Password Synchronization Plugin Guide

/ ForgeRock Identity Management 6.5

Latest update: 6.5.1.0

Lana Frost
Nabil Maynard

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

Abstract

Guide to configuring and integrating the password synchronization plugins into your IDM deployment.

Table of Contents

Preface

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

1. About This Guide

In this guide you will learn how to use password synchronization plugins to synchronize passwords between ForgeRock Identity Management (IDM) and an LDAP server, either ForgeRock Directory Services (DS) or Active Directory.

This guide is written for systems integrators building solutions based on ForgeRock Identity Management services. This guide describes the two password synchronization plugins, and shows you how to set up and configure the plugins as part of your IDM deployment.

2. Accessing Documentation Online

ForgeRock publishes comprehensive documentation online:

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

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

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

3. Using the ForgeRock.org Site

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

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

Chapter 1

Synchronizing Passwords Between IDM and an LDAP Server

Password synchronization ensures uniform password changes across the resources that store the password. After password synchronization, a user can authenticate with the same password on each resource. No centralized directory or authentication server is required for performing authentication. Password synchronization reduces the number of passwords users need to remember, so they can use fewer, stronger passwords.

IDM can propagate passwords to the resources that store a user's password. In addition, you can download plugins from the ForgeRock BackStage download site to intercept and synchronize passwords that are changed natively in ForgeRock Directory Services (DS) and Active Directory.

If you use these plugins to synchronize passwords, set up password policy enforcement on the LDAP resource, rather than on IDM. Alternatively, ensure that all password policies that are enforced are identical to prevent password updates on one resource from being rejected by IDM or by another resource.

The password synchronization plugin intercepts password changes on the LDAP resource before the passwords are stored in encrypted form. The plugin then sends the intercepted password value to IDM, using an HTTP POST request to patch the corresponding managed user object.

Note

The plugins do not use the LDAP connector to transmit passwords, but send a generic HTTP POST request with a `patch-by-query` action, similar to the following:

```
HTTP POST /managed/user?_action=patch-by-query&uid=bjensen&password=MyPassw0rd
```

If the IDM instance is unavailable when a password is changed in either DS or Active Directory, the respective password plugin intercepts the change, encrypts the password, and stores the encrypted password in a JSON file. The plugin then checks whether the IDM instance is available, at a predefined interval. When IDM becomes available, the plugin performs a PATCH on the managed user record, to replace the password with the encrypted password stored in the JSON file.

To be able to synchronize passwords, both password synchronization plugins require that the corresponding managed user object exist in the IDM repository.

The following sections describe how to use the password synchronization plugin for DS, and the corresponding plugin for Active Directory.

Chapter 2

Synchronizing Passwords With ForgeRock Directory Services (DS)

The DS password synchronization plugin intercepts passwords that are changed natively in the DS server and propagates these password changes to IDM. The password synchronization plugin captures password changes in clear text, encrypts them, and transmits them to IDM. If IDM is unavailable when a password change occurs, the password change is queued for subsequent retry.

The password synchronization plugin requires keys to encrypt changed passwords and certificates to secure communication between DS and IDM. If you do not have existing certificates generated by a Certificate Authority, you can use the certificates that are generated when you set up the DS and IDM servers.

The procedures in this section assume that both IDM and DS are installed and running.

2.1. Establishing Secure Communication Between IDM and DS

The password synchronization plugin encrypts passwords using IDM's public key. IDM then uses its private key to decrypt the password.

This section describes how to export IDM's certificate, containing its public key, to DS so that the password synchronization plugin can use the public key to encrypt the password. The same certificate is used by the plugin to trust the SSL certificate that is provided by IDM.

IDM generates a self-signed certificate the first time it starts up. This procedure uses the self-signed certificate for demonstration purposes. In a production environment, you should use a certificate that has been signed by a Certificate Authority (CA).

There are three possible modes of communication between the DS password synchronization plugin and IDM:

- *SSL Authentication.* In this case, you must import IDM's certificate into the DS truststore (either the self-signed certificate that is generated the first time IDM starts, or a CA-signed certificate).

  For more information, see "To Import the IDM Certificate into the DS Truststore".

- *Mutual SSL Authentication.* In this case, you must import the IDM certificate into the DS truststore, as described in "To Import the IDM Certificate into the DS Truststore", *and* import

the DS CA certificate into the IDM truststore, as described in "To Import the DS Certificate into the IDM Truststore". You must also add the DS SSL certificate DN as a value of the `allowedAuthenticationIdPatterns` property in your project's `conf/authentication.json` file. Mutual SSL authentication is the default configuration of the password synchronization plugin, and the one described in this procedure.

- *HTTP Basic Authentication.* In this case, the connection is secured using a username and password, rather than any exchange of certificates. IDM supports basic authentication for testing purposes only. Do *not* use basic authentication in production. To configure the plugin for basic authentication, set the following properties in the plugin configuration:

  - `openidm-url`
  - `openidm-username`
  - `openidm-password`

  For more information, see "Installing and Configuring the Password Synchronization Plugin". Note that the password sync plugin also requires the IDM certificate to encrypt the password such that it can be decrypted when it is replayed on IDM. Therefore, even if you use HTTP basic authentication, you must import the IDM certificate into the DS truststore, as described in "To Import the IDM Certificate into the DS Truststore".

To Import the IDM Certificate into the DS Truststore

IDM generates a self-signed certificate the first time it starts up. This procedure uses the self-signed certificate to demonstrate how to get the password synchronization plugin up and running. In a production environment, you should use a certificate that has been signed by a Certificate Authority (CA).

The default Java truststore contains signing certificates from well-known CAs. If your CA certificate is not in the default truststore, or if you are using a self-signed certificate, import it into a DS truststore as described here.

Note

The IDM self-signed certificate uses the domain alias `openidm-localhost`. If you are testing the plugin on your local machine, add that alias to your `/etc/hosts` file, for example:

```
127.0.0.1       localhost openidm-localhost
```

DS does not enable a trust manager provider by default. For DS to trust the IDM certificate, you must enable a trust manager provider and reference it from the password plugin configuration.

1. Export the IDM generated self-signed certificate to a file, as follows:

```
$ keytool \
 -export \
 -alias openidm-localhost \
 -file openidm-localhost.crt \
 -keystore /path/to/openidm/security/keystore.jceks \
 -storetype jceks
Enter keystore password: changeit
Certificate stored in file <openidm-localhost.crt>
```

The default IDM keystore password is `changeit`.

2. Import the self-signed certificate into a DS truststore. The following command creates a new truststore file and imports the certificate:

```
$ keytool \
 -import \
 -alias openidm-localhost \
 -file openidm-localhost.crt \
 -keystore /path/to/opendj/config/truststore \
 -storepass:file /path/to/opendj/config/keystore.pin \
 -storetype PKCS12 \
 -noprompt
Certificate was added to keystore
```

3. (Optional) Check that the IDM certificate is in the DS truststore:

```
keytool \
 -list -v \
 -keystore /path/to/opendj/config/truststore \
 -storepass:file /path/to/opendj/config/keystore.pin
...
Alias name: openidm-localhost
Creation date: Nov 5, 2019
Entry type: trustedCertEntry

...
```

4. Add a trust manager provider to access the truststore:

```
$ /path/to/opendj/bin/dsconfig \
 create-trust-manager-provider \
 --hostname localhost \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --type file-based \
 --provider-name "PKCS12 Trust Manager" \
 --set enabled:true \
 --set trust-store-file:/path/to/opendj/config/truststore \
 --set trust-store-pin:"&{file:config/keystore.pin}" \
 --trustAll \
 --no-prompt
The File Based Trust Manager Provider was created successfully
```

To Import the DS Certificate into the IDM Truststore

For mutual authentication, you must import the DS certificate into the IDM truststore.

DS generates a self-signed certificate when you set up communication over LDAPS. This procedure uses the self-signed certificate to get the password synchronization plugin up and running. In a production environment, use a certificate that has been signed by a Certificate Authority.

1. Export the generated DS self-signed certificate to a file, as follows:

```
$ keytool \
 -export \
 -alias server-cert \
 -file server-cert.crt \
 -keystore /path/to/opendj/config/keystore \
 -storepass:file /path/to/opendj/config/keystore.pin
Certificate stored in file <server-cert.crt>
```

2. Import the DS self-signed certificate into the IDM truststore:

```
$ keytool \
 -importcert \
 -alias ds-server-cert \
 -keystore /path/to/openidm/security/truststore \
 -storepass changeit \
 -file server-cert.crt
Certificate was added to keystore
```

3. Restart IDM:

```
/path/to/openidm/shutdown.sh; /path/to/openidm/startup.sh
```

2.2. Installing and Configuring the Password Synchronization Plugin

The following steps install the password synchronization plugin on a DS directory server that is running on the same host as IDM (localhost). If you are running DS on a different host, use the fully qualified domain name instead of `localhost`.

You must use the plugin version that corresponds to your IDM and DS versions. For more information, see "Supported Password Synchronization Plugins" in the *Release Notes*. This procedure assumes that you are using IDM 6.5 and DS 6.5.

1. Download the password synchronization plugin (`DS-IDM-account-change-notification-handler-6.5.0.zip`) from the ForgeRock BackStage download site.

2. Extract the contents of the .zip file to the directory where DS is installed:

```
$ unzip ~/Downloads/DS-IDM-account-change-notification-handler-6.5.0.zip -d /path/to/opendj/
Archive:  DS-IDM-account-change-notification-handler-6.5.0.zip
   creating: opendj/
   ...
```

3. Restart DS to load the additional schema from the password synchronization plugin:

```
$ /path/to/opendj/bin/stop-ds --restart
Stopping Server...
[11/Nov/2019:12:51:39 +0200] category=com.forgerock.opendj.ldap.config.config severity=NOTICE
 msgID=571
 msg=Loaded extension from file '/path/to/opendj/lib/extensions/opendj-openidm-account-change-
notification-handler-6.5.0-sources.jar'...
 ...
[11/Nov/2019:12:51:42 +0200] category=CORE severity=NOTICE msgID=135
 msg=The Directory Server has started successfully
```

4. Configure the password synchronization plugin.

   Change the values of the parameters in the following example to match your DS and IDM deployments:

```
/path/to/opendj/bin/dsconfig \
 create-account-status-notification-handler \
 --type openidm \
 --handler-name OpenIDM\ Notification\ Handler \
 --set enabled:true \
 --set openidm-url:https://openidm-localhost:8444/openidm/managed/user \
 --set attribute-type:entryUUID \
 --set attribute-type:uid \
 --set password-attribute:password \
 --set query-id:for-userName \
 --set log-file:logs/pwsync \
 --set update-interval:0s \
 --set ssl-cert-nickname:ssl-key-pair \
 --set private-key-alias:openidm-localhost \
 --set request-retry-attempts:0 \
 --set certificate-subject-dn:CN=openidm-localhost,O=OpenIDM\ Self-Signed\ Certificate,OU=None,L=None
,ST=None,C=None \
 --set key-manager-provider:PKCS12 \
 --set trust-manager-provider:PKCS12 \
 --hostname localhost \
 --port 4444 \
 --bindDn uid=admin \
 --trustAll \
 --bindPassword password \
 --no-prompt
The Openidm Account Status Notification Handler was created successfully
```

   The configurable properties of the plugin are as follows:

   `enabled`

   Specifies whether the plugin is enabled.

`openidm-url`

The endpoint at which the plugin should find IDM managed user accounts.

Example: `https://localhost:8444/openidm/managed/user`

For HTTP basic authentication, specify the `http` protocol in the URL, and a non-mutual authentication port, for example `http://localhost:8080/openidm/managed/user`.

`attribute-type`

Specifies zero or more attribute types that the plugin will send along with the password change. If no attribute types are specified, only the DN and the new password will be synchronized to IDM.

Suggested values: `entryUUID` and `uid`

`password-attribute`

The attribute in IDM that stores user passwords. This property is used to construct the patch request on the IDM managed user object.

Suggested value: `password`

`query-id`

The query-id for the patch-by-query request. This query must be defined in the repository configuration.

Default value: `for-userName`

`log-file`

The directory where plugin log files are written, and where modified passwords are written when the plugin cannot contact IDM. Passwords in this directory will be encrypted.

Suggested value: `logs/pwsync`

Note that this setting has no effect if the `update-interval` is set to `0 seconds`.

`update-interval`

The interval, in seconds, at which password changes are propagated to IDM. If this value is 0, updates are made synchronously in the foreground, and no encrypted passwords are stored in the configured `log-file`.

`ssl-cert-nickname`

The client key alias.

Example: `server-cert`

`private-key-alias`

The alias of the private key that should be used by IDM to decrypt the JSON object.

Example: `openidm-localhost`

`request-retry-attempts`

The number of times the plugin will attempt a synchronization request if the first attempt fails. If this value is zero, the request is not retried. If the value is more than zero, the request will be retried that number of additional times before the request is removed from the queue. Note that if `ds-cfg-update-interval` is set to zero, the request will not be retried regardless of what this property is set to.

In cases where a request fails for a transient reason (such as when IDM is down or the connection is timing out), the number of retry attempts will not decrement. The reason the request failed will be included in logs, and attempts will continue to occur until IDM responds.

`certificate-subject-dn`

The certificate subject DN of the IDM private key. The previous example assumes that you are using the self-signed certificate that is generated when IDM first starts.

Example: `CN=localhost, O=OpenIDM Self-Signed Certificate, OU=None, L=None, ST=None, C=None`

`key-manager-provider`

The DS key manager provider. The key manager provider specified here must be enabled.

Example: `"Default Key Manager"`

`trust-manager-provider`

The DS trust manager provider. The trust manager provider specified here must be enabled.

Default value: `"PKCS12 Trust Manager"`

5. Restart DS for the new configuration to take effect:

```
/path/to/opendj/bin/stop-ds --restart
Stopping Server..
.
..
.
...The Directory Server has started successfully
```

6. Adjust your DS password policy configuration to use the password synchronization plugin.

The following command adjusts the default password policy:

```
$ /path/to/opendj/bin/dsconfig \
 set-password-policy-prop \
 --port 4444 \
 --hostname localhost \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --policy-name "Default Password Policy" \
 --set account-status-notification-handler:"OpenIDM Notification Handler" \
 --no-prompt \
 --trustAll
```

To Configure IDM for Password Synchronization

The password synchronization plugin uses client certificate authentication to authenticate to IDM. If your project does not already have client certificate authentication configured, you must add the authentication module to your server configuration. You must also update your security configuration to add the IDM key alias.

1. Add the `CLIENT_CERT` authentication module to your project's `conf/authentication.json` file and set the `allowedAuthenticationIdPatterns` property to the certificate DN of the DS SSL certificate.

   The following example assumes that you are using the self-signed DS certificate:

   ```
   more /path/to/openidm/project-dir/conf/authentication.json
   "authModules" : [
       ...
       {
           "name" : "CLIENT_CERT",
           "properties" : {
               "queryOnResource" : "managed/user",
               "defaultUserRoles" : [
                   "internal/role/openidm-cert",
                   "internal/role/openidm-authorized"
               ],
               "allowedAuthenticationIdPatterns" : [
                   ".*CN=localhost, O=OpenDJ RSA Self-Signed Certificate.*"
               ]
           },
           "enabled" : true
       },
       ...
   ]
   ```

   For more information about client certificate authentication, see "Authenticating With Client Certificates" in the *Integrator's Guide*.

2. Update the IDM secret store (`conf/secrets.json`) to add the `openidm-localhost` alias to the `idm.default` secretId:

```
"mappings": [
    {
        "secretId" : "idm.default",
        "types": [ "ENCRYPT", "DECRYPT" ],
        "aliases": [ "&{openidm.config.crypto.alias|openidm-sym-default}", "openidm-localhost" ]
    },
    ...
]
```

For more information about secret stores, see "Configuration Options in `secrets.json`" in the *Integrator's Guide*.

Password synchronization should now be configured and working.

You can test that the setup has been successful as follows:

1. Change a user password in DS.

   The new password should be synchronized to the corresponding IDM managed user account.

2. Make sure that the `PASSTHROUGH_AUTHENTICATION` module is disabled (to ensure that the user is not authenticating with her DS credentials) and that the `MANAGED_USER` module is enabled (so that the user can authenticate with her managed user credentials).

3. You should now be able to log into the Self Service UI (https://localhost:8443/#login/) as that user ID, with the new password.

2.3. Updating the DS Password Synchronization Plugin

Additional steps may be necessary when updating the DS password synchronization plugin after upgrading DS. Check the corresponding Knowledge Base article for more information.

2.4. Uninstalling the DS Password Synchronization Plugin

To uninstall the plugin, change the DS configuration as follows:

1. Reset your DS password policy configuration so that it no longer uses the password synchronization plugin.

   The following command resets the default password policy:

```
$ cd /path/to/opendj/bin
$ ./dsconfig \
 set-password-policy-prop \
 --port 4444 \
 --hostname `hostname` \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --policy-name "Default Password Policy" \
 --reset account-status-notification-handler \
 --no-prompt
```

2. Delete the IDM Notification Handler from the DS configuration:

```
$ ./dsconfig \
 delete-account-status-notification-handler \
 --port 4444 \
 --hostname `hostname` \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --handler-name "OpenIDM Notification Handler" \
 --no-prompt
The Account Status Notification Handler was deleted successfully
```

3. Remove the password synchronization plugin from the DS extensions:

```
$ cd /path/to/opendj
$ rm lib/extensions/opendj-openidm-account-change-notification-handler*
```

4. Restart DS for the new configuration to take effect:

```
$ cd /path/to/opendj/bin
$ ./stop-ds --restart
```

Chapter 3

Synchronizing Passwords With Active Directory

Use the Active Directory password synchronization plugin to synchronize passwords between IDM and Active Directory (on systems running at least Microsoft Windows Server 2003).

Install the plugin on Active Directory domain controllers (DCs) to intercept password changes, and send the password values to IDM over an encrypted channel. You must have Administrator privileges to install the plugin. In a clustered Active Directory environment, you must install the plugin on all DCs.

3.1. Installing the Active Directory Password Synchronization Plugin

The following steps install the password synchronization on an Active directory server:

1. Download the Active Directory password synchronization plugin from the ForgeRock BackStage download site.
2. Install the plugin using one of the following methods:
   - Double-click the setup file to launch the installation wizard.
   - Alternatively, from a command-line, start the installation wizard with the `idm-setup.exe` command. To save the settings in a configuration file, use the `/saveinf` switch as follows:
     ```
     PS C:\path\to\dir> idm-setup.exe /saveinf=C:\temp\adsync.inf
     ```
   - If you have a configuration file with installation parameters, you can install the password plugin in silent mode as follows:
     ```
     PS C:\path\to\dir> idm-setup.exe /verysilent /loadinf=C:\temp\adsync.inf
     ```
3. Provide the following information during the installation. You must accept the license agreement shown to proceed with the installation:

   OpenIDM Connection information

   - *OpenIDM URL.* Enter the URL where IDM is deployed, including the query that targets each user account. For example:

```
https://localhost:8444/openidm/managed/user?_action=patch&_queryId=for-userName&uid=
${samaccountname}
```

- *OpenIDM User Password attribute.* The password attribute for the `managed/user` object, such as `password`.

  If the `password` attribute does not exist in the IDM `managed/user` object, the password sync service will return an error when it attempts to replay a password update that has been made in Active Directory. If your managed user objects do not include passwords, you can add an `onCreate` script to the Active Directory > Managed Users mapping that sets an empty password when managed user accounts are created. The following excerpt of a sample `sync.json` file shows such a script in the mapping:

```
"mappings" : [
 {
   "name" : "systemAdAccounts_managedUser",
   "source" : "system/ad/account",
   "target" : "managed/user",
   "properties" : [
     {
       "source" : "sAMAccountName",
       "target" : "userName"
     }
   ],
   "onCreate" : {
     "type" : "text/javascript",
     "source" : "target.password=''"
   },
...
```

  The onCreate script creates an empty password in the `managed/user` object, so that the password attribute exists and can be patched.

OpenIDM Authentication Parameters

Provide the following information:

- *User name.* Enter the name of an administrative user that can authenticate to IDM, for example, `openidm-admin`.
- *Password.* Enter the password of the user that authenticates to IDM, for example, `openidm-admin`.
- *Select authentication type.* Select the type of authentication that Active Directory will use to authenticate to IDM.

  For plain HTTP authentication, select `OpenIDM Header`. For SSL mutual authentication, select `Certificate`.

Certificate authentication settings

If you selected `Certificate` as the authentication type on the previous screen, specify the details of the certificate that will be used for authentication.

- *Select Certificate file.* Browse to select the certificate file that Active Directory will use to authenticate to IDM. The certificate file must be configured with an appropriate encoding, cryptographic hash function, and digital signature. The plugin can read a public or a private key from a PKCS12 archive file.

  For production purposes, you should use a certificate that has been issued by a Certificate Authority. For testing purposes, you can generate a self-signed certificate. Whichever certificate you use, you must import that certificate into the IDM truststore.

  To generate a self-signed certificate for Active Directory, follow these steps:

  1. On the Active Directory host, generate a private key, which will be used to generate a self-signed certificate with the alias `ad-pwd-plugin-localhost`:

     ```
     > keytool.exe ^
      -genkey ^
      -alias ad-pwd-plugin-localhost ^
      -keyalg rsa ^
      -dname "CN=localhost, O=AD-pwd-plugin Self-Signed Certificate" ^
      -keystore keystore.jceks ^
      -storetype JCEKS
     Enter keystore password: changeit
     Re-enter new password: changeit
     Enter key password for <ad-pwd-plugin-localhost>
           <RETURN if same as keystore password>
     ```

  2. Now use the private key, stored in the `keystore.jceks` file, to generate the self-signed certificate:

     ```
     > keytool.exe ^
      -selfcert ^
      -alias ad-pwd-plugin-localhost ^
      -validity 365 ^
      -keystore keystore.jceks ^
      -storetype JCEKS ^
      -storepass changeit
     ```

  3. Export the certificate. In this case, the **keytool** command exports the certificate in a PKCS12 archive file format, used to store a private key with a certificate:

```
> keytool.exe ^
 -importkeystore ^
 -srckeystore keystore.jceks ^
 -srcstoretype jceks ^
 -srcstorepass changeit ^
 -srckeypass changeit ^
 -srcalias ad-pwd-plugin-localhost ^
 -destkeystore ad-pwd-plugin-localhost.p12 ^
 -deststoretype PKCS12 ^
 -deststorepass changeit ^
 -destkeypass changeit ^
 -destalias ad-pwd-plugin-localhost ^
 -noprompt
```

4. The PKCS12 archive file is named `ad-pwd-plugin-localhost.p12`. Import the contents of the keystore contained in this file to the system that hosts IDM. To do so, import the PKCS12 file into the IDM keystore file, named `truststore`, in the `/path/to/openidm/security` directory.

   On the machine that is running IDM, enter the following command:

```
$ keytool \
 -importkeystore \
 -srckeystore /path/to/ad-pwd-plugin-localhost.p12 \
 -srcstoretype PKCS12 \
 -destkeystore truststore \
 -deststoretype JKS
```

- *Password to open the archive file with the private key and certificate.* Specify the keystore password (`changeit`, in the previous example).

Password Encryption settings

Provide the details of the certificate that will be used to encrypt password values.

- *Select certificate file.* Browse to select the certificate that will be used for password encryption. The certificate must be in PKCS12 format.

  For evaluation purposes, you can use a self-signed certificate, as described earlier. For production purposes, you should use a certificate that has been issued by a Certificate Authority.

  Whichever certificate you use, the certificate must be imported into the IDM keystore, so that IDM can locate the key with which to decrypt the data. To import the certificate into the IDM keystore, `keystore.jceks`, run the following command on the IDM host (UNIX):

```
$ keytool \
 -importkeystore \
 -srckeystore /path/to/ad-pwd-plugin-localhost.p12 \
 -srcstoretype PKCS12 \
 -destkeystore /path/to/openidm/security/keystore.jceks \
 -deststoretype jceks
```

- *Private key alias*. Specify the alias for the certificate, such as `ad-pwd-plugin-localhost`. The password sync plugin sends the alias when communicating with IDM, which uses the alias to retrieve the corresponding private key in IDM's keystore.

  Update the IDM secret store (`conf/secrets.json`) to add this certificate alias to the `idm.default` secretId:

  ```
  "mappings": [
      {
          "secretId": "idm.default",
          "types": [ "ENCRYPT", "DECRYPT" ],
          "aliases": [ "&{openidm.config.crypto.alias|openidm-sym-default}","ad-pwd-plugin-
  localhost" ] },
          ...
      }
  ]
  ```

- *Password to open certificate file*. Specify the password to access the PFX keystore file, such as `changeit`, from the previous example.

- *Select encryption standard*. Specify the encryption standard that will be used when encrypting the password value (AES-128, AES-192, or AES-256).

  If you are using Oracle JDK 8 and you select an encryption key type greater than AES-128, you must install the Unlimited JCE Policy, *on the machine on which the password sync plugin is installed*. To install the unlimited JCE Policy, follow these steps:

  - Download the JCE zip file for Java 8 from the Oracle Technology Network site.
  - Locate the `lib\security` folder of your JRE, for example, `C:\Program Files\Java\jre8\lib\security`.
  - Remove the following `.jar` files from the `lib\security` folder:

    `local_policy.jar`
    `US_export_policy.jar`

  - Unzip the JCE zip file and copy the two `_policy.jar` files to the `lib\security` folder of your JRE.
  - If the password sync plugin is already running, you must restart it for the installation of the JCE policy files to take effect.

Data storage

Provide the details for the storage of encrypted passwords in the event that IDM is not available when a password modification is made.

- Select a secure directory in which the JSON files that contain encrypted passwords are queued. The server should prevent access to this folder, except access by the `Password Sync service`. The path name cannot include spaces.
- *Directory poll interval (seconds).* Enter the number of seconds between calls to check whether IDM is available, for example, `60`, to poll IDM every minute.

Log storage

Provide the details of the messages that should be logged by the plugin.

- Select the location to which messages should be logged. The path name cannot include spaces.
- *Select logging level.* Select the severity of messages that should be logged, either `error`, `info`, `warning`, `fatal`, or `debug`.

Select Destination Location

Setup installs the plugin in the location you select, by default `C:\Program Files\OpenIDM Password Sync`.

4. After running the installation wizard, restart the computer.
5. If you selected to authenticate over plain HTTP in the previous step, your setup is now complete.

   If you selected to authenticate with mutual authentication, complete this step.

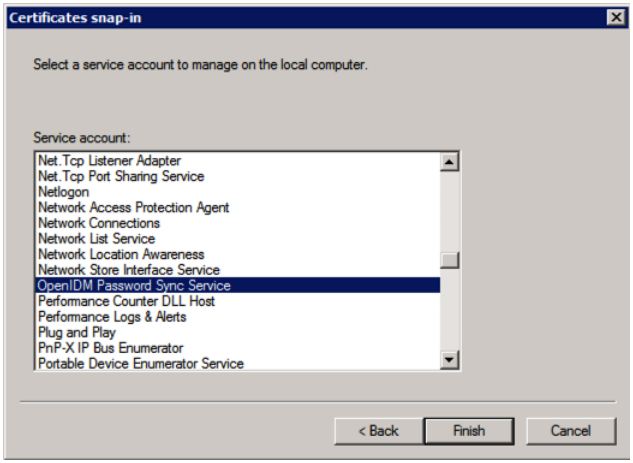
   - The Password Sync Service uses Windows certificate stores to verify IDM's identity. The certificate that IDM uses must therefore be added to the list of trusted certificates on the Windows machine.

     In a production environment, use a certificate that has been issued by a certificate authority. For test purposes, you can use the self-signed certificate that is generated by IDM on first startup.

     To add the IDM certificate to the list of trusted certificates, use the Microsoft Management Console.

     1. Select Start and type `mmc` in the Search field.
     2. In the Console window, select File > Add/Remove Snap-in.
     3. From the left hand column, select Certificates and click Add.
     4. Select My user account, and click Finish.
     5. Repeat the previous two steps for Service account and Computer account.

        For Service account, select Local computer, then select OpenIDM Password Sync Service.



For Computer account, select Local computer.

6. Click Finish when you have added the three certificate snap-ins.
7. Still in the Microsoft Management Console, expand Certificates - Current User > Personal and select Certificates.
8. Select Action > All Tasks > Import to open the Certificate Import Wizard.
9. Browse for the IDM certificate. If you have exported IDM's self-signed certificate, the certificate is `openidm-localhost.crt`.
10. Enter the Password for the certificate (`changeit` by default, if you use the IDM self-signed certificate).
11. Accept the default for the Certificate Store.
12. Click Finish to complete the import.
13. Repeat the previous six steps to import the certificate for:

```
Certificates - Current User > Trusted Root Certification Authorities
Certificates - Service > OpenIDM Password Sync\Personal
Certificates - Service > OpenIDM Password Sync\Trusted Root Certification Authorities
Certificates > Local Computer > Personal
Certificates > Local Computer > Trusted Root Certification Authorities
```

Password synchronization should now be configured and working. To test that the setup was successful, change a user password in Active Directory. That password should be synchronized to the corresponding IDM managed user account, and you should be able to query the user's own entry in IDM using the new password.

3.2. Changing the Password Synchronization Plugin Configuration After Installation

If you need to change any settings after installation, access the settings using the Registry Editor under HKEY_LOCAL_MACHINE > SOFTWARE > ForgeRock > OpenIDM > PasswordSync.

For information about creating registry keys, see the corresponding Windows documentation.

You can change the following registry keys to reconfigure the plugin:

Keys to set the method of authentication

- `authType` sets the authentication type.

  For plain HTTP or SSL authentication using IDM headers, set `authType` to `idm`.

  For SSL mutual authentication using a certificate, set `authType` to `cert`.

  By default, the plugin does not validate the IDM certificate. To configure this validation, set the following registry key: `netSslVerifyPeer = True`.

- `authToken0` sets the username or certificate path for authentication.

  For example, for plain HTTP or SSL authentication, set `authToken0` to `openidm-admin`.

  For SSL mutual authentication, set `authToken0` to the certificate path, for example `path/to/certificate/cert.p12`. Only PKCS12 format certificates are supported.

- `authToken1` sets the password for authentication.

  For example, for plain HTTP or SSL authentication, set `authToken1` to `openidm-admin`.

  For SSL mutual authentication, set `authToken1` to the password to the keystore.

Keys to set encryption of captured passwords

- `certFile` sets the path to the keystore used for encrypting captured passwords, for example `path/to/keystore.p12`. Only PKCS12 keystores are supported.
- `certPassword` sets the password to the keystore.
- `keyAlias` specifies the alias that is used to encrypt passwords.
- `keyType` sets the cypher algorithm, for example `aes128`

Keys to set encryption of sensitive registry values

For security reasons, you should encrypt the values of the `authToken1` and `certPassword` keys. These values are encrypted automatically when the plugin is installed, but when you change the settings, you can encrypt the values manually by setting the `encKey` registry key.

Note

If you do not want to encrypt the values of the `authToken1` and `certPassword` keys, you *must* remove the `encKey` from the registry, otherwise the plugin will use the value stored in that key to decrypt those values (even if they include an empty string).

To encrypt the values of the `authToken1` and `certPassword` keys:

1. Optionally, generate a new encryption key by running the following command:

   ```
   idmsync.exe --key
   ```

2. Encrypt the values of the sensitive registry keys as follows:

   ```
   idmsync.exe --encrypt "key-value" "authToken1Value"
   idmsync.exe --encrypt "key-value" "certPasswordValue"
   ```

3. Replace the existing values of the `encyKey`, `authToken1` and `certPassword` keys with the values you generated in the previous step.

   If you do not want to generate a new encryption key, skip the first step and use the existing encryption key from the registry.

Keys to set the IDM connection information

The password synchronization plugin assumes that the Active Directory user attribute is `sAMAccountName`. The default attribute will work in most deployments. If you cannot use the `sAMAccountName` attribute to identify the Active Directory user, set the following registry keys on your Active Directory server, specifying an alternative attribute. These examples use the `employeeId` attribute instead of `sAMAccountName`:

- `userAttribute = employeeId`
- `userSearchFilter = (&(objectClass=user)(sAMAccountName=%s))`
- `idmURL = https://localhost:8444/openidm/managed/user?_action=patch&_queryId=for-userName&uid=${employeeId}`

Keys to set the behavior when IDM is unavailable

When IDM is unavailable, or when an update fails, the password synchronization plugin stores the user password change a JSON file on the Active Directory system and attempts to resend the password change at regular intervals.

After installation, you can change the behaviour by setting the following registry keys:

Also the netTimeout in milliseconds can be set.

- `dataPath` - the location where the plugin stores the unsent changes. When any unsent changes have been delivered successfully, files in this path are deleted. The plugin creates one file for

each user. This means that if a user changes his password three times in a row, you will see only one file containing the last change.

- `pollEach` - the interval (in seconds) at which the plugin attempts to resend the changes.
- `netTimeout` - the length of time (in milliseconds) after which the plugin stops attempting a connection.

Keys to set the logging configuration

- `logPath` sets the path to the log file.

  If you change this parameter, you *must* restart the machine for the new setting to take effect. If you change the `logPath` and do not restart the machine, the service will write the logs to the new location but the sync module will continue to write logs to the old location until the machine is restarted.

- `logSize` - the maximum log size (in Bytes) before the log is rotated. When the log file reaches this size, it is renamed `idm.log.0` and a new `idm.log` file is created.
- `logLevel` sets the logging level, `debug`, `info`, `warning`, `error`, or `fatal`.

Key to configure support for older IDM versions

If the `idm2only` key is set to `true`, the plugin uses an old version of the patch request. This key *must not* exist in the registry for IDM versions 3.0 and later.

If you change any of the registry keys associated with the password synchronization plugin, run the **idmsync.exe --validate** command to make sure that all of the keys have appropriate values.

The password synchronization plugin is installed and run as a service named OpenIDM Password Sync Service. You can use the Windows Service Manager to start and stop the service. To start or stop the plugin manually, run the **idmsync.exe --start** or **idmsync.exe --stop** command.

3.3. Uninstalling the Active Directory Password Synchronization Plugin

You can uninstall the Active Directory Password Synchronization plugin from multiple locations:

- Uninstall from the Windows Control Panel (Control Panel > Programs and Features, select `OpenIDM Password Sync` from the list and select Uninstall).
- Run the uninstaller (`unins000.exe`) found in the OpenIDM Password Sync install directory (by default, `C:\Program Files\OpenIDM Password Sync`).

After the uninstaller has run, Windows will prompt you to restart. Restart to complete the uninstall process.

3.3.1. Removing Installed Authentication Certificates

If you selected to authenticate with mutual authentication, you can manually remove the IDM certificates you installed using the following steps:

1. Open the Microsoft Management Console, expand Certificates - Current User > Personal, and select Certificates.
2. Delete the previously installed certificate from the list of certificates.
3. Repeat this process for:

   ```
   Certificates - Current User > Trusted Root Certification Authorities
   Certificates > Local Computer > Personal
   Certificates > Local Computer > Trusted Root Certification Authorities
   ```

4. If the OpenIDM Password Sync service is still listed with stored certificates:
   1. Select File > Add/Remove Snap-in.
   2. Select Certificates - OpenIDM Password Sync from the column on the right and select Remove.

Chapter 4

Troubleshooting Password Sync

While default configuration settings work in many circumstances, some configurations introduce additional complexity that you should keep in mind during development. This chapter covers some known issues and how to address them.

4.1. Preventing Infinite Loops

Note

For the sake of simplicity, we reference Active Directory in this chapter, but all instructions are also applicable to DS unless otherwise noted.

In cases where the Active Directory syncs passwords with IDM in both directions, it is possible for a password update in Active Directory to trigger a password sync with IDM that is then sent as an update to the same Active Directory service that triggered the sync. This can cause an infinite loop, where Active Directory and IDM are constantly updating the password and telling the other system to do the same.

This can be prevented by making a few changes to your configuration:

1. Update the user object in `managed.json` to add two new fields: `adPassword`, which should match the parameters for `password`; and `adPasswordChange`:

```
...
"adPassword" : {
    "description" : "",
    "title" : "adPassword",
    "viewable" : true,
    "searchable" : false,
    "userEditable" : true,
    "policies" : [],
    ...
"adPasswordChange" : {
    "description" : "",
    "title" : "AD Flag Password Change",
    "viewable" : true,
    "searchable" : false,
    "userEditable" : false,
    "policies" : [ ],
    "returnByDefault" : false,
    "minLength" : "",
    "pattern" : "",
    "type" : "string"
},
```

2. Update the `onUpdate` script referenced in `managed.json` to include a check comparing the updated `adPassword` with the old value, and updating `password` if there's a mismatch. Note that in cases where the password was changed outside of Active Directory, we also set `adPasswordChange` to true:

```
...
var clear_old_object = openidm.decrypt(oldObject);
var clear_new_object = openidm.decrypt(newObject);

if (clear_new_object.adPassword != clear_old_object.adPassword) {
    newObject.password = newObject.adPassword;
    newObject.adPasswordChange = 'false';
    console.log('AD Plugin Changing the Password in OpenIDM');
} else if (clear_old_object.password != clear_new_object.password) {
    newObject.adPassword = newObject.password;
    newObject.adPasswordChange = 'true';
    console.log('User changing their password in OpenIDM ');
}
...
```

3. Update the mapping between IDM and Active Directory in `sync.json` to also check `adPasswordChange`. If `adPasswordChange` is true, the change needs to be sent to Active Directory:

```
...
{
    "target" : "__PASSWORD__",
    "source" : "password",
    "condition" : {
        "type" : "text/javascript",
        "globals" : { },
        "source" : "object.password != null && object.adPasswordChange == 'true';"
    }
},
```

The DS password sync plugin defaults to `password` as the IDM user password attribute to sync. In order to use the solution outlined above, you will need to change the user password attribute the DS sync plugin is using to something unique (such as `ldapPassword`).

By default, the Active Directory password sync plugin installer sets `adPassword` as the IDM user password attribute to sync. Since this is already a distinct user password attribute from the `password` attribute found in the default user managed object, no further changes are needed to use this solution.

IDM Glossary

correlation query	A correlation query specifies an expression that matches existing entries in a source repository to one or more entries on a target repository. While a correlation query may be built with a script, it is *not* a correlation script. As noted in "Correlating Source Objects With Existing Target Objects" in the *Integrator's Guide*, you can set up a query definition, such as `_queryId`, `_queryFilter`, or `_queryExpression`, possibly with the help of a `linkQualifier`.
correlation script	A correlation script matches existing entries in a source repository, and returns the IDs of one or more matching entries on a target repository. While it skips the intermediate step associated with a `correlation query`, a correlation script can be relatively complex, based on the operations of the script.
entitlement	An entitlement is a collection of attributes that can be added to a user entry via roles. As such, it is a specialized type of `assignment`. A user or device with an entitlement gets access rights to specified resources. An entitlement is a property of a managed object.
JCE	Java Cryptographic Extension, which is part of the Java Cryptography Architecture, provides a framework for encryption, key generation, and digital signatures.
JSON	JavaScript Object Notation, a lightweight data interchange format based on a subset of JavaScript syntax. For more information, see the JSON site.

JSON Pointer	A JSON Pointer defines a string syntax for identifying a specific value within a JSON document. For information about JSON Pointer syntax, see the JSON Pointer RFC.
JWT	JSON Web Token. As noted in the JSON Web Token draft IETF Memo, "JSON Web Token (JWT) is a compact URL-safe means of representing claims to be transferred between two parties." For IDM, the JWT is associated with the `JWT_SESSION` authentication module.
managed object	An object that represents the identity-related data managed by IDM. Managed objects are configurable, JSON-based data structures that IDM stores in its pluggable repository. The default configuration of a managed object is that of a user, but you can define any kind of managed object, for example, groups or roles.
mapping	A policy that is defined between a source object and a target object during reconciliation or synchronization. A mapping can also define a trigger for validation, customization, filtering, and transformation of source and target objects.
OSGi	A module system and service platform for the Java programming language that implements a complete and dynamic component model. For a good introduction, see the OSGi site. Currently only the Apache Felix container is supported.
reconciliation	During reconciliation, comparisons are made between managed objects and objects on source or target systems. Reconciliation can result in one or more specified actions, including, but not limited to, synchronization.
resource	An external system, database, directory server, or other source of identity data to be managed and audited by the identity management system.
REST	Representational State Transfer. A software architecture style for exposing resources, using the technologies and protocols of the World Wide Web. REST describes how distributed data objects, or resources, can be defined and addressed.
role	IDM distinguishes between two distinct role types - provisioning roles and authorization roles. For more information, see "*Working With Managed Roles*" in the *Integrator's Guide*.
source object	In the context of reconciliation, a source object is a data object on the source system, that IDM scans before attempting to find a corresponding object on the target system. Depending on the defined mapping, IDM then adjusts the object on the target system (target object).

synchronization	The synchronization process creates, updates, or deletes objects on a target system, based on the defined mappings from the source system. Synchronization can be scheduled or on demand.
system object	A pluggable representation of an object on an external system. For example, a user entry that is stored in an external LDAP directory is represented as a system object in IDM for the period during which IDM requires access to that entry. System objects follow the same RESTful resource-based design principles as managed objects.
target object	In the context of reconciliation, a target object is a data object on the target system, that IDM scans after locating its corresponding object on the source system. Depending on the defined mapping, IDM then adjusts the target object to match the corresponding source object.