



Password Synchronization Plugin Guide

/ ForgeRock Identity Management 7.1

Latest update: 7.1.6

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

Copyright © 2017-2020 ForgeRock AS.

Abstract

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



This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License.

To view a copy of this license, visit <https://creativecommons.org/licenses/by-nc-nd/3.0/> or send a letter to Creative Commons, 444 Castro Street, Suite 900, Mountain View, California, 94041, USA.

© Copyright 2010-2020 ForgeRock, Inc. All rights reserved. ForgeRock is a registered trademark of ForgeRock, Inc. Other marks appearing herein may be trademarks of their respective owners.

This product or document is protected by copyright and distributed under licenses restricting its use, copying, and distribution. No part of this product or document may be reproduced in any form by any means without prior written authorization of ForgeRock and its licensors, if any.

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESSED OR IMPLIED CONDITIONS, REPRESENTATIONS, AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

DejaVu Fonts

Bitstream Vera Fonts Copyright

Copyright (c) 2003 by Bitstream, Inc. All Rights Reserved. Bitstream Vera is a trademark of Bitstream, Inc.

Permission is hereby granted, free of charge, to any person obtaining a copy of the fonts accompanying this license ("Fonts") and associated documentation files (the "Font Software"), to reproduce and distribute the Font Software, including without limitation the rights to use, copy, merge, publish, distribute, and/or sell copies of the Font Software, and to permit persons to whom the Font Software is furnished to do so, subject to the following conditions:

The above copyright and trademark notices and this permission notice shall be included in all copies of one or more of the Font Software typefaces.

The Font Software may be modified, altered, or added to, and in particular the designs of glyphs or characters in the Fonts may be modified and additional glyphs or characters may be added to the Fonts, only if the fonts are renamed to names not containing either the words "Bitstream" or the word "Vera".

This License becomes null and void to the extent applicable to Fonts or Font Software that has been modified and is distributed under the "Bitstream Vera" names.

The Font Software may be sold as part of a larger software package but no copy of one or more of the Font Software typefaces may be sold by itself.

THE FONT SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT OF COPYRIGHT, PATENT, TRADEMARK, OR OTHER RIGHT. IN NO EVENT SHALL BITSTREAM OR THE GNOME FOUNDATION BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, INCLUDING ANY GENERAL, SPECIAL, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF THE USE OR INABILITY TO USE THE FONT SOFTWARE OR FROM OTHER DEALINGS IN THE FONT SOFTWARE.

Except as contained in this notice, the names of Gnome, the Gnome Foundation, and Bitstream Inc., shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Font Software without prior written authorization from the Gnome Foundation or Bitstream Inc., respectively. For further information, contact: fonts at gnome dot org.

Arev Fonts Copyright

Copyright (c) 2006 by Tavmjong Bah. All Rights Reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of the fonts accompanying this license ("Fonts") and associated documentation files (the "Font Software"), to reproduce and distribute the modifications to the Bitstream Vera Font Software, including without limitation the rights to use, copy, merge, publish, distribute, and/or sell copies of the Font Software, and to permit persons to whom the Font Software is furnished to do so, subject to the following conditions:

The above copyright and trademark notices and this permission notice shall be included in all copies of one or more of the Font Software typefaces.

The Font Software may be modified, altered, or added to, and in particular the designs of glyphs or characters in the Fonts may be modified and additional glyphs or characters may be added to the Fonts, only if the fonts are renamed to names not containing either the words "Tavmjong Bah" or the word "Arev".

This License becomes null and void to the extent applicable to Fonts or Font Software that has been modified and is distributed under the "Tavmjong Bah Arev" names.

The Font Software may be sold as part of a larger software package but no copy of one or more of the Font Software typefaces may be sold by itself.

THE FONT SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT OF COPYRIGHT, PATENT, TRADEMARK, OR OTHER RIGHT. IN NO EVENT SHALL TAVMJONG BAH BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, INCLUDING ANY GENERAL, SPECIAL, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF THE USE OR INABILITY TO USE THE FONT SOFTWARE OR FROM OTHER DEALINGS IN THE FONT SOFTWARE.

Except as contained in this notice, the name of Tavmjong Bah shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Font Software without prior written authorization from Tavmjong Bah. For further information, contact: tavmjong @ free . fr.

FontAwesome Copyright

Copyright (c) 2017 by Dave Gandy, <https://fontawesome.com/>.

This Font Software is licensed under the SIL Open Font License, Version 1.1. See <https://opensource.org/licenses/OFL-1.1>.




Table of Contents

Overview	iv
1. Password Synchronization Plugins	1
2. Synchronize Passwords With ForgeRock Directory Services (DS)	2
Setting Up IDM and DS to Demonstrate Password Synchronization	2
Establishing Secure Communication Between IDM and DS	5
Installing and Configuring the DS Password Synchronization Plugin	10
Trying the DS Password Synchronization Plugin	17
Updating the DS Password Synchronization Plugin	19
Uninstalling the DS Password Synchronization Plugin	19
3. Synchronize Passwords With Active Directory	21
Installing the Active Directory Password Synchronization Plugin	21
Changing the Password Synchronization Plugin Configuration After Installation ..	28
Uninstalling the Active Directory Password Synchronization Plugin	31
IDM Glossary	33

Overview

Password synchronization ensures uniform password changes across the resources that store the password. This guide shows you 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.

Quick Start

 Password Sync Plugins Discover how password synchronization works in IDM and learn about the plugins provided for synchronizing passwords with LDAP servers.	 Synchronize With DS Use the ForgeRock Directory Services password synchronization plugin.	 Synchronize With AD Use the Active Directory password synchronization plugin.
---	--	--

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>.

The ForgeRock Common REST API works across the platform to provide common ways to access web resources and collections of resources.

Chapter 1

Password Synchronization Plugins

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** action.

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.

Chapter 2

Synchronize 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. The examples that follow use the keys generated when you set up the DS and IDM servers.

Setting Up IDM and DS to Demonstrate Password Synchronization

The following examples prepare a demonstration of password synchronization from DS to IDM. After this preparation:

- DS and IDM are installed and running on your computer, with default security settings.

In particular, both servers have key pairs used later in the demonstration:

- DS has a generated TLS key pair (alias: `ssl-key-pair` and certificate subject DN: `CN=DS, O=ForgeRock.com`) signed by the DS deployment key-based CA (certificate subject DN: `CN=Deployment key, O=ForgeRock.com`).

DS uses the certificate to set up TLS connections, and to authenticate to IDM.

- IDM has a generated key pair, alias `openidm-localhost`. The certificate is self-signed, and has certificate subject DN `CN=openidm-localhost, O=OpenIDM Self-Signed Certificate, OU=None, L=None, ST=None, C=None`.

DS uses the public key certificate to encrypt passwords before sending them to IDM.

- IDM does not otherwise synchronize DS data with its own.

This lets you confirm that DS, not synchronization, provides the updated password to IDM.

- DS and IDM both have a user account for Barbara Jensen.

After you have configured password synchronization, when Barbara Jensen's password changes in DS, DS sends the change to IDM.

Prepare IDM

1. Update your hosts file.

The IDM self-signed certificate uses the domain alias `openidm-localhost`. When testing the DS plugin on your computer, add the alias to your `/etc/hosts` file:

```
127.0.0.1      localhost openidm-localhost
```

2. Unzip IDM.

3. Start IDM:

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

4. Add Barbara Jensen's account to IDM:

```
curl \
--request PUT \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept: application/json" \
--header "If-None-Match: *" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--data
  {"userName": "bjensen", "password": "Password1", "mail": "bjensen@example.com", "sn": "Jensen", "givenName": "Barbara"}
"http://localhost:8080/openidm/managed/user/bjensen"
{
  "_id": "bjensen",
  "_rev": "revision",
  "userName": "bjensen",
  "mail": "bjensen@example.com",
  "sn": "Jensen",
  "givenName": "Barbara",
  "accountStatus": "active",
  "effectiveAssignments": [],
  "effectiveRoles": []
}
```

Prepare DS

1. Download the DS 7.1 .zip distribution from the [ForgeRock BackStage download site](#).

2. Unzip DS:

```
unzip -q ~/Downloads/DS-7.1.zip -d /path/to
```

3. Generate a DS deployment key that you use for DS setup and managing keys:

```
/path/to/openssh/bin/dskeymgr create-deployment-key --deploymentKeyPassword password  
your-deployment-key
```

4. Set up and start DS with data from an IDM example that includes Barbara Jensen's entry:

```
/path/to/openssh/setup \  
--serverId evaluation-only \  
--deploymentKey your-deployment-key \  
--deploymentKeyPassword password \  
--rootUserDn uid=admin \  
--rootUserPassword password \  
--hostname localhost \  
--adminConnectorPort 4444 \  
--ldapsPort 1636 \  
--profile ds-user-data \  
--set ds-user-data/baseDn:dc=com \  
--set ds-user-data/ldifFile:/path/to/openssh/samples/sync-with-ldap/data/Example.ldif \  
--start \  
--acceptLicense
```

The sample data includes an entry for Barbara Jensen. The rest of the sample configuration is not used here.

5. Check that the directory superuser can read Barbara Jensen's entry in the directory:

```
/path/to/openssh/bin/ldapsearch \  
--port 1636 \  
--useSSL \  
--usePkcs12TrustStore /path/to/openssh/config/keystore \  
--trustStorePasswordFile /path/to/openssh/config/keystore.pin \  
--hostname localhost \  
--bindDn uid=admin \  
--bindPassword password \  
--baseDn dc=com \  
"(uid=bjensen)"  
dn: uid=bjensen,ou=People,dc=example,dc=com  
objectClass: person  
objectClass: organizationalPerson  
objectClass: inetOrgPerson  
objectClass: top  
cn: Barbara Jensen  
description: Created for OpenIDM  
givenName: Barbara  
mail: bjensen@example.com  
sn: Jensen  
telephoneNumber: 1-360-229-7105  
uid: bjensen  
userPassword: {PBKDF2-HMAC-SHA256}10:hash
```

Notice that this entry differs from the account you added to IDM. However, the user identifier is **bjensen** in both cases. This will let IDM identify Barbara Jensen's account as the one whose password has changed when it receives the notification from DS.

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.

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

SSL Authentication

For this communication mode, 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).

Mutual SSL Authentication

Note

Mutual SSL authentication is the default configuration of the password synchronization plugin.

For this communication mode, you must:

- Import the IDM certificate into the DS truststore.
- Import the DS CA certificate into the IDM truststore.
- Add the DS certificate subject DN as a value of the `allowedAuthenticationIdPatterns` property in your project's `conf/authentication.json` file.

AM Bearer Tokens

When you use IDM and AM together as a platform, configure the password synchronization plugin to use AM bearer tokens for authentication.

For this communication mode, you must:

- Import the IDM certificate into the DS truststore.
- Import the DS CA certificate into the IDM truststore.
- Configure the password synchronization plugin to accept AM bearer tokens.

HTTP Basic Authentication

Warning

IDM supports basic authentication for *testing purposes only*. Do not use basic authentication in production.

For this mode, the connection is secured using a username and password, rather than any exchange of certificates. Because the password sync plugin requires the IDM certificate to encrypt/decrypt passwords, you must import the IDM certificate into the DS truststore.

For this communication mode, you must:

- Import the IDM certificate into the DS truststore.
- Set the following properties in the plugin configuration:
 - `openidm-url`
 - `openidm-username`
 - `openidm-password`

Enable DS to Trust the IDM Certificate

The first time IDM starts, it generates a self-signed certificate. This procedure uses the self-signed certificate to demonstrate how to get the password synchronization plugin up and running. In a production environment, 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 the DS keystore, as described here.

1. Export the IDM 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 \  
-storepass changeit  
Certificate stored in file <openidm-localhost.crt>
```

The default IDM keystore password is `changeit`.

2. Import the self-signed certificate into the DS keystore:

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

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

```
keytool \
-list \
-keystore /path/to/openswim/config/keystore \
-storepass:file /path/to/openswim/config/keystore.pin
...
openidm-localhost, date, trustedCertEntry,
Certificate fingerprint (SHA-256): fingerprint
...
```

Enable IDM to Trust DS Certificates

For mutual SSL authentication, you must also import a trusted DS certificate into the IDM truststore, either a trusted CA certificate, or the CA certificate that is generated by the DS deployment key and password. For more information, see [Deployment Keys](#) in the *DS Security Guide*. This procedure uses the CA certificate generated by the DS deployment key and password.

1. Run the following command on your DS server to export the CA certificate to a file. Substitute the values for `--deploymentKey` and `--deploymentKeyPassword` with the deployment key and password that you used when you set up the DS server:

```
/path/to/openswim/bin/dskeymgr \
export-ca-cert \
--deploymentKey your-deployment-key \
--deploymentKeyPassword password \
--outputFile ds-ca-cert.pem
```

2. Import the DS CA certificate into the IDM truststore:

```
keytool \
-importcert \
-alias ds-ca-cert \
-keystore /path/to/openidm/security/truststore \
-storepass changeit \
-file ds-ca-cert.pem
Owner: CN=Deployment key, O=ForgeRock.com
Issuer: CN=Deployment key, O=ForgeRock.com
...
Trust this certificate? [no]: yes
Certificate was added to keystore
```

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

```
keytool \  
-list \  
-keystore /path/to/openidm/security/truststore \  
-storepass changeit  
...  
ds-ca-cert, date, trustedCertEntry,  
Certificate fingerprint (SHA-256): fingerprint  
...
```

4. Restart IDM:

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

Configure the Password Synchronization Plugin to Accept AM Bearer Tokens

This procedure uses the [Platform Setup Guide](#) as the basis for setting up IDM to use AM bearer tokens for authentication, and may need adjustment for your specific environment.

1. Perform [Platform Setup - Deployment Two - Shared Identity Store](#), using the following settings during the applicable steps:

+ *Configuration Store*

- adminConnectorPort 4444
- ldapPort 1389
- enableStartTls
- ldapsPort 1636
- httpsPort 8443
- replicationPort 8989
- deploymentKey *your-deployment-key*

+ *Identity Store*

- adminConnectorPort 4445
- ldapPort 1390
- ldapsPort 1637
- replicationPort 8990

- deploymentKey *your-deployment-key*

+ Tomcat - AM

- port: 8080
- redirects: 8444

+ resolver/boot.properties

- openidm.port.http=8081
- openidm.port.https=8445
- openidm.port.mutualauth=8446
- openidm.host=openidm.example.com
- openidm.auth.clientauthonlyports=8446

2. Configure a `ds-password-sync-plugin` OAuth Client for the Password Sync Plugin:
 - a. If you're not currently logged in to the AM console as the `amAdmin` user, log in.
 - b. In the Top Level Realm, select Applications > OAuth 2.0 > Clients, and click Add Client.
 - c. Enter the following details:
 - Client ID: `ds-password-sync-plugin`
 - Client secret: `ds-password-sync-plugin`
 - Scopes: `fr:idm:*`, `openid`, `am-introspect-all-tokens`, `am-introspect-all-tokens-any-realm`
 - d. Click Create.
 - e. On the Advanced tab:
 - Token Endpoint Authentication Method: Select `client_secret_basic`.
 - Grant Types: Add `Client Credentials`.
 - f. Click Save Changes.

3. If you haven't performed the following procedures, do that now:
 - Import the IDM certificate into the DS truststore.
 - Import the DS CA certificate into the IDM truststore.
4. Add `openidm-localhost` to the `idm.default` mapping alias array in your project's `conf/secrets.json` file:

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

5. Log in to the IDM Admin UI as `amAdmin` and create a new managed/user:
 - a. From the navigation bar of the Admin UI, select Manage > User, and then click + New User.
 - b. On the New User page, enter the Username `ds-password-sync-plugin`, other information, as necessary, and click Save.
 - c. On the `ds-password-sync-plugin` user page, select the Authorization Roles tab.
 - d. Click Add Authorization Roles, select all of the following roles, and then click Add:

```
openidm-admin  
openidm-authorized  
openidm-cert  
openidm-reg
```

6. Add a `staticUserMapping` for the `ds-password-sync-plugin` user to the `conf/authentication.json` file:

```
{  
  "subject" : "ds-password-sync-plugin",  
  "localUser" : "managed/user/ds-password-sync-plugin",  
  "roles" : [  
    "internal/role/openidm-authorized",  
    "internal/role/openidm-admin"  
  ]  
}
```

Installing and Configuring the DS 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 7.1, DS 7.1, and version 7.1 of the password synchronization plugin.

Depending on whether you are using IDM with AM, select one of the following plugin installation and configuration procedures:

+ *Install the Plugin and Password Synchronization Configuration Without AM Bearer Tokens*

1. Download the password synchronization plugin.
2. Extract the .zip file contents to the DS installation directory:

```
unzip ~/Downloads/DS-IDM-account-change-notification-handler-7.1.zip -d /path/to/openssl/
```

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

```
/path/to/openssl/bin/stop-ds --restart
Stopping Server...
...msg=Loaded extension from file '/path/to/openssl/lib/extensions/openssl-openidm-account-change-
notification-handler-7.1.jar'
...
...msg=The Directory Server has started successfully
```

4. Configure the password synchronization plugin:

```
/path/to/openssl/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 private-key-alias:openidm-localhost \
--set certificate-subject-dn:"CN=openidm-localhost, O=OpenIDM Self-Signed Certificate, OU=None,
L=None, ST=None, C=None" \
--set ssl-cert-nickname:ssl-key-pair \
--set key-manager-provider:PKCS12 \
--set trust-manager-provider:PKCS12 \
--set password-attribute:password \
--set attribute-type:entryUUID \
--set attribute-type:uid \
--set query-id:for-username \
--set log-file:logs/pwsync \
--set update-interval:5s \
--set request-retry-attempts:5000 \
--hostname localhost \
--port 4444 \
--bindDn uid=admin \
--bindPassword password \
--usePkcs12TrustStore /path/to/openssl/config/keystore \
--trustStorePasswordFile /path/to/openssl/config/keystore.pin \
--no-prompt
The Openidm Account Status Notification Handler was created successfully
```

Adapt the settings to match your DS and IDM deployments:

Setting	Details
<code>enabled</code>	<p>Enables the plugin.</p> <p>Leave this setting as shown.</p>
<code>openidm-url</code>	<p>The endpoint where the plugin finds IDM managed user accounts.</p> <p>Port <code>8444</code> is the IDM default for mutual TLS connections.</p>
<code>private-key-alias</code>	<p>The IDM private key, used to decrypt the JSON objects from DS that contain passwords.</p> <p>The example references the default IDM private key of the self-signed key pair generated at setup time.</p>
<code>certificate-subject-dn</code>	<p>The certificate subject DN for the IDM public key.</p> <p>The DS plugin encrypts JSON objects with the IDM public key, so this must match the certificate for the IDM private key specified for the <code>private-key-alias</code> property.</p> <p>The example shows the subject DN of the default IDM self-signed certificate.</p>
<code>ssl-cert-nickname</code>	<p>The alias of the DS TLS certificate used to authenticate to IDM.</p> <p>The example uses the default DS server certificate generated at setup time.</p>
<code>key-manager-provider</code>	<p>The provider for the keystore where DS finds its TLS key pair specified in <code>ssl-cert-nickname</code>.</p> <p>The example uses the default DS provider configured at setup time.</p>
<code>trust-manager-provider</code>	<p>The provider for the keystore where DS finds the IDM public key specified in <code>certificate-subject-dn</code>.</p> <p>The example uses the default DS provider configured at setup time, and updated in "Enable DS to Trust the IDM Certificate".</p>
<code>password-attribute</code>	<p>The name of the password field in the JSON that DS sends to IDM for a password change.</p> <p>This attribute type must be defined in the managed object schema in IDM, and it must have either the user password or auth password syntax.</p>
<code>attribute-type</code>	<p>LDAP attributes that the DS plugin sends to IDM with the password. IDM can use these to uniquely identify the user, even if the user's account has moved.</p> <p>If no attribute types are specified, DS sends only the DN and the new password to IDM.</p>
<code>query-id</code>	<p>The query-id for the patch-by-query request.</p>

Setting	Details
	Leave this setting as shown.
<code>log-file</code>	<p>The DS directory where the plugin writes log files containing encrypted passwords before notifying IDM.</p> <p>This setting has no effect in the example, where the <code>update-interval</code> is zero seconds.</p>
<code>update-interval</code>	<p>The interval at which the DS plugin sends password changes to IDM.</p> <p>If this value is zero, the plugin sends updates synchronously. No encrypted passwords are stored in the configured <code>log-file</code> directory. The plugin does not retry failed requests, irrespective of the <code>request-retry-attempts</code> setting.</p>
<code>request-retry-attempts</code>	<p>The number of times the plugin attempts a synchronization request if the first attempt fails. If this value is zero, the request is not retried. If the value is greater than zero, the plugin retries the specified number of times before giving up and removing the request from its queue.</p> <p>When a request fails due to a transient condition, such as failure to contact IDM, or a connection timeout, the plugin does not decrement the number of retry attempts. The plugin logs a message with the reason the request failed, and continues to retry until IDM responds.</p>

- Restart DS for the new configuration to take effect:

```
/path/to/openssl/bin/stop-ds --restart
```

- Update DS password policies to use the password synchronization plugin.

The following example updates the default DS password policy:

```
/path/to/openssl/bin/dsconfig \
set-password-policy-prop \
--policy-name "Default Password Policy" \
--set account-status-notification-handler:"OpenIDM Notification Handler" \
--hostname localhost \
--port 4444 \
--bindDN uid=admin \
--bindPassword password \
--usePkcs12TrustStore /path/to/openssl/config/keystore \
--trustStorePasswordFile /path/to/openssl/config/keystore.pin \
--no-prompt
```

For details on configuring DS password policies, see [Passwords](#) in the *DS Security Guide*.

Configure IDM for Password Synchronization

The password synchronization plugin uses client certificate authentication to authenticate to IDM. You must also update your security configuration to add the IDM key alias.

1. If your authentication configuration (You can manage the authentication configuration over REST at the `config/authentication` endpoint, or directly in the `conf/authentication.json` file.) does not already include client certificate authentication, configure it as follows:
 - a. Add the `CLIENT_CERT` authentication module to your authentication configuration (You can manage the authentication configuration over REST at the `config/authentication` endpoint, or directly in the `conf/authentication.json` file.).
 - b. Set the `allowedAuthenticationIdPatterns` property to the certificate DN of the DS SSL certificate (`ssl-key-pair` by default).
 - c. Add `internal/role/openidm-cert` to the array of `defaultUserRoles`.

The following example assumes that you are using the default DS `ssl-key-pair` certificate that has a certificate subject DN of `CN=DS, O=ForgeRock`:

```
"authModules" : [
  ...
  {
    "name" : "CLIENT_CERT",
    "properties" : {
      "queryOnResource" : "managed/user",
      "defaultUserRoles" : [
        "internal/role/openidm-cert",
        "internal/role/openidm-authorized"
      ],
      "allowedAuthenticationIdPatterns" : [
        ".*CN=DS, O=ForgeRock.com.*"
      ]
    },
    "enabled" : true
  },
  ...
]
```

For more information about client certificate authentication, see "CLIENT_CERT" in the *Authentication and Authorization Guide*.

2. Update the IDM secret store (`conf/secrets.json`) to add the alias used in the `private-key-alias` plugin setting 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 "Configuring Secret Stores" in the *Security Guide*.

+ Install the Plugin and Password Synchronization Configuration With AM Bearer Tokens

1. Download the password synchronization plugin.
2. Extract the .zip file contents to the DS identity store installation directory:

```
unzip ~/Downloads/DS-IDM-account-change-notification-handler-7.1.zip -d /path/to/openshift-identity/
```

3. Configure the password synchronization plugin to use AM bearer token authentication:

```
/path/to/openshift-identity/bin/dsconfig \
create-account-status-notification-handler \
--type openidm \
--handler-name "OpenIDM Notification Handler" \
--set certificate-subject-dn:"CN=openidm-localhost,O=OpenIDM Self-Signed  
Certificate,OU=None,L=None,ST=None,C=None" \
--set enabled:true \
--set attribute-type:entryUUID \
--set attribute-type:uid \
--set trust-manager-provider:PKCS12 \
--set key-manager-provider:PKCS12 \
--hostname identities.example.com \
--port 4445 \
--bindDn uid=admin \
--trustAll \
--bindPassword str0ngAdm1nPa55word \
--set log-file:logs/pwsync \
--set password-attribute:password \
--set query-id:for-username \
--set private-key-alias:openidm-localhost \
--set openidm-url:https://openidm-localhost:8445/openidm/managed/user \
--set oauth2-access-token-url:http://am.example.com:8080/openam/oauth2/realms/root/access_token \
--set oauth2-scope:"openid fr:idm:*" \
--set oauth2-client-id:ds-password-sync-plugin \
--set oauth2-client-secret:ds-password-sync-plugin \
--set request-retry-attempts:5000 \
--set update-interval:5s \
--no-prompt
The Openidm Account Status Notification Handler was created successfully
```

Note

Adapt the above settings to match your DS, IDM, and AM deployments.

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

```
/path/to/openshift-identity/bin/stop-ds --restart
```

5. Configure the DS password policy to use the password synchronization plugin. The following example updates the default DS password policy:

```
/path/to/openshift-identity/bin/dsconfig \
set-password-policy-prop \
--policy-name "Default Password Policy" \
--set account-status-notification-handler:"OpenIDM Notification Handler" \
--hostname identities.example.com \
--port 4445 \
--bindDN uid=admin \
--bindPassword strongAdminPa55word \
--usePkcs12TrustStore /path/to/openshift-identity/config/keystore \
--trustStorePasswordFile /path/to/openshift-identity/config/keystore.pin \
--no-prompt
```

For details on configuring DS password policies, see *Passwords in the DS Security Guide*.

6. Generate an AM bearer token. For example:

```
curl -k \
--request POST \
--user "ds-password-sync-plugin:ds-password-sync-plugin" \
--data "grant_type=client_credentials" \
--data "scope=openid fr:idm:*" \
"http://am.example.com:8080/openam/oauth2/realms/root/access_token"
{
  "access_token": "access_token",
  "scope": "openid fr:idm:*",
  "id_token": "id_token",
  "token_type": "Bearer",
  "expires_in": 3599
}
```

7. Optionally, to test the AM bearer token, create a new managed user using the token as the authorization. For example:

```
curl -v \
--header "Authorization: Bearer access_token" \
--header "accept: application/json" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--request POST "http://openidm-localhost:8081/openidm/managed/user?_action=create" \
--data '{"userName":"jdoe", "password": "6fNcHgBF", "mail": "jdoe@example.com", "sn": "Doe",
"givenName": "Jane"}'
{
  "_id": "7884b8ab-a226-4ee5-b9b9-0718f5a19335",
  "_rev": "00000000f527116e",
  "userName": "jdoe",
  "accountStatus": "active",
  "givenName": "Jane",
  "sn": "Doe",
  "mail": "jdoe@example.com"
}
```

Trying the DS Password Synchronization Plugin

With the plugin installed and configured, and with secure communications enabled between DS and IDM, you can test that the setup has been successful as follows:

1. Change a user password in DS:

```
/path/to/openssl/bin/ldappasswordmodify \
--hostname localhost \
--port 1636 \
--useSsl \
--usePkcs12TrustStore /path/to/openssl/config/keystore \
--trustStorePasswordFile /path/to/openssl/config/keystore.pin \
--bindDN uid=admin \
--bindPassword password \
--authzID dn:uid=bjensen,ou=people,dc=example,dc=com \
--newPassword Chngth5pwd
The LDAP password modify operation was successful
```


The message **The LDAP password modify operation was successful** only indicates that the password change succeeded for DS. This does not mean that DS has propagated the change to IDM.

When you have successfully updated the password in DS, DS attempts to synchronize the change to the corresponding IDM managed user account.

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

+ *Show login process*


```
chromium --ignore-certificate-errors https://localhost:8443/#login
```



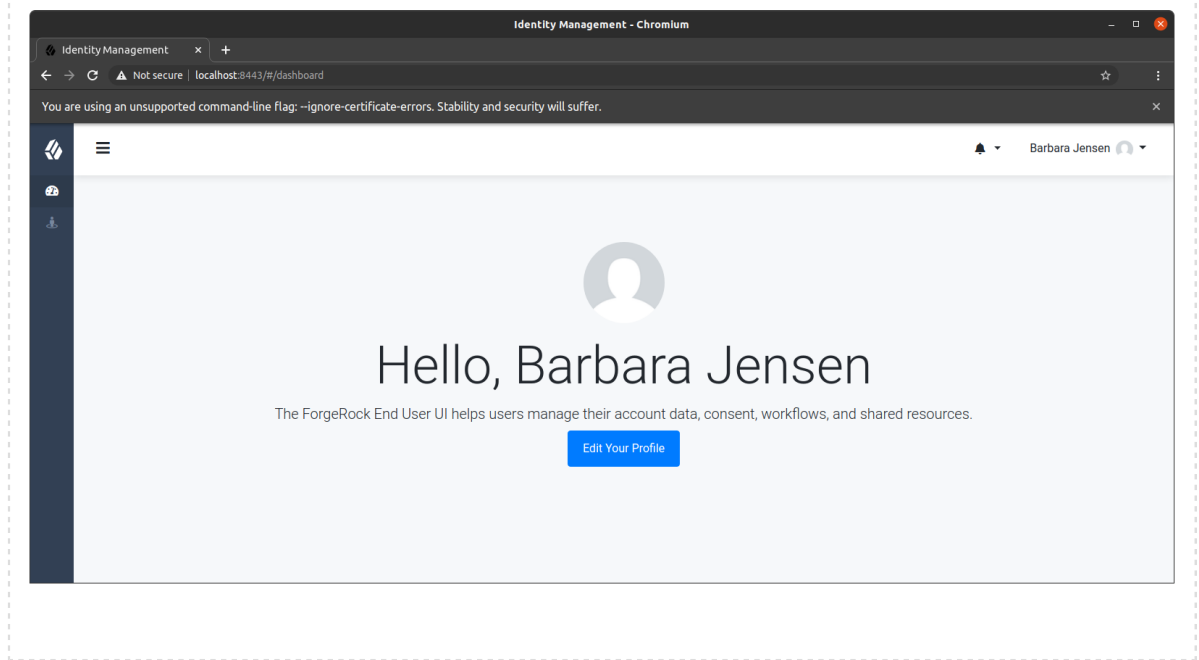
Sign In

Username
bjensen

Password
Chngth5pwd



Sign In



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.

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:

```
/path/to/openssl/bin/dsconfig \  
set-password-policy-prop \  
--policy-name "Default Password Policy" \  
--reset account-status-notification-handler \  
--hostname localhost \  
--port 4444 \  
--bindDN uid=admin \  
--bindPassword password \  
--usePkcs12TrustStore /path/to/openssl/config/keystore \  
--trustStorePasswordFile /path/to/openssl/config/keystore.pin \  
--no-prompt
```

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

```
/path/to/openssl/bin/dsconfig \  
delete-account-status-notification-handler \  
--handler-name "OpenIDM Notification Handler" \  
--hostname localhost \  
--port 4444 \  
--bindDN uid=admin \  
--bindPassword password \  
--usePkcs12TrustStore /path/to/openssl/config/keystore \  
--trustStorePasswordFile /path/to/openssl/config/keystore.pin \  
--no-prompt  
The Account Status Notification Handler was deleted successfully
```

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

```
rm /path/to/openssl/lib/extensions/openssl-openidm-account-change-notification-handler*
```

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

```
/path/to/openssl/bin/stop-ds --restart
```


Chapter 3

Synchronize 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 2012 R2).

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.

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.
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&_queryFilter=username+eq+'${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`.

- OAuth2 Access Token URL

If you are using the authentication type `OAuth2 Access Token`, enter the token URL; for example:

```
https://am.example.com/am/oauth2/realms/root/access_token
```

- OAuth2 Scope

If you are using the authentication type **OAuth2 Access Token**, enter the OAuth2 token scope; for example **fr:idm:\***.

- Select authentication type

Select the authentication type that Active Directory will use to authenticate to IDM:

- To use plain HTTP authentication, select OpenIDM Header.
- To use SSL mutual authentication, select Certificate.
- To use AM bearer tokens, select OAuth2 Access Token.

## 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 PKCS #12 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 PKCS #12 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 PKCS #12 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 PKCS #12 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 format must be PKCS #12.

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"] },
 ...
]
```

For more information about secret stores, see "Configuring Secret Stores" in the *Security Guide*.

- 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).

## 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 the folder in which Service will store its output data files

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 folder in which Service will store its log files

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 **Certificate** as the authentication type during setup, complete the following sub-steps; otherwise, your setup is now complete.

- 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.

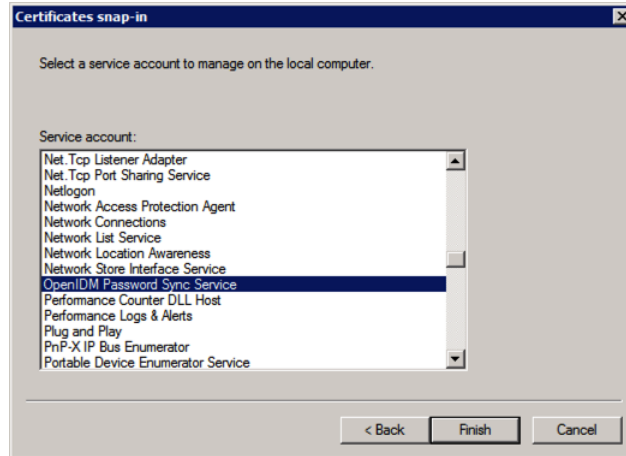
In a production environment, use a certificate that has been issued by a certificate authority. For test purposes, you can use the self-signed certificate that is generated by IDM on first startup.

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.

## 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 PKCS #12 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 PKCS #12 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 `encKey`, `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&_queryFilter=uid+eq+${employeeId}`

You can set the registry key `userSearchFilterStrict` to control the behavior of `userSearchFilter`:

- `userSearchFilterStrict = true` - requires `userSearchFilter` to return a value, which you can use to filter out users/groups from being password-synced.
- `userSearchFilterStrict = false` - this is the default behavior. `userSearchFilter` is only used to look up the Active Directory user attribute, and when it fails, password sync is still attempted with a default attribute.

## 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:

- `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.
- `maxFileRetry` - the maximum number of password change retry attempts after which the plugin stops attempting to send changes.
- `netTimeout` - the length of time (in milliseconds) before the plugin stops attempting a connection.
- `pollEach` - the interval (in seconds) between each attempt to send changes.

## 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.

## Key to configure infinite loop prevention

### Note

This feature requires AD Password Synchronization Plugin version 1.4.0 or later. Because version 1.4.0 can fail to make a secure connection with certain Windows versions, ForgeRock recommends using a later version.

When Active Directory syncs passwords with IDM bidirectionally, it is possible to enter an infinite loop, where Active Directory and IDM are constantly updating the password and telling the other system to do the same. To help prevent this situation, you can set the `pwdChangeInterval` key to the number of seconds that must elapse between password updates.

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.

## 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.

## 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.

# IDM Glossary

|                    |                                                                                                                                                                                                                                                                                                                                                                                    |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| correlation query  | A correlation query specifies an expression that matches existing entries in a source repository to one or more entries in a target repository. A correlation query might be built with a script, but it is not the same as a correlation script. For more information, see <i>"Correlating Source Objects With Existing Target Objects"</i> in the <i>Synchronization Guide</i> . |
| 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 <b>correlation query</b> , 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 <b>assignment</b> . 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 <a href="#">JSON site</a> .                                                                                                                                                                                                                |
| 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 <a href="#">JSON Pointer RFC</a> .                                                                                                                                                                                              |

|                 |                                                                                                                                                                                                                                                                                                                             |
|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| JWT             | JSON Web Token. As noted in the <a href="#">JSON Web Token draft IETF Memo</a> , "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 <a href="#">JWT_SESSION</a> 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 more information, see <a href="#">What is OSGi?</a> Currently, only the <a href="#">Apache Felix</a> 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 <a href="#">"Managed Roles"</a> in the <i>Object Modeling Guide</i> .                                                                                                                             |
| 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.                                                      |