in favor of __SECONDARY_EMAILS__ attribute

LDAP connector

- OPENICF-2544: LiveSync timestamp strategy may lose changes when remote handler returns `false`

PingOne connector

- OPENICF-2507: Initial release of the PingOne connector. Refer to PingOne connector for more information.

SAP connector

- OPENICF-2410: Additional attributes in the Profile Object Type
- OPENICF-2411: Additional attributes in the Activity Groups Object Type

Scripted REST connector

- OPENICF-1917: Support for throttling

Webex Connector

- OPENICF-2047: Initial release of the Webex connector. Refer to Webex Connector for more information.

Updated connectors without change details

> **Note**
>
> Connectors without change details can include security, formatting, and other internal-facing fixes.

- AWS connector
- Kerberos connector

1.5.20.20 Connectors

Updated connectors with change details

Database Table connector

- OPENICF-2606: Schema is unnecessarily regenerated for every operation.

Google Apps connector

- OPENICF-2194: PATCH remove operation doesn't update the object when both the field and value are provided.
- OPENICF-2351: Include 503 errors in the retry logic for GoogleApps connector.
- OPENICF-2490: Requests hang if the Google Admin SDK API has not been enabled within the configured Google Project.

Microsoft Graph API connector

- OPENICF-2593: Upgrade azure-identity dependency to latest version.

Salesforce connector

- OPENICF-2626: A duplicate header sent by the connector prevents successful OAuth flow.

SCIM connector

- OPENICF-2575: Running liveSync for object classes other than the Account object results in an error.
- OPENICF-2601: Inject common attributes within dynamically generated schemas for all resource types.

Updated connectors without change details

> **Note**
>
> Connectors without change details can include security, formatting, and other internal-facing fixes.

- Adobe Marketing Cloud connector
- Marketo connector
- ScriptedREST connector
- ScriptedSQL connector

1.5.20.19 Connectors

Updated connectors with change details

SCIM connector

- OPENICF-1296: HTTP Status and Error Response Handling.
- OPENICF-2574: Authorization header contains an extra space which breaks client_credentials flow.
- OPENICF-2579: TestOp should catch all exceptions thrown by the initial attempt to read the alternate ServiceProviderConfig endpoint.

Updated connectors without change details

> **Note**
>
> Connectors without change details can include security, formatting, and other internal-facing fixes.

- Microsoft Graph API connector

1.5.20.18 Connectors

Updated connectors with change details

Dropbox connector

- OPENICF-2354 Missing property messages.

Google Apps connector

- OPENICF-2487 License assignment account attribute should be an array of strings.

LDAP connector

- OPENICF-2296: Bad IP address for the LDAP host should be caught, and a 503 error code should be returned by IDM.
- OPENICF-2401: `queryFilter` true or false against isActive attribute returns all results.
- OPENICF-2526: Specify a negative offset (in seconds) to be applied to the timestamp token when querying for changes on the remote LDAP server using the `timestampSyncOffset` configuration property.
- OPENICF-2555: Ability to define custom octet string attributes using the `customOctetStringAttributes` configuration property.

Microsoft Graph API connector

- OPENICF-2006: Clicking on Azure AD connector for the first time throws a 500 error.
- OPENICF-2027: Support single quotation marks in query filters.
- OPENICF-2140: Info level logging is overused for this connector.

Salesforce connector

- OPENICF-1527: Returns a generic ConnectorException 'Error: 400' on expired/revoked refresh_token.
- OPENICF-2246: Implement support for Client Credentials Grant type. Refer to Configure the Salesforce connector.
- OPENICF-2266: User schema is not cached.
- OPENICF-2505: createFullConfig NPEs when supportedObjectTypes contains FeatureLicense.

SAP connector

- OPENICF-2371: Scripts for SAP HR searching and filtering.
- OPENICF-2465: Prevent activity group assignment from being deleted when the assignment is end-dated.
- OPENICF-2480: SAP Central User Administration (CUA) support.

SAP HANA Database connector

Initial release of the SAP HANA Database connector. Refer to SAP HANA Database connector for more information.

- OPENICF-2368: SAP HANA Database connector.

SCIM connector

- OPENICF-1528: Salesforce returns a generic ConnectorException 'Error: 400' on expired/revoked refresh_token.
- OPENICF-2472: access_token validation checked on `issued_at` claim instead of `expires_in` for refresh_token grant.
- OPENICF-2500: Extension attributes not flattened when converted to ConnectorObject.
- OPENICF-2504: Map JSON integer type to Java Long.

Updated connectors without change details

> **Note**
>
> Connectors without change details can include security, formatting, and other internal-facing fixes.

- Adobe Marketing Cloud connector
- AS400 connector
- AWS connector
- Box connector
- Cerner connector
- CSV connector
- Database Table connector
- DocuSign connector
- Epic connector
- GCP connector
- HubSpot connector
- IBM RACF connector
- Oracle EBS connector
- Peoplesoft connector
- SAP S/4HANA connector
- SAP SuccessFactors connector
- ScriptedREST connector
- ScriptedSQL connector
- ServiceNow connector
- Workday connector

1.5.20.17 Connectors

No public changes were made specific to this connector, though a new version was released.

No public changes were made specific to this connector, though a new version was released.

No public changes were made specific to this connector, though a new version was released.

- OPENICF-1723: Clarify usage of `proxyUri` configuration property
- OPENICF-900: Implement the `/Schemas` endpoint discovery
- OPENICF-2297: Roles attribute should be a list of Strings, not a list of Objects
- OPENICF-2482: Dynamic schema does not default to static schema on all exceptions
- OPENICF-2483: Creating a user with special attributes fails with dynamically generated schema
- OPENICF-2484: PUT w/schemas attribute fails for Providers that support Patch
- OPENICF-2448: HTTP Client fails to handle OAuth errors
- OPENICF-2453: Persist optional refresh_token issued upon successful access_token renewal

No public changes were made specific to this connector, though a new version was released.

1.5.20.16 Connectors

No public changes were made specific to this connector, though a new version was released.

No public changes were made specific to this connector, though a new version was released.

- OPENICF-2356: GoogleApps Connector doesn't allow listing of licenses
- OPENICF-2394: Align Scripted Connector templates

No public changes were made specific to this connector, though a new version was released.

- OPENICF-2400: Kerberos Search operation logs incorrect operation type
- OPENICF-2394: Align Scripted Connector templates
- OPENICF-2394: Align Scripted Connector templates
- OPENICF-2355: MSGraphAPI Connector doesn't support assigning `servicePlans` to an Azure user
- OPENICF-2394: Align Scripted Connector templates
- OPENICF-2357: Salesforce Connector doesn't allow listing of licenses
- OPENICF-2035: SAP Account Object Type attributes
- OPENICF-2036: SAP Role Object Type Attributes
- OPENICF-2037: SAP UM Profile Object Type Attributes
- OPENICF-2292: Group Object Type attributes
- OPENICF-2350: R3 script uses deprecated methods to parse date
- OPENICF-2360: NPE getting SAP configuration
- OPENICF-2377: Active Group memberships should not sync activity group name

- OPENICF-2379: Should not retrieve, display, or allow manipulation of password hashing attributes
- OPENICF-2386: Router should not be a required attribute
- OPENICF-2388: Must throw an error upon user create/update/delete error
- OPENICF-2394: Align Scripted Connector templates
- OPENICF-2397: Add pagination
- OPENICF-2419: Timestamp filtering support
- OPENICF-2432: Default location for the ScriptRoots is incorrect
- OPENICF-2435: Respect boolean response from search result handler
- OPENICF-2452: Filter CODVN, CODVC, and CODVS from User LOGONDATA
- OPENICF-2459: Query with `_queryFilter=true` no longer returns full user object
- OPENICF-2430: Search and Sync operations do not respect handler result
- OPENICF-2394: Align Scripted Connector templates
- OPENICF-2429: Search and Sync operations do not respect handler result
- OPENICF-2394: Align Scripted Connector templates
- OPENICF-2394: Align Scripted Connector templates
- OPENICF-2438: `externalFieldAndParameterCriteria` config parameter should not be set to null by default

1.5.20.15 Connectors

No public changes were made specific to this connector, though a new version was released.

- OPENICF-2308: Database Table Connector - Possible regression of OPENICF-903
- OPENICF-1987: ORA-00933 - SQL command not properly ended error using Database Table Connector

Initial release of the Dropbox connector. Refer to Dropbox connector for more information.

- OPENICF-2051: Dropbox connector
- OPENICF-2306: MS Graph API Connector: Creating and updating applications with certificates fails
- OPENICF-2269: MS Graph API Connector: Implement application role assignments
- OPENICF-1964: MS Graph API Connector: Add the ability to handle User’s Contacts object
- OPENICF-2315: MS Graph API Connector: otherMails attribute should be an array of strings
- OPENICF-2343: Cannot delete a list of PermissionSetAssignments
- OPENICF-2320: SCIM Connector: totalResults is not used when query is using paging
- OPENICF-2321: SCIM Connector: pagedResultsOffset is not used properly

- OPENICF-2325: SCIM Connector: HTTP error 429 should have a more explicit message
- OPENICF-2323: SCIM Connector: prevent query with sorting when the Service Provider does not accept sorting
- OPENICF-1916: SCIM Connector: Support for throttling

No public changes were made specific to this connector, though a new version was released.

No public changes were made specific to this connector, though a new version was released.

1.5.20.14 Connectors

- OPENICF-2236 - AS400 Connector: does not expose all the AS400ConnectionPool configuration properties
- OPENICF-2252: GoogleApps Connector: Unable to configure connector via UI
- OPENICF-2225: LDAP Connector: syncToken nativeType to be configurable / updated - mismatch with DS type stops livesync

No public changes were made specific to this connector, though a new version was released.

- OPENICF-1976: MS Graph API Connector: Ability to create guest users
- OPENICF-2208: MS Graph API Connector: add the ability to read "application" and "servicePrincipal" object
- OPENICF-2238: MS Graph API Connector: unable to retrieve roles
- OPENICF-2247: MS Graph API Connector: Query filters on collections and filters requiring advanced query parameters cause errors
- OPENICF-2248: MS Graph API Connector: Implement role assignment and role eligibility schedules
- OPENICF-2251: MS Graph API Connector: __ACCOUNT__ data listing fails in native console for assignedLicenses
- OPENICF-2257: MS Graph API Connector: Clicking Role Assignment in Data tab throws a Graph API error
- OPENICF-2267: MS Graph API Connector: Proxy -→ Java.lang.ClassCastException: class okhttp3.OkHttpClient cannot be cast to class com.azure.core.http.HttpClient (okhttp3.OkHttpClient and com.azure.core.http.HttpClient are in unnamed module of loader
- OPENICF-2270: MS Graph API Connector: Adding API permissions to applications fails
- OPENICF-2271: MS Graph API Connector: proxy basic auth not implemented but referenced
- OPENICF-2275: MS Graph API Connector: Refactor connector new object handlers and UnsupportedOperationException handling

Initial release of the EBS connector. Refer to Oracle EBS connector for more information.

- OPENICF-1781: EBS Connector V1.0
- OPENICF-2311: PeopleSoft Connector: Remove embedded `psft-2.0` and `psjoa-1.0` Jar files
- OPENICF-2176 - Salesforce Connector: Support Feature License Elements as List on User Object
- OPENICF-1922 SCIM Connector: PATCH operation should use `path` attribute for "add" and "replace"

- OPENICF-2241: SCIM Connector: Service Provider Config settings don’t work for Salesforce

1.5.20.12 Connectors

Initial release of the AS400 connector. Refer to AS400 connector for more information.

- OPENICF-2192: NPE when updating LicenseAssignments through a user update
- OPENICF-2117: Hide Alternate Emails from the schema
- OPENICF-2195: Intermittent NPE when we try to read newly created user
- OPENICF-400: LDAP connector should be able to properly handle reading the AD tokenGroups attribute
- OPENICF-2033: PeopleSoft Connector v2.0
- OPENICF-2183: Exception when SAP connector is running in OpenIDM
- OPENICF-2007: SAP SuccessFactors v2
- OPENICF-1916: Support for throttling
- OPENICF-2207: Ability to define Accept: and Content-Type: HTTP headers
- OPENICF-2030: Connector breaks when workerID is empty when using RCS
- OPENICF-2150: Ability to add field and parameter to the request criteria

1.5.20.11 Connectors

No public changes were made specific to this connector, though a new version was released.

No public changes were made specific to this connector, though a new version was released.

No public changes were made specific to this connector, though a new version was released.

- OPENICF-1960: Cerner Connector v2

No public changes were made specific to this connector, though a new version was released.

No public changes were made specific to this connector, though a new version was released.

No public changes were made specific to this connector, though a new version was released.

Initial release of the GCP connector. Refer to Google Cloud Platform connector for more information.

- OPENICF-1749: GCP Connector
- OPENICF-2039: GoogleApps Connector: missing some user attributes
- OPENICF-2040: GoogleApps Connector: Manage role attributes
- OPENICF-2041: GoogleApps Connector: Group attributes
- OPENICF-2064: Google Apps Connector: Query the Google Workspace instance for Licenses

- OPENICF-2066: GoogleApps Connector: Ability to query Roles and RoleAssignments
- OPENICF-2136: Google Apps Connector: Exponential Back off for reading google objects required

No public changes were made specific to this connector, though a new version was released.

- OPENICF-1762: IBM RACF API Connector

> **Note**
>
> There was a previous RACF connector, which is deprecated. Users of the previous RACF connector should migrate to the new connector.

- OPENICF-1856: LDAP Connector: Assignment of static group to IDM User fails to assign it on LDAP side if user is already a member of a Dynamic Group on LDAP side
- OPENICF-2089: LDAP Connector: ldapGroups membership does not take into account nested membership of other groups
- OPENICF-2108: LDAP Connector: slow group membership updates with unindexed member/uniqueMember attributes in DS
- OPENICF-2126: Assignment Issue: Managed User to DS Groups Failure to Select Target Group

No public changes were made specific to this connector, though a new version was released.

- OPENICF-2068: MSGraphAPI Connector: Implement Azure AD Directory Roles support
- OPENICF-2088: MSGraphAPI Connector: Implement Azure AD custom role creation

No public changes were made specific to this connector, though a new version was released.

No public changes were made specific to this connector, though a new version was released.

No public changes were made specific to this connector, though a new version was released.

No public changes were made specific to this connector, though a new version was released.

- OPENICF-2112: SCIM Connector: caseSensitive
- OPENICF-2113: SCIM Connector: problem with "issuedAt" from OAuth neg
- OPENICF-2114: SCIM Connector: use authenticationBasic as an option for OAuth neg
- OPENICF-2125: SCIM Connector: Fix Filter

No public changes were made specific to this connector, though a new version was released.

- OPENICF-2130: ServiceNow connector query results do not match what is returned from API

No public changes were made specific to this connector, though a new version was released.

1.5.20.9 Connectors

No public changes were made specific to this connector, though a new version was released.

- OPENICF-1614: MS Graph API Connector: Livesync on user/group does not populate membership
- OPENICF-1858: MS Graph API Connector: Add Group Owners management
- OPENICF-1675: SAP Connector: Groovy deps should be embedded
- OPENICF-2071: SAP Connector: Cannot update ACTIVITY GROUPS for users

1.5.20.8 Connectors

- OPENICF-1935: CSV Connector: generates a stacktrace for Read Only permission files
- OPENICF-1969: CSV Connector: Update csv connector parsing library
- OPENICF-1258: CSV Connector: stripping empty strings, replacing with nulls.

No public changes were made specific to this connector, though a new version was released.

- OPENICF-2038: Google Apps Connector: Updating user’s group membership may return NPE
- OPENICF-1977: LDAP Connector: Detect CA LDAP directory server
- OPENICF-1606: MS Graph API Connector: Upgrade to MS Graph Java SDK v3
- OPENICF-1807: MS Graph API Connector: Better handle failure of hard delete
- OPENICF-1819: MS Graph API Connector: "performHardDelete" should be set to false by default

No public changes were made specific to this connector, though a new version was released.

- OPENICF-2002: Salesforce Connector: syncFailureHandler can exceed maxRetries

No public changes were made specific to this connector, though a new version was released.

1.5.20.7 Connectors

Initial release of the AWS IAM connector. Refer to Amazon Web Services (AWS) connector for more information.

- OPENICF-1780: AWS IAM Connector

No public changes were made specific to this connector, though a new version was released.

No public changes were made specific to this connector, though a new version was released.

- OPENICF-1897: LDAP Connector: Add support for nested AD groups

No public changes were made specific to this connector, though a new version was released.

Initial release of the Oracle PeopleSoft connector. Refer to PeopleSoft connector for more information.

- OPENICF-1748: PeopleSoft Connector
- OPENICF-1812: SalesForce Connector: syncFailureHandler maxRetries is not working

Initial release of the SAP S/4HANA connector. Refer to SAP S/4HANA connector for more information.

- OPENICF-1782: SAP Hana Connector

No public changes were made specific to this connector, though a new version was released.

1.5.20.6 Connectors

Initial release of the Cerner connector. Refer to Cerner connector for more information.

- OPENICF-1737: Cerner Connector
- OPENICF-1818: Epic V2 Connector
- OPENICF-1878: Epic Connector: Query filter not matching uid returns HTTP 404
- OPENICF-1181: Google Apps Connector: Unable to delete custom attributes
- OPENICF-1901: LDAP Connector: Reduce JVM garbage from ConnectorObjectBuilder and AttributeBuilder
- OPENICF-1833: Update MongoDB driver to the latest for compatibility with newer versions of MongoDB

1.5.20.5 Connectors

No public changes were made specific to this connector, though a new version was released.

- OPENICF-1711: Database Table Connector - ORA-22816 error when using Oracle trigger

Initial release of the Epic connector. Refer to Epic connector for more information.

- OPENICF-1750: Epic Connector
- OPENICF-1808: Google Apps Connector: when user is provisioned using a role assignment, group isn't set correctly
- OPENICF-1859: LDAP Connector: _memberId is not returned with AD & liveSync if attribute range is used

No public changes were made specific to this connector, though a new version was released.

No public changes were made specific to this connector, though a new version was released.

No public changes were made specific to this connector, though a new version was released.

No public changes were made specific to this connector, though a new version was released.

No public changes were made specific to this connector, though a new version was released.

- OPENICF-1822: SuccessFactors should not require PEM formatted file on disk

1.5.20.4 Connectors

No public changes were made specific to this connector, though a new version was released.

No public changes were made specific to this connector, though a new version was released.

1.5.20.3 Connectors

- OPENICF-1692: Database Table Connector: throwing a null pointer exception
- OPENICF-1716: Google Apps Connector: Add recoveryEmail and recoveryPhone attributes for User
- OPENICF-1731: LDAP Connector: Escape characters (\) not properly handled on delete and updates ops

No public changes were made specific to this connector, though a new version was released.

No public changes were made specific to this connector, though a new version was released.

No public changes were made specific to this connector, though a new version was released.

1.5.20.2 Connectors

- OPENICF-1677: CSV Connector returns pagedResultsCookie for queries with _pageSize=0.
- OPENICF-1666: LDAP Connector: ldapGroups should restrict membership to the specified contexts.
- OPENICF-1656: MS Graph API Connector: Unable to update onPremisesExtensionAttributes.
- OPENICF-1687: MS Graph API Connector: Should be able to work behind an HTTP Proxy.
- OPENICF-1698: MS Graph API Connector: get the cause of exception if test() fails.
- OPENICF-1689: Workday Connector: Workers transaction logs are filtered.
- OPENICF-1691: Workday Connector: Reduce Garbage collection when building connector objects.

1.5.20.1 Connectors

> **Note**
>
> 1.5.20.1 is a limited release, where only the Database Table Connector was released to Backstage.

- OPENICF-1477: Database Table Connector: ORA-01000: maximum open cursors exceeded
- OPENICF-1596: PSQLException: FATAL: terminating connection due to idle-in-transaction timeout

1.5.20.0 Connectors

- OPENICF-1560: LDAP Connector: RFE Disable Paged Results Control

- OPENICF-1586: LDAP Connector: Timestamp sync strategy: Synchronization filters are not used properly
- OPENICF-1553: MongoDB Connector: convertBSONtoICF() does not traverse Arrays.
- OPENICF-1538: MS Graph API Connector: Sync() does not work
- OPENICF-1541: MS Graph API Connector: Add ConsistencyLevel: eventual' header and $count=true for endsWith filter
- OPENICF-1557: MS Graph API Connector: Handle user employeeHireDate attribute and Calendar data type
- OPENICF-1558: MS Graph API Connector: Make sure sortKey is supported by the objectClass
- OPENICF-1559: MS Graph API Connector: Implement Authenticate() call
- OPENICF-1595: MS Graph API Connector: test() should connect to the MS Graph endpoint to validate the connectionThe following known issues will be addressed in a later release:
- OPENICF-1614: MS Graph API Connector: Livesync on user/group does not populate membership
- OPENICF-1615: MS Graph API Connector: Deleting Azure AD group works but throws HTTP 500
- OPENICF-1589: SCIM Connector: NPE caused by exception not properly handled
- OPENICF-1591: SCIM Connector: Parsing OAuth response should not fail on unknown properties
- OPENICF-1598: SCIM Connector: NPE when updating attribute with null value
- OPENICF-1600: SCIM Connector: unknown attributes in a query result should not throw parsing exception
- OPENICF-1601: SCIM Connector: Implement a global connection timeout property

1.5.19.6 Connectors

No issues specific any connectors were addressed in this release.

1.5.19.5 Connectors

- OPENICF-1530: system?_action=createFullConfig validation does not return consistent errors
- OPENICF-1510: Errors in Database Table Connector docs
- OPENICF-1523: ScriptedGroovy connectors fail to load in IDM 7.x when embedded Groovy version does not match IDM Groovy version
- OPENICF-1493: MS Graph API Connector: add the ability to read/assign license for the user
- OPENICF-1499: MS Graph API Connector: remove the maximumConnections property
- OPENICF-1507: MS Graph API Connector: add the ability to read subscribedSku object
- OPENICF-1525: MS Graph API Connector: replace the default Graph SDK logger

- OPENICF-1526: MS Graph API Connector: add the ability to read Team objects
- OPENICF-1522: Salesforce Connector : implement StatefulConfiguration to allow persistence of accessToken in memory
- OPENICF-1518: SCIM connector: Http client ConnectionManager is not set properly
- OPENICF-1504: Workday Connector: SyncToken should be updated even if no events
- OPENICF-1506: Workday Connector: SyncToken should be set to tenant timestamp after call to sync()
- OPENICF-1508: Workday Connector: Query on SCR objects should not include date range as a search criteria

1.5.19.4 Connectors

No issues specific any connectors were addressed in this release.

1.5.19.3 Connectors

- OPENICF-1475: MS Graph API Connector: the 'manager' only returns the id and not the full object
- OPENICF-1481: MS Graph API Connector: add the ability to assign/remove user’s manager
- OPENICF-1483: MS Graph API Connector: can’t remove all groups a user belongs to
- OPENICF-1471: SalesForce Connector: should not implement PoolableConnector interface

1.5.19.2 Connectors

- OPENICF-1448: LDAP Connector: Enabling changelog livesync for oracle unified directory (OUD)
- OPENICF-1466: LDAP Connector: Update filterWithOrInsteadOfAnd to apply to timestamp and Active Directory liveSync
- OPENICF-1470: LDAP Connector: Null Check in ADUserAccounControl.addControl
- OPENICF-1472: LDAP Connector: Data not synced from AD to IDM via livesync on __ALL__ object
- OPENICF-1469: MS Graph API Connector: implement a read/write rate limiter
- OPENICF-1401: SCIM Connector: Align exceptions for not configured (blank/null) configurationProperties

1.5.19.1 Connectors

- OPENICF-1446: MS Graph API Connector: implement PoolableConnector
- OPENICF-1352: Salesforce connector: pagination and cookies not working properly
- OPENICF-1444: SCIM connector - provide support for 'scope'
- OPENICF-1433: SSH connector: Kerberos username prompt for public key and password auth

- OPENICF-1445: SSH connector: Stale or disconnected SSH sessions are not detected when borrowing from the pool
- OPENICF-1383: Workday Connector: Upgrade to API v35.0
- OPENICF-1419: Workday Connector: Implement Service Center Representative object type
- OPENICF-1426: Workday Connector: Ability to update email for Service Center Representative object
- OPENICF-1432: Workday Connector: Implement OR filter
- OPENICF-1447: Workday Connector: add the Contingent_Worker_ID as a search criteria

1.5.19.0 Connectors

> **Note**
>
> Starting in version 1.5.19.0, ICF connectors that previously had external library dependencies now have those dependencies bundled inside the connector.

Initial release of the MS Graph API Connector.

- OPENICF-1388: LDAP Connector 1.5.5.0 throws java.lang.NoSuchMethodError on Java 8
- OPENICF-1396: OPENIDM-15448 changes seemingly broke querying ldap via the data tab
- OPENICF-1414: Scripted Groovy (v3) based connectors fail to load with IDM releases prior to 7.0

1.5.18.0 Connectors

> **Note**
>
> Starting in version 1.5.18.0, the ICF Connector Framework and all connectors bundled with IDM share a unified version number.

No issues specific any connectors were addressed in this release.

Java RCS release notes

Subscribe for automatic updates: ICF release notes RSS Feed

Refer to Connector framework release notes for details regarding any changes to the ICF Connector Framework that may affect RCS behavior.

Downloads are available on Backstage.

> **Important**
>
> Updates to the Java RCS can also include security, formatting, and other internal-facing fixes.

1.5.20.30 Java RCS

- **OPENICF-2272: The Java RCS download now includes a sample dockerfile at `path/to/openicf/docker/Dockerfile`. Learn more in Deploy Java RCS in a Docker container.**

1.5.20.29 Java RCS

- **OPENICF-1724: Fixed error message when uninstalling non-existent Java RCS Windows service.**

1.5.20.28 Java RCS

- **OPENICF-2153: Ability to output `CAUD_TRANSACTION_ID` in the RCS logs.**
- **OPENICF-2616: The default truststore is now the RCS security/trustStore previously introduced by OPENICF-2152.**
- **OPENICF-2970: Simplified default `ConnectorServer.properties` and added sample configurations available in `conf/samples`:**
  - `ConnectorServer.properties.cloud-client`
  - `ConnectorServer.properties.default-parameters`
  - `ConnectorServer.properties.onprem-client`
  - `ConnectorServer.properties.onprem-server`
- **OPENICF-2972: Removed the `/setDefaults` command.**

1.5.20.27 Java RCS

- **OPENICF-2969: The default RCS `webSocketConnections` are reduced from `3` to `2`.**

1.5.20.26 Java RCS

- **OPENICF-2942: You can launch Java RCS in a Docker container with multiple values (comma-separated) defined for `connectorserver.url` in `OPENICF_OPTS`.**

1.5.20.23 Java RCS

Java 17 required

Running Java RCS requires Java 17.

More bundled connectors

Java RCS now bundles the following additional connectors:

- AS400
- Cerner
- Epic
- IBM RACF
- MongoDB
- Oracle EBS
- Peoplesoft
- SAP
- SAP HANA DB

1.5.20.22 Java RCS

- OPENICF-2640: If remote IDM process is stopped, Websocket connections increase until IDM process is back.

1.5.20.21 Java RCS

- OPENICF-2228: `logback.xml` moved to `conf/` directory.
- OPENICF-2152: Provide a default SSL truststore file.
- OPENICF-2511: Connection to IDM becomes dysfunctional after a period of inactivity in RCS.
- OPENICF-2643: Timeout waiting to acquire a websocket to send a message has been decreased from 2 minutes to 30 seconds.
- OPENICF-2644: NPE may be thrown on WebSocketConnectionGroup shutdown.
- OPENICF-2154: RCS now logs any connector exception to the log file and console.

1.5.20.18 Java RCS

- OPENICF-1638: The default `logback.xml` logging configuration rolls log files daily. Refer to Rolling log policy.

- OPENICF-2547: New local connector facade created --> Method: newConnectorFacadeInstance.

1.5.20.15 Java RCS

- OPENICF-2336: Java RCS: Change the default connector.groupCheckInterval=900 seconds to 60 seconds.

1.5.20.14 Java RCS

- OPENICF-1418: Java RCS: Invalid interval properties not handled properly for client mode.
- OPENICF-2181: Java RCS: Housekeeping task should log which endpoint/instance it is working with.
- OPENICF-2274: Java RCS: Response to unknown protobuf request should contain RCS version.

1.5.20.12 Java RCS

- OPENICF-1473: Java RCS: ConnectorServer.properties template should include config for FRAAS.
- OPENICF-1889: Java RCS: Include relevant defaults for RCS config.

1.5.20.11 Java RCS

- OPENICF-2132: Java RCS: docker-entrypoint.sh uses -run instead of -service to start the RCS.
- OPENICF-2137: Java RCS: When running in -service mode, version is not displayed at startup.
- OPENICF-2174: Java RCS: Incompatible with AM macaroons: Unrecognized field "expireTime".

1.5.20.9 Java RCS

Bundled connectors were updated, though no changes to the remote connector server were made.

1.5.20.8 Java RCS

- OPENICF-2000: potential log flooding resulting from operation cancel request messages for LocalOperations which have already completed.

1.5.20.7 Java RCS

- OPENICF-1883: Java RCS: Improve stability of RCS WebSocket connection management.
- OPENICF-1975: Java RCS: Increase default heap size from 512m to 1g.
- OPENICF-1925: Java RCS: require explicitly set property to enable agent deployment.

1.5.20.6 Java RCS

- OPENICF-1832: Java RCS: High CPU usage when running as a service.

1.5.20.5 Java RCS

- OPENICF-1855: Investigate handling query 'poison pill' termination via recon automatic retry upon exception receipt.

1.5.20.4 Java RCS

- OPENICF-1726: Java RCS: OAuth access token should be cached and reused till expired.
- OPENICF-1744: Java RCS: Unable to run RCS with Marketo connector using a different groovy version.
- OPENICF-1796: Java RCS: NPE if connectorserver.url has a bad hostname

1.5.20.3 Java RCS

- OPENICF-1725: Java RCS: classPath issue in JAVA_DLL when running as a service on Windows.
- OPENICF-1730: Client ConnectorInfos cache not refreshed upon RCS instance restart when using RCS Agent.
- OPENICF-1743: Java RCS: windows service starts up and stops abruptly.
- OPENICF-1751: Sporadic issues managing RCS-hosted connectors through IDM Native Admin Console.
- OPENICF-1783: Java RCS: Rename the windows service name.
- OPENICF-1792: Java RCS: message hostId missing and causing a connection drop.
- OPENICF-1746: Java RCS: Should display its current version in console and jar files should have their version in file name.
- OPENICF-1764: Java RCS: on Windows, ConnectorServer.bat /setKey does not work.
- OPENICF-1774: Java RCS: upgrade Procrun to latest version for RCS as a Windows service.

1.5.20.2 Java RCS

- OPENICF-1655: Java RCS: When using TLS, the RCS does not work behind a proxy.

1.5.20.0 Java RCS

- OPENICF-1366: Java Connector Server: /setDefaults does not revert config to default properly.
- OPENICF-1502: RCS: requests not cancelled when websocket closes.
- OPENICF-1540: RCS: requests bearer token from AM, but doesn’t look for error status code in response.

- OPENICF-1544: Fix double-checked locking in WebSocketConnectionGroup.
- OPENICF-1549: Update default ConnectorServer.properties.
- OPENICF-1555: Clarify locking behavior in ConnectorServer for Grizzly server lifecycle.
- OPENICF-1561: RCS: Reduce log level for common debug messages.

1.5.19.6 Java RCS

- OPENIDM-16178: IDM recon would fail w/ remote Java connector server.

1.5.19.5 Java RCS

- OPENICF-1516: Failed ICF Search Query confuses total number of search results.
- OPENICF-1520: Java RCS: Connection groups can accumulate many more websockets than they should have.

1.5.19.4 Java RCS

- OPENICF-1485: Java RCS: Non operational ConnectionGroup should be closed and removed.
- OPENICF-1486: Java RCS: Connection housekeeping task may stop running.
- OPENICF-1494: Java RCS: Housekeeping task gets blocked.
- OPENICF-1500: Java RCS: Improve default logging.

1.5.19.3 Java RCS

- OPENICF-1482: Java RCS: fails to reestablish connections to IDM after IDM is restarted.

1.5.19.2 Java RCS

- OPENICF-1467: RCS: endless loops on connection loss and shutdown.

1.5.19.1 Java RCS

No issues specific to the Remote Connector Server were addressed in this release.

1.5.19.0 Java RCS

- OPENICF-1393: Java Connector Server: useSSL property use should be clarified.
- OPENICF-1394: missing connectorserver.scope in connectorserver property file.
- OPENICF-1395: Investigate and clean up the following start up error message.

- OPENICF-1397: Java Connector Server: javax.net.ssl trustStore and keyStore properties should be set.
- OPENICF-1399: restarting IDM with active RCS causes RCS to decrement websocket connection count.
- OPENICF-1400: Java Connector Server: Property name usessl should match docs and code.
- OPENICF-1404: Java connector server proxy config for port is incorrect.
- OPENICF-1407: Java RCS: Incorrect url in Debug message of HttpRequestPacket header for non-SSL.
- OPENICF-1408: Java RCS: NPE when we set proxyHost for client mode.

1.5.18.0 Java RCS

- OPENICF-1371: Java Connector server does not always reestablish closed websockets.
- OPENICF-1390: Java RCS: Prevent use of websockets that are about to be closed.
- OPENICF-1392: Java Connector Server: TTL should be in seconds.

.NET RCS release notes

Subscribe for automatic updates: ICF release notes RSS Feed

Refer to Connector framework release notes for details regarding any changes to the ICF Connector Framework that can affect RCS behavior.

Downloads are available on Backstage.

> **Note**
>
> Unless you have a specific need for the .NET version of the remote connector server (RCS), such as needing to use the PowerShell connector toolkit, we recommend using the Java-based remote connector server instead.

1.5.7.0 .NET RCS

Connection improvements

- .NET remote connector server should be able to initiate connection to IDM (OPENICF-731)
- Client mode should support IDM authentication (OPENICF-1311)
- Unable to start in client mode when no intervals used (OPENICF-1314)
- When we attempt to stop in client mode, the connection is re-initiated (OPENICF-1315)
- ConnectorObject should default the Name to Uid if Name is not present (OPENICF-1318)
- Add the ability to connect to multiple IDM endpoints (OPENICF-1376)
- Connection TTL should be in seconds (OPENICF-1626)
- ConnectionGroup fixes for improved connection handling (OPENICF-1630)
- Handle failure HTTP status codes when requesting OAuth 2.0 tokens (OPENICF-1631)
- Fix handshake timing problem (OPENICF-1682)
- Prevent use of websockets that are about to be closed (OPENICF-1685)
- Ensure that IDM gets notification that a websocket is about to be closed (OPENICF-1700)
- Stagger connection starts if webSocketConnections > 1 (OPENICF-1706)
- SocketClosingSoonException introduces null values that break protobuf3 (OPENICF-2001)
- Improve stability of RCS WebSocket connection management (OPENICF-2008)
- If OAuth token endpoint is defined, .NET RCS still tries to use Basic Auth to connect to ID Cloud (OPENICF-2188)
- Support for HTTP proxy authentication (OPENICF-2197)
- Closing WebSockets are not handled properly (OPENICF-2217)

Configuration improvements

- Separate config properties in the ConnectorServerService.exe.Config (OPENICF-1313)

- Make Pong interval configurable (OPENICF-1362)
- Update default properties values (OPENICF-1628)
- Support for hostId (OPENICF-1512)
- Align HTTP proxy property names with Java RCS (OPENICF-2204)

PowerShell connector now included with .NET connector server

- Embed the PowerShell connector with the .NET connector server (OPENICF-1906)
- Align PowerShell connector version number with the .NET RCS version (OPENICF-1962)
- Integrate the PowerShell samples in the project (OPENICF-1970)
- PowerShell connector: Query might return HTTP 500 when sorting by some properties (OPENICF-2205)
- AD PowerShell samples should filter __NAME__ as a sort key (OPENICF-2172)

Dependency updates and cleanup

- Update and cleanup some dependencies. (OPENICF-1963, OPENICF-1971)
- Upgrade protocol buffer version and package (OPENICF-1836, OPENICF-2173)
- Upgrade .NET framework (OPENICF-1707)
- Fix the Wix project, get rid of legacy dlls (OPENICF-1913)
- Exception upon start due to a missing dependency (OPENICF-1951)

General fixes and improvements

- Sporadic issues managing RCS-hosted connectors through IDM Native Admin Console (OPENICF-2011)
- Query filter on name attribute with pageSize and pagedResultsCookie returns HTTP 500 (OPENICF-1954)
- PagedResultsCookie should be set to null if empty when deserialized from protobuf message (OPENICF-1679)

Connector framework release notes

Subscribe for automatic updates: ICF release notes RSS Feed

Important

Updates to the connector framework can also include security, formatting, and other internal-facing fixes.

1.5.20.30 Framework

No public changes were made to the framework, though a new version was released.

1.5.20.29 Framework

No public changes were made to the framework, though a new version was released.

1.5.20.28 Framework

No public changes were made to the framework, though a new version was released.

1.5.20.26 Framework

- OPENICF-2973: Resolves a race condition within the Java Framework that could result in Groovy `ClassLoader` failures at runtime.
- OPENICF-2751: You can configure global operation rate limits on a per-operation basis for any connector. Learn more in Operation rate limits.

1.5.20.25 Framework

No public changes were made to the framework, though a new version was released.

1.5.20.24 Framework

- OPENICF-2882: The connector framework lets you define nested objects (map objects) in the provisioner `configurationProperties`.

1.5.20.23 Framework

No public changes were made to the framework, though a new version was released.

1.5.20.22 Framework

No public changes were made to the framework, though a new version was released.

1.5.20.21 Framework

- OPENICF-2642: Align Jetty servlet WebSocketConnectionGroup check interval with default Java RCS value.

1.5.20.18 Framework

No public changes were made to the framework, though a new version was released.

1.5.20.15 Framework

- OPENICF-2384: Java Framework: Allow __PASSWORD__ removal via null values.

1.5.20.11 Framework

No public changes were made to the framework, though a new version was released.

1.5.20.8 Framework

- OPENICF-1998: Local/RemoteRequest congruence checks should throw a retryable exception upon failure.

1.5.20.7 Framework

- OPENICF-1883: Java RCS: Improve stability of RCS WebSocket connection management.

1.5.20.6 Framework

- OPENIDM-17535: IDM stack releases that include bundled connectors should continue to work with existing provisioner configuration.

1.5.20.5 Framework

- OPENICF-1855: Investigate handling query 'poison pill' termination via recon automatic retry upon exception receipt.

1.5.20.4 Framework

No public changes were made to the framework, though a new version was released.

1.5.20.3 Framework

- OPENICF-1704: Framework: resetConnectorInfos does not implement intent.
- OPENICF-1730: Client ConnectorInfos cache not refreshed upon RCS instance restart when using RCS Agent.

- OPENICF-1735: Upgrade to groovy 3.0.9.

1.5.20.0 Framework

> **Note**
>
> For a list of security issues addressed in this release, refer to the related Security Advisory in the Knowledge Base.

- OPENICF-1566: Framework: ICF Jetty servlet default maxMessageSize is too small.

1.5.19.6 Framework

No issues specific to the ICF Connector Framework were addressed in this release.

1.5.19.5 Framework

No issues specific to the ICF Connector Framework were addressed in this release.

1.5.19.4 Framework

No issues specific to the ICF Connector Framework were addressed in this release.

1.5.19.3 Framework

No issues specific to the ICF Connector Framework were addressed in this release.

1.5.19.2 Framework

No issues specific to the ICF Connector Framework were addressed in this release.

1.5.19.1 Framework

No issues specific to the ICF Connector Framework were addressed in this release.

1.5.19.0 Framework

> **Note**
>
> Starting in version 1.5.19.0, ICF connectors that previously had external library dependencies now have those dependencies bundled inside the connector.

- OPENICF-1413: Use framework version 1.5.11.0 for ldap-connector to support Java8-compatible release.
- OPENICF-1414: Scripted Groovy (v3) based connectors fail to load with IDM releases prior to 7.0.

1.5.18.0 Framework

> **Note**
>
> Starting in version 1.5.18.0, the ICF Connector Framework and all connectors bundled with IDM share a unified version number.

No issues specific to the ICF Connector Framework were addressed in this release.

Deprecation

The following functionality is deprecated and likely to be removed in a future release.

1.5.20.30

ServiceNow connector

- The `sys_id` ServiceNow connector attribute is deprecated and replaced with the `__NAME__` attribute.
- The `user` object type in the ServiceNow schema is deprecated. Use the native `__ACCOUNT__` object type instead.

1.5.20.21

Google Apps connector

The `__SECONDARY_EMAIL__` user attribute is deprecated. Use the newer attribute `__SECONDARY_EMAILS__`. These two attributes are mutually exclusive.

Earlier than 1.5.18.0

`JAVA_TYPE_DATE` attribute type

Support for the native attribute type `JAVA_TYPE_DATE` is deprecated and will be removed in a future release. This property-level extension is an alias for `string`. Any dates assigned to this extension should be formatted per ISO 8601.

Changed functionality

The following changes may impact existing deployments when you update. Adjust existing scripts, files, configurations, and so on, as necessary.

Connectors

1.5.20.29

Minimum RCS and framework versions

The following connectors now use map objects in their configuration and require RCS and framework versions 1.5.20.24 or later:

- Multiple CSV connector
- SaaS REST Connector
- Workday connector

Removed properties

We removed the following runtime configuration properties:

Connector	Removed configuration properties
Adobe Marketing Cloud connector	`accessToken`
Epic connector	`accessToken` `tokenValidity`
Marketo connector	`accessToken` `tokenExpiration`

1.5.20.22

Database Table connector

- OPENICF-2679: Reduce log level of many operations.

DocuSign connector

- OPENICF-2557: DocuSign connector v2 causes incompatibility with the Synchronize data between IDM and DocuSign sample.

RCS

1.5.20.24

- OPENICF-2882: Support for nested objects (map objects) in the provisioner `configurationProperties`. Any connector that supports map objects must use this RCS version or later.

1.5.20.23

Java 17 required

Running Java RCS requires Java 17.

1.5.20.21

Logging configuration file

The default location for `logback.xml` was moved from `lib/framework/` to `conf/`. You can now edit the path and filename, refer to Logging configuration file.

Framework

1.5.20.24

- OPENICF-2882: The connector framework lets you define nested objects (map objects) in the provisioner `configurationProperties`. Any connector that supports map objects must use this framework version or later.

Known issues

This topic lists issues that remain open at the time of release.

ICF/Connector issues

- OPENICF-1365: LDAP Connector: Triggered livesync using timestamps on a custom object returns HTTP 500
- OPENICF-1905: Database Table Connector: Error when using __NAME__ and pr operator in queryFilter
- OPENICF-1991: Java RCS: No logging when we start RCS with /run and then /install as a Windows service
- OPENICF-2223: MSGraphAPI Connector: query filter using `pr` doesn't work
- OPENICF-2234: ScriptedSQL Connector: Throws "Unable to load FastStringService"
- OPENICF-2235: AS400 Connector: connectionTimeout setting is incorrectly applied to the maxLifetime pooled connections
- OPENICF-2258: MSGraphAPI Connector: Clicking on Directory Role Template gives oData error
- OPENICF-2265: MS Graph API Connector: Invalid filter clause when paging certain filtered results
- OPENICF-2289: SCIM Connector: Update operation fails on Salesforce using scimv2
- OPENICF-2302: LDAP Connector: createFullConfig doesn't throw a uniform error when invalid connection details provided
- OPENICF-2319: SCIM Connector: GoTo system returns non-404 code when trying to read a deleted record
- OPENICF-2349: ServiceNow Connector: query filter with complex expression including negation `!` doesn't work
- OPENICF-2369: MSGraphAPI Connector: Attributes embedded in the additionalDataManager should be exposed upon request
- OPENICF-2399: HubSpot Connector: can return wrong OWNER when single querying
- OPENICF-2403: Marketo Connector: cannot get the list of all leads when the result set is paged by the external system
- OPENICF-2416: SAP Connector: InternalServerError thrown when requesting _pagedResultsOffset which exceeds number of available records
- OPENICF-2495: SCIM Connector: Do not log failure to retrieve AccessToken issued_at time at SEVERE level
- OPENICF-2516: SAP Connector: Unsupported Filter operators are not rejected by the connector
- OPENICF-2518: SAP Connector: Info level logging is overused in this connector
- OPENICF-2539: Dropbox connector: improve error handling that throws java.lang.IllegalStateException
- OPENICF-2541: LDAP Connector: switching between changelog and timestamp livesync throws HTTP 500
- OPENICF-2629: SaasCommon: HTTP client default headers need to be defined per operation basis
- OPENICF-2670: SaaS Common: Admin UI allows saving bad configuration values
- OPENICF-2677: SCIM Connector: attribute duplication on PATCH caused by presence of __NAME__ and `userName` or `displayName` on provisioner file for V2

- **OPENICF-2686: SuccessFactors Connector: default headers should not be defined at HTTP client level**
- **OPENICF-2763: MSGraphAPI Connector: Reconciliation on repo user fails to update target**
- **OPENICF-2775: SCIM Connector: Query on Salesforce displayName attribute throws an error**
- **OPENICF-2793: Adobe Admin Console Connector: Schema incorrectly contains both the ICF __NAME__ attribute and it's associated native source attribute**
- **OPENICF-2794: DocuSign Connector: Schema incorrectly contains both the ICF __NAME__ attribute and it's associated native source attribute**
- **OPENICF-2884: Framework: remove `Script` as valid connectorConfiguration property type**
- **OPENICF-2917: AWS Connector: List on Inline Policy throws an error**
- **OPENICF-2974: Groovy Connector: An invalid path containing multiple slash characters in the scriptRoots is not properly validated**
- **OPENICF-2997: SCIM Connector: Doesn't detect custom resourceType when used against scim.dev**
- **OPENICF-3043: GoogleApps Connector: Test operation should test connectivity to Google via supplied config**
- **OPENICF-3080: Box Connector: Consolidate HTTP client initializer with SaaSCommon**
- **OPENICF-3087: Java RCS: A runtime exception in the Main thread upon RCS startup is not being logged**
- **OPENICF-3089: Remove connector reference to test code on several connectors**
- **OPENICF-3094: Consolidate and improve configuration validation exceptions and handling**
- **OPENICF-3096: SCIM Connector: BeyondTrust patch remove on multivalued map throws 400 [invalidFilter]**
- **OPENICF-3146: Incorrect HTTP 500 response when retrieving invalid or non-existent object via MS Graph API connector**
- **OPENICF-3148: AWS IAM Connector: Query filter on non-existing id returns HTTP 404**
- **OPENICF-3149: Box Connector: Query filter on non-existing id returns HTTP 404**
- **OPENICF-3150: DocuSign Connector: Query filter on non-existing id returns HTTP 404**
- **OPENICF-3151: Duo Connector: Query filter on non-existing id returns HTTP 404**
- **OPENICF-3152: PingOne Connector: Query filter on non-existing id returns HTTP 404**
- **OPENICF-3153: Rest Connector: Query filter on non-existing id returns HTTP 404**
- **OPENICF-3154: Salesforce Connector: Query filter on non-existing id returns HTTP 404**
- **OPENICF-3164: CSV Connector: Validate newlineString property**
- **OPENICF-3165: MSGraphApi Connector: Query filter on non-existing id returns HTTP 404**