OpenICF Connector Configuration Reference

Version 1.5

Lana Frost
László Hordós
Mark Craig

ForgeRock AS
33 New Montgomery St.,
Suite 1500
San Francisco, CA 94105, USA
+1 415-599-1100 (US)
www.forgerock.com

Abstract

Compiled reference documentation that describes all the configurable properties for the connectors that are *supported and tested with OpenIDM 4.0.0*. Note that additional connectors, and the corresponding configuration reference material, are available on the OpenICF Connectors site.

Table of Contents

Preface

This guide shows you how to work with and develop OpenICF connectors, which decouple applications from data resources.

1 Who Should Use this Guide

This guide is written for Java and web developers who use OpenICF to connect to resources from their applications, and who build OpenICF connectors and connector servers.

2 Formatting Conventions

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

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

Command-line, terminal sessions are formatted as follows:

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

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

Program listings are formatted as follows:

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

3 Accessing Documentation Online

ForgeRock publishes comprehensive documentation online:

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

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

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

Core documentation therefore follows a three-phase review process designed to eliminate errors:

- Product managers and software architects review project documentation design with respect to the readers' software lifecycle needs.

- Subject matter experts review proposed documentation changes for technical accuracy and completeness with respect to the corresponding software.

- Quality experts validate implemented documentation changes for technical accuracy, completeness in scope, and usability for the readership.

The review process helps to ensure that documentation published for a ForgeRock release is technically accurate and complete.

Fully reviewed, published core documentation is available at http://backstage.forgerock.com/. Use this documentation when working with a ForgeRock Identity Platform release.

4 Joining the ForgeRock Community

Visit the Community resource center where you can find information about each project, download trial builds, browse the resource catalog, ask and answer questions on the forums, find community events near you, and find the source code for open source software.

Chapter 1
CSV File Connector Installation Instructions

This chapter describes how to install the CSV File Connector and any prerequisites specific to its use.

1.1 Before You Install the 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 can serve as a change log, often containing only those user records that have changed.

1.2 Installing the CSV File Connector

The CSV File Connector is provided in the `openidm/connectors/csvfile-connector-1.5.0.0.jar` file, for local use. If you are running the connector remotely, copy the connector jar file to the `bundles` directory on the Java connector server. No additional installation steps are required for the CSV File Connector.

OpenIDM provides a sample CSV File Connector configuration at `openidm/samples/sample4/conf/provisioner.openicf-csv.json`. Edit that sample to specify *at least* the path to the CSV file (`"filePath"`) and the attribute that will serve as the primary key (`"uniqueAttribute"`) for the user accounts. Additional configuration properties are as described in the *Configuration chapter*.

Chapter 2
CSV File Connector Configuration

This chapter describes the structure and configuration of the CSV File Connector, the operations that are supported by the connector, and the connector schema.

The CSV File Connector does not support connector pooling.

2.1 CSV File Connector Reference Object

The CSV File Connector has the following unique identifiers, expressed here in JSON format.

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

You can use OpenIDM to generate this configuration automatically when you configure the connector. Alternatively, you can copy this section and paste it directly into your connector configuration file (`provisioner.openicf-`*`connector-name`*`.json`).

2.2 OpenICF Interfaces Implemented by the CSV File Connector

The CSV File Connector implements the following OpenICF interfaces.

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

Batch
: Execute a series of operations in a single request.

Create
: Creates an object and its `uid`.

Delete
: Deletes an object, referenced by its `uid`.

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

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

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

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

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

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

Test
: Tests the connector configuration. Testing a configuration checks all elements of the environment that are referred to by the configuration are

available. For example, the connector might make a physical connection to a host that is specified in the configuration to verify that it exists and that the credentials that are specified in the configuration are valid.

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

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

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

2.3 CSV File Connector Configuration

The CSV File Connector has the following configurable properties.

2.3.1 Configuration Properties

Property	**Type**	**Default**	**Encrypted** [a]	**Required** [b]
`csvFile`	`File`	`null`		Yes
Full path to the CSV file				
`headerUid`	`String`	`uid`		No
Name of the uid column as found in the CSV file				
`quoteCharacter`	`String`	`"`		No
Character used to quote fields				
`headerPassword`	`String`	`password`		No
Name of the password column as found in the CSV file				
`fieldDelimiter`	`String`	`,`		No
Character used to delimit columnar fields				

Property	Type	Default	Encrypted [a]	Required [b]
`syncFileRetentionCount`	`int`	`3`		No
Number of sync history files to retain				
`newlineString`	`String`			No
Character(s) used to terminate a line in the CSV file				
`headerName`	`String`	`username`		No
Name of the user name column as found in the CSV file				

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

Chapter 3

Database Table Connector Installation Instructions

This chapter describes how to install the Database Table Connector and any prerequisites specific to its use.

3.1 Before You Install the Database Table Connector

The Database Table Connector enables provisioning to a single table in a JDBC database. Before you set up the Database Table Connector, your JDBC database must be up and running, and the required JDBC driver must be available in the `openidm/bundle` directory.

Download the driver that corresponds to your database:

- For a MySQL database, download MySQL Connector/J, version 5.1 or later from the MySQL website.
- For an MS SQL database, download the JDBC Driver 4.0 for SQL Server (`sqljdbc_4.0.2206.100_enu.tar.gz`) from Microsoft's download site.
- For an Oracle DB database, create an Oracle DB driver from two separate jar files, as described in To Set Up OpenIDM With Oracle Database in the *OpenIDM Installation Guide*.

3.2 Installing the Database Table Connector

The Database Table Connector is provided in the `openidm/connectors/databasetable-connector-1.1.0.1.jar` file, for local use. If you are running the connector remotely, copy the connector jar file to the `bundles` directory on the Java connector server.

OpenIDM provides a sample Database Table Connector configuration at `openidm/samples/provisioners/provisioner.openicf-contractordb.json`. The corresponding data definition language file is provided in `openidm/samples/provisioners/provisioner.openicf-contractordb.sql`. Edit this sample configuration, to specify at least the following properties.

- The JDBC `database` that contains the table to which you are provisioning
- The `table` in the database that contains the user accounts
- The `keyColumn` value that is used as the unique identifier for rows in the table

Additional configuration properties are as described in the *Configuration chapter*.

3.3 Configuring Connection Pooling

The Database Table Connector supports connection pooling, which can substantially improve the performance of the connector. The basic connection pooling configuration is described in the *Connection Pooling Configuration Appendix*.

Chapter 4

Database Table Connector Configuration

This chapter describes the structure and configuration of the Database Table Connector, the operations that are supported by the connector, and the connector schema.

The Database Table Connector supports connector pooling for improved performance and scalability. For information about configuring connector pooling, see the Configuring Connector Pooling.

4.1 Database Table Connector Reference Object

The Database Table Connector has the following unique identifiers, expressed here in JSON format.

```
"connectorRef" : {
      "bundleName"    : "org.forgerock.openicf.connectors.databasetable-connector",
      "bundleVersion" : "1.1.0.1",
      "connectorName" : "org.identityconnectors.databasetable.DatabaseTableConnector"
      }
```

You can use OpenIDM to generate this configuration automatically when you configure the connector. For more information, see *Configuring Connectors* in the *OpenIDM Integrator's Guide*. Alternatively, you can copy this section and paste it directly into your connector configuration file (`provisioner.openicf-`*`connector-name`*`.json`).

4.2 OpenICF Interfaces Implemented by the Database Table Connector

The Database Table Connector implements the following OpenICF interfaces.

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

Create
: Creates an object and its `uid`.

Delete
: Deletes an object, referenced by its `uid`.

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

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

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

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

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

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

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

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

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

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

4.3 Database Table Connector Configuration

The Database Table Connector has the following configurable properties.

4.3.1 Configuration Properties

Property	**Type**	**Default**	[a]	**Required** [b]	
`quoting`	`String`				
Select whether database column names for this resource should be quoted, and the quoting characters. By default, database column names are not quoted (None). For other selections (Single, Double, Back, or Brackets), column names will appear between single quotes, double quotes, back quotes, or brackets in the SQL generated to access the database.					
`host`	`String`				
Enter the name of the host where the database is running.					
`port`	`String`				
Enter the port number the database server is listening on.					

Property	Type	Default	[a]	Required [b]	
user	String				
Enter the name of the mandatory Database user with permission to account table.					
password	GuardedSt	null			
Enter a user account that has permission to access accounts table.					
database	String				
Enter the name of the database on the database server that contains the table.					
table	String				
Enter the name of the table in the database that contains the accounts.					
keyColumn	String				
This mandatory column value will be used as the unique identifier for rows in the table.					
passwordColumn	String				
Enter the name of the column in the table that will hold the password values. If empty, no validation on resource and passwords are activated.					
jdbcDriver	String	oracle. jdbc. driver. OracleDriv			
Specify the JDBC Driver class name. Oracle is oracle.jdbc.driver.OracleDriver. MySQL is					

Property	Type	Default	[a]	Required[b]	
org.gjt.mm.mysql.Driver. Could be empty if datasource is provided.					
`jdbcUrlTemplate`	`String`	`jdbc:oracl %h:%p:%d`			
Specify the JDBC Driver Connection URL. Oracle template is jdbc:oracle:thin:@[host]:[port(1521)]:[DB]. MySQL template is jdbc:mysql://[host]:[port(3306)]/[db], for more info, read the JDBC driver documentation. Could be empty if datasource is provided.					
`enableEmptyString`	`boolean`	`false`			
Select to enable support for writing an empty strings, 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.					
`rethrowAllSQLExcepti`	`boolean`	`true`			
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`			
Select to retrieve Timestamp data type of the columns in java.sql.Timestamp format from the database table.					
`allNative`	`boolean`	`false`			
Select to retrieve all data type of the columns in a native format from the database table.					
`validConnectionQuery`	`String`	`null`			
There can be specified the check connection alive query. If empty, default implementation will test it using the switch on/off					

Property	Type	Default	[a]	Required [b]
the autocommit. Some select 1 from dummy table could be more efficient.				
`changeLogColumn`	`String`			Sync
The change log column store the latest change time. Providing this value the Sync capabilities are activated.				
`datasource`	`String`			
Enter the JDBC Data Source Name/Path to connect to the Oracle server. If specified, connector will only try to connect using Datasource and ignore other resource parameters specified.The example value is: jdbc/SampleDataSourceName				
`jndiProperties`	`String[]`	`null`		
Could be empty or enter the JDBC JNDI Initial context factory, context provider in a format: key = value.				

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

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

Chapter 5

Groovy Connector Toolkit Installation Instructions

This chapter describes how to install the Groovy Connector Toolkit and any prerequisites specific to its use.

5.1 Before You Install the Groovy Connector Toolkit

The Groovy Connector Toolkit is a generic scripted connector that enables you to run Groovy scripts on any external resource. The Groovy Connector Toolkit is provided with OpenIDM, in the JAR `openidm/connectors/groovy-connector-1.4.2.0.jar`.

Sample scripted connector implementations are provided in the Maven repository. The following sample implementations are provided:

Scripted SQL Connector
: The Scripted SQL Connector uses Groovy scripts to interact with a JDBC database.

Scripted REST Connector
: The Scripted REST Connector enables you to connect to any resource over HTTP/REST. The connector creates the HTTP/REST context (specifying the content type, authentication mode, encoding, and so on), and manages the connection.

The connector relies on the Groovy scripting language and its RESTClient package. The Groovy scripts are responsible for sending requests and processing results.

The following sample Groovy script creates a new user in OpenDJ, using OpenDJ's REST API:

```
connection.put(
    path: '/users/' + name,
    headers: ['If-None-Match': '*'],
    contentType: JSON,
    requestContentType: JSON,
    body: JsonOutput.toJson(
        [_id : name,
            name : [
                familyName: "Doe",
                givenName : "John"
            ],
            displayName: "John Doe"
        ]
    )
);
```

Scripted CREST Connector

The Scripted CREST sample is a generic implementation that takes a schema configuration file to define the attribute mapping from the OpenICF connector object to the CREST resource. In the sample, the schema configuration file has the same syntax as the OpenIDM provisioner configuration file (for example, `provisioner.openicf-scriptedcrest.json`), which defines the mapping between OpenIDM and the OpenICF connector object. Most CRUD operations should work with the sample scripts, however the Scripted CREST sample is not intended to work "out of the box". It is expected that the scripts will be customized to address the requirements of your deployment. The sample scripts are a good starting point on which to base your customization.

Depending on the implementation that you use, the Groovy Connector Toolkit has specific dependencies, as described in the following sections. If you use the connector that is bundled with OpenIDM, the required OSGi-ready dependencies are bundled and do not have to be downloaded. If you download the connector outside of OpenIDM, you must download the dependencies required for your connector implementation.

5.1.1 ScriptedSQL Connector Dependencies

The Scripted SQL Connector dependencies should be placed in either the `lib` folder, or the `bundle` folder, depending on the required OSGi compatibility. If the dependency is "OSGi-ready" it can be placed in the `bundle` folder, otherwise it must be placed in the `lib` folder. The OSGi-ready jars described here, require Java version 7.

Non OSGi-ready jars:
: Create a `lib/` folder in your OpenIDM installation directory.

```
$ mkdir /path/to/openidm/lib
```

Download the following dependencies and copy them to the `openidm/lib` folder.

- Apache Tomcat Juli, `tomcat-juli-7.0.55.jar` (org.apache.tomcat:tomcat-juli:7.0.55)
- Tomcat JDBC Pool Package, `tomcat-jdbc-7.0.53.jar` (org.apache.tomcat:tomcat-jdbc:7.0.53)

OSGi-ready jars:
: Download the following dependencies and place them in the `openidm/bundle` folder.

- Tomcat Core Logging Package (Juli), `tomcat-juli-8.0.12.jar` (org.apache.tomcat:juli:8.0.12)
- OSGi-ready Tomcat JDBC Pool Package, `tomcat-jdbc-8.0.12.jar` (org.apache.tomcat:tomcat-jdbc:8.0.12)

The Groovy Connector Toolkit scripted SQL implementation uses Groovy scripts to interact with a JDBC database. Before you set up the connector, your JDBC database must be up and running. The required JDBC driver must be available in either the `openidm/bundle` directory (if it is OSGi-ready) or in the `openidm/lib` directory (if it is not OSGi-ready).

Download the driver that corresponds to your database:

- For a MySQL database, download MySQL Connector/J, version 5.1 or later from the MySQL website.
- For an MS SQL database, download the JDBC Driver 4.0 for SQL Server (`sqljdbc_4.0.2206.100_enu.tar.gz`) from Microsoft's download site.
- For an Oracle DB database, create an Oracle DB driver from two separate jar files, as described in To Set Up OpenIDM With Oracle Database in the *OpenIDM Installation Guide*.

5.1.2 Scripted REST Connector Dependencies

Download the following dependencies and copy them to the `openidm/bundle` folder.

- HttpComponents Client (OSGi bundle), `httpclient-osgi-4.3.6.jar` (org.apache.httpcomponents:httpclient-osgi:4.3.6)
- HttpComponents Core (OSGi bundle), `httpcore-osgi-4.3.2.jar` (org.apache.httpcomponents:httpcore-osgi:4.3.2)

5.1.3 ScriptedCREST Connector Dependencies

Download the following dependencies and copy them to the `openidm/bundle` folder.

- HttpComponents AsyncClient (OSGi bundle), `httpasyncclient-osgi-4.0.2.jar` (org.apache.httpcomponents:httpasyncclient-osgi:4.0.2)
- HttpComponents Client (OSGi bundle), `httpclient-osgi-4.3.3.jar` (org.apache.httpcomponents:httpclient-osgi:4.3.3)
- HttpComponents Core (OSGi bundle), `httpcore-osgi-4.3.2.jar` (org.apache.httpcomponents:httpcore-osgi:4.3.2)

5.2 Installing the Groovy Connector Toolkit

The Groovy Connector Toolkit is provided in the `openidm/connectors/groovy-connector-1.4.2.0.jar` file, for local use. If you are running the connector remotely, copy the connector jar file to the `openicf/bundles` directory on the Java connector server. Also, copy any dependencies (described in the previous section) to the `lib` directory on the remote connector server. Generate a connector configuration for your Groovy connector implementation.

A sample connector configuration for a scripted SQL implementation is provided in `/path/to/openidm/samples/sample3`. The following excerpt of the configuration shows the connector bundle details and the properties that are used to connect to the JDBC database:

```
{
   "name" : "scriptedsql",
   "connectorRef" : {
       "bundleName" : "org.forgerock.openicf.connectors.groovy-connector",
       "bundleVersion" : "1.4.2.0",
       "connectorName" : "org.forgerock.openicf.connectors.scriptedsql.ScriptedSQLConnector"
   },
   ...
   "configurationProperties" : {
        "username" : "root",
        "password" : "",
        "driverClassName" : "com.mysql.jdbc.Driver",
        "url" : "jdbc:mysql://localhost:3306/HRDB",
        "autoCommit" : true,
        "reloadScriptOnExecution" : false,
        "authenticateScriptFileName" : "AuthenticateScript.groovy",
        "createScriptFileName" : "CreateScript.groovy",
        "testScriptFileName" : "TestScript.groovy",
        "searchScriptFileName" : "SearchScript.groovy",
        "deleteScriptFileName" : "DeleteScript.groovy",
        "updateScriptFileName" : "UpdateScript.groovy",
        "syncScriptFileName" : "SyncScript.groovy",
        "schemaScriptFileName" : "SchemaScript.groovy",
        "classpath" : [
            "&{launcher.project.location}/tools"
        ]
    },
...
```

The Groovy scripts required for the sample are located in the `path/to/openidm/samples/sample3/tools` directory and can be customized for your deployment.

Edit the `"configurationProperties"` in the connector configuration file to match your JDBC database.

For details of all the configurable properties for this connector, see the *Configuration chapter*.

5.3 Configuring Connector Pooling

The Groovy Connector Toolkit supports connection pooling, which can substantially improve the performance of the connector. The basic connection pooling configuration is described in the *Connection Pooling Configuration Appendix*.

5.4 Configuring Scripted CREST Connector Pooling

TO BE WRITTEN MANUALLY

Chapter 6

Scripted Groovy Connector Configuration

This chapter describes the structure and configuration of the Scripted Groovy Connector, the operations that are supported by the connector, and the connector schema.

The Scripted Groovy Connector does not support connector pooling.

6.1 Scripted Groovy Connector Reference Object

The Scripted Groovy Connector has the following unique identifiers, expressed here in JSON format.

```
"connectorRef" : {
      "bundleName"    : "org.forgerock.openicf.connectors.groovy-connector",
      "bundleVersion" : "1.4.2.0",
      "connectorName" : "org.forgerock.openicf.connectors.groovy.ScriptedConnector"
      }
```

You can use OpenIDM to generate this configuration automatically when you configure the connector. Alternatively, you can copy this section and paste it directly into your connector configuration file (`provisioner.openicf-`*`connector-name`*`.json`).

6.2 OpenICF Interfaces Implemented by the Scripted Groovy Connector

The Scripted Groovy Connector implements the following OpenICF interfaces.

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

Create
: Creates an object and its `uid`.

Delete
: Deletes an object, referenced by its `uid`.

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

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

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

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

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

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

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

Test
: Tests the connector configuration. Testing a configuration checks all elements of the environment that are referred to by the configuration are

available. For example, the connector might make a physical connection to a host that is specified in the configuration to verify that it exists and that the credentials that are specified in the configuration are valid.

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

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

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

6.3 Scripted Groovy Connector Configuration

The Scripted Groovy Connector has the following configurable properties.

6.3.1 Operation Script Files Properties

Property	**Type**	**Default**	**Encrypted** [a]	**Required** [b]
`authenticateScriptFileName`	`String`	`null`		Authenticate
The name of the file used to perform the AUTHENTICATE operation.				
`deleteScriptFileName`	`String`	`null`		Delete
The name of the file used to perform the DELETE operation.				
`schemaScriptFileName`	`String`	`null`		Schema
The name of the file used to perform the SCHEMA operation.				
`customizerScriptFileName`	`String`	`null`		No
The script used to customize some function of the connector. Read the documentation for more details.				
`resolveUsernameScriptFileNam`	`String`	`null`		Resolve Username

Property	Type	Default	Encrypted [a]	Required [b]
The name of the file used to perform the RESOLVE_USERNAME operation.				
`testScriptFileName`	`String`	`null`		Test
The name of the file used to perform the TEST operation.				
`updateScriptFileName`	`String`	`null`		Update
The name of the file used to perform the UPDATE operation.				
`searchScriptFileName`	`String`	`null`		Get Search
The name of the file used to perform the SEARCH operation.				
`scriptOnResourceScriptFileNa`	`String`	`null`		Script On Resource
The name of the file used to perform the RUNSCRIPTONRESOURCE operation.				
`createScriptFileName`	`String`	`null`		Create
The name of the file used to perform the CREATE operation.				
`syncScriptFileName`	`String`	`null`		Sync
The name of the file used to perform the SYNC operation.				

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

6.3.2 Groovy Engine configuration Properties

Property	Type	Default	Encrypted [a]	Required [b]
`warningLevel`	`int`	1		No
Warning Level of the compiler				

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

Property	Type	Default	Encrypted [a]	Required [b]
`verbose`	`boolean`	`false`		No
If true, the compiler should produce action information				
`recompileGroovySource`	`boolean`	`false`		No
If set to true recompilation is enabled				
`tolerance`	`int`	`10`		No
The error tolerance, which is the number of non-fatal errors (per unit) that should be tolerated before compilation is aborted.				

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

6.3.3 Configuration Properties

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

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

Chapter 7

Scripted Poolable Groovy Connector Configuration

This chapter describes the structure and configuration of the Scripted Poolable Groovy Connector, the operations that are supported by the connector, and the connector schema.

The Scripted Poolable Groovy Connector supports connector pooling for improved performance and scalability. For information about configuring connector pooling, see the Configuring Connector Pooling.

7.1 Scripted Poolable Groovy Connector Reference Object

The Scripted Poolable Groovy Connector has the following unique identifiers, expressed here in JSON format.

```
"connectorRef" : {
      "bundleName"    : "org.forgerock.openicf.connectors.groovy-connector",
      "bundleVersion" : "1.4.2.0",
      "connectorName" : "org.forgerock.openicf.connectors.groovy.ScriptedPoolableConnector"
      }
```

You can use OpenIDM to generate this configuration automatically when you configure the connector. Alternatively, you can copy this section and paste it directly into your connector configuration file (`provisioner.openicf-`*`connector-name`*`.json`).

7.2 OpenICF Interfaces Implemented by the Scripted Poolable Groovy Connector

The Scripted Poolable Groovy Connector implements the following OpenICF interfaces.

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

Create
: Creates an object and its `uid`.

Delete
: Deletes an object, referenced by its `uid`.

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

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

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

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

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

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

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

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

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

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

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

7.3 Scripted Poolable Groovy Connector Configuration

The Scripted Poolable Groovy Connector has the following configurable properties.

7.3.1 Operation Script Files Properties

Property	Type	Default	Encrypted [a]	Required [b]
`authenticateScriptFileName`	`String`	`null`		Authenticate
The name of the file used to perform the AUTHENTICATE operation.				
`deleteScriptFileName`	`String`	`null`		Delete
The name of the file used to perform the DELETE operation.				
`schemaScriptFileName`	`String`	`null`		Schema
The name of the file used to perform the SCHEMA operation.				

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

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

7.3.2 Groovy Engine configuration Properties

Property	Type	Default	Encrypted [a]	Required [b]
`warningLevel`	`int`	`1`		No
Warning Level of the compiler				
`minimumRecompilationInterval`	`int`	`100`		No
Sets the minimum of time after a script can be recompiled.				
`scriptRoots`	`String[]`	`null`		Yes
The root folder to load the scripts from. If the value is null or empty the classpath value is used.				
`debug`	`boolean`	`false`		No
If true, debugging code should be activated				
`targetDirectory`	`File`	`null`		No
Directory into which to write classes.				
`disabledGlobalASTTransformat`	`String[]`	`null`		No
Sets a list of global AST transformations which should not be loaded even if they are defined in META-INF/org.codehaus.groovy.transform.ASTTransformation files. By default, none is disabled.				
`classpath`	`String[]`	`[]`		No
Classpath for use during compilation.				
`scriptExtensions`	`String[]`	`['groovy']`		No
Description is not available				
`sourceEncoding`	`String`	`UTF-8`		No
Encoding for source files				

Property	Type	Default	Encrypted [a]	Required [b]
`scriptBaseClass`	`String`	`null`		No
Base class name for scripts (must derive from Script)				
`verbose`	`boolean`	`false`		No
If true, the compiler should produce action information				
`recompileGroovySource`	`boolean`	`false`		No
If set to true recompilation is enabled				
`tolerance`	`int`	`10`		No
The error tolerance, which is the number of non-fatal errors (per unit) that should be tolerated before compilation is aborted.				

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

7.3.3 Configuration Properties

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

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

Chapter 8
Scripted REST Connector Configuration

This chapter describes the structure and configuration of the Scripted REST Connector, the operations that are supported by the connector, and the connector schema.

The Scripted REST Connector does not support connector pooling.

8.1 Scripted REST Connector Reference Object

The Scripted REST Connector has the following unique identifiers, expressed here in JSON format.

```
"connectorRef" : {
      "bundleName"    : "org.forgerock.openicf.connectors.groovy-connector",
      "bundleVersion" : "1.4.2.0",
      "connectorName" : "org.forgerock.openicf.connectors.scriptedrest.ScriptedRESTConnector"
      }
```

You can use OpenIDM to generate this configuration automatically when you configure the connector. Alternatively, you can copy this section and paste it directly into your connector configuration file (`provisioner.openicf-`*`connector-name`*`.json`).

8.2 OpenICF Interfaces Implemented by the Scripted REST Connector

The Scripted REST Connector implements the following OpenICF interfaces.

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

Create
: Creates an object and its `uid`.

Delete
: Deletes an object, referenced by its `uid`.

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

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

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

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

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

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

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

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

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

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

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

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

8.3 Scripted REST Connector Configuration

The Scripted REST Connector has the following configurable properties.

8.3.1 Groovy Engine configuration Properties

Property	**Type**	**Default**	**Encrypted** [a]	**Required** [b]
`minimumRecompilationInterval`	`int`	`100`		No
Sets the minimum of time after a script can be recompiled.				
`scriptRoots`	`String[]`	`null`		Yes
The root folder to load the scripts from. If the value is null or empty the classpath value is used.				
`debug`	`boolean`	`false`		No
If true, debugging code should be activated				
`disabledGlobalASTTransformat`	`String[]`	`null`		No
Sets a list of global AST transformations which should not be loaded even if they are defined in META-INF/org.codehaus.groovy.transform.ASTTransformation files. By default, none is disabled.				
`scriptExtensions`	`String[]`	`['groovy']`		No

Property	Type	Default	Encrypted [a]	Required [b]
Description is not available				
`sourceEncoding`	`String`	`UTF-8`		No
Encoding for source files				
`scriptBaseClass`	`String`	`null`		No
Base class name for scripts (must derive from Script)				
`verbose`	`boolean`	`false`		No
If true, the compiler should produce action information				
`tolerance`	`int`	`10`		No
The error tolerance, which is the number of non-fatal errors (per unit) that should be tolerated before compilation is aborted.				
`warningLevel`	`int`	`1`		No
Warning Level of the compiler				
`targetDirectory`	`File`	`null`		No
Directory into which to write classes.				
`classpath`	`String[]`	`[]`		No
Classpath for use during compilation.				
`recompileGroovySource`	`boolean`	`false`		No
If set to true recompilation is enabled				

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

8.3.2 Operation Script Files Properties

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

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

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

8.3.3 Configuration Properties

Property	Type	Default	Encrypted [a]	Required [b]
`serviceAddress`	`URI`	`null`		Yes
Description is not available				
`customConfiguration`	`String`	`null`		No
Custom Configuration script for Groovy ConfigSlurper				
`customSensitiveConfiguratior`	`GuardedString`	`null`	Yes	No
Custom Sensitive Configuration script for Groovy ConfigSlurper				
`defaultAuthMethod`	`String`	`BASIC`		No
Description is not available				
`proxyAddress`	`URI`	`null`		No
Description is not available				
`defaultRequestHeaders`	`String[]`	`null`		No

Property	Type	Default	Encrypted [a]	Required [b]
Description is not available				
`defaultContentType`	`String`	`application/json`		No
Description is not available				

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

8.3.4 Basic Configuration Properties Properties

Property	Type	Default	Encrypted [a]	Required [b]
`username`	`String`	`null`		No
Description is not available				
`password`	`GuardedString`	`null`	Yes	No
An example GuardedString property				

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

Chapter 9

Scripted CREST Connector Configuration

This chapter describes the structure and configuration of the Scripted CREST Connector, the operations that are supported by the connector, and the connector schema.

The Scripted CREST Connector supports connector pooling for improved performance and scalability. For information about configuring connector pooling, see the Configuring Connector Pooling.

9.1 Scripted CREST Connector Reference Object

The Scripted CREST Connector has the following unique identifiers, expressed here in JSON format.

```
"connectorRef" : {
      "bundleName"    : "org.forgerock.openicf.connectors.groovy-connector",
      "bundleVersion" : "1.4.2.0",
      "connectorName" : "org.forgerock.openicf.connectors.scriptedcrest.ScriptedCRESTConnector"
      }
```

You can use OpenIDM to generate this configuration automatically when you configure the connector. Alternatively, you can copy this section and paste it directly into your connector configuration file (`provisioner.openicf-`*`connector-name`*`.json`).

9.2 OpenICF Interfaces Implemented by the Scripted CREST Connector

The Scripted CREST Connector implements the following OpenICF interfaces.

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

Create
: Creates an object and its `uid`.

Delete
: Deletes an object, referenced by its `uid`.

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

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

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

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

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

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

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

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

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

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

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

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

9.3 Scripted CREST Connector Configuration

The Scripted CREST Connector has the following configurable properties.

9.3.1 Groovy Engine configuration Properties

Property	**Type**	**Default**	**Encrypted** [a]	**Required** [b]
`minimumRecompilationInterval`	`int`	`100`		No
Sets the minimum of time after a script can be recompiled.				
`scriptRoots`	`String[]`	`null`		Yes
The root folder to load the scripts from. If the value is null or empty the classpath value is used.				
`debug`	`boolean`	`false`		No
If true, debugging code should be activated				
`disabledGlobalASTTransformat`	`String[]`	`null`		No
Sets a list of global AST transformations which should not be loaded even if they are defined in META-INF/org.codehaus.groovy.transform.ASTTransformation files. By default, none is disabled.				
`scriptExtensions`	`String[]`	`['groovy']`		No

Property	Type	Default	Encrypted [a]	Required [b]
Description is not available				
`sourceEncoding`	`String`	`UTF-8`		No
Encoding for source files				
`scriptBaseClass`	`String`	`null`		No
Base class name for scripts (must derive from Script)				
`verbose`	`boolean`	`false`		No
If true, the compiler should produce action information				
`tolerance`	`int`	`10`		No
The error tolerance, which is the number of non-fatal errors (per unit) that should be tolerated before compilation is aborted.				
`warningLevel`	`int`	`1`		No
Warning Level of the compiler				
`targetDirectory`	`File`	`null`		No
Directory into which to write classes.				
`classpath`	`String[]`	`[]`		No
Classpath for use during compilation.				
`recompileGroovySource`	`boolean`	`false`		No
If set to true recompilation is enabled				

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

9.3.2 Operation Script Files Properties

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

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

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

9.3.3 Configuration Properties

Property	Type	Default	Encrypted [a]	Required [b]
serviceAddress	URI	null		Yes
Description is not available				
customConfiguration	String	null		No
Custom Configuration script for Groovy ConfigSlurper				
customSensitiveConfiguratior	GuardedString	null	Yes	No
Custom Sensitive Configuration script for Groovy ConfigSlurper				
defaultAuthMethod	String	BASIC		No
Description is not available				
proxyAddress	URI	null		No
Description is not available				

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

9.3.4 Basic Configuration Properties Properties

Property	Type	Default	Encrypted [a]	Required [b]
`username`	`String`	`null`		No
Description is not available				
`password`	`GuardedString`	`null`	Yes	No
An example GuardedString property				

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

Chapter 10
Scripted SQL Connector Configuration

This chapter describes the structure and configuration of the Scripted SQL Connector, the operations that are supported by the connector, and the connector schema.

The Scripted SQL Connector does not support connector pooling.

10.1 Scripted SQL Connector Reference Object

The Scripted SQL Connector has the following unique identifiers, expressed here in JSON format.

```
"connectorRef" : {
      "bundleName"    : "org.forgerock.openicf.connectors.groovy-connector",
      "bundleVersion" : "1.4.2.0",
      "connectorName" : "org.forgerock.openicf.connectors.scriptedsql.ScriptedSQLConnector"
      }
```

You can use OpenIDM to generate this configuration automatically when you configure the connector. Alternatively, you can copy this section and paste it directly into your connector configuration file (`provisioner.openicf-`*`connector-name`*`.json`).

10.2 OpenICF Interfaces Implemented by the Scripted SQL Connector

The Scripted SQL Connector implements the following OpenICF interfaces.

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

Create
: Creates an object and its `uid`.

Delete
: Deletes an object, referenced by its `uid`.

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

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

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

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

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

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

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

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

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

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

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

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

10.3 Scripted SQL Connector Configuration

The Scripted SQL Connector has the following configurable properties.

10.3.1 Groovy Engine configuration Properties

Property	**Type**	**Default**	**Encrypted** [a]	**Required** [b]
`minimumRecompilationInterval`	`int`	`100`		No
Sets the minimum of time after a script can be recompiled.				
`scriptBaseClass`	`String`	`null`		No
Base class name for scripts (must derive from Script)				
`verbose`	`boolean`	`false`		No
If true, the compiler should produce action information				
`tolerance`	`int`	`10`		No
The error tolerance, which is the number of non-fatal errors (per unit) that should be tolerated before compilation is aborted.				
`classpath`	`String[]`	`[]`		No
Classpath for use during compilation.				

Property	Type	Default	Encrypted [a]	Required [b]
`recompileGroovySource`	`boolean`	`false`		No
If set to true recompilation is enabled				
`scriptRoots`	`String[]`	`null`		Yes
The root folder to load the scripts from. If the value is null or empty the classpath value is used.				
`debug`	`boolean`	`false`		No
If true, debugging code should be activated				
`disabledGlobalASTTransformat`	`String[]`	`null`		No
Sets a list of global AST transformations which should not be loaded even if they are defined in META-INF/org.codehaus.groovy.transform.ASTTransformation files. By default, none is disabled.				
`scriptExtensions`	`String[]`	`['groovy']`		No
Description is not available				
`sourceEncoding`	`String`	`UTF-8`		No
Encoding for source files				
`warningLevel`	`int`	`1`		No
Warning Level of the compiler				
`targetDirectory`	`File`	`null`		No
Directory into which to write classes.				

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

10.3.2 Configuration Properties

Property	Type	Default	Encrypted [a]	Required [b]
`password`	`String`	`null`	Yes	No
The connection password to be passed to our JDBC driver to establish a connection. Note that method DataSource.getConnection(username,password) by default will not use credentials passed into the method, but will use the ones configured here. See alternateUsernameAllowed property for more details.				
`customConfiguration`	`String`	`null`		No
Custom Configuration script for Groovy ConfigSlurper				
`customSensitiveConfiguratio`	`GuardedString`	`null`	Yes	No
Custom Sensitive Configuration script for Groovy ConfigSlurper				
`maxIdle`	`int`	`100`		No
The maximum number of connections that should be kept in the pool at all times. Default value is maxActive:100 Idle connections are checked periodically (if enabled) and connections that been idle for longer than minEvictableIdleTimeMillis will be released. (also see testWhileIdle)				
`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.sql.Connection object. The default value is null.				
`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.				
`validationQuery`	`String`	`null`		No

Property	Type	Default	Encrypted [a]	Required [b]
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)				
`testOnConnect`	`boolean`	`false`		No
Description is not available				
`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.				
`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.				
`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.				
`minIdle`	`int`	`10`		No
The minimum number of established connections that should be kept in the pool at all times. The connection pool can shrink below this number if validation queries fail. Default value is derived from initialSize:10 (also see testWhileIdle)				
`dataSourceJNDI`	`String`	`null`		No
The JNDI name for a data source to be looked up in JNDI and then used to establish connections to the database. See the dataSource attribute. Default value is null				

Property	Type	Default	Encrypted [a]	Required [b]
`validationInterval`	`long`	`30000`		No
avoid excess validation, only run validation at most at this frequency - time in milliseconds. If a connection is due for validation, but has been validated previously within this interval, it will not be validated again. The default value is 30000 (30 seconds).				
`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.				
`accessToUnderlyingConnectior`	`boolean`	`true`		No
Property not used. Access can be achieved by calling unwrap on the pooled connection. see javax.sql.DataSource interface, or call getConnection through reflection or cast the object as javax.sql.PooledConnection				
`url`	`String`	`null`		No
Description is not available				
`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)				
`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				

Property	**Type**	**Default**	**Encrypted** [a]	**Required** [b]
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.				
`initSQL`	`String`	`null`		No
A custom query to be run when a connection is first created. The default value is null.				
`validatorClassName`	`String`	`null`		No
The name of a class which implements the org.apache.tomcat.jdbc.pool.Validator interface and provides a no-arg constructor (may be implicit). If specified, the class will be used to create a Validator instance which is then used instead of any validation query to validate connections. The default value is null. An example value is com.mycompany.project.SimpleValidator.				
`defaultCatalog`	`String`	`null`		No
The default catalog of connections created by this pool.				
`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				
`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.				
`useDisposableConnectionFacad`	`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 [a]	Required [b]
`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				
`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.				
`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.				
`numTestsPerEvictionRun`	`int`	`0`		No
Property not used in tomcat-jdbc-pool.				
`name`	`String`	`Tomcat Connection Pool[1-2107444648]`		No
Description is not available				
`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)				

Property	Type	Default	Encrypted [a]	Required [b]
`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.)				
`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.				
`jmxEnabled`	`boolean`	`true`		No
Register the pool with JMX or not. The default value is true.				
`validationQueryTimeout`	`int`	`-1`		No
The timeout in seconds before a connection validation queries fail. This works by calling java.sql.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.				
`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)				
`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.				
`useLock`	`boolean`	`false`		No
Description is not available				

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

Property	Type	Default	Encrypted [a]	Required [b]
connections from applications that fail to close a connection. See also logAbandoned The default value is false.				
timeBetweenEvictionRunsMilli	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).				
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).				
initialSize	int	10		No
The initial number of connections that are created when the pool is started. Default value is 10				
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.				

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

10.3.3 Operation Script Files Properties

Property	Type	Default	Encrypted [a]	Required [b]
updateScriptFileName	String	null		Update
The name of the file used to perform the UPDATE operation.				
searchScriptFileName	String	null		Get Search
The name of the file used to perform the SEARCH operation.				

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

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

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

Chapter 11

Generic JNDI based LDAP Connector Installation Instructions

This chapter describes how to install the Generic JNDI based LDAP Connector and any prerequisites specific to its use.

11.1 Before You Install the Generic JNDI based LDAP Connector

The Generic JNDI based LDAP Connector enables you to provision to any LDAP v3 compliant directory server. Before you use the connector, ensure that your directory server is up and running and that the connection details in the `conf/provisioner.openicf-ldap.json` file match those of your directory server deployment.

The Generic JNDI based LDAP Connector is supported with any LDAP V3 directory server, and for use with OpenIDM 2.1 and 3.x. The connector works with the OpenICF framework versions 1.1 and 1.4.

11.2 Installing the Generic JNDI based LDAP Connector

The Generic JNDI based LDAP Connector is provided in the `openidm/connectors/ldap-connector-1.4.1.0.jar` file, for local use. If you are running the connector remotely, copy the connector jar file to the `bundles` directory on the Java connector server.

A sample Generic JNDI based LDAP Connector configuration is provided in `openidm/samples/provisioners/provisioner.openicf-ldap.json`. Edit this file to match your LDAP directory deployment, and copy the file to the `openidm/conf` directory. Edit the configuration file, specifying at least the connection details to your LDAP server. Additional configuration properties are as described in the *Configuration chapter*.

11.3 Configuring Connector Pooling

The Generic JNDI based LDAP Connector supports connection pooling, which can substantially improve the performance of the connector. The basic connection pooling configuration is described in the *Connection Pooling Configuration Appendix*.

Specifically, for the Generic JNDI based LDAP Connector, the value of the `"maxObjects"` and `"maxIdle"` properties *must* be the same. The best performance results with this connector have been observed when these properties have been increased to `40` respectively.

Chapter 12
LDAP Connector Configuration

This chapter describes the structure and configuration of the LDAP Connector, the operations that are supported by the connector, and the connector schema.

The LDAP Connector supports connector pooling for improved performance and scalability. For information about configuring connector pooling, see the Configuring Connector Pooling.

12.1 LDAP Connector Reference Object

The LDAP Connector has the following unique identifiers, expressed here in JSON format.

```
"connectorRef" : {
      "bundleName"    : "org.forgerock.openicf.connectors.ldap-connector",
      "bundleVersion" : "1.4.1.0",
      "connectorName" : "org.identityconnectors.ldap.LdapConnector"
      }
```

You can use OpenIDM to generate this configuration automatically when you configure the connector. Alternatively, you can copy this section and paste it directly into your connector configuration file (`provisioner.openicf-`*`connector-name`*`.json`).

12.2 OpenICF Interfaces Implemented by the LDAP Connector

The LDAP Connector implements the following OpenICF interfaces.

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

Create
: Creates an object and its `uid`.

Delete
: Deletes an object, referenced by its `uid`.

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

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

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

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

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

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

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

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

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

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

12.3 LDAP Connector Configuration

The LDAP Connector has the following configurable properties.

12.3.1 Configuration Properties

Property	Type	Default	Encrypted [a]	Required [b]
`accountSynchronizationFilter`	`String`	`null`		Sync
An optional LDAP filter for the objects to synchronize. Because the change log is for all objects, this filter updates only objects that match the specified filter. If you specify a filter, an object will be synchronized only if it matches the filter and includes a synchronized object class.				
`passwordAttributeToSynchroni`	`String`	`null`		Sync
The name of the password attribute to synchronize when performing password synchronization.				
`synchronizePasswords`	`boolean`	`false`		Sync
If true, the connector will synchronize passwords. The Password Capture Plugin needs to be installed for password synchronization to work.				
`removeLogEntryObjectClassFro`	`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.				

Property	Type	Default	Encrypted [a]	Required [b]
`modifiersNamesToFilterOut`	`String[]`	`[]`		Sync
The list of names (DNs) to filter from the changes. Changes with the attribute "modifiersName" that match entries in this list will be filtered out. The standard value is the administrator name used by this adapter, to prevent loops. Entries should be of the format "cn=Directory Manager".				
`passwordDecryptionKey`	`GuardedByteAr`	`null`	Yes	Sync
The key to decrypt passwords with when performing password synchronization.				
`groupSynchronizationFilter`	`String`	`null`		Sync
An optional LDAP filter for the objects to synchronize. Because the change log is for all objects, this filter updates only objects that match the specified filter. If you specify a filter, an object will be synchronized only if it matches the filter and includes a synchronized object class.				
`credentials`	`GuardedString`	`null`	Yes	No
Password for the principal.				
`changeLogBlockSize`	`int`	`100`		Sync
The number of change log entries to fetch per query.				
`baseContextsToSynchronize`	`String[]`	`[]`		Sync
One or more starting points in the LDAP tree that will be used to determine if a change should be synchronized. The base contexts attribute will be used to synchronize a change if this property is not set.				
`attributesToSynchronize`	`String[]`	`[]`		Sync
The names of the attributes to synchronize. This ignores updates from the change log if they do not update any of the named attributes. For example, if only "department" is listed, then only changes that affect "department" will be processed. All other updates are ignored. If blank (the default), then all changes are processed.				
`changeNumberAttribute`	`String`	`changeNumber`		Sync

Property	Type	Default	Encrypted [a]	Required [b]
The name of the change number attribute in the change log entry.				
`passwordDecryptionInitializa`	`GuardedByteAr`	`null`	Yes	Sync
The initialization vector to decrypt passwords with when performing password synchronization.				
`filterWithOrInsteadOfAnd`	`boolean`	`false`		Sync
Normally the filter used to fetch change log entries is an and-based filter retrieving an interval of change entries. If this property is set, the filter will or together the required change numbers instead.				
`useTimestampsForSync`	`boolean`	`false`		Sync
If true, the connector will use the createTimestamp and modifyTimestamp system attributes to detect changes (Create/Update) on the directory instead of native change detection mechanism (cn=changelog on OpenDJ or Update Sequence Number -USN- on Active Directory for instance). Default value is false.				
`objectClassesToSynchronize`	`String[]`	`['inetOrgPers`		Sync
The object classes to synchronize. The change log is for all objects; this filters updates to just the listed object classes. You should not list the superclasses of an object class unless you intend to synchronize objects with any of the superclass values. For example, if only "inetOrgPerson" objects should be synchronized, but the superclasses of "inetOrgPerson" ("person", "organizationalperson" and "top") should be filtered out, then list only "inetOrgPerson" here. All objects in LDAP are subclassed from "top". For this reason, you should never list "top", otherwise no object would be filtered.				
`port`	`int`	`389`		No
TCP/IP port number used to communicate with the LDAP server.				
`vlvSortAttribute`	`String`	`uid`		No
Specify the sort attribute to use for VLV indexes on the resource.				
`passwordAttribute`	`String`	`userPassword`		No

Property	Type	Default	Encrypted [a]	Required [b]
The name of the LDAP attribute that holds the password. When changing a users password, the new password is set to this attribute.				
`useBlocks`	`boolean`	`false`		No
Specifies whether to use block-based LDAP controls, like the simple paged results or VLV control. When performing search operations on large numbers of entries, the entries are returned in blocks to reduce the amount of memory used by the operation.				
`maintainPosixGroupMembershi`	`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.				
`failover`	`String[]`	`[]`		No
List all servers that should be used for failover in case the preferred server fails. If the preferred server fails, JNDI will connect to the next available server in the list. List all servers in the form of "ldap://ldap.example.com:389/", which follows the standard LDAP v3 URLs described in RFC 2255. Only the host and port parts of the URL are relevant in this setting.				
`ssl`	`boolean`	`false`		No
Select the check box to connect to the LDAP server using SSL.				
`getGroupMemberId`	`boolean`	`false`		No
Specifies whether to add an extra _memberId attribute to get the group members __UID__				
`referralsHandling`	`String`	`follow`		No
Defines how to handle LDAP referrals. Possible values can be follow, ignore or throw.				
`principal`	`String`	`null`		No
The distinguished name with which to authenticate to the LDAP server.				

Property	Type	Default	Encrypted [a]	Required [b]
`baseContexts`	`String[]`	`[]`		No
One or more starting points in the LDAP tree that will be used when searching the tree. Searches are performed when discovering users from the LDAP server or when looking for the groups of which a user is a member.				
`readSchema`	`boolean`	`true`		No
If true, the connector will read the schema from the server. If false, the connector will provide a default schema based on the object classes in the configuration. This property must be true in order to use extended object classes.				
`authType`	`String`	`simple`		No
The authentication mechanism to use: Simple or SASL-GSSAPI. Defaults to "simple".				
`accountObjectClasses`	`String[]`	`['top', 'person', 'organizatio 'inetOrgPers`		No
The default list of object classes that will be used when creating new user objects in the LDAP tree. This can be overridden by specifying the user object classes during the Create operation.				
`accountUserNameAttributes`	`String[]`	`['uid', 'cn']`		No
Attribute or attributes which holds the account's user name. They will be used when authenticating to find the LDAP entry for the user name to authenticate.				
`host`	`String`	`null`		No
The name or IP address of the host where the LDAP server is running.				
`groupMemberAttribute`	`String`	`uniqueMember`		No
The name of the group attribute that will be updated with the distinguished name of the user when the user is added to the group.				

Property	Type	Default	Encrypted [a]	Required [b]
`passwordHashAlgorithm`	`String`	`null`		No
Indicates the algorithm that the Identity system should use to hash the password. Currently supported values are SSHA, SHA, SMD5, MD5 and WIN-AD (when AD is the target). A blank value indicates that the system will not hash passwords. This will cause clear text passwords to be stored in LDAP unless the LDAP server performs the hash (as Forgerocks OpenDJ does, for example).				
`accountSearchFilter`	`String`	`null`		No
An optional LDAP filter to control which accounts are returned from the LDAP resource. If no filter is specified, only accounts that include all specified object classes are returned.				
`usePagedResultControl`	`boolean`	`false`		No
When enabled, the LDAP Paged Results control is preferred over the VLV control when retrieving entries.				
`blockSize`	`int`	`100`		No
The maximum number of entries that can be in a block when retrieving entries in blocks.				
`startTLS`	`boolean`	`false`		No
Specifies whether to use the startTLS operation to initiate a TLS/SSL session.				
`groupObjectClasses`	`String[]`	`['top', 'groupOfUniq`		No
The default list of object classes that will be used when creating new group objects in the LDAP tree. This can be overridden by specifying the group object classes during the Create operation.				
`uidAttribute`	`String`	`entryUUID`		No
The name of the LDAP attribute that is mapped to the OpenICF UID attribute.				
`groupSearchFilter`	`String`	`null`		No

Property	Type	Default	Encrypted [a]	Required [b]
An optional LDAP filter to control which groups are returned from the LDAP resource. If no filter is specified, only groups that include all specified object classes are returned.				
`maintainLdapGroupMembership`	`boolean`	`false`		No
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.				
`respectResourcePasswordPolic`	`boolean`	`false`		No
When this resource is specified in a Login Module (i.e., this resource is a pass-through authentication target) and the resource's password policy is configured for change-after-reset, a user whose resource account password has been administratively reset will be required to change that password after successfully authenticating.				

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

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

Appendix A. OpenICF Interfaces

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

A.1 AttributeNormalizer

Normalize attributes to ensure consistent filtering.

A.2 Authenticate

Provides simple authentication with two parameters, presumed to be a user name and password. If the connector does not implement the AuthenticateOp interface it can not be used in OpenIDM to provide pass-through authentication.

A.3 Batch

Execute a series of operations in a single request. If a resource does not support batch operations, the connector will not implement the batch operation interface. The OpenICF framework will still support batched requests but the operations will be executed iteratively through the connector.

A.4 Connector Event

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

A.5 Create

Create an object and return its uid.

A.6 Delete

Delete an object by its uid.

A.7 Get

Get an object by its uid.

A.8 PoolableConnector

Use pools of target resources.

A.9 Resolve Username

Resolve an object to its uid based on its username.

A.10 Schema

Describe supported object types, operations, and options.

A.11 Script on Connector

Allow script execution on connector.

A.12 Script On Resource

Allow script execution on the resource.

A.13 Search

Allow searches for resource objects.

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

A.14 Sync

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

A.15 Sync Event

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

A.16 Test

Test the connection configuration, including connecting to the resource.

A.17 Update

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

A.18 Update Attribute Values

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

Appendix B. OpenICF Operation Options

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

B.1 Scope

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

- SCOPE_OBJECT
- SCOPE_ONE_LEVEL
- SCOPE_SUBTREE

B.2 Container

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

B.3 Run as User

An option to use with Script on Resource and possibly others that specifies an account under which to execute the script/operation. The specified account will appear to have performed any action that the script/operation performs.

B.4 Run with Password

An option to use with Script on Resource and possibly others that specifies a password under which to execute the script/operation.

B.5 Attributes to Get

Determines which attributes to retrieve during Search and Sync. This option overrides the default behavior, which is for the connector to return exactly the set of attributes that are identified as returned by default in the schema for that connector. This option allows a client application to request additional attributes that would not otherwise not be returned (generally because such attributes are more expensive for a connector to fetch and to format) and/or to request only a subset of the attributes that would normally be returned.

B.6 Paged Results Cookie

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

B.7 Paged Results Offset

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

B.8 Page Size

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

B.9 Sort Keys

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

B.10 Fail on Error

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

B.11 Require Serial

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

Appendix C. Connection Pooling Configuration

Certain connectors support the ability to be pooled. For a pooled connector, OpenICF maintains a pool of connector instances and reuses these instances for multiple provisioning and reconciliation operations. When an operation must be executed, an existing connector instance is taken from the connector pool. If no connector instance exists, a new instance is initialized. When the operation has been executed, the connector instance is released back into the connector pool, ready to be used for a subsequent operation.

For an unpooled connector, a new connector instance is initialized for every operation. When the operation has been executed, OpenICF disposes of the connector instance.

Because the initialization of a connector is an expensive operation, reducing the number of connector initializations can substantially improve performance.

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

- `"maxObjects"` - the maximum number of connector instances in the pool (both idle and active). The default value is `10` instances.
- `"maxIdle"` - the maximum number of idle connector instances in the pool. The default value is `10` idle instances.
- `"maxWait"` - the maximum period to wait for a free connector instance to become available before failing. The default period is `150000` milliseconds, or 15 seconds.

- `"minEvictableIdleTimeMillis"` - the minimum period to wait before evicting an idle connector instance from the pool. The default period is `120000` milliseconds, or 12 seconds.
- `"minIdle"` - the minimum number of idle connector instances in the pool. The default value is `1` instance.

Appendix D. Release Levels & Interface Stability

This appendix includes ForgeRock definitions for product release levels and interface stability.

D.1 ForgeRock Product Release Levels

ForgeRock defines Major, Minor, and Maintenance product release levels. The release level is reflected in the version number. The release level tells you what sort of compatibility changes to expect.

Table D.1. Release Level Definitions

Release Label	Version Numbers	Characteristics
Major	Version: x[.0.0] (trailing 0s are optional)	• Bring major new features, minor features, and bug fixes • Can include

Release Label	Version Numbers	Characteristics
		changes even to Stable interfaces • Can remove previously Deprecated functionality, and in rare cases remove Evolving functionality that has not been explicitly Deprecated • Include changes present in previous Minor and Maintenance releases
Minor	Version: x.y[.0] (trailing 0s are optional)	• Bring minor features, and bug fixes • Can include backwards-compatible changes to Stable interfaces in the same Major

Release Label	Version Numbers	Characteristics
		release, and incompatible changes to Evolving interfaces • Can remove previously Deprecated functionality • Include changes present in previous Minor and Maintenance releases
Maintenance	Version: x.y.z	• Bring bug fixes • Are intended to be fully compatible with previous versions from the same Minor release

D.2 ForgeRock Product Interface Stability

ForgeRock products support many protocols, APIs, GUIs, and command-line interfaces. Some of these interfaces are standard and very stable. Others offer new functionality that is continuing to evolve.

ForgeRock acknowledges that you invest in these interfaces, and therefore must know when and how ForgeRock expects them to change. For that reason, ForgeRock defines interface stability labels and uses these definitions in ForgeRock products.

Table D.2. Interface Stability Definitions

Stability Label	Definition
Stable	This documented interface is expected to undergo backwards-compatible changes only for major releases. Changes may be announced at least one minor release before they take effect.
Evolving	This documented interface is continuing to evolve and so is expected to change, potentially in backwards-incompatible ways even in a minor release. Changes are documented at the time of product release. While new protocols and APIs are still in the process of standardization, they are Evolving. This applies for example to recent Internet-Draft implementations, and also to newly developed functionality.
Deprecated	This interface is deprecated and likely to be removed in a future release. For previously stable interfaces, the change was likely announced in a previous release. Deprecated interfaces will be removed from ForgeRock products.
Removed	This interface was deprecated in a previous release and has now been removed from the product.
Internal/ Undocumented	Internal and undocumented interfaces can change without notice. If you depend on one of these interfaces, contact ForgeRock support or email info@forgerock.com to discuss your needs.

Index