



Platform Setup Guide

/ ForgeRock Identity Platform 7

Latest update: 7.0.0

Copyright © 2020-2021 ForgeRock AS.

Abstract

Guide to setting up the ForgeRock Identity Platform™ locally.



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.

ForgeRock® and ForgeRock Identity Platform™ are trademarks of ForgeRock Inc. or its subsidiaries in the U.S. and in other countries. Trademarks are the property of their respective owners.

UNLESS OTHERWISE MUTUALLY AGREED BY THE PARTIES IN WRITING, LICENSOR OFFERS THE WORK AS-IS AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE WORK, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF TITLE, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR THE ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO SUCH EXCLUSION MAY NOT APPLY TO YOU.

EXCEPT TO THE EXTENT REQUIRED BY APPLICABLE LAW, IN NO EVENT WILL LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES ARISING OUT OF THIS LICENSE OR THE USE OF THE WORK, EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

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 NONINFRINGEMENT 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 NONINFRINGEMENT 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. Deployment Overview	1
Component Interaction	2
Server Settings	3
2. Deployment One - Separate Identity Stores	5
Set up Your Data Stores	5
Set Up a Container	7
Secure Connections	7
Configure AM	9
Set up IDM	40
3. Deployment Two - Shared Identity Store	44
Set up Your Data Stores	44
Set Up a Container	46
Secure Connections	46
Configure AM	48
Set up IDM	79
4. Deploy the Platform UIs	84
5. Migrate From a Full Stack Deployment	88
6. HSMs and ForgeRock Software	90
On Protecting Secrets	90
Performance	91
How ForgeRock Services Interact with HSMs	92
HSM Features and ForgeRock Services	94
Configure ForgeRock Services to Use an HSM	96

Overview

The ForgeRock Identity Platform is the only offering for access management, identity management, user-managed access, directory services, and an identity gateway, designed and built as a single, unified platform.

This guide lets you set up the platform components without using the ForgeRock Identity Cloud.

Quick Start

 <p>Start Here</p> <p>Learn about the two sample deployments shown in this guide.</p>	 <p>Deployment One</p> <p>Set up the ForgeRock Identity Platform with separate identity stores between AM and IDM.</p>
 <p>Deployment Two</p> <p>Set up the ForgeRock Identity Platform with a DS identity store that is shared between AM and IDM.</p>	 <p>Platform UIs</p> <p>Learn about the three Platform UIs and deploy them with either Deployment One or Deployment Two.</p>

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.

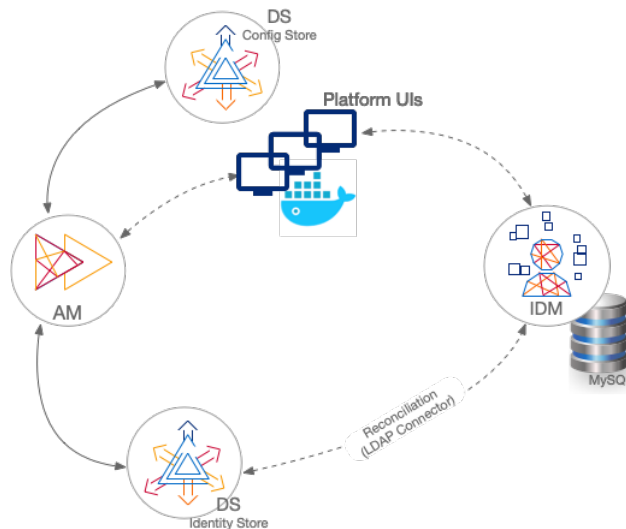
Chapter 1

Deployment Overview

This guide shows two sample deployments:

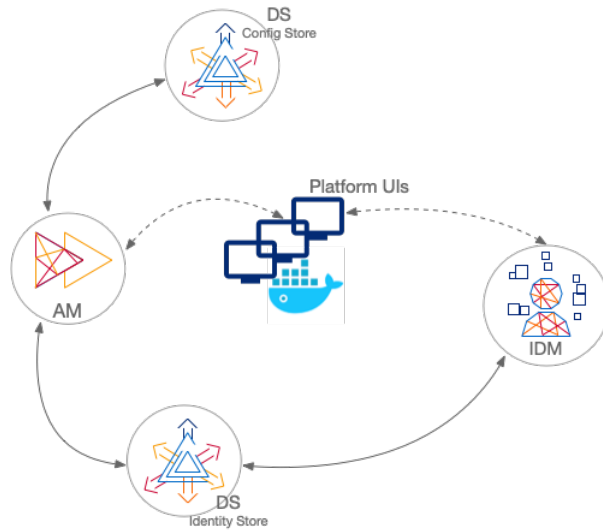
"Deployment One - Separate Identity Stores"

This deployment has an external DS instance configured as the AM configuration store, and a second external DS instance configured as the AM identity store. The IDM repository is an external JDBC database. The sample was tested with MySQL. The deployment uses an LDAP connector to synchronize the identities between IDM and AM:



"Deployment Two - Shared Identity Store"

This deployment has an external DS instance configured as the AM configuration store. The AM and IDM servers share an external DS instance as the identity store, and no synchronization configuration is required:

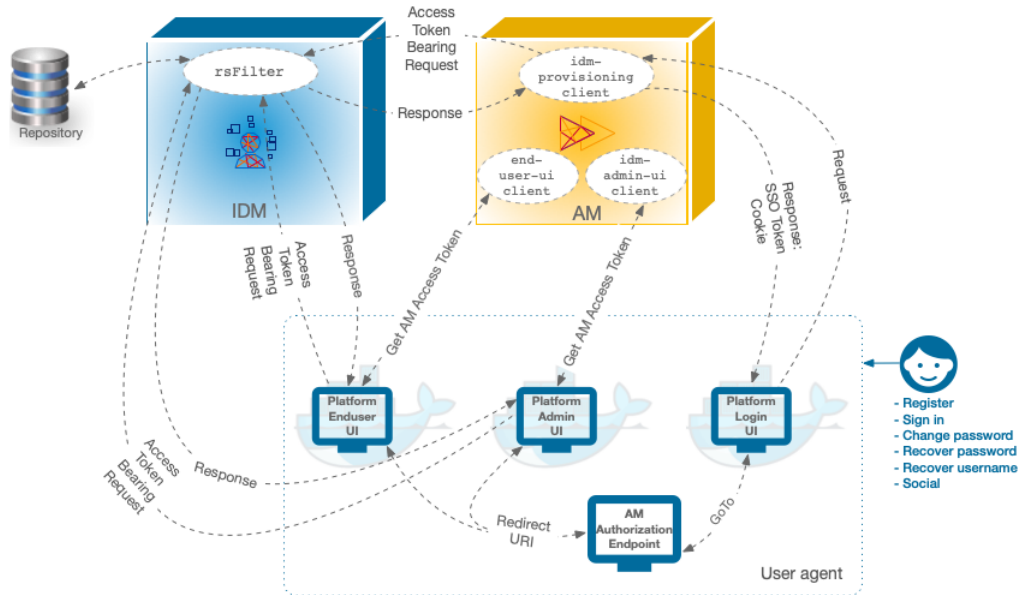


Important

In both deployments, the Platform UIs run in separate Docker containers. If you want to use the Platform UIs, get Docker before you start.

Component Interaction

A platform configuration relies on multiple components working together. The following image shows how the AM OAuth 2 clients interact with the IDM resource server filter (**rsFilter**) to grant access through the Platform UIs:



1. The Platform UIs send a request to the AM Authorization Endpoint.
2. If the end user is authenticated, the user agent is redirected back to the UI, according to the Redirection URI request parameter.
3. If the end user is not authenticated, the AM Authorization Endpoint redirects the user agent to the Platform Login UI.
4. After successful authentication, the Platform Login UI redirects the user agent back to the AM Authorization Endpoint, according to the GoTo request parameter.

Server Settings

This guide assumes that all servers are deployed on their own hosts, with the following server settings. Adjust the settings to match your own deployment.

Important

If you are testing these deployments on a single host machine, add aliases for the machine names in your `/etc/hosts` file and *change the port numbers to avoid conflicts*.

If you are deploying the entire platform on a single host, the recommended alternative is to use the ForgeOps Cloud Developer's Kit (CDK) on Minikube.

- AM host: `am.example.com`

- AM port: 8080
- External DS configuration store host: config.example.com
- External DS configuration store ports:

```
adminConnectorPort 4444
ldapPort 1389
ldapsPort 1636
replicationPort 8989
```

- External DS identity store host: identities.example.com

These settings apply to both the separate and shared DS identity stores.

- External DS identity store ports:

```
adminConnectorPort 4444
ldapPort 1389
ldapsPort 1636
replicationPort 8989
```

- IDM host: openidm.example.com
- IDM ports: HTTP 8080, HTTPS 8443
- Platform Admin UI: http://localhost:8082
- Platform Login UI: http://localhost:8083
- Platform End User UI: http://localhost:8888

Chapter 2

Deployment One - Separate Identity Stores

This deployment assumes that you are using the following data stores:

- An external DS instance as the AM configuration store.
- A separate external DS instance as the AM identity store.
- A MySQL repository as the IDM data store.

Note

The IDM End User UI is not supported in a platform deployment, as it does not support authentication through AM. You can use the Platform UIs with this deployment, or create your own UI that supports authentication through AM.

- "Set up Your Data Stores"
- "Set Up a Container"
- "Secure Connections"
- "Configure AM"
- "Set up IDM"

Set up Your Data Stores

Configuration Store

1. Set up a DS server as an external configuration data store, using the `am-config` setup profile.

For more information about this step, see [setup profiles](#) in the *DS Installation Guide*.

This command sets up the config store with the parameters listed in "Server Settings":

```
/path/to/openssl/setup \  
--deploymentKeyPassword password \  
--rootUserDN uid=admin \  
--rootUserPassword strongAdminPa55word \  
--monitorUserPassword strongMonitorPa55word \  
--hostname config.example.com \  
--adminConnectorPort 4444 \  
--ldapPort 1389 \  
--enableStartTls \  
--ldapsPort 1636 \  
--httpsPort 8443 \  
--replicationPort 8989 \  
--profile am-config \  
--set am-config/amConfigAdminPassword:5up35tr0ng \  
--acceptLicense
```

Make a note of the generated deployment key. You will need it to export the server certificate later in this procedure. You can set the deployment key as a variable in this terminal window. For example:

```
export DEPLOYMENT_KEY=deployment-key
```

2. Start the DS server:

```
/path/to/openssl/bin/start-ds
```

Identity Store

- Set up a DS server as an AM identity store, using the `am-identity-store` setup profile.

For more information about this step, see [setup profiles in the DS Installation Guide](#).

This command sets up the identity store with the parameters listed in "Server Settings":

```
/path/to/openssl/setup \  
--deploymentKeyPassword password \  
--rootUserDN uid=admin \  
--rootUserPassword strongAdminPa55word \  
--monitorUserPassword strongMonitorPa55word \  
--hostname identities.example.com \  
--adminConnectorPort 4444 \  
--ldapPort 1389 \  
--enableStartTls \  
--ldapsPort 1636 \  
--replicationPort 8989 \  
--profile am-identity-store \  
--set am-identity-store/amIdentityStoreAdminPassword:5up35tr0ng \  
--acceptLicense
```

Make a note of the generated deployment key. You will need it to export the server certificate later in this procedure. You can set the deployment key as a variable in this terminal window. For example:

```
export DEPLOYMENT_KEY=deployment-key
```

Start the DS server:

```
/path/to/openssl/bin/start-ds
```

Set Up a Container

- Install a Java container to deploy AM.

This guide assumes that you are using Apache Tomcat.

For instructions on setting up Tomcat, see *Preparing Apache Tomcat in the AM Installation Guide*.

Secure Connections

Important

From DS 7 onwards, you *must* secure connections to DS servers.

1. Configure your AM container for SSL connections.
2. Create a new directory that will house a dedicated truststore for AM:

```
mkdir -p /path/to/openam-security/
```

3. Make a copy of your JDK's default truststore; for example, `$JAVA_HOME/lib/security/cacerts`, name it `truststore`, and place it in the directory you created:

```
cp $JAVA_HOME/lib/security/cacerts /path/to/openam-security/truststore
```

Note

The default password of the `lib/security/cacerts` truststore is `changeit`. You should change this password in a production environment.

4. *On each DS server*, export the DS server certificate.

You must run these commands in the same terminal window where you set the `DEPLOYMENT_KEY` variable.

On `config.example.com`:

```
/path/to/openssl/bin/openssl dskeygen export-ca-cert \
--deploymentKey $DEPLOYMENT_KEY \
--deploymentKeyPassword password \
--alias config-ca-cert \
--outputFile config-ca-cert.pem
```

On `identities.example.com`:

```
/path/to/openssl/bin/openssl dskeygen export-ca-cert \
--deploymentKey $DEPLOYMENT_KEY \
--deploymentKeyPassword password \
--alias identities-ca-cert \
--outputFile identities-ca-cert.pem
```

- Import each DS server certificate into the new AM truststore. If you are not testing this example on a single host, you might need to copy each certificate file onto the AM host machine first:

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

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

- Configure the truststore in Apache Tomcat so that AM can access it.

Append the truststore settings to the `CATALINA_OPTS` variable in the `$CATALINA_BASE/bin/setenv.sh` file.

For example:

```
CATALINA_OPTS="-Djavax.net.ssl.trustStore=/path/to/openam-security/truststore\
-Djavax.net.ssl.trustStorePassword=changeit\
-Djavax.net.ssl.trustStoreType=jks"
```

Refer to your specific container's documentation for information on configuring truststores.

Configure AM

When your external data stores are configured, follow these procedures to configure AM with the ForgeRock Identity Platform:

- "Install AM"
- "Configure OAuth Clients"
- "Configure Authentication Trees"
- "Map Authentication Trees"
- "Configure an OAuth 2.0 Provider Service"
- "Configure an IDM Provisioning Service"
- "Enable the Password Update Tree (optional)"
- "Enable CORS Support"

Install AM

1. Follow the instructions in the *AM Installation Guide* to download AM. Make sure you download the `.zip` file, not just the `.war` file.
2. Follow the instructions in the *AM Installation Guide* to prepare your environment, and prepare a web application container.

Use Apache Tomcat as the application container, listening on the default port (`8080`).

3. Copy the AM `.war` file to deploy in Apache Tomcat as `am.war`:

```
cp AM-7.0.0.war /path/to/tomcat/webapps/am.war
```

4. Start Tomcat if it is not already running.
5. Navigate to the deployed AM application; for example, `http://am.example.com:8080/am/`.
6. Select Create New Configuration to create a custom configuration.
7. Accept the license agreement and click Continue.
8. Set a password for the default user, `amAdmin`.

This guide assumes that the `amAdmin` password is `Passw0rd`.

9. On the Server Settings screen, enter your AM server settings; for example:

- Server URL: `http://am.example.com:8080`
- Cookie Domain: `example.com`
- Platform Locale: `en_US`
- Configuration Directory: `/path/to/openam`

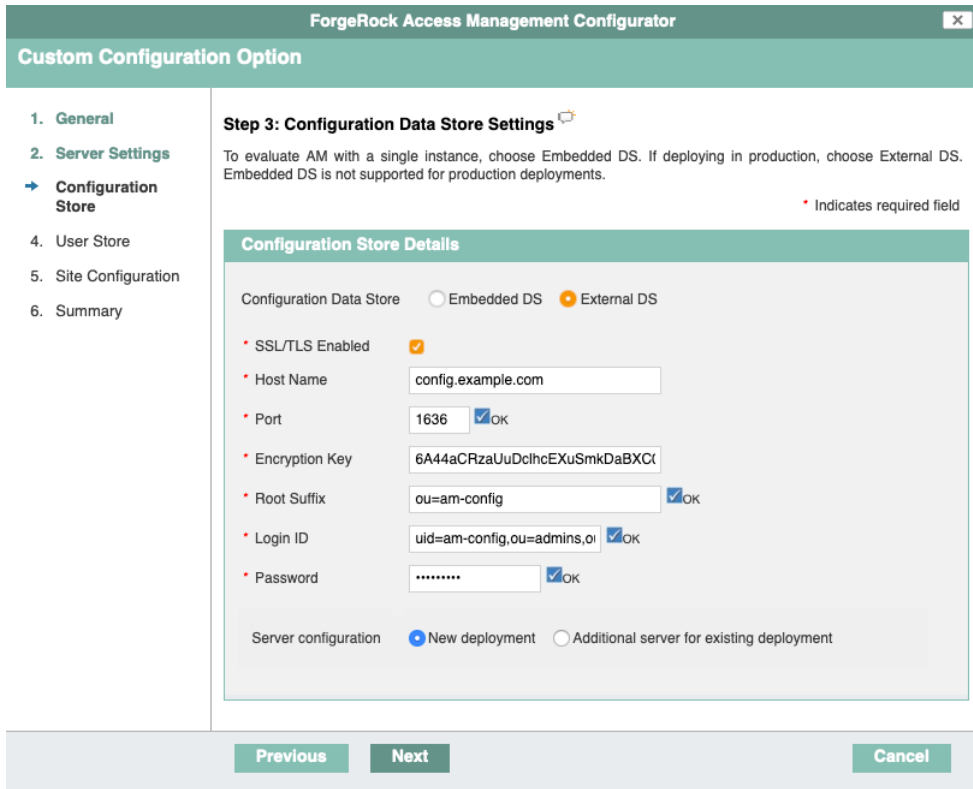
10. On the Configuration Data Store Settings screen, select External DS, and enter the details for the DS instance that you set up as a configuration store.

This list reflects the DS configuration store installed with the listed "Server Settings".

- SSL/TLS: `Enabled`

TLS is required in a production deployment.

- Host Name: `config.example.com`
- Port: `1636`
- Encryption Key: (generated encryption key)
- Root Suffix: `ou=am-config`
- Login ID: `uid=am-config,ou=admins,ou=am-config`
- Password: `5up35tr0ng`
- Server configuration: `New deployment`



11. On the User Data Store Settings screen, select External User Data Store, and enter the details for the DS instance that you set up as an identity store.

This list reflects the DS identity store installed with the listed "Server Settings".

- User Data Store Type: **ForgeRock Directory Services (DS)**
- SSL/TLS: **Enabled**

TLS is required in a production deployment.

- Host Name: **identities.example.com**
- Port: **1636**
- Root Suffix: **ou=identities**
- Login ID: **uid=am-identity-bind-account,ou=admins,ou=identities**

- Password: `5up35tr0ng`

12. On the Site Configuration screen, select No.

13. Click Create Configuration.

Configure OAuth Clients

This procedure configures *four* OAuth 2.0 clients:

1. If you're not currently logged in to the AM console as the `amAdmin` user, log in.
2. Configure an `idm-provisioning` client to make calls to IDM:
 - a. In the Top Level Realm, select Applications > OAuth 2.0 > Clients, and click Add Client.
 - b. Enter the following details:
 - Client ID: `idm-provisioning`
 - Client secret: `openidm`

- Scopes: `fr:idm:*`
- c. Click Create.
 - d. On the Advanced tab:
 - Response Types: Check that `token` is present (it should be there by default).
 - Grant Types: Remove `Authorization Code` and add `Client Credentials`.
 - e. Click Save Changes.
3. Configure an `idm-resource-server` client to introspect the access token:
 - a. In the Top Level Realm, select Applications > OAuth 2.0 > Clients, and click Add Client.
 - b. Enter the following details:
 - Client ID: `idm-resource-server`
 - Client secret: `password`

The value of this field must match the `clientSecret` that you will set in the `rsFilter` module in the IDM authentication configuration (`/path/to/openidm/conf/authentication.json`) during your IDM setup.

 - Scopes: `am-introspect-all-tokens am-introspect-all-tokens-any-realm`
 - c. Click Create.
 4. Configure an `idm-admin-ui` client that will be used by the Platform Admin UI:
 - a. In the Top Level Realm, select Applications > OAuth 2.0 > Clients, and click Add Client.
 - b. Enter the following details:
 - Client ID: `idm-admin-ui`
 - Client Secret: (no client secret is required)
 - Redirection URIs: `http://openidm.example.com:8080/platform/appAuthHelperRedirect.html http://openidm.example.com:8080/platform/sessionCheck.html http://openidm.example.com:8080/admin/appAuthHelperRedirect.html http://openidm.example.com:8080/admin/sessionCheck.html http://localhost:8082/appAuthHelperRedirect.html http://localhost:8082/sessionCheck.html`
 - Scopes: `openid fr:idm:*`

Note

At a minimum, the scopes that you set here must include the scopes that you will set in the `rsFilter` authentication configuration (`/path/to/openidm/conf/authentication.json`) during your IDM setup.

- c. Click Create.
 - d. On the Core tab:
 - Client type: Select `Public`.Click Save Changes.
 - e. On the Advanced tab:
 - Grant Types: Add `Implicit`.
 - Token Endpoint Authentication Method: Select `none`.
 - Implied consent: Enable.Click Save Changes.
5. Configure an `end-user-ui` client that will be used by the Platform End User UI:
- a. In the Top Level Realm, select Applications > OAuth 2.0 > Clients, and click Add Client.
 - b. Enter the following details:
 - Client ID: `end-user-ui`
 - Client Secret: (no client secret is required)
 - Redirection URIs: `http://openidm.example.com:8080/enduser/appAuthHelperRedirect.html` `http://openidm.example.com:8080/enduser/sessionCheck.html` `http://localhost:8888/appAuthHelperRedirect.html` `http://localhost:8888/sessionCheck.html`
 - Scopes: `openid fr:idm:*`

Note

At a minimum, the scopes that you set here must include the scopes that you will set in the `rsFilter` authentication configuration (`/path/to/openidm/conf/authentication.json`) during your IDM setup.

- c. Click Create.
- d. On the Core tab:

- Client type: Select **Public**.

Click Save Changes.

e. On the Advanced tab:

- Grant Types: Add **Implicit**.
- Token Endpoint Authentication Method: Select **none**.
- Implied Consent: Enable.

Click Save Changes.

Configure Authentication Trees

The platform deployment relies on three authentication trees to enable authentication through AM. When you extract the AM `.zip` file, you will get a `sample-trees-7.0.0.zip` file that contains a number of sample authentication trees, in JSON files. Use the Amster command-line utility to import the platform authentication trees into your AM configuration:

1. Extract the `sample-trees-7.0.0.zip` file and list the sample trees in the `/path/to/openam-samples/root/AuthTree` directory:

```
ls /path/to/openam-samples/root/AuthTree
Agent.json PlatformForgottenUsername.json
Example.json PlatformLogin.json
Facebook-ProvisionIDMAccount.json PlatformProgressiveProfile.json
Google-AnonymousUser.json PlatformRegistration.json
Google-DynamicAccountCreation.json PlatformResetPassword.json
HmacOneTimePassword.json PlatformUpdatePassword.json
PersistentCookie.json RetryLimit.json
```

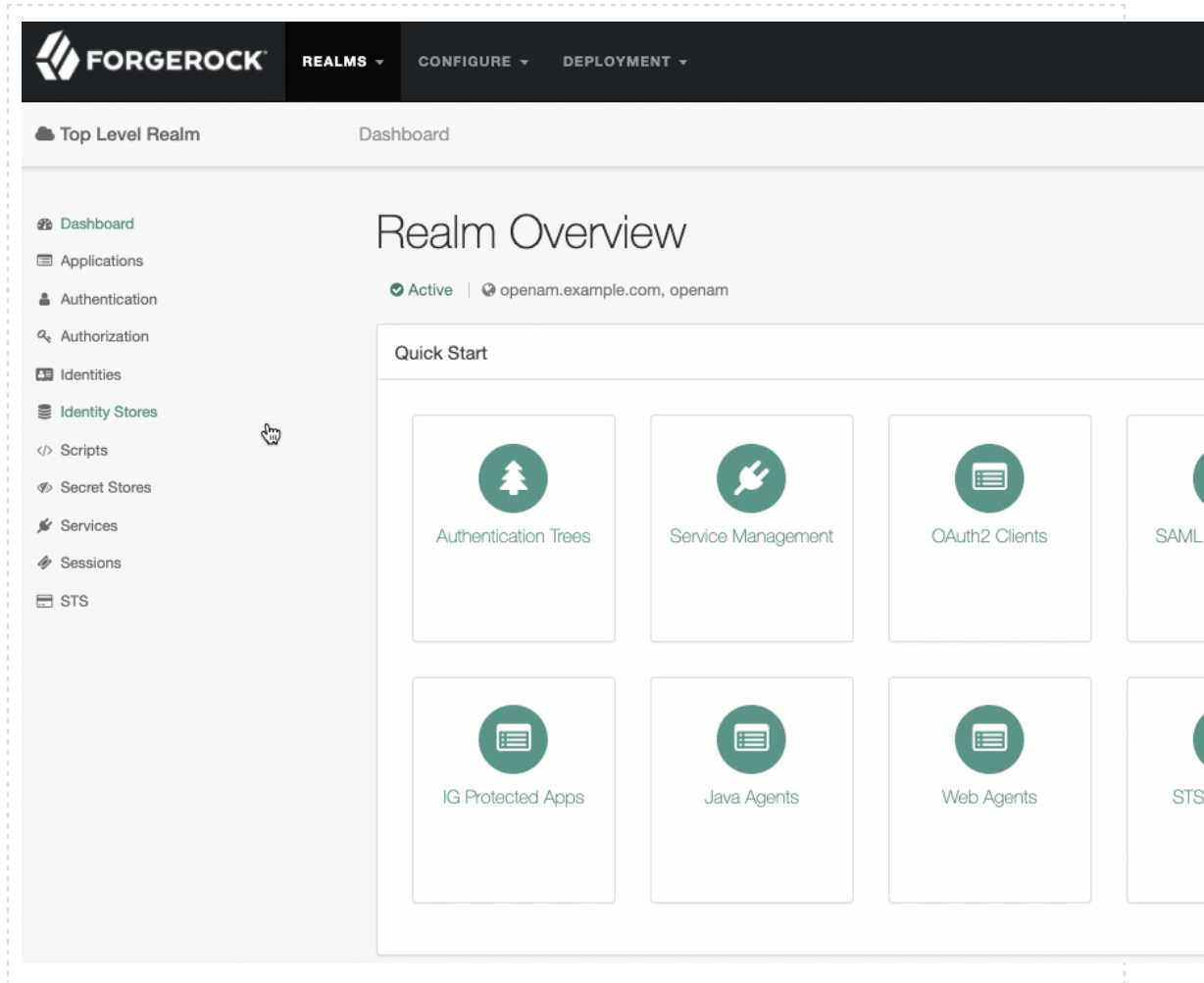
2. Download and install Amster.
3. Start Amster, then connect to your AM instance:

```
./amster
Amster OpenAM Shell (7.0.0 build @build.number@, JVM: 11.0.4)
Type ':help' or ':h' for help.
-----
am> connect --interactive http://am.example.com:8080/am
Sign in
User Name: amAdmin
Password: *****
amster am.example.com:8080>
```

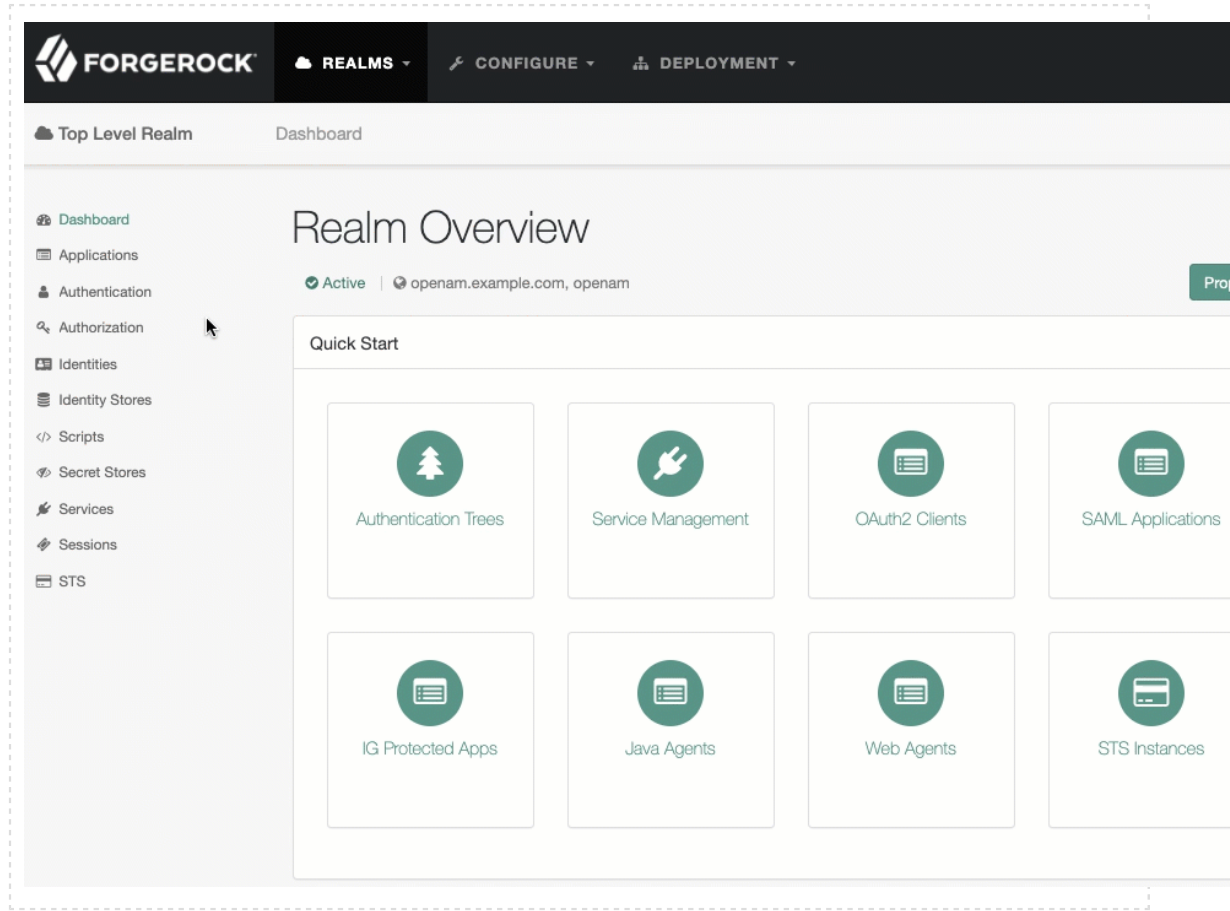
4. Import the sample authentication trees and nodes:

```
amster am.example.com:8080> import-config --path /path/to/openam-samples/root
Importing directory /path/to/openam-samples/root/AcceptTermsAndConditions
Imported /path/to/openam-samples/root/AcceptTermsAndConditions/b4a0e915-
c15d-4b83-9c9d-18347d645976.json
...
Import completed successfully
```

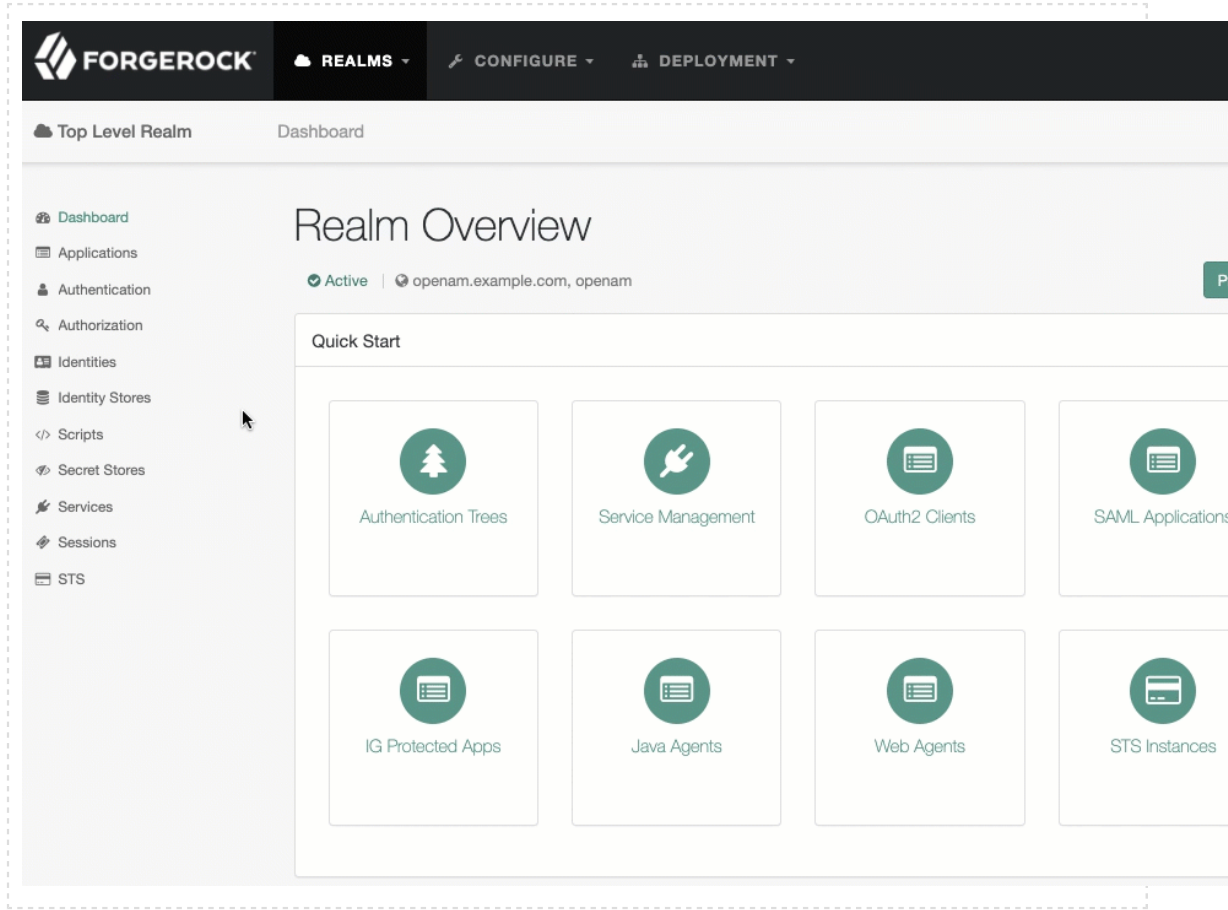
5. If you're not currently logged in to the AM console as the `amAdmin` user, log in.
6. Configure the `PlatformRegistration` tree:
 - a. In the Top Level Realm, select Authentication > Trees, and click on PlatformRegistration.
 - b. On the `PlatformRegistration` tree, add a `Success URL` node between `Increment Login Count` and `Success`, and set its value to `http://localhost:8888`.
 - + *Show Me*



- c. Click Save.
7. Configure the `PlatformLogin` tree:
- a. In the Top Level Realm, select Authentication > Trees, and click on PlatformLogin.
 - b. On the `PlatformLogin` tree, add a `Success URL` node between `Inner Tree Evaluator` and `Success`, and set its value to `http://localhost:8888`.
- + *Show Me*



- c. Click Save.
8. Configure the `PlatformResetPassword` tree:
- a. In the Top Level Realm, select Authentication > Trees, and click on PlatformResetPassword.
 - b. On the `PlatformResetPassword` tree, add a `Success URL` node between `Patch Object` and `Success`, and set its value to `http://localhost:8888`.
- + Show Me



c. Click Save.

9. For the authentication trees that require email, set the External Login Page URL.

In the Top Level Realm, select Authentication > Settings, and click the General tab. Set External Login Page URL to <http://localhost:8083>, then click Save Changes.

Map Authentication Trees

Map the platform trees to the corresponding Self-Service endpoints. For more information about this step, see "Configure Self-Service Trees Endpoints" in the *Platform Self-Service Guide*.

1. From the top menu, select Configure > Global Services > Self Service Trees.
2. Add the following Tree Mappings:

Key	Value
registration	PlatformRegistration
login	PlatformLogin
resetPassword	PlatformResetPassword

Self Service Trees

Realm Defaults

Tree Mapping ?

registration	<input type="text" value="PlatformRegistration"/>	✕
login	<input type="text" value="PlatformLogin"/>	✕
resetPassword	<input type="text" value="PlatformResetPassword"/>	✕

+ Add

3. Click Save Changes.

Configure an OAuth 2.0 Provider Service

1. If you're not currently logged in to the AM console as the `amAdmin` user, log in.
2. In the Top Level Realm, select Services, and click Add a Service.
3. Under Choose a service type, select OAuth2 Provider.
4. For Client Registration Scope Whitelist, add the following scopes:
 - `fr:ldm:*`
 - `am-introspect-all-tokens`
 - `openid`
5. Click Create.
6. On the Advanced tab, check that the following item is listed under Response Type Plugins, and add it if not:


```
id_token|org.forgerock.openidconnect.IdTokenResponseTypeHandler
```

7. Click Save Changes.
8. On the Consent tab, enable Allow Clients to Skip Consent.
9. Click Save Changes.

Configure an IDM Provisioning Service

1. If you're not currently logged in to the AM console as the `amAdmin` user, log in.
2. From the top menu, select Configure > Global Services > IDM Provisioning.
3. Set the following fields:
 - Enabled
 - Deployment URL: `http://openidm.example.com:8080`
 - Deployment Path: `openidm`
 - IDM Provisioning Client: `idm-provisioning`
4. Click Save Changes.

Enable the Password Update Tree (optional)

To let end users update their own passwords (using the `PlatformUpdatePassword` tree), add the `idm-provisioning` service as an authorized client of the `OAuth2 Provider` service.

You cannot use the UI for this step—you must update the OAuth 2.0 configuration over REST:

1. Use the following request to get the token ID:

```
curl \
--request POST \
--header "Content-Type: application/json" \
--header "X-OpenAM-Username: amAdmin" \
--header "X-OpenAM-Password: Password" \
--header "Accept-API-Version: resource=2.0, protocol=1.0" \
'http://am.example.com:8080/am/json/realms/root/authenticate'
{
  "tokenId": "tokenId",
  "successUrl": "/openam/console",
  "realm": "/"
}
```

2. Make a GET request to the `oauth-oidc` endpoint to return the current OAuth configuration:
+ *Example Curl Request*

```

curl \
-X GET \
-H 'Accept-API-Version: protocol=1.0,resource=1.0' \
-H 'X-Requested-With: XMLHttpRequest' \
-H 'Origin: http://am.example.com:8080' \
-H 'Referer: http://am.example.com:8080/am/ui-admin/' \
-H 'Cookie: amlbcookie=01; iPlanetDirectoryPro=tokenId' \
"http://am.example.com:8080/am/json/realms/root/realms-config/services/oauth-oidc"
{
  "core0Auth2Config": {
    "refreshTokenLifetime": 604800,
    "accessTokenLifetime": 3600,
    "usePolicyEngineForScope": false,
    "codeLifetime": 120,
    "issueRefreshTokenOnRefreshedToken": true,
    "macaroonTokensEnabled": false,
    "issueRefreshToken": true,
    "accessTokenModificationScript": "d22f9a0c-426a-4466-b95e-d0f125b0d5fa",
    "statelessTokensEnabled": false
  },
  "core0IDCConfig": {
    "supportedIDTokenEncryptionMethods": [
      "A256GCM",
      "A192GCM",
      "A128GCM",
      "A128CBC-HS256",
      "A192CBC-HS384",
      "A256CBC-HS512"
    ],
    "jwtTokenLifetime": 3600,
    "supportedClaims": [],
    "supportedIDTokenEncryptionAlgorithms": [
      "ECDH-ES+A256KW",
      "ECDH-ES+A192KW",
      "RSA-OAEP",
      "ECDH-ES+A128KW",
      "RSA-OAEP-256",
      "A128KW",
      "A256KW",
      "ECDH-ES",
      "dir",
      "A192KW"
    ],
    "supportedIDTokenSigningAlgorithms": [
      "PS384",
      "RS384",
      "ES384",
      "HS256",
      "HS512",
      "ES256",
      "RS256",
      "HS384",
      "ES512",
      "PS256",
      "PS512",
      "RS512"
    ]
  },
}

```

```

    "oidcClaimsScript": "36863ffb-40ec-48b9-94b1-9a99f71cc3b5"
  },
  "advancedOAuth2Config": {
    "supportedScopes": [
      "openid",
      "fr:idm:*",
      "am-introspect-all-tokens"
    ],
    "tlsCertificateRevocationCheckingEnabled": false,
    "codeVerifierEnforced": "false",
    "tokenSigningAlgorithm": "HS256",
    "authenticationAttributes": [
      "uid"
    ],
    "passwordGrantAuthService": "[Empty]",
    "defaultScopes": [],
    "tlsClientCertificateHeaderFormat": "URLENCODED_PEM",
    "scopeImplementationClass": "org.forgerock.openam.oauth2.OpenAMScopeValidator",
    "responseTypeClasses": [
      "code|org.forgerock.oauth2.core.AuthorizationCodeResponseTypeHandler",
      "id_token|org.forgerock.openidconnect.IdTokenResponseTypeHandler",
      "device_code|org.forgerock.oauth2.core.TokenResponseTypeHandler",
      "token|org.forgerock.oauth2.core.TokenResponseTypeHandler"
    ],
    "tlsCertificateBoundAccessTokensEnabled": true,
    "hashSalt": "changeme",
    "moduleMessageEnabledInPasswordGrant": false,
    "tokenEncryptionEnabled": false,
    "tokenCompressionEnabled": false,
    "grantTypes": [
      "implicit",
      "urn:iETF:params:oauth:grant-type:saml2-bearer",
      "refresh_token",
      "password",
      "client_credentials",
      "urn:iETF:params:oauth:grant-type:device_code",
      "authorization_code",
      "urn:openid:params:grant-type:ciba",
      "urn:iETF:params:oauth:grant-type:uma-ticket",
      "urn:iETF:params:oauth:grant-type:jwt-bearer"
    ],
    "displayNameAttribute": "cn",
    "macaroonTokenFormat": "V2",
    "supportedSubjectTypes": [
      "public"
    ]
  },
  "advancedOIDCConfig": {
    "storeOpsTokens": true,
    "defaultACR": [],
    "supportedRequestParameterEncryptionEnc": [
      "A256GCM",
      "A192GCM",
      "A128GCM",
      "A128CBC-HS256",
      "A192CBC-HS384",
      "A256CBC-HS512"
    ],
    "claimsParameterSupported": false,
  }
}

```

```

"amrMappings": {},
"supportedUserInfoEncryptionEnc": [
  "A256GCM",
  "A192GCM",
  "A128GCM",
  "A128CBC-HS256",
  "A192CBC-HS384",
  "A256CBC-HS512"
],
"authorisedIdmDelegationClients": [],
"alwaysAddClaimsToToken": false,
"supportedUserInfoSigningAlgorithms": [
  "ES384",
  "HS256",
  "HS512",
  "ES256",
  "RS256",
  "HS384",
  "ES512"
],
"supportedRequestParameterEncryptionAlgorithms": [
  "ECDH-ES+A256KW",
  "ECDH-ES+A192KW",
  "ECDH-ES+A128KW",
  "RSA-OAEP",
  "RSA-OAEP-256",
  "A128KW",
  "A256KW",
  "ECDH-ES",
  "dir",
  "A192KW"
],
"supportedTokenIntrospectionResponseEncryptionEnc": [
  "A256GCM",
  "A192GCM",
  "A128GCM",
  "A128CBC-HS256",
  "A192CBC-HS384",
  "A256CBC-HS512"
],
"supportedTokenIntrospectionResponseSigningAlgorithms": [
  "PS384",
  "RS384",
  "EdDSA",
  "ES384",
  "HS256",
  "HS512",
  "ES256",
  "RS256",
  "HS384",
  "ES512",
  "PS256",
  "PS512",
  "RS512"
],
"authorisedOpenIdConnectSSOClients": [],
"idTokenInfoClientAuthenticationEnabled": true,
"supportedRequestParameterSigningAlgorithms": [
  "PS384",

```

```

    "RS384",
    "ES384",
    "HS256",
    "HS512",
    "ES256",
    "RS256",
    "HS384",
    "ES512",
    "PS256",
    "PS512",
    "RS512"
  ],
  "supportedUserInfoEncryptionAlgorithms": [
    "ECDH-ES+A256KW",
    "ECDH-ES+A192KW",
    "RSA-OAEP",
    "ECDH-ES+A128KW",
    "RSA-OAEP-256",
    "A128KW",
    "A256KW",
    "ECDH-ES",
    "dir",
    "A192KW"
  ],
  "supportedTokenIntrospectionResponseEncryptionAlgorithms": [
    "ECDH-ES+A256KW",
    "ECDH-ES+A192KW",
    "RSA-OAEP",
    "ECDH-ES+A128KW",
    "RSA-OAEP-256",
    "A128KW",
    "A256KW",
    "ECDH-ES",
    "dir",
    "A192KW"
  ],
  "supportedTokenEndpointAuthenticationSigningAlgorithms": [
    "PS384",
    "RS384",
    "ES384",
    "HS256",
    "HS512",
    "ES256",
    "RS256",
    "HS384",
    "ES512",
    "PS256",
    "PS512",
    "RS512"
  ],
  "loaMapping": {}
},
"clientDynamicRegistrationConfig": {
  "dynamicClientRegistrationSoftwareStatementRequired": false,
  "dynamicClientRegistrationScope": "dynamic_client_registration",
  "requiredSoftwareStatementAttestedAttributes": [
    "redirect_uris"
  ],
  "generateRegistrationAccessTokens": true,

```

```

    "allowDynamicRegistration": false
  },
  "cibaConfig": {
    "supportedCibaSigningAlgorithms": [
      "ES256",
      "PS256"
    ],
    "cibaAuthReqIdLifetime": 600,
    "cibaMinimumPollingInterval": 2
  },
  "consent": {
    "enableRemoteConsent": false,
    "supportedRcsRequestSigningAlgorithms": [
      "PS384",
      "RS384",
      "ES384",
      "HS256",
      "HS512",
      "ES256",
      "RS256",
      "HS384",
      "ES512",
      "PS256",
      "PS512",
      "RS512"
    ],
    "supportedRcsResponseSigningAlgorithms": [
      "PS384",
      "RS384",
      "ES384",
      "HS256",
      "HS512",
      "ES256",
      "RS256",
      "HS384",
      "ES512",
      "PS256",
      "PS512",
      "RS512"
    ],
    "clientsCanSkipConsent": true,
    "supportedRcsRequestEncryptionAlgorithms": [
      "ECDH-ES+A256KW",
      "ECDH-ES+A192KW",
      "RSA-OAEP",
      "ECDH-ES+A128KW",
      "RSA-OAEP-256",
      "A128KW",
      "A256KW",
      "ECDH-ES",
      "dir",
      "A192KW"
    ],
    "supportedRcsResponseEncryptionMethods": [
      "A256GCM",
      "A192GCM",
      "A128GCM",
      "A128CBC-HS256",
      "A192CBC-HS384"
    ]
  }
}

```

```

    "A256CBC-HS512"
  ],
  "supportedRcsRequestEncryptionMethods": [
    "A256GCM",
    "A192GCM",
    "A128GCM",
    "A128CBC-HS256",
    "A192CBC-HS384",
    "A256CBC-HS512"
  ],
  "remoteConsentServiceId": "[Empty]",
  "supportedRcsResponseEncryptionAlgorithms": [
    "ECDH-ES+A256KW",
    "ECDH-ES+A192KW",
    "ECDH-ES+A128KW",
    "RSA-OAEP",
    "RSA-OAEP-256",
    "A128KW",
    "A256KW",
    "ECDH-ES",
    "dir",
    "A192KW"
  ]
},
"deviceCodeConfig": {
  "devicePollInterval": 5,
  "deviceCodeLifetime": 300
},
"_id": "",
"_type": {
  "_id": "oauth-oidc",
  "name": "OAuth2 Provider",
  "collection": false
}
}
}

```

3. Make a PUT request to the `oauth-oidc` endpoint to submit the JSON payload returned in the previous step, replacing the value of the `authorisedIdmDelegationClients` property under `advancedOIDCConfig` with `idm-provisioning`:

```
"authorisedIdmDelegationClients": [ "idm-provisioning"],
```

+ *Example Curl Request*

```

curl \
--request PUT \
-H 'User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:70.0) Gecko/20100101 Firefox/70.0' \
-H 'Accept: application/json, text/javascript, */*; q=0.01' \
-H 'Accept-Language: en-US' \
--compressed \
-H 'Content-Type: application/json' \
-H 'Accept-API-Version: protocol=1.0,resource=1.0' \
-H 'X-Requested-With: XMLHttpRequest' \
-H 'Cache-Control: no-cache' \

```

```

-H 'Origin: http://am.example.com:8080' \
-H 'DNT: 1' \
-H 'Connection: keep-alive' \
-H 'Referer: http://am.example.com:8080/am/ui-admin/' \
-H 'Cookie: amlbcookie=01; iPlanetDirectoryPro=tokenId' \
--data '{
  "core0Auth2Config": {
    "refreshTokenLifetime": 604800,
    "accessTokenLifetime": 3600,
    "usePolicyEngineForScope": false,
    "codeLifetime": 120,
    "issueRefreshTokenOnRefreshedToken": true,
    "macaroonTokensEnabled": false,
    "issueRefreshToken": true,
    "accessTokenModificationScript": "d22f9a0c-426a-4466-b95e-d0f125b0d5fa",
    "statelessTokensEnabled": false
  },
  "core0IDCConfig": {
    "supportedIDTokenEncryptionMethods": [
      "A256GCM",
      "A192GCM",
      "A128GCM",
      "A128CBC-HS256",
      "A192CBC-HS384",
      "A256CBC-HS512"
    ],
    "jwtTokenLifetime": 3600,
    "supportedClaims": [],
    "supportedIDTokenEncryptionAlgorithms": [
      "ECDH-ES+A256KW",
      "ECDH-ES+A192KW",
      "RSA-OAEP",
      "ECDH-ES+A128KW",
      "RSA-OAEP-256",
      "A128KW",
      "A256KW",
      "ECDH-ES",
      "dir",
      "A192KW"
    ],
    "supportedIDTokenSigningAlgorithms": [
      "PS384",
      "RS384",
      "ES384",
      "HS256",
      "HS512",
      "ES256",
      "RS256",
      "HS384",
      "ES512",
      "PS256",
      "PS512",
      "RS512"
    ],
    "oidcClaimsScript": "36863ffb-40ec-48b9-94b1-9a99f71cc3b5"
  },
  "advanced0Auth2Config": {
    "supportedScopes": [
      "openid",

```



```

        "fr:idm:*",
        "am-introspect-all-tokens"
    ],
    "tlsCertificateRevocationCheckingEnabled": false,
    "codeVerifierEnforced": "false",
    "tokenSigningAlgorithm": "HS256",
    "authenticationAttributes": [
        "uid"
    ],
    "passwordGrantAuthService": "[Empty]",
    "defaultScopes": [],
    "tlsClientCertificateHeaderFormat": "URLENCODED_PEM",
    "scopeImplementationClass": "org.forgerock.openam.oauth2.OpenAMScopeValidator",
    "responseTypeClasses": [
        "code|org.forgerock.oauth2.core.AuthorizationCodeResponseTypeHandler",
        "id_token|org.forgerock.openidconnect.IdTokenResponseTypeHandler",
        "device_code|org.forgerock.oauth2.core.TokenResponseTypeHandler",
        "token|org.forgerock.oauth2.core.TokenResponseTypeHandler"
    ],
    "tlsCertificateBoundAccessTokensEnabled": true,
    "hashSalt": "changeme",
    "moduleMessageEnabledInPasswordGrant": false,
    "tokenEncryptionEnabled": false,
    "tokenCompressionEnabled": false,
    "grantTypes": [
        "implicit",
        "urn:iETF:params:oauth:grant-type:saml2-bearer",
        "refresh_token",
        "password",
        "client_credentials",
        "urn:iETF:params:oauth:grant-type:device_code",
        "authorization_code",
        "urn:openid:params:grant-type:ciba",
        "urn:iETF:params:oauth:grant-type:uma-ticket",
        "urn:iETF:params:oauth:grant-type:jwt-bearer"
    ],
    "displayNameAttribute": "cn",
    "macaroonTokenFormat": "V2",
    "supportedSubjectTypes": [
        "public"
    ]
},
"advancedOIDCConfig": {
    "storeOpsTokens": true,
    "authorisedIdmDelegationClients": ["idm-provisioning"],
    "defaultACR": [],
    "supportedRequestParameterEncryptionEnc": [
        "A256GCM",
        "A192GCM",
        "A128GCM",
        "A128CBC-HS256",
        "A192CBC-HS384",
        "A256CBC-HS512"
    ],
    "claimsParameterSupported": false,
    "amrMappings": {},
    "supportedUserInfoEncryptionEnc": [
        "A256GCM",
        "A192GCM",
    ]
}

```

```

        "A128GCM",
        "A128CBC-HS256",
        "A192CBC-HS384",
        "A256CBC-HS512"
    ],
    "alwaysAddClaimsToToken": false,
    "supportedUserInfoSigningAlgorithms": [
        "ES384",
        "HS256",
        "HS512",
        "ES256",
        "RS256",
        "HS384",
        "ES512"
    ],
    "supportedRequestParameterEncryptionAlgorithms": [
        "ECDH-ES+A256KW",
        "ECDH-ES+A192KW",
        "ECDH-ES+A128KW",
        "RSA-OAEP",
        "RSA-OAEP-256",
        "A128KW",
        "A256KW",
        "ECDH-ES",
        "dir",
        "A192KW"
    ],
    "supportedTokenIntrospectionResponseEncryptionEnc": [
        "A256GCM",
        "A192GCM",
        "A128GCM",
        "A128CBC-HS256",
        "A192CBC-HS384",
        "A256CBC-HS512"
    ],
    "supportedTokenIntrospectionResponseSigningAlgorithms": [
        "PS384",
        "RS384",
        "EdDSA",
        "ES384",
        "HS256",
        "HS512",
        "ES256",
        "RS256",
        "HS384",
        "ES512",
        "PS256",
        "PS512",
        "RS512"
    ],
    "authorisedOpenIdConnectSSOClients": [],
    "idTokenInfoClientAuthenticationEnabled": true,
    "supportedRequestParameterSigningAlgorithms": [
        "PS384",
        "RS384",
        "ES384",
        "HS256",
        "HS512",
        "ES256",
    ]

```

```

        "RS256",
        "HS384",
        "ES512",
        "PS256",
        "PS512",
        "RS512"
    ],
    "supportedUserInfoEncryptionAlgorithms": [
        "ECDH-ES+A256KW",
        "ECDH-ES+A192KW",
        "RSA-OAEP",
        "ECDH-ES+A128KW",
        "RSA-OAEP-256",
        "A128KW",
        "A256KW",
        "ECDH-ES",
        "dir",
        "A192KW"
    ],
    "supportedTokenIntrospectionResponseEncryptionAlgorithms": [
        "ECDH-ES+A256KW",
        "ECDH-ES+A192KW",
        "RSA-OAEP",
        "ECDH-ES+A128KW",
        "RSA-OAEP-256",
        "A128KW",
        "A256KW",
        "ECDH-ES",
        "dir",
        "A192KW"
    ],
    "supportedTokenEndpointAuthenticationSigningAlgorithms": [
        "PS384",
        "RS384",
        "ES384",
        "HS256",
        "HS512",
        "ES256",
        "RS256",
        "HS384",
        "ES512",
        "PS256",
        "PS512",
        "RS512"
    ],
    "loaMapping": {}
},
"clientDynamicRegistrationConfig": {
    "dynamicClientRegistrationSoftwareStatementRequired": false,
    "dynamicClientRegistrationScope": "dynamic_client_registration",
    "requiredSoftwareStatementAttestedAttributes": [
        "redirect_uris"
    ],
    "generateRegistrationAccessTokens": true,
    "allowDynamicRegistration": false
},
"cibaConfig": {
    "supportedCibaSigningAlgorithms": [
        "ES256",

```

```

        "PS256"
    ],
    "cibaAuthReqIdLifetime": 600,
    "cibaMinimumPollingInterval": 2
},
"consent": {
    "enableRemoteConsent": false,
    "supportedRcsRequestSigningAlgorithms": [
        "PS384",
        "RS384",
        "ES384",
        "HS256",
        "HS512",
        "ES256",
        "RS256",
        "HS384",
        "ES512",
        "PS256",
        "PS512",
        "RS512"
    ],
    "supportedRcsResponseSigningAlgorithms": [
        "PS384",
        "RS384",
        "ES384",
        "HS256",
        "HS512",
        "ES256",
        "RS256",
        "HS384",
        "ES512",
        "PS256",
        "PS512",
        "RS512"
    ],
    "clientsCanSkipConsent": true,
    "supportedRcsRequestEncryptionAlgorithms": [
        "ECDH-ES+A256KW",
        "ECDH-ES+A192KW",
        "RSA-OAEP",
        "ECDH-ES+A128KW",
        "RSA-OAEP-256",
        "A128KW",
        "A256KW",
        "ECDH-ES",
        "dir",
        "A192KW"
    ],
    "supportedRcsResponseEncryptionMethods": [
        "A256GCM",
        "A192GCM",
        "A128GCM",
        "A128CBC-HS256",
        "A192CBC-HS384",
        "A256CBC-HS512"
    ],
    "supportedRcsRequestEncryptionMethods": [
        "A256GCM",
        "A192GCM",

```

```

        "A128GCM",
        "A128CBC-HS256",
        "A192CBC-HS384",
        "A256CBC-HS512"
    ],
    "remoteConsentServiceId": "[Empty]",
    "supportedRcsResponseEncryptionAlgorithms": [
        "ECDH-ES+A256KW",
        "ECDH-ES+A192KW",
        "ECDH-ES+A128KW",
        "RSA-OAEP",
        "RSA-OAEP-256",
        "A128KW",
        "A256KW",
        "ECDH-ES",
        "dir",
        "A192KW"
    ]
},
"deviceCodeConfig": {
    "devicePollInterval": 5,
    "deviceCodeLifetime": 300
},
"_id": "",
"_type": {
    "_id": "oauth-oidc",
    "name": "OAuth2 Provider",
    "collection": false
}
} \
"http://am.example.com:8080/am/json/realms/root/realms-config/services/oauth-oidc"
{
    "coreOAuth2Config": {
        "refreshTokenLifetime": 604800,
        "accessTokenLifetime": 3600,
        "usePolicyEngineForScope": false,
        "codeLifetime": 120,
        "issueRefreshTokenOnRefreshedToken": true,
        "macaroonTokensEnabled": false,
        "issueRefreshToken": true,
        "accessTokenModificationScript": "d22f9a0c-426a-4466-b95e-d0f125b0d5fa",
        "statelessTokensEnabled": false
    },
    "coreIDCCConfig": {
        "supportedIDTokenEncryptionMethods": [
            "A256GCM",
            "A192GCM",
            "A128GCM",
            "A128CBC-HS256",
            "A192CBC-HS384",
            "A256CBC-HS512"
        ],
        "jwtTokenLifetime": 3600,
        "supportedClaims": [],
        "supportedIDTokenEncryptionAlgorithms": [
            "ECDH-ES+A256KW",
            "ECDH-ES+A192KW",
            "RSA-OAEP",
            "ECDH-ES+A128KW",

```

```

    "RSA-OAEP-256",
    "A128KW",
    "A256KW",
    "ECDH-ES",
    "dir",
    "A192KW"
  ],
  "supportedIDTokenSigningAlgorithms": [
    "PS384",
    "RS384",
    "ES384",
    "HS256",
    "HS512",
    "ES256",
    "RS256",
    "HS384",
    "ES512",
    "PS256",
    "PS512",
    "RS512"
  ],
  "oidcClaimsScript": "36863ffb-40ec-48b9-94b1-9a99f71cc3b5"
},
"advancedOAuth2Config": {
  "supportedScopes": [
    "openid",
    "fr:idm:*",
    "am-introspect-all-tokens"
  ],
  "tlsCertificateRevocationCheckingEnabled": false,
  "codeVerifierEnforced": "false",
  "tokenSigningAlgorithm": "HS256",
  "authenticationAttributes": [
    "uid"
  ],
  "passwordGrantAuthService": "[Empty]",
  "defaultScopes": [],
  "tlsClientCertificateHeaderFormat": "URLENCODED_PEM",
  "scopeImplementationClass": "org.forgerock.openam.oauth2.OpenAMScopeValidator",
  "responseTypeClasses": [
    "code|org.forgerock.oauth2.core.AuthorizationCodeResponseTypeHandler",
    "id_token|org.forgerock.openidconnect.IdTokenResponseTypeHandler",
    "device_code|org.forgerock.oauth2.core.TokenResponseTypeHandler",
    "token|org.forgerock.oauth2.core.TokenResponseTypeHandler"
  ],
  "tlsCertificateBoundAccessTokensEnabled": true,
  "hashSalt": "changeme",
  "moduleMessageEnabledInPasswordGrant": false,
  "tokenEncryptionEnabled": false,
  "tokenCompressionEnabled": false,
  "grantTypes": [
    "implicit",
    "urn:ietf:params:oauth:grant-type:saml2-bearer",
    "refresh_token",
    "password",
    "client_credentials",
    "urn:ietf:params:oauth:grant-type:device_code",
    "authorization_code",
    "urn:openid:params:grant-type:ciba",

```

```

    "urn:ietf:params:oauth:grant-type:uma-ticket",
    "urn:ietf:params:oauth:grant-type:jwt-bearer"
  ],
  "displayNameAttribute": "cn",
  "macaroonTokenFormat": "V2",
  "supportedSubjectTypes": [
    "public"
  ]
},
"advancedOIDCConfig": {
  "storeOpsTokens": true,
  "defaultACR": [],
  "supportedRequestParameterEncryptionEnc": [
    "A256GCM",
    "A192GCM",
    "A128GCM",
    "A128CBC-HS256",
    "A192CBC-HS384",
    "A256CBC-HS512"
  ],
  "claimsParameterSupported": false,
  "amrMappings": {},
  "supportedUserInfoEncryptionEnc": [
    "A256GCM",
    "A192GCM",
    "A128GCM",
    "A128CBC-HS256",
    "A192CBC-HS384",
    "A256CBC-HS512"
  ],
  "authorisedIdmDelegationClients": ["idm-provisioning"],
  "alwaysAddClaimsToToken": false,
  "supportedUserInfoSigningAlgorithms": [
    "ES384",
    "HS256",
    "HS512",
    "ES256",
    "RS256",
    "HS384",
    "ES512"
  ],
  "supportedRequestParameterEncryptionAlgorithms": [
    "ECDH-ES+A256KW",
    "ECDH-ES+A192KW",
    "ECDH-ES+A128KW",
    "RSA-OAEP",
    "RSA-OAEP-256",
    "A128KW",
    "A256KW",
    "ECDH-ES",
    "dir",
    "A192KW"
  ],
  "supportedTokenIntrospectionResponseEncryptionEnc": [
    "A256GCM",
    "A192GCM",
    "A128GCM",
    "A128CBC-HS256",
    "A192CBC-HS384",

```

```

    "A256CBC-HS512"
  ],
  "supportedTokenIntrospectionResponseSigningAlgorithms": [
    "PS384",
    "RS384",
    "EdDSA",
    "ES384",
    "HS256",
    "HS512",
    "ES256",
    "RS256",
    "HS384",
    "ES512",
    "PS256",
    "PS512",
    "RS512"
  ],
  "authorisedOpenIdConnectSSOClients": [],
  "idTokenInfoClientAuthenticationEnabled": true,
  "supportedRequestParameterSigningAlgorithms": [
    "PS384",
    "RS384",
    "ES384",
    "HS256",
    "HS512",
    "ES256",
    "RS256",
    "HS384",
    "ES512",
    "PS256",
    "PS512",
    "RS512"
  ],
  "supportedUserInfoEncryptionAlgorithms": [
    "ECDH-ES+A256KW",
    "ECDH-ES+A192KW",
    "RSA-OAEP",
    "ECDH-ES+A128KW",
    "RSA-OAEP-256",
    "A128KW",
    "A256KW",
    "ECDH-ES",
    "dir",
    "A192KW"
  ],
  "supportedTokenIntrospectionResponseEncryptionAlgorithms": [
    "ECDH-ES+A256KW",
    "ECDH-ES+A192KW",
    "RSA-OAEP",
    "ECDH-ES+A128KW",
    "RSA-OAEP-256",
    "A128KW",
    "A256KW",
    "ECDH-ES",
    "dir",
    "A192KW"
  ],
  "supportedTokenEndpointAuthenticationSigningAlgorithms": [
    "PS384",

```



```

    "RS384",
    "ES384",
    "HS256",
    "HS512",
    "ES256",
    "RS256",
    "HS384",
    "ES512",
    "PS256",
    "PS512",
    "RS512"
  ],
  "loaMapping": {}
},
"clientDynamicRegistrationConfig": {
  "dynamicClientRegistrationSoftwareStatementRequired": false,
  "dynamicClientRegistrationScope": "dynamic_client_registration",
  "requiredSoftwareStatementAttestedAttributes": [
    "redirect_uris"
  ],
  "generateRegistrationAccessTokens": true,
  "allowDynamicRegistration": false
},
"cibaConfig": {
  "supportedCibaSigningAlgorithms": [
    "ES256",
    "PS256"
  ],
  "cibaAuthReqIdLifetime": 600,
  "cibaMinimumPollingInterval": 2
},
"consent": {
  "enableRemoteConsent": false,
  "supportedRcsRequestSigningAlgorithms": [
    "PS384",
    "RS384",
    "ES384",
    "HS256",
    "HS512",
    "ES256",
    "RS256",
    "HS384",
    "ES512",
    "PS256",
    "PS512",
    "RS512"
  ],
  "supportedRcsResponseSigningAlgorithms": [
    "PS384",
    "RS384",
    "ES384",
    "HS256",
    "HS512",
    "ES256",
    "RS256",
    "HS384",
    "ES512",
    "PS256",
    "PS512",

```

```

    "RS512"
  ],
  "clientsCanSkipConsent": true,
  "supportedRcsRequestEncryptionAlgorithms": [
    "ECDH-ES+A256KW",
    "ECDH-ES+A192KW",
    "RSA-OAEP",
    "ECDH-ES+A128KW",
    "RSA-OAEP-256",
    "A128KW",
    "A256KW",
    "ECDH-ES",
    "dir",
    "A192KW"
  ],
  "supportedRcsResponseEncryptionMethods": [
    "A256GCM",
    "A192GCM",
    "A128GCM",
    "A128CBC-HS256",
    "A192CBC-HS384",
    "A256CBC-HS512"
  ],
  "supportedRcsRequestEncryptionMethods": [
    "A256GCM",
    "A192GCM",
    "A128GCM",
    "A128CBC-HS256",
    "A192CBC-HS384",
    "A256CBC-HS512"
  ],
  "remoteConsentServiceId": "[Empty]",
  "supportedRcsResponseEncryptionAlgorithms": [
    "ECDH-ES+A256KW",
    "ECDH-ES+A192KW",
    "ECDH-ES+A128KW",
    "RSA-OAEP",
    "RSA-OAEP-256",
    "A128KW",
    "A256KW",
    "ECDH-ES",
    "dir",
    "A192KW"
  ]
},
"deviceCodeConfig": {
  "devicePollInterval": 5,
  "deviceCodeLifetime": 300
},
"_id": "",
"_type": {
  "_id": "oauth-oidc",
  "name": "OAuth2 Provider",
  "collection": false
}
}

```

```
}
```

Enable CORS Support

Cross-origin resource sharing (CORS) lets user agents make requests across domains.

1. If you're not currently logged in to the AM console as the `amAdmin` user, log in.
2. From the top menu, select Configure > Global Services > CORS Service.
3. On the Secondary Configurations tab, click Add a Secondary Configuration.
4. On the New Configuration screen, enter the following values:

- Name: `Cors Configuration`

- Accepted Origins:

```
http://localhost:8083  
http://localhost:8082  
http://localhost:8888  
http://openidm.example.com:8080  
https://openidm.example.com:8443
```

- Accepted Methods:

```
HEAD  
DELETE  
POST  
GET  
PUT  
PATCH
```

- Accepted Headers:

```
authorization  
x-openidm-username  
if-none-match  
x-openidm-nosession  
x-openidm-password  
accept-api-version  
x-requested-with  
content-type  
if-match  
cache-control  
user-agent
```

- Exposed Headers: `WWW-Authenticate`

5. Click Create.
6. On the Cors Configuration screen, set the following values:
 - Enable the CORS filter: Enable
 - Max Age: 600
 - Allow Credentials: Enable
7. Click Save Changes.

Set up IDM

This procedure sets up IDM with an external MySQL repository. The procedure assumes that IDM is installed according to the listed "Server Settings".

1. Follow the instructions in the *IDM Installation Guide* to install and run IDM.
2. Edit the `/path/to/openidm/resolver/boot.properties` file to set the hostname:

```
openidm.host=openidm.example.com
```
3. Configure your IDM repository. This procedure was tested with a MySQL repository. Follow the instructions in the *IDM Installation Guide* to set up a MySQL repository.
4. (Optional) Configure social authentication.

In your project's `conf/managed.json` file, add an `aliasList` property to the `user` object:

```

{
  "objects": [
    {
      "name": "user",
      ...
      "schema": {
        "properties": {
          ...
          "aliasList": {
            "title": "User Alias Names List",
            "description": "List of identity aliases used primarily to record social IdP subjects for
this user",
            "type": "array",
            "items": {
              "type": "string",
              "title": "User Alias Names Items"
            },
            "viewable": false,
            "searchable": false,
            "userEditable": true,
            "returnByDefault": false,
            "isVirtual": false
          }
        }
      }
    }
    ...
  ]
}

```

5. Change the authentication mechanism to `rsFilter` only:

- Replace the default `conf/authentication.json` file with this file.
- Check that the `clientSecret` matches the `Client secret` that you set for the `idm-resource-server` client in AM (see "Configure OAuth Clients").
- Check that the `scopes` match the `Scope(s)` that you set for the `idm-admin-ui` and `end-user-ui` clients in AM (see "Configure OAuth Clients").

For more information about the `rsFilter` authentication module, see the [IDM Security Guide](#).

6. Edit the IDM Admin UI configuration so that you can still authenticate through the IDM Admin UI:

- a. In your `conf/ui-configuration.json` file, insert a `platformSettings` object into the `configuration` object:

```
{
  "configuration" : {
    "platformSettings" : {
      "adminOauthClient" : "idm-admin-ui",
      "adminOauthClientScopes" : "openid fr:idm:*",
      "amUrl" : "http://am.example.com:8080/am",
      "loginUrl" : ""
    }
  }
}
```

- b. In your `conf/ui.context-admin.json` file, check that `X-Frame-Options` is set to `SAMEORIGIN`:

+ *Sample `ui.context-admin.json`*

```
{
  "enabled" : true,
  "urlContextRoot" : "/admin",
  "defaultDir" : "${idm.install.dir}/ui/admin/default",
  "extensionDir" : "${idm.install.dir}/ui/admin/extension",
  "responseHeaders" : {
    "X-Frame-Options" : "SAMEORIGIN"
  }
}
```

You should now be able to access the IDM Admin UI at `http://openidm.example.com:8080/admin`. When you log in to the Admin UI, use the default AM administrative user (`amAdmin`), and not `openidm-admin`.

7. Configure the CORS servlet filter.

Replace the default `conf/servletfilter-cors.json` file with this file.

8. Configure synchronization between the IDM repository and the AM identity store.

- a. Add a configuration for the LDAP connector.

Create a configuration file named `provisioner.openicf-ldap.json` in the `/path/to/openidm/conf` directory. Use this file as a template.

Pay particular attention to the connection properties, `host`, `port`, `principal`, and `credentials`. These must match the configuration of the DS server that you set up as the identity store.

- b. Add a mapping between IDM managed user objects, and AM identities stored in DS.

Create a mapping file named `sync.json` in the `/path/to/openidm/conf` directory. Use the following sample configuration file as a template:

Create a mapping file named `sync.json` in the `/path/to/openidm/conf` directory. Use this file as a template.

- Secure the connection to the DS server. This step assumes that you have set up the identity store, and exported the DS CA certificate from `identities.example.com` CA certificate from `identities.example.com` (as shown in Step 4 of "Secure Connections").

Import the DS CA certificate into the IDM truststore:

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

- (Optional) If you want to use the `PlatformForgottenUsername` or `PlatformResetPassword` trees, configure outbound email.

Note

After you have installed the Platform UI, you can configure email through the UI at <http://openidm.example.com:8080/admin>.

IDM is now configured. Move on to setting up the Platform UI.

Chapter 3

Deployment Two - Shared Identity Store

This deployment assumes that you are using the following data stores:

- An external DS instance as the AM configuration store.
- A separate external DS instance that is shared between AM and IDM as the identity store.

Note

The IDM End User UI is not supported in a platform deployment, as it does not support authentication through AM. You can use the Platform UIs with this deployment, or create your own UI that supports authentication through AM.

- "Set up Your Data Stores"
- "Set Up a Container"
- "Secure Connections"
- "Configure AM"
- "Set up IDM"

Set up Your Data Stores

Configuration Store

1. Set up a DS server as an external configuration data store, using the `am-config` setup profile.

For more information about this step, see setup profiles in the *DS Installation Guide*.

This command sets up the config store with the parameters listed in "Server Settings":


```
/path/to/openssl/setup \  
--deploymentKeyPassword password \  
--rootUserDN uid=admin \  
--rootUserPassword strongAdminPa55word \  
--monitorUserPassword strongMonitorPa55word \  
--hostname config.example.com \  
--adminConnectorPort 4444 \  
--ldapPort 1389 \  
--enableStartTls \  
--ldapsPort 1636 \  
--httpsPort 8443 \  
--replicationPort 8989 \  
--profile am-config \  
--set am-config/amConfigAdminPassword:5up35tr0ng \  
--acceptLicense
```

Make a note of the generated deployment key. You will need it to export the server certificate later in this procedure. You can set the deployment key as a variable in this terminal window. For example:

```
export DEPLOYMENT_KEY=deployment-key
```

2. Start the DS server:

```
/path/to/openssl/bin/start-ds
```

Identity Store

- Set up a DS server as a shared identity store, using the `am-identity-store` and `idm-repo` setup profiles.

For more information about this step, see setup profiles in the *DS Installation Guide*.

This command sets up the identity store with the parameters listed in "Server Settings":

```
/path/to/openssl/setup \  
--deploymentKeyPassword password \  
--rootUserDN uid=admin \  
--rootUserPassword strongAdminPa55word \  
--monitorUserPassword strongMonitorPa55word \  
--hostname identities.example.com \  
--adminConnectorPort 4444 \  
--ldapPort 1389 \  
--enableStartTls \  
--ldapsPort 1636 \  
--replicationPort 8989 \  
--profile am-identity-store \  
--set am-identity-store/amIdentityStoreAdminPassword:5up35tr0ng \  
--profile idm-repo \  
--set idm-repo/domain:forgerock.io \  
--acceptLicense
```

Make a note of the generated deployment key. You will need it to export the server certificate later in this procedure. You can set the deployment key as a variable in this terminal window. For example:

```
export DEPLOYMENT_KEY=deployment-key
```

Start the DS server:

```
/path/to/openssh/bin/start-ds
```

Set Up a Container

- Install a Java container to deploy AM.

This guide assumes that you are using Apache Tomcat.

For instructions on setting up Tomcat, see [Preparing Apache Tomcat in the AM Installation Guide](#).

Secure Connections

Important

From DS 7 onwards, you *must* secure connections to DS servers.

1. Configure your AM container for SSL connections.
2. Create a new directory that will house a dedicated truststore for AM:

```
mkdir -p /path/to/openam-security/
```

3. Make a copy of your JDK's default truststore; for example, `$JAVA_HOME/lib/security/cacerts`, name it `truststore`, and place it in the directory you created:

```
cp $JAVA_HOME/lib/security/cacerts /path/to/openam-security/truststore
```

Note

The default password of the `lib/security/cacerts` truststore is `changeit`. You should change this password in a production environment.

4. *On each DS server*, export the DS server certificate.

You must run these commands in the same terminal window where you set the `DEPLOYMENT_KEY` variable.

On `config.example.com`:

```
/path/to/openssl/bin/dskeymgr export-ca-cert \  
--deploymentKey $DEPLOYMENT_KEY \  
--deploymentKeyPassword password \  
--alias config-ca-cert \  
--outputFile config-ca-cert.pem
```

On `identities.example.com`:

```
/path/to/openssl/bin/dskeymgr export-ca-cert \  
--deploymentKey $DEPLOYMENT_KEY \  
--deploymentKeyPassword password \  
--alias identities-ca-cert \  
--outputFile identities-ca-cert.pem
```

5. Import each DS server certificate into the new AM truststore. If you are not testing this example on a single host, you might need to copy each certificate file onto the AM host machine first:

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

6. Configure the truststore in Apache Tomcat so that AM can access it.

Append the truststore settings to the `CATALINA_OPTS` variable in the `$CATALINA_BASE/bin/setenv.sh` file.

For example:

```
CATALINA_OPTS="-Djavax.net.ssl.trustStore=/path/to/openam-security/truststore\  
-Djavax.net.ssl.trustStorePassword=changeit\  
-Djavax.net.ssl.trustStoreType=jks"
```

Refer to your specific container's documentation for information on configuring truststores.

Configure AM

When your external data stores are configured, follow these procedures to configure AM with the ForgeRock Identity Platform:

- "Install AM"
- "Adjust the Identity Store Configuration for IDM"
- "Configure OAuth Clients"
- "Configure Authentication Trees"
- "Map Authentication Trees"
- "Configure an OAuth 2.0 Provider Service"
- "Configure an IDM Provisioning Service"
- "Enable the Password Update Tree (optional)"
- "Enable CORS Support"

Install AM

1. Follow the instructions in the *AM Installation Guide* to download AM. Make sure you download the `.zip` file, not just the `.war` file.
2. Follow the instructions in the *AM Installation Guide* to prepare your environment, and prepare a web application container.

Use Apache Tomcat as the application container, listening on the default port (`8080`).

3. Copy the AM `.war` file to deploy in Apache Tomcat as `am.war`:

```
cp AM-7.0.0.war /path/to/tomcat/webapps/am.war
```

4. Start Tomcat if it is not already running.
5. Navigate to the deployed AM application; for example, `http://am.example.com:8080/am/`.
6. Select Create New Configuration to create a custom configuration.
7. Accept the license agreement, and click Continue.
8. Set a password for the default user, `amAdmin`.

This guide assumes that the `amAdmin` password is `Passw0rd`.

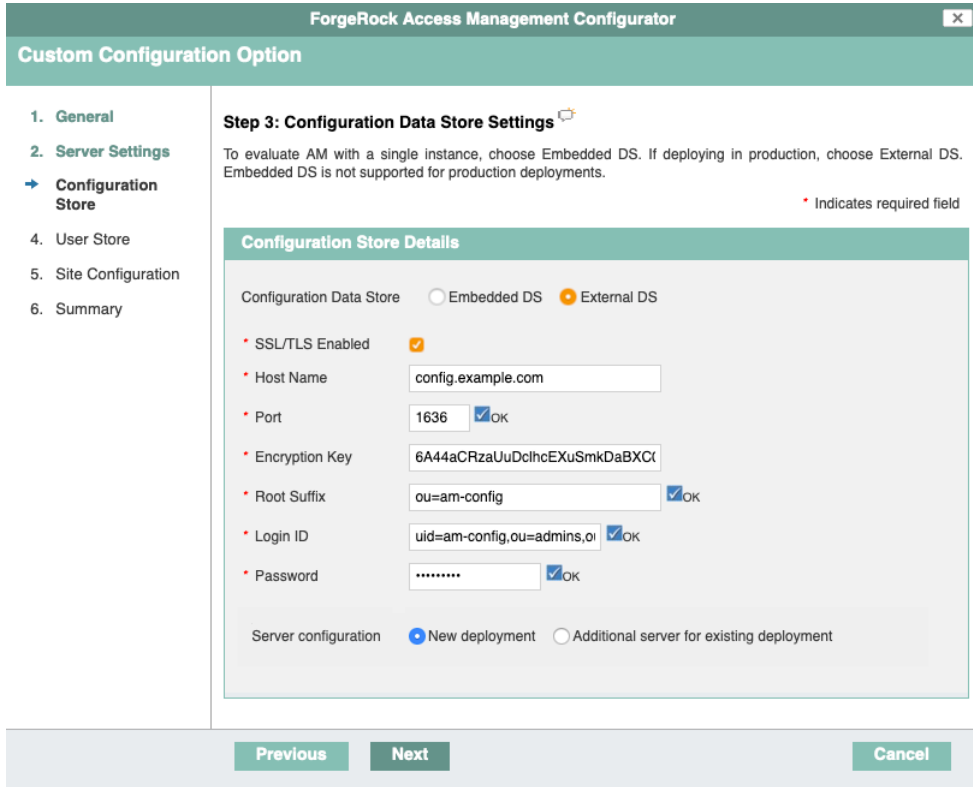
9. On the Server Settings screen, enter your AM server settings; for example:
 - Server URL: `http://am.example.com:8080`
 - Cookie Domain: `example.com`
 - Platform Locale: `en_US`
 - Configuration Directory: `/path/to/openam`
10. On the Configuration Data Store Settings screen, select External DS, and enter the details for the DS instance that you set up as a configuration store.

This list reflects the DS configuration store installed with the listed "Server Settings".

- SSL/TLS: `Enabled`

TLS is required in a production deployment.

- Host Name: `config.example.com`
- Port: `1636`
- Encryption Key: (generated encryption key)
- Root Suffix: `ou=am-config`
- Login ID: `uid=am-config,ou=admins,ou=am-config`
- Password: `5up35tr0ng`
- Server configuration: `New deployment`



11. On the User Data Store Settings screen, select External User Data Store, and enter the details for the DS instance that you set up as an identity store.

This list reflects the DS identity store installed with the listed "Server Settings".

- User Data Store Type: **ForgeRock Directory Services (DS)**
- SSL/TLS: **Enabled**

TLS is required in a production deployment.

- Host Name: **identities.example.com**
- Port: **1636**
- Root Suffix: **ou=identities**
- Login ID: **uid=am-identity-bind-account,ou=admins,ou=identities**

- Password: `5up35tr0ng`

ForgeRock Access Management Configurator

Custom Configuration Option

1. General
2. Server Settings
3. Configuration Store
➔ **User Store**
5. Site Configuration
6. Summary

Step 4: User Data Store Settings

You can store user data in the embedded configuration data store for evaluation purposes. For production deployments you will need to use an external user data store. Please note that the Policy Service and LDAP Authentication Module are configured to use the Directory Administrator DN and Password provided here.

Embedded User Data Store (DS)
 External User Data Store

*Indicates required field

User Store Details

*User Data Store Type
 ForgeRock Directory Services (DS) Oracle Directory Server Enterprise Edition
 AD with Domain Name Active Directory with Host and Port
 IBM Tivoli Directory Server Active Directory Application Mode
 ForgeRock DS For IAM

*SSL/TLS Enabled

*Directory Name

*Port OK

*Root Suffix OK

*Login ID OK

*Password OK

12. On the Site Configuration screen, select No, then click Next.

13. Review the Configurator Summary Details, then click Create Configuration.

Adjust the Identity Store Configuration for IDM

1. If you're not currently logged in to the AM console as the `amAdmin` user, log in.
2. In the Top Level Realm, select Identity Stores then click OpenDJ.
3. On the Server Settings tab, set LDAPv3 Plug-in Search Scope to `SCOPE_ONE`, then click Save Changes.
4. On the User Configuration tab, set LDAP Users Search Attribute to `fr-idm-uuid`, then click Save Changes.
5. On the Authentication Configuration tab, check that the Authentication Naming Attribute is set to `uid`, then click Save Changes.

Configure OAuth Clients

This procedure configures *four* OAuth 2.0 clients:

1. If you're not currently logged in to the AM console as the `amAdmin` user, log in.
2. Configure an `idm-provisioning` client to make calls to IDM:
 - a. In the Top Level Realm, select Applications > OAuth 2.0 > Clients, and click Add Client.
 - b. Enter the following details:
 - Client ID: `idm-provisioning`
 - Client secret: `openidm`
 - Scopes: `fr:idm:*`
 - c. Click Create.
 - d. On the Advanced tab:
 - Response Types: Check that `token` is present (it should be there by default).
 - Grant Types: Remove `Authorization Code` and add `Client Credentials`.
 - e. Click Save Changes.
3. Configure an `idm-resource-server` client to introspect the access token:
 - a. In the Top Level Realm, select Applications > OAuth 2.0 > Clients, and click Add Client.
 - b. Enter the following details:
 - Client ID: `idm-resource-server`
 - Client secret: `password`

The value of this field must match the `clientSecret` that you will set in the `rsFilter` module in the IDM authentication configuration (`/path/to/openidm/conf/authentication.json`) during your IDM setup.

 - Scopes: `am-introspect-all-tokens am-introspect-all-tokens-any-realm`
 - c. Click Create.
4. Configure an `idm-admin-ui` client that will be used by the Platform Admin UI:
 - a. In the Top Level Realm, select Applications > OAuth 2.0 > Clients, and click Add Client.
 - b. Enter the following details:

- Client ID: `idm-admin-ui`
- Client Secret: (no client secret is required)
- Redirection URIs: `http://openidm.example.com:8080/platform/appAuthHelperRedirect.html` `http://openidm.example.com:8080/platform/sessionCheck.html` `http://openidm.example.com:8080/admin/appAuthHelperRedirect.html` `http://openidm.example.com:8080/admin/sessionCheck.html` `http://localhost:8082/appAuthHelperRedirect.html` `http://localhost:8082/sessionCheck.html`
- Scopes: `openid fr:idm:*`

Note

At a minimum, the scopes that you set here must include the scopes that you will set in the `rsFilter` authentication configuration (`/path/to/openidm/conf/authentication.json`) during your IDM setup.

- Click Create.
 - On the Core tab:
 - Client type: Select `Public`.Click Save Changes.
 - On the Advanced tab:
 - Grant Types: Add `Implicit`.
 - Token Endpoint Authentication Method: Select `none`.
 - Implied consent: Enable.Click Save Changes.
- Configure an `end-user-ui` client that will be used by the Platform End User UI:
 - In the Top Level Realm, select Applications > OAuth 2.0 > Clients, and click Add Client.
 - Enter the following details:
 - Client ID: `end-user-ui`
 - Client Secret: (no client secret is required)
 - Redirection URIs: `http://openidm.example.com:8080/enduser/appAuthHelperRedirect.html` `http://openidm.example.com:8080/enduser/sessionCheck.html` `http://localhost:8888/appAuthHelperRedirect.html` `http://localhost:8888/sessionCheck.html`

- Scopes: `openid fr:idm:*`

Note

At a minimum, the scopes that you set here must include the scopes that you will set in the `rsFilter` authentication configuration (`/path/to/openidm/conf/authentication.json`) during your IDM setup.

c. Click Create.

d. On the Core tab:

- Client type: Select `Public`.

Click Save Changes.

e. On the Advanced tab:

- Grant Types: Add `Implicit`.
- Token Endpoint Authentication Method: Select `none`.
- Implied Consent: Enable.

Click Save Changes.

Configure Authentication Trees

The platform deployment relies on three authentication trees to enable authentication through AM. When you extract the AM `.zip` file, you will get a `sample-trees-7.0.0.zip` file that contains a number of sample authentication trees, in JSON files. Use the Amster command-line utility to import the platform authentication trees into your AM configuration:

1. Extract the `sample-trees-7.0.0.zip` file and list the sample trees in the `/path/to/openam-samples/root/AuthTree` directory:

```
ls /path/to/openam-samples/root/AuthTree
Agent.json PlatformForgottenUsername.json
Example.json PlatformLogin.json
Facebook-ProvisionIDMAccount.json PlatformProgressiveProfile.json
Google-AnonymousUser.json PlatformRegistration.json
Google-DynamicAccountCreation.json PlatformResetPassword.json
HmacOneTimePassword.json PlatformUpdatePassword.json
PersistentCookie.json RetryLimit.json
```

2. Download and install Amster.

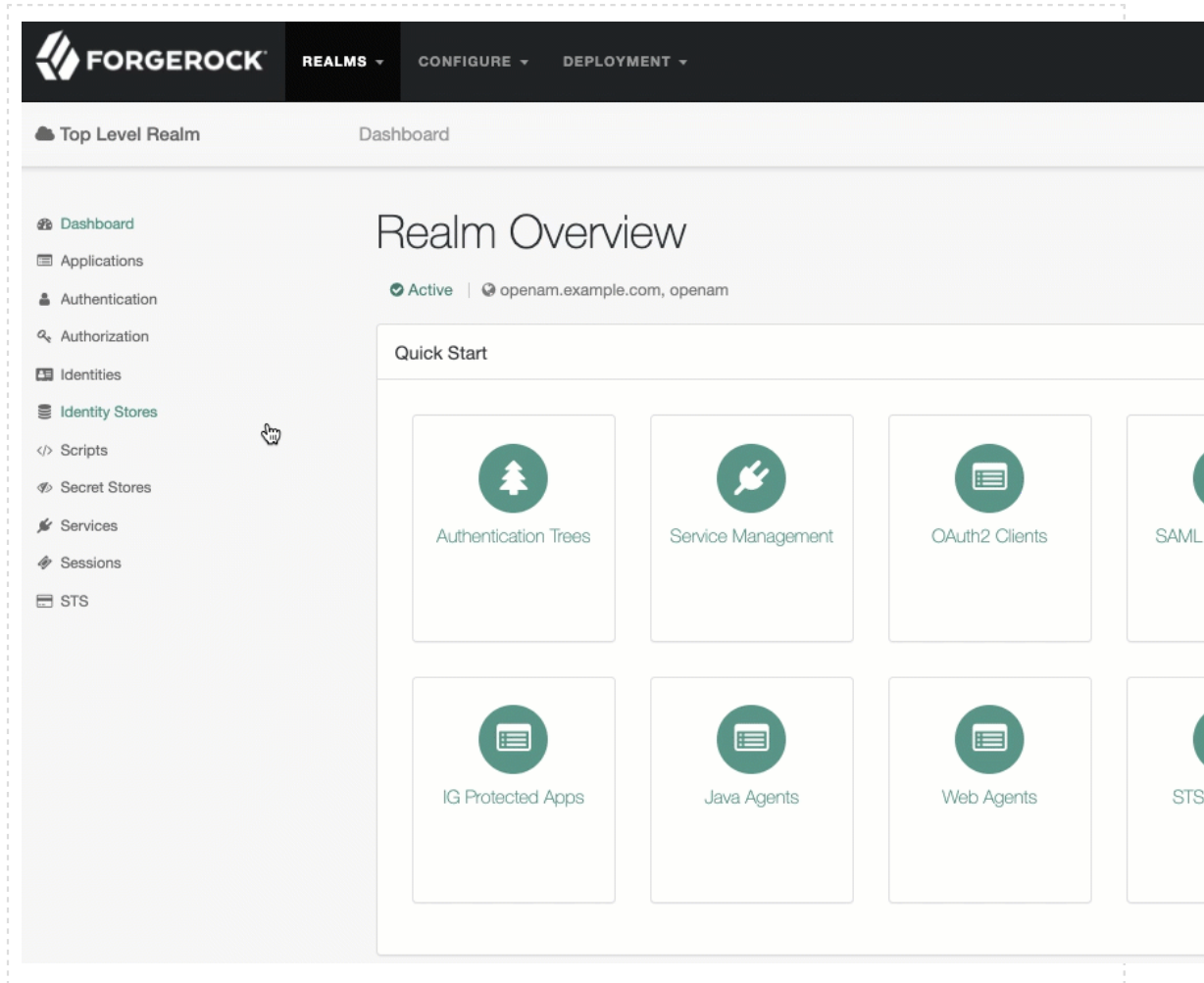
3. Start Amster, then connect to your AM instance:

```
./amster
Amster OpenAM Shell (7.0.0 build @build.number@, JVM: 11.0.4)
Type ':help' or ':h' for help.
-----
am> connect --interactive http://am.example.com:8080/am
Sign in
User Name: amAdmin
Password: *****
amster am.example.com:8080>
```

4. Import the sample authentication trees and nodes:

```
amster am.example.com:8080> import-config --path /path/to/openam-samples/root
Importing directory /path/to/openam-samples/root/AcceptTermsAndConditions
Imported /path/to/openam-samples/root/AcceptTermsAndConditions/b4a0e915-
c15d-4b83-9c9d-18347d645976.json
...
Import completed successfully
```

5. If you're not currently logged in to the AM console as the `amAdmin` user, log in.
6. Configure the `PlatformRegistration` tree:
- In the Top Level Realm, select Authentication > Trees, and click on PlatformRegistration.
 - On the `PlatformRegistration` tree, add a `Success URL` node between `Increment Login Count` and `Success`, and set its value to `http://localhost:8888`.
+ *Show Me*

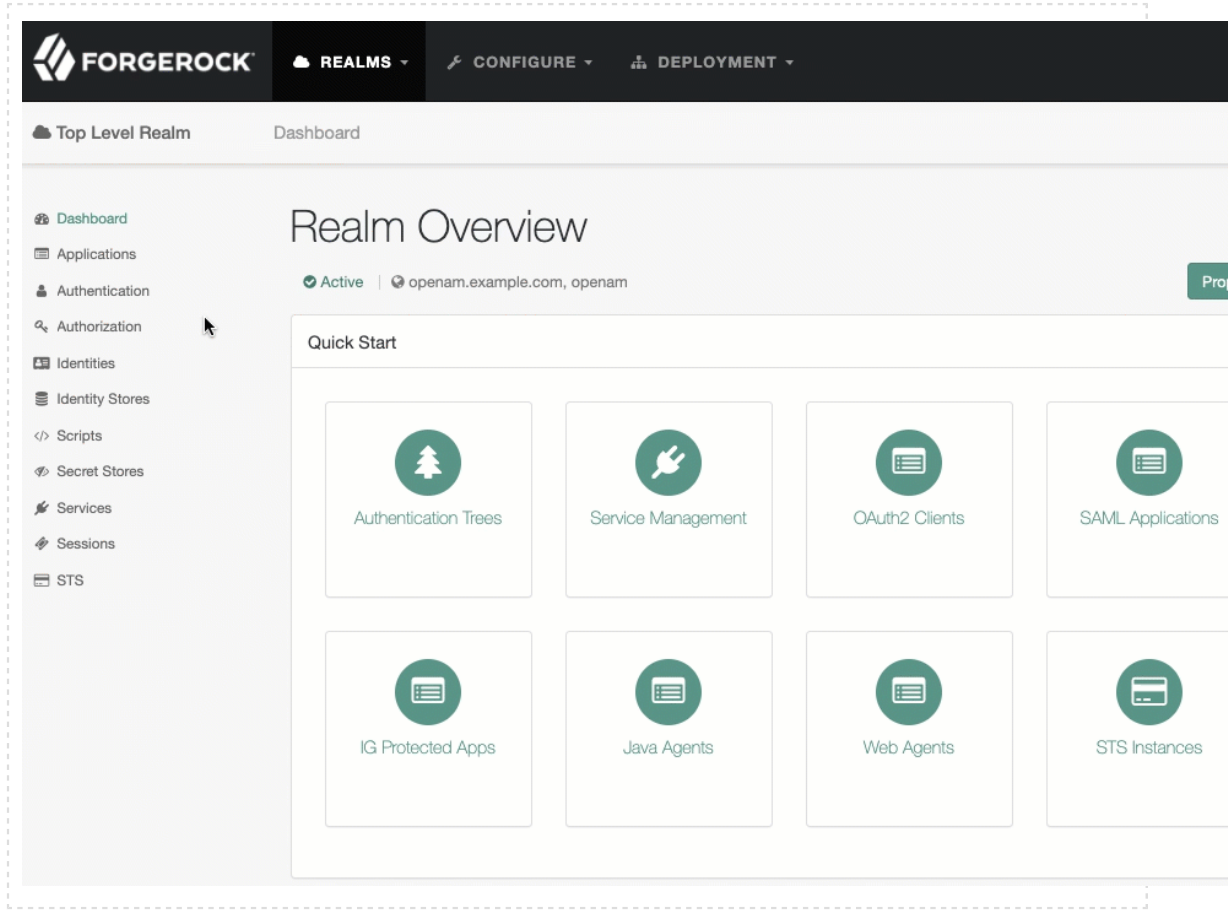


c. Click Save.

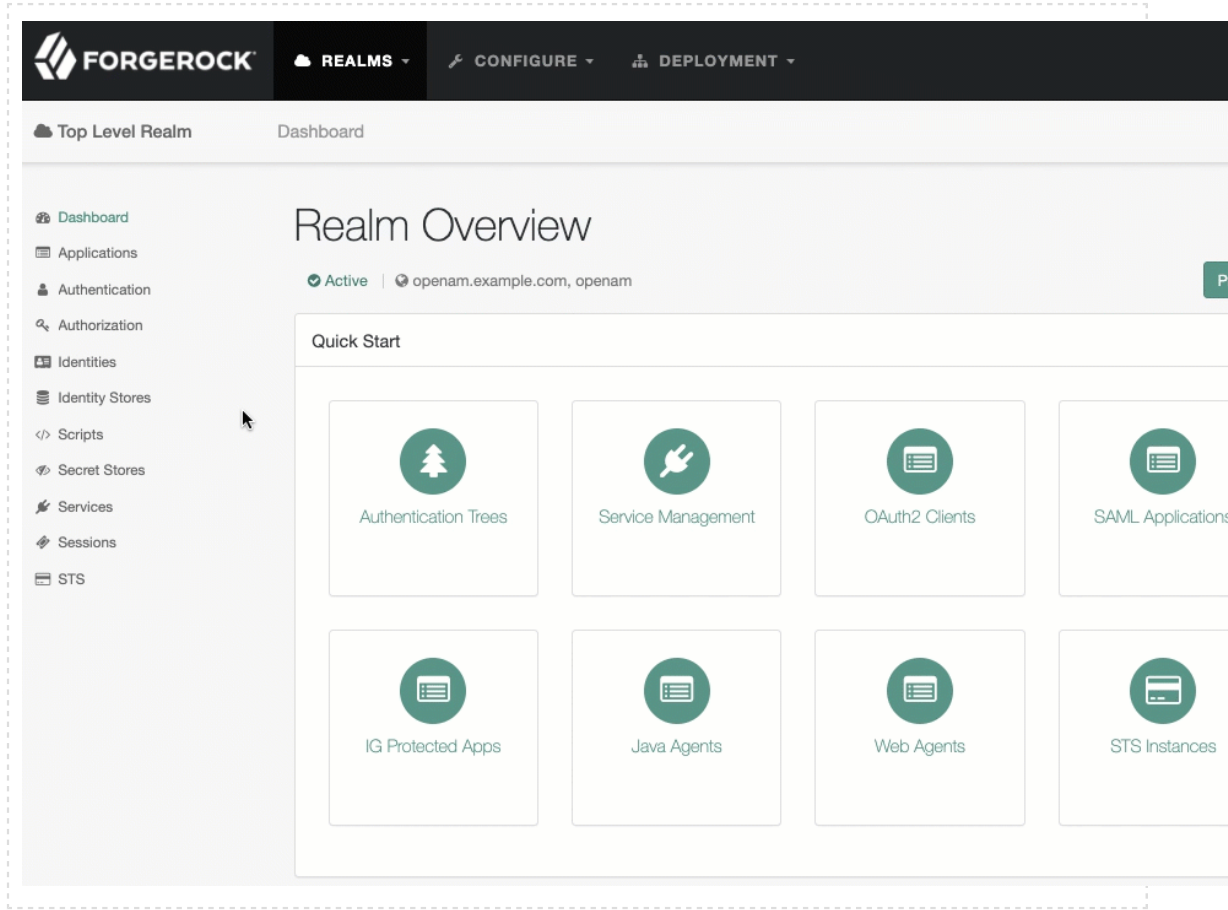
7. Configure the PlatformLogin tree:

- a. In the Top Level Realm, select Authentication > Trees, and click on PlatformLogin.
- b. On the PlatformLogin tree, add a Success URL node between Inner Tree Evaluator and Success, and set its value to http://localhost:8888.

+ Show Me



- c. Click Save.
8. Configure the `PlatformResetPassword` tree:
- a. In the Top Level Realm, select Authentication > Trees, and click on PlatformResetPassword.
 - b. On the `PlatformResetPassword` tree, add a `Success URL` node between `Patch Object` and `Success`, and set its value to `http://localhost:8888`.
- + Show Me



c. Click Save.

9. For the authentication trees that require email, set the External Login Page URL.

In the Top Level Realm, select Authentication > Settings, and click the General tab. Set External Login Page URL to <http://localhost:8083>, then click Save Changes.

Map Authentication Trees

Map the platform trees to the corresponding Self-Service endpoints. For more information about this step, see "Configure Self-Service Trees Endpoints" in the *Platform Self-Service Guide*.

1. From the top menu, select Configure > Global Services > Self Service Trees.
2. Add the following Tree Mappings:

Key	Value
registration	PlatformRegistration
login	PlatformLogin
resetPassword	PlatformResetPassword

Self Service Trees

Realm Defaults

Tree Mapping ?

registration	<input type="text" value="PlatformRegistration"/>	✕
login	<input type="text" value="PlatformLogin"/>	✕
resetPassword	<input type="text" value="PlatformResetPassword"/>	✕
<input type="text" value="Key"/>	<input type="text" value="Value"/>	<input type="button" value="+ Add"/>

3. Click Save Changes.

Configure an OAuth 2.0 Provider Service

1. If you're not currently logged in to the AM console as the `amAdmin` user, log in.
2. In the Top Level Realm, select Services, and click Add a Service.
3. Under Choose a service type, select OAuth2 Provider.
4. For Client Registration Scope Whitelist, add the following scopes:
 - `fr:ldm:*`
 - `am-introspect-all-tokens`
 - `openid`
5. Click Create.
6. On the Advanced tab, check that the following item is listed under Response Type Plugins, and add it if not:

```
id_token|org.forgerock.openidconnect.IdTokenResponseTypeHandler
```

7. Click Save Changes.
8. On the Consent tab, enable Allow Clients to Skip Consent.
9. Click Save Changes.

Configure an IDM Provisioning Service

1. If you're not currently logged in to the AM console as the `amAdmin` user, log in.
2. From the top menu, select Configure > Global Services > IDM Provisioning.
3. Set the following fields:
 - Enabled
 - Deployment URL: `http://openidm.example.com:8080`
 - Deployment Path: `openidm`
 - IDM Provisioning Client: `idm-provisioning`
4. Click Save Changes.

Enable the Password Update Tree (optional)

To let end users update their own passwords (using the `PlatformUpdatePassword` tree), add the `idm-provisioning` service as an authorized client of the `OAuth2 Provider` service.

You cannot use the UI for this step—you must update the OAuth 2.0 configuration over REST:

1. Use the following request to get the token ID:

```
curl \
--request POST \
--header "Content-Type: application/json" \
--header "X-OpenAM-Username: amAdmin" \
--header "X-OpenAM-Password: Password" \
--header "Accept-API-Version: resource=2.0, protocol=1.0" \
'http://am.example.com:8080/am/json/realms/root/authenticate'
{
  "tokenId": "tokenId",
  "successUrl": "/openam/console",
  "realm": "/"
}
```

2. Make a GET request to the `oauth-oidc` endpoint to return the current OAuth configuration:
+ *Example Curl Request*


```

curl \
-X GET \
-H 'Accept-API-Version: protocol=1.0,resource=1.0' \
-H 'X-Requested-With: XMLHttpRequest' \
-H 'Origin: http://am.example.com:8080' \
-H 'Referer: http://am.example.com:8080/am/ui-admin/' \
-H 'Cookie: amlbcookie=01; iPlanetDirectoryPro=tokenId' \
"http://am.example.com:8080/am/json/realms/root/realms-config/services/oauth-oidc"
{
  "core0Auth2Config": {
    "refreshTokenLifetime": 604800,
    "accessTokenLifetime": 3600,
    "usePolicyEngineForScope": false,
    "codeLifetime": 120,
    "issueRefreshTokenOnRefreshedToken": true,
    "macaroonTokensEnabled": false,
    "issueRefreshToken": true,
    "accessTokenModificationScript": "d22f9a0c-426a-4466-b95e-d0f125b0d5fa",
    "statelessTokensEnabled": false
  },
  "core0IDCConfig": {
    "supportedIDTokenEncryptionMethods": [
      "A256GCM",
      "A192GCM",
      "A128GCM",
      "A128CBC-HS256",
      "A192CBC-HS384",
      "A256CBC-HS512"
    ],
    "jwtTokenLifetime": 3600,
    "supportedClaims": [],
    "supportedIDTokenEncryptionAlgorithms": [
      "ECDH-ES+A256KW",
      "ECDH-ES+A192KW",
      "RSA-OAEP",
      "ECDH-ES+A128KW",
      "RSA-OAEP-256",
      "A128KW",
      "A256KW",
      "ECDH-ES",
      "dir",
      "A192KW"
    ],
    "supportedIDTokenSigningAlgorithms": [
      "PS384",
      "RS384",
      "ES384",
      "HS256",
      "HS512",
      "ES256",
      "RS256",
      "HS384",
      "ES512",
      "PS256",
      "PS512",
      "RS512"
    ]
  },
}

```

```

    "oidcClaimsScript": "36863ffb-40ec-48b9-94b1-9a99f71cc3b5"
  },
  "advancedOAuth2Config": {
    "supportedScopes": [
      "openid",
      "fr:idm:*",
      "am-introspect-all-tokens"
    ],
    "tlsCertificateRevocationCheckingEnabled": false,
    "codeVerifierEnforced": "false",
    "tokenSigningAlgorithm": "HS256",
    "authenticationAttributes": [
      "uid"
    ],
    "passwordGrantAuthService": "[Empty]",
    "defaultScopes": [],
    "tlsClientCertificateHeaderFormat": "URLENCODED_PEM",
    "scopeImplementationClass": "org.forgerock.openam.oauth2.OpenAMScopeValidator",
    "responseTypeClasses": [
      "code|org.forgerock.oauth2.core.AuthorizationCodeResponseTypeHandler",
      "id_token|org.forgerock.openidconnect.IdTokenResponseTypeHandler",
      "device_code|org.forgerock.oauth2.core.TokenResponseTypeHandler",
      "token|org.forgerock.oauth2.core.TokenResponseTypeHandler"
    ],
    "tlsCertificateBoundAccessTokensEnabled": true,
    "hashSalt": "changeme",
    "moduleMessageEnabledInPasswordGrant": false,
    "tokenEncryptionEnabled": false,
    "tokenCompressionEnabled": false,
    "grantTypes": [
      "implicit",
      "urn:iETF:params:oauth:grant-type:saml2-bearer",
      "refresh_token",
      "password",
      "client_credentials",
      "urn:iETF:params:oauth:grant-type:device_code",
      "authorization_code",
      "urn:openid:params:grant-type:ciba",
      "urn:iETF:params:oauth:grant-type:uma-ticket",
      "urn:iETF:params:oauth:grant-type:jwt-bearer"
    ],
    "displayNameAttribute": "cn",
    "macaroonTokenFormat": "V2",
    "supportedSubjectTypes": [
      "public"
    ]
  },
  "advancedOIDCConfig": {
    "storeOpsTokens": true,
    "defaultACR": [],
    "supportedRequestParameterEncryptionEnc": [
      "A256GCM",
      "A192GCM",
      "A128GCM",
      "A128CBC-HS256",
      "A192CBC-HS384",
      "A256CBC-HS512"
    ],
    "claimsParameterSupported": false,
  }
}

```

```

"amrMappings": {},
"supportedUserInfoEncryptionEnc": [
  "A256GCM",
  "A192GCM",
  "A128GCM",
  "A128CBC-HS256",
  "A192CBC-HS384",
  "A256CBC-HS512"
],
"authorisedIdmDelegationClients": [],
"alwaysAddClaimsToToken": false,
"supportedUserInfoSigningAlgorithms": [
  "ES384",
  "HS256",
  "HS512",
  "ES256",
  "RS256",
  "HS384",
  "ES512"
],
"supportedRequestParameterEncryptionAlgorithms": [
  "ECDH-ES+A256KW",
  "ECDH-ES+A192KW",
  "ECDH-ES+A128KW",
  "RSA-OAEP",
  "RSA-OAEP-256",
  "A128KW",
  "A256KW",
  "ECDH-ES",
  "dir",
  "A192KW"
],
"supportedTokenIntrospectionResponseEncryptionEnc": [
  "A256GCM",
  "A192GCM",
  "A128GCM",
  "A128CBC-HS256",
  "A192CBC-HS384",
  "A256CBC-HS512"
],
"supportedTokenIntrospectionResponseSigningAlgorithms": [
  "PS384",
  "RS384",
  "EdDSA",
  "ES384",
  "HS256",
  "HS512",
  "ES256",
  "RS256",
  "HS384",
  "ES512",
  "PS256",
  "PS512",
  "RS512"
],
"authorisedOpenIdConnectSSOClients": [],
"idTokenInfoClientAuthenticationEnabled": true,
"supportedRequestParameterSigningAlgorithms": [
  "PS384",

```

```

"RS384",
"ES384",
"HS256",
"HS512",
"ES256",
"RS256",
"HS384",
"ES512",
"PS256",
"PS512",
"RS512"
],
"supportedUserInfoEncryptionAlgorithms": [
"ECDH-ES+A256KW",
"ECDH-ES+A192KW",
"RSA-OAEP",
"ECDH-ES+A128KW",
"RSA-OAEP-256",
"A128KW",
"A256KW",
"ECDH-ES",
"dir",
"A192KW"
],
"supportedTokenIntrospectionResponseEncryptionAlgorithms": [
"ECDH-ES+A256KW",
"ECDH-ES+A192KW",
"RSA-OAEP",
"ECDH-ES+A128KW",
"RSA-OAEP-256",
"A128KW",
"A256KW",
"ECDH-ES",
"dir",
"A192KW"
],
"supportedTokenEndpointAuthenticationSigningAlgorithms": [
"PS384",
"RS384",
"ES384",
"HS256",
"HS512",
"ES256",
"RS256",
"HS384",
"ES512",
"PS256",
"PS512",
"RS512"
],
"loaMapping": {}
},
"clientDynamicRegistrationConfig": {
"dynamicClientRegistrationSoftwareStatementRequired": false,
"dynamicClientRegistrationScope": "dynamic_client_registration",
"requiredSoftwareStatementAttestedAttributes": [
"redirect_uris"
],
"generateRegistrationAccessTokens": true,

```

```

    "allowDynamicRegistration": false
  },
  "cibaConfig": {
    "supportedCibaSigningAlgorithms": [
      "ES256",
      "PS256"
    ],
    "cibaAuthReqIdLifetime": 600,
    "cibaMinimumPollingInterval": 2
  },
  "consent": {
    "enableRemoteConsent": false,
    "supportedRcsRequestSigningAlgorithms": [
      "PS384",
      "RS384",
      "ES384",
      "HS256",
      "HS512",
      "ES256",
      "RS256",
      "HS384",
      "ES512",
      "PS256",
      "PS512",
      "RS512"
    ],
    "supportedRcsResponseSigningAlgorithms": [
      "PS384",
      "RS384",
      "ES384",
      "HS256",
      "HS512",
      "ES256",
      "RS256",
      "HS384",
      "ES512",
      "PS256",
      "PS512",
      "RS512"
    ],
    "clientsCanSkipConsent": true,
    "supportedRcsRequestEncryptionAlgorithms": [
      "ECDH-ES+A256KW",
      "ECDH-ES+A192KW",
      "RSA-OAEP",
      "ECDH-ES+A128KW",
      "RSA-OAEP-256",
      "A128KW",
      "A256KW",
      "ECDH-ES",
      "dir",
      "A192KW"
    ],
    "supportedRcsResponseEncryptionMethods": [
      "A256GCM",
      "A192GCM",
      "A128GCM",
      "A128CBC-HS256",
      "A192CBC-HS384"
    ]
  }
}

```

```

    "A256CBC-HS512"
  ],
  "supportedRcsRequestEncryptionMethods": [
    "A256GCM",
    "A192GCM",
    "A128GCM",
    "A128CBC-HS256",
    "A192CBC-HS384",
    "A256CBC-HS512"
  ],
  "remoteConsentServiceId": "[Empty]",
  "supportedRcsResponseEncryptionAlgorithms": [
    "ECDH-ES+A256KW",
    "ECDH-ES+A192KW",
    "ECDH-ES+A128KW",
    "RSA-OAEP",
    "RSA-OAEP-256",
    "A128KW",
    "A256KW",
    "ECDH-ES",
    "dir",
    "A192KW"
  ]
},
"deviceCodeConfig": {
  "devicePollInterval": 5,
  "deviceCodeLifetime": 300
},
"_id": "",
"_type": {
  "_id": "oauth-oidc",
  "name": "OAuth2 Provider",
  "collection": false
}
}
}

```

3. Make a PUT request to the `oauth-oidc` endpoint to submit the JSON payload returned in the previous step, replacing the value of the `authorisedIdmDelegationClients` property under `advancedOIDCConfig` with `idm-provisioning`:

```
"authorisedIdmDelegationClients": [ "idm-provisioning"],
```

+ *Example Curl Request*

```

curl \
--request PUT \
-H 'User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:70.0) Gecko/20100101 Firefox/70.0' \
-H 'Accept: application/json, text/javascript, */*; q=0.01' \
-H 'Accept-Language: en-US' \
--compressed \
-H 'Content-Type: application/json' \
-H 'Accept-API-Version: protocol=1.0,resource=1.0' \
-H 'X-Requested-With: XMLHttpRequest' \
-H 'Cache-Control: no-cache' \

```

```

-H 'Origin: http://am.example.com:8080' \
-H 'DNT: 1' \
-H 'Connection: keep-alive' \
-H 'Referer: http://am.example.com:8080/am/ui-admin/' \
-H 'Cookie: amlbcookie=01; iPlanetDirectoryPro=tokenId' \
--data '{
  "core0Auth2Config": {
    "refreshTokenLifetime": 604800,
    "accessTokenLifetime": 3600,
    "usePolicyEngineForScope": false,
    "codeLifetime": 120,
    "issueRefreshTokenOnRefreshedToken": true,
    "macaroonTokensEnabled": false,
    "issueRefreshToken": true,
    "accessTokenModificationScript": "d22f9a0c-426a-4466-b95e-d0f125b0d5fa",
    "statelessTokensEnabled": false
  },
  "core0IDCConfig": {
    "supportedIDTokenEncryptionMethods": [
      "A256GCM",
      "A192GCM",
      "A128GCM",
      "A128CBC-HS256",
      "A192CBC-HS384",
      "A256CBC-HS512"
    ],
    "jwtTokenLifetime": 3600,
    "supportedClaims": [],
    "supportedIDTokenEncryptionAlgorithms": [
      "ECDH-ES+A256KW",
      "ECDH-ES+A192KW",
      "RSA-OAEP",
      "ECDH-ES+A128KW",
      "RSA-OAEP-256",
      "A128KW",
      "A256KW",
      "ECDH-ES",
      "dir",
      "A192KW"
    ],
    "supportedIDTokenSigningAlgorithms": [
      "PS384",
      "RS384",
      "ES384",
      "HS256",
      "HS512",
      "ES256",
      "RS256",
      "HS384",
      "ES512",
      "PS256",
      "PS512",
      "RS512"
    ],
    "oidcClaimsScript": "36863ffb-40ec-48b9-94b1-9a99f71cc3b5"
  },
  "advanced0Auth2Config": {
    "supportedScopes": [
      "openid",

```

```

        "fr:idm:*",
        "am-introspect-all-tokens"
    ],
    "tlsCertificateRevocationCheckingEnabled": false,
    "codeVerifierEnforced": "false",
    "tokenSigningAlgorithm": "HS256",
    "authenticationAttributes": [
        "uid"
    ],
    "passwordGrantAuthService": "[Empty]",
    "defaultScopes": [],
    "tlsClientCertificateHeaderFormat": "URLENCODED_PEM",
    "scopeImplementationClass": "org.forgerock.openam.oauth2.OpenAMScopeValidator",
    "responseTypeClasses": [
        "code|org.forgerock.oauth2.core.AuthorizationCodeResponseTypeHandler",
        "id_token|org.forgerock.openidconnect.IdTokenResponseTypeHandler",
        "device_code|org.forgerock.oauth2.core.TokenResponseTypeHandler",
        "token|org.forgerock.oauth2.core.TokenResponseTypeHandler"
    ],
    "tlsCertificateBoundAccessTokensEnabled": true,
    "hashSalt": "changeme",
    "moduleMessageEnabledInPasswordGrant": false,
    "tokenEncryptionEnabled": false,
    "tokenCompressionEnabled": false,
    "grantTypes": [
        "implicit",
        "urn:iETF:params:oauth:grant-type:saml2-bearer",
        "refresh_token",
        "password",
        "client_credentials",
        "urn:iETF:params:oauth:grant-type:device_code",
        "authorization_code",
        "urn:openid:params:grant-type:ciba",
        "urn:iETF:params:oauth:grant-type:uma-ticket",
        "urn:iETF:params:oauth:grant-type:jwt-bearer"
    ],
    "displayNameAttribute": "cn",
    "macaroonTokenFormat": "V2",
    "supportedSubjectTypes": [
        "public"
    ]
},
"advancedOIDCConfig": {
    "storeOpsTokens": true,
    "authorisedIdmDelegationClients": ["idm-provisioning"],
    "defaultACR": [],
    "supportedRequestParameterEncryptionEnc": [
        "A256GCM",
        "A192GCM",
        "A128GCM",
        "A128CBC-HS256",
        "A192CBC-HS384",
        "A256CBC-HS512"
    ],
    "claimsParameterSupported": false,
    "amrMappings": {},
    "supportedUserInfoEncryptionEnc": [
        "A256GCM",
        "A192GCM",

```



```

        "A128GCM",
        "A128CBC-HS256",
        "A192CBC-HS384",
        "A256CBC-HS512"
    ],
    "alwaysAddClaimsToToken": false,
    "supportedUserInfoSigningAlgorithms": [
        "ES384",
        "HS256",
        "HS512",
        "ES256",
        "RS256",
        "HS384",
        "ES512"
    ],
    "supportedRequestParameterEncryptionAlgorithms": [
        "ECDH-ES+A256KW",
        "ECDH-ES+A192KW",
        "ECDH-ES+A128KW",
        "RSA-OAEP",
        "RSA-OAEP-256",
        "A128KW",
        "A256KW",
        "ECDH-ES",
        "dir",
        "A192KW"
    ],
    "supportedTokenIntrospectionResponseEncryptionEnc": [
        "A256GCM",
        "A192GCM",
        "A128GCM",
        "A128CBC-HS256",
        "A192CBC-HS384",
        "A256CBC-HS512"
    ],
    "supportedTokenIntrospectionResponseSigningAlgorithms": [
        "PS384",
        "RS384",
        "EdDSA",
        "ES384",
        "HS256",
        "HS512",
        "ES256",
        "RS256",
        "HS384",
        "ES512",
        "PS256",
        "PS512",
        "RS512"
    ],
    "authorisedOpenIdConnectSSOClients": [],
    "idTokenInfoClientAuthenticationEnabled": true,
    "supportedRequestParameterSigningAlgorithms": [
        "PS384",
        "RS384",
        "ES384",
        "HS256",
        "HS512",
        "ES256",
    ]

```

```

        "RS256",
        "HS384",
        "ES512",
        "PS256",
        "PS512",
        "RS512"
    ],
    "supportedUserInfoEncryptionAlgorithms": [
        "ECDH-ES+A256KW",
        "ECDH-ES+A192KW",
        "RSA-OAEP",
        "ECDH-ES+A128KW",
        "RSA-OAEP-256",
        "A128KW",
        "A256KW",
        "ECDH-ES",
        "dir",
        "A192KW"
    ],
    "supportedTokenIntrospectionResponseEncryptionAlgorithms": [
        "ECDH-ES+A256KW",
        "ECDH-ES+A192KW",
        "RSA-OAEP",
        "ECDH-ES+A128KW",
        "RSA-OAEP-256",
        "A128KW",
        "A256KW",
        "ECDH-ES",
        "dir",
        "A192KW"
    ],
    "supportedTokenEndpointAuthenticationSigningAlgorithms": [
        "PS384",
        "RS384",
        "ES384",
        "HS256",
        "HS512",
        "ES256",
        "RS256",
        "HS384",
        "ES512",
        "PS256",
        "PS512",
        "RS512"
    ],
    "loaMapping": {}
},
"clientDynamicRegistrationConfig": {
    "dynamicClientRegistrationSoftwareStatementRequired": false,
    "dynamicClientRegistrationScope": "dynamic_client_registration",
    "requiredSoftwareStatementAttestedAttributes": [
        "redirect_uris"
    ],
    "generateRegistrationAccessTokens": true,
    "allowDynamicRegistration": false
},
"cibaConfig": {
    "supportedCibaSigningAlgorithms": [
        "ES256",

```

```

        "PS256"
    ],
    "cibaAuthReqIdLifetime": 600,
    "cibaMinimumPollingInterval": 2
  },
  "consent": {
    "enableRemoteConsent": false,
    "supportedRcsRequestSigningAlgorithms": [
      "PS384",
      "RS384",
      "ES384",
      "HS256",
      "HS512",
      "ES256",
      "RS256",
      "HS384",
      "ES512",
      "PS256",
      "PS512",
      "RS512"
    ],
    "supportedRcsResponseSigningAlgorithms": [
      "PS384",
      "RS384",
      "ES384",
      "HS256",
      "HS512",
      "ES256",
      "RS256",
      "HS384",
      "ES512",
      "PS256",
      "PS512",
      "RS512"
    ],
    "clientsCanSkipConsent": true,
    "supportedRcsRequestEncryptionAlgorithms": [
      "ECDH-ES+A256KW",
      "ECDH-ES+A192KW",
      "RSA-OAEP",
      "ECDH-ES+A128KW",
      "RSA-OAEP-256",
      "A128KW",
      "A256KW",
      "ECDH-ES",
      "dir",
      "A192KW"
    ],
    "supportedRcsResponseEncryptionMethods": [
      "A256GCM",
      "A192GCM",
      "A128GCM",
      "A128CBC-HS256",
      "A192CBC-HS384",
      "A256CBC-HS512"
    ],
    "supportedRcsRequestEncryptionMethods": [
      "A256GCM",
      "A192GCM",

```

```

    "A128GCM",
    "A128CBC-HS256",
    "A192CBC-HS384",
    "A256CBC-HS512"
  ],
  "remoteConsentServiceId": "[Empty]",
  "supportedRcsResponseEncryptionAlgorithms": [
    "ECDH-ES+A256KW",
    "ECDH-ES+A192KW",
    "ECDH-ES+A128KW",
    "RSA-OAEP",
    "RSA-OAEP-256",
    "A128KW",
    "A256KW",
    "ECDH-ES",
    "dir",
    "A192KW"
  ]
},
"deviceCodeConfig": {
  "devicePollInterval": 5,
  "deviceCodeLifetime": 300
},
"_id": "",
"_type": {
  "_id": "oauth-oidc",
  "name": "OAuth2 Provider",
  "collection": false
}
}' \
"http://am.example.com:8080/am/json/realms/root/realms-config/services/oauth-oidc"
{
  "coreOAuth2Config": {
    "refreshTokenLifetime": 604800,
    "accessTokenLifetime": 3600,
    "usePolicyEngineForScope": false,
    "codeLifetime": 120,
    "issueRefreshTokenOnRefreshedToken": true,
    "macaroonTokensEnabled": false,
    "issueRefreshToken": true,
    "accessTokenModificationScript": "d22f9a0c-426a-4466-b95e-d0f125b0d5fa",
    "statelessTokensEnabled": false
  },
  "coreIDCConfig": {
    "supportedIDTokenEncryptionMethods": [
      "A256GCM",
      "A192GCM",
      "A128GCM",
      "A128CBC-HS256",
      "A192CBC-HS384",
      "A256CBC-HS512"
    ],
    "jwtTokenLifetime": 3600,
    "supportedClaims": [],
    "supportedIDTokenEncryptionAlgorithms": [
      "ECDH-ES+A256KW",
      "ECDH-ES+A192KW",
      "RSA-OAEP",
      "ECDH-ES+A128KW",

```

```

    "RSA-OAEP-256",
    "A128KW",
    "A256KW",
    "ECDH-ES",
    "dir",
    "A192KW"
  ],
  "supportedIDTokenSigningAlgorithms": [
    "PS384",
    "RS384",
    "ES384",
    "HS256",
    "HS512",
    "ES256",
    "RS256",
    "HS384",
    "ES512",
    "PS256",
    "PS512",
    "RS512"
  ],
  "oidcClaimsScript": "36863ffb-40ec-48b9-94b1-9a99f71cc3b5"
},
"advancedOAuth2Config": {
  "supportedScopes": [
    "openid",
    "fr:idm:*",
    "am-introspect-all-tokens"
  ],
  "tlsCertificateRevocationCheckingEnabled": false,
  "codeVerifierEnforced": "false",
  "tokenSigningAlgorithm": "HS256",
  "authenticationAttributes": [
    "uid"
  ],
  "passwordGrantAuthService": "[Empty]",
  "defaultScopes": [],
  "tlsClientCertificateHeaderFormat": "URLENCODED_PEM",
  "scopeImplementationClass": "org.forgerock.openam.oauth2.OpenAMScopeValidator",
  "responseTypeClasses": [
    "code|org.forgerock.oauth2.core.AuthorizationCodeResponseTypeHandler",
    "id_token|org.forgerock.openidconnect.IdTokenResponseTypeHandler",
    "device_code|org.forgerock.oauth2.core.TokenResponseTypeHandler",
    "token|org.forgerock.oauth2.core.TokenResponseTypeHandler"
  ],
  "tlsCertificateBoundAccessTokensEnabled": true,
  "hashSalt": "changeme",
  "moduleMessageEnabledInPasswordGrant": false,
  "tokenEncryptionEnabled": false,
  "tokenCompressionEnabled": false,
  "grantTypes": [
    "implicit",
    "urn:ietf:params:oauth:grant-type:saml2-bearer",
    "refresh_token",
    "password",
    "client_credentials",
    "urn:ietf:params:oauth:grant-type:device_code",
    "authorization_code",
    "urn:openid:params:grant-type:ciba",

```

```

    "urn:ietf:params:oauth:grant-type:uma-ticket",
    "urn:ietf:params:oauth:grant-type:jwt-bearer"
  ],
  "displayNameAttribute": "cn",
  "macaroonTokenFormat": "V2",
  "supportedSubjectTypes": [
    "public"
  ]
},
"advancedOIDCConfig": {
  "storeOpsTokens": true,
  "defaultACR": [],
  "supportedRequestParameterEncryptionEnc": [
    "A256GCM",
    "A192GCM",
    "A128GCM",
    "A128CBC-HS256",
    "A192CBC-HS384",
    "A256CBC-HS512"
  ],
  "claimsParameterSupported": false,
  "amrMappings": {},
  "supportedUserInfoEncryptionEnc": [
    "A256GCM",
    "A192GCM",
    "A128GCM",
    "A128CBC-HS256",
    "A192CBC-HS384",
    "A256CBC-HS512"
  ],
  "authorisedIdmDelegationClients": ["idm-provisioning"],
  "alwaysAddClaimsToToken": false,
  "supportedUserInfoSigningAlgorithms": [
    "ES384",
    "HS256",
    "HS512",
    "ES256",
    "RS256",
    "HS384",
    "ES512"
  ],
  "supportedRequestParameterEncryptionAlgorithms": [
    "ECDH-ES+A256KW",
    "ECDH-ES+A192KW",
    "ECDH-ES+A128KW",
    "RSA-OAEP",
    "RSA-OAEP-256",
    "A128KW",
    "A256KW",
    "ECDH-ES",
    "dir",
    "A192KW"
  ],
  "supportedTokenIntrospectionResponseEncryptionEnc": [
    "A256GCM",
    "A192GCM",
    "A128GCM",
    "A128CBC-HS256",
    "A192CBC-HS384",

```

```

"A256CBC-HS512"
],
"supportedTokenIntrospectionResponseSigningAlgorithms": [
  "PS384",
  "RS384",
  "EdDSA",
  "ES384",
  "HS256",
  "HS512",
  "ES256",
  "RS256",
  "HS384",
  "ES512",
  "PS256",
  "PS512",
  "RS512"
],
"authorisedOpenIdConnectSSOClients": [],
"idTokenInfoClientAuthenticationEnabled": true,
"supportedRequestParameterSigningAlgorithms": [
  "PS384",
  "RS384",
  "ES384",
  "HS256",
  "HS512",
  "ES256",
  "RS256",
  "HS384",
  "ES512",
  "PS256",
  "PS512",
  "RS512"
],
"supportedUserInfoEncryptionAlgorithms": [
  "ECDH-ES+A256KW",
  "ECDH-ES+A192KW",
  "RSA-OAEP",
  "ECDH-ES+A128KW",
  "RSA-OAEP-256",
  "A128KW",
  "A256KW",
  "ECDH-ES",
  "dir",
  "A192KW"
],
"supportedTokenIntrospectionResponseEncryptionAlgorithms": [
  "ECDH-ES+A256KW",
  "ECDH-ES+A192KW",
  "RSA-OAEP",
  "ECDH-ES+A128KW",
  "RSA-OAEP-256",
  "A128KW",
  "A256KW",
  "ECDH-ES",
  "dir",
  "A192KW"
],
"supportedTokenEndpointAuthenticationSigningAlgorithms": [
  "PS384",

```

```

    "RS384",
    "ES384",
    "HS256",
    "HS512",
    "ES256",
    "RS256",
    "HS384",
    "ES512",
    "PS256",
    "PS512",
    "RS512"
  ],
  "loaMapping": {}
},
"clientDynamicRegistrationConfig": {
  "dynamicClientRegistrationSoftwareStatementRequired": false,
  "dynamicClientRegistrationScope": "dynamic_client_registration",
  "requiredSoftwareStatementAttestedAttributes": [
    "redirect_uris"
  ],
  "generateRegistrationAccessTokens": true,
  "allowDynamicRegistration": false
},
"cibaConfig": {
  "supportedCibaSigningAlgorithms": [
    "ES256",
    "PS256"
  ],
  "cibaAuthReqIdLifetime": 600,
  "cibaMinimumPollingInterval": 2
},
"consent": {
  "enableRemoteConsent": false,
  "supportedRcsRequestSigningAlgorithms": [
    "PS384",
    "RS384",
    "ES384",
    "HS256",
    "HS512",
    "ES256",
    "RS256",
    "HS384",
    "ES512",
    "PS256",
    "PS512",
    "RS512"
  ],
  "supportedRcsResponseSigningAlgorithms": [
    "PS384",
    "RS384",
    "ES384",
    "HS256",
    "HS512",
    "ES256",
    "RS256",
    "HS384",
    "ES512",
    "PS256",
    "PS512",
    "RS512"
  ]
}

```



```

"RS512"
],
"clientsCanSkipConsent": true,
"supportedRcsRequestEncryptionAlgorithms": [
  "ECDH-ES+A256KW",
  "ECDH-ES+A192KW",
  "RSA-OAEP",
  "ECDH-ES+A128KW",
  "RSA-OAEP-256",
  "A128KW",
  "A256KW",
  "ECDH-ES",
  "dir",
  "A192KW"
],
"supportedRcsResponseEncryptionMethods": [
  "A256GCM",
  "A192GCM",
  "A128GCM",
  "A128CBC-HS256",
  "A192CBC-HS384",
  "A256CBC-HS512"
],
"supportedRcsRequestEncryptionMethods": [
  "A256GCM",
  "A192GCM",
  "A128GCM",
  "A128CBC-HS256",
  "A192CBC-HS384",
  "A256CBC-HS512"
],
"remoteConsentServiceId": "[Empty]",
"supportedRcsResponseEncryptionAlgorithms": [
  "ECDH-ES+A256KW",
  "ECDH-ES+A192KW",
  "ECDH-ES+A128KW",
  "RSA-OAEP",
  "RSA-OAEP-256",
  "A128KW",
  "A256KW",
  "ECDH-ES",
  "dir",
  "A192KW"
]
},
"deviceCodeConfig": {
  "devicePollInterval": 5,
  "deviceCodeLifetime": 300
},
"_id": "",
"_type": {
  "_id": "oauth-oidc",
  "name": "OAuth2 Provider",
  "collection": false
}
}

```

```
}
```

Enable CORS Support

Cross-origin resource sharing (CORS) lets user agents make requests across domains.

1. If you're not currently logged in to the AM console as the `amAdmin` user, log in.
2. From the top menu, select Configure > Global Services > CORS Service.
3. On the Secondary Configurations tab, click Add a Secondary Configuration.
4. On the New Configuration screen, enter the following values:

- Name: `Cors Configuration`

- Accepted Origins:

```
http://localhost:8083  
http://localhost:8082  
http://localhost:8888  
http://openidm.example.com:8080  
https://openidm.example.com:8443
```

- Accepted Methods:

```
HEAD  
DELETE  
POST  
GET  
PUT  
PATCH
```

- Accepted Headers:

```
authorization  
x-openidm-username  
if-none-match  
x-openidm-nosession  
x-openidm-password  
accept-api-version  
x-requested-with  
content-type  
if-match  
cache-control  
user-agent
```

- Exposed Headers: `WWW-Authenticate`

5. Click Create.
6. On the Cors Configuration screen, set the following values:
 - Enable the CORS filter: Enable
 - Max Age: 600
 - Allow Credentials: Enable
7. Click Save Changes.

Set up IDM

This procedure sets up IDM with an external DS instance as the repository. The DS instance is shared with AM as the identity store. The procedure assumes that both IDM and the DS instance have been installed with the listed "Server Settings".

1. Follow the instructions in the *IDM Installation Guide* to install IDM.
2. Edit the `/path/to/openidm/resolver/boot.properties` file to set the hostname:

```
openidm.host=openidm.example.com
```

3. Configure the shared IDM repository.

This step assumes that you have set up the identity store, and exported the DS CA certificate from `identities.example.com` (as shown in Step 4 of "Secure Connections").

- a. Import the DS CA certificate into the IDM truststore:

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

- b. Replace the default `conf/repo.ds.json` file with this file.

Important

Replace the values of `fileBasedTrustManagerFile` and `fileBasedTrustManagerPasswordFile` with the path to your IDM truststore and truststore password file.

Check that the properties under `ldapConnectionFactories` match the DS server that you set up as the identity store.

It is worth starting IDM at this point, to make sure that it can connect to the external DS repository. You must see `OpenIDM ready` in the output:

```
/path/to/openidm/startup.sh  
-> OpenIDM version "7.0.0" build-info  
OpenIDM ready
```

4. Update the `user` object in your managed object configuration by making the following changes to `conf/managed.json`.

All changes are made to the object `user > schema > properties`:

```
{  
  "objects": [  
    {  
      "name": "user",  
      ...  
      "schema": {  
        "properties": {  
          ...  
        }  
      }  
    }  
  ]  
}
```

- a. Remove encryption from the `password` property. Remove:

```
"encryption" : {  
  "purpose": "idm.password.encryption"  
}
```

- b. Add a `cn` property to the `user` object:

```

{
  "objects": [
    {
      "name": "user",
      ...
      "schema": {
        "properties": {
          "_id": {
            ...
          },
          "cn": {
            "title": "Common Name",
            "description": "Common Name",
            "type": "string",
            "viewable": false,
            "searchable": false,
            "userEditable": false,
            "scope": "private",
            "isPersonal": true,
            "isVirtual": true,
            "onStore": {
              "type": "text/javascript",
              "source": "object.cn || (object.givenName + ' ' + object.sn)"
            }
          },
          ...
        }
      }
    }
  ]
}

```

c. (Optional) Configure social authentication.

Add an `aliasList` property to the `user` object:

```

"aliasList": {
  "title": "User Alias Names List",
  "description": "List of identity aliases used primarily to record social IdP subjects for this user",
  "type": "array",
  "items": {
    "type": "string",
    "title": "User Alias Names Items"
  },
  "viewable": false,
  "searchable": false,
  "userEditable": true,
  "returnByDefault": false,
  "isVirtual": false
}

```

5. Change the authentication mechanism to `rsFilter` only:

- Replace the default `conf/authentication.json` file with this file.

- Check that the `clientSecret` matches the `Client secret` that you set for the `idm-resource-server` client in AM (see "Configure OAuth Clients").
- Check that the `scopes` match the `Scope(s)` that you set for the `idm-admin-ui` and `end-user-ui` clients in AM (see "Configure OAuth Clients").

For more information about the `rsFilter` authentication module, see the IDM Security Guide.

6. Edit the IDM Admin UI configuration so that you can still authenticate through the IDM Admin UI:
 - a. In your `/path/to/openidm/conf/ui-configuration.json` file, insert a `platformSettings` object into the `configuration` object:

```
{
  "configuration" : {
    ...
    "platformSettings" : {
      "adminOAuthClient" : "idm-admin-ui",
      "adminOAuthClientScopes" : "openid fr:idm:*",
      "amUrl" : "http://am.example.com:8080/am",
      "loginUrl" : ""
    }
  }
}
```

- b. In your `conf/ui.context-admin.json` file, check that `X-Frame-Options` is set to `SAMEORIGIN`:

+ *Sample `ui.context-admin.json`*

```
{
  "enabled" : true,
  "urlContextRoot" : "/admin",
  "defaultDir" : "&{idm.install.dir}/ui/admin/default",
  "extensionDir" : "&{idm.install.dir}/ui/admin/extension",
  "responseHeaders" : {
    "X-Frame-Options" : "SAMEORIGIN"
  }
}
```

7. Configure the CORS servlet filter.

Replace the default `conf/servletfilter-cors.json` file with this file.

8. If you have not already started IDM, start it now, and test that you can access the IDM Admin UI at `http://openidm.example.com:8080/admin`. When you log in to the Admin UI, use the default AM administrative user (`amAdmin`), and not `openidm-admin`.
9. (Optional) If you want to use the `PlatformForgottenUsername` or `PlatformResetPassword` trees, configure outbound email.

Note

After you have installed the Platform UI, you can configure email through the UI at <http://openidm.example.com:8080/admin>.

IDM is now configured. Move on to setting up the Platform UI.

Chapter 4

Deploy the Platform UIs

The ForgeRock Identity Platform™ includes three UIs:

Admin UI

Lets platform administrators manage applications, identities, end user journeys, and so on.

Warning

The Admin UI is currently in *preview mode*. It is an evolving interface, and will be fully supported in the near future.

End User UI

Lets end users manage certain aspects of their own accounts.

Important

This UI is *not* the same as the standalone IDM End User UI. The platform End User UI is designed specifically for an integrated platform deployment. The IDM End User UI should be used only in a standalone IDM deployment.

Login UI

A unified login UI that lets both administrators and end users log in to the platform.

The three UIs are provided as separate Docker images.

Download and Run the Platform UI

1. Download the Docker images:
 - a. Download the Admin UI:
 - b. Download the Enduser UI:
 - c. Download the Login UI:

```
docker pull gcr.io/forgerock-io/platform-admin-ui:7.0.0
```

```
docker pull gcr.io/forgerock-io/platform-enduser-ui:7.0.0
```



```
docker pull gcr.io/forgerock-io/platform-login-ui:7.0.0
```

2. Create an environment file named `platform_env` that will be passed to the `docker` command.

The file should have the following contents:

```
AM_URL=http://am.example.com:8080/am
AM_ADMIN_URL=http://am.example.com:8080/am/ui-admin
IDM_REST_URL=http://openidm.example.com:8080/openidm
IDM_ADMIN_URL=http://openidm.example.com:8080/admin
IDM_UPLOAD_URL=http://openidm.example.com:8080/upload
IDM_EXPORT_URL=http://openidm.example.com:8080/export
LOGIN_UI_URL=http://localhost:8083/#/service/PlatformLogin
ENDUSER_UI_URL=http://localhost:8888
PLATFORM_ADMIN_URL=http://localhost:8082
ENDUSER_CLIENT_ID=end-user-ui
ADMIN_CLIENT_ID=idm-admin-ui
THEME=default
PLATFORM_UI_LOCALE=en
```

3. List the Docker images to find their IDs:

```
docker image list
```

REPOSITORY	TAG	IMAGE ID	...
gcr.io/forgerock-io/platform-admin-ui	7.0.0	82e714f77229	...
gcr.io/forgerock-io/platform-login-ui	7.0.0	cbb85cd78931	...
gcr.io/forgerock-io/platform-enduser-ui	7.0.0	b990cf769697	...

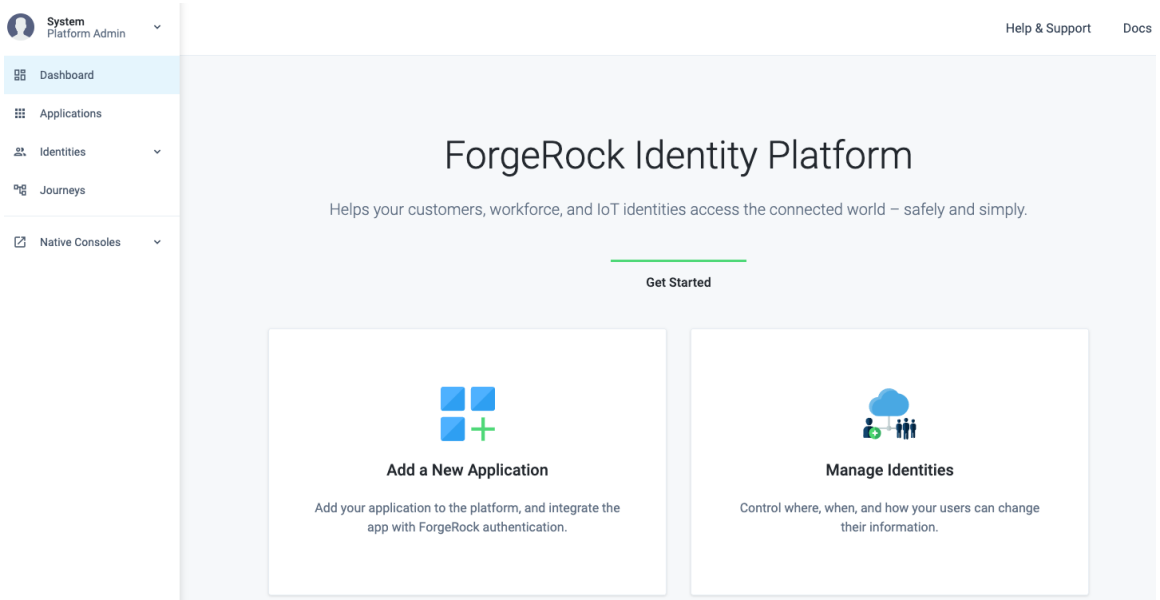
4. Start each Docker image respectively, specifying the environment file:

- a. Change to the the directory in which you saved the `platform_env` file, then start the Admin UI:

```
docker run -t -i -p 8082:8080 --env-file=platform_env 82e714f77229
Replacing env vars in JS

Setting AM URL as http://am.example.com:8080/am
Setting AM ADMIN URL as http://am.example.com:8080/am/ui-admin
Setting IDM REST URL as http://openidm.example.com:8080/openidm
Setting IDM ADMIN URL as http://openidm.example.com:8080/admin
...
```

Check that the Admin UI is running by going to <http://localhost:8082>:

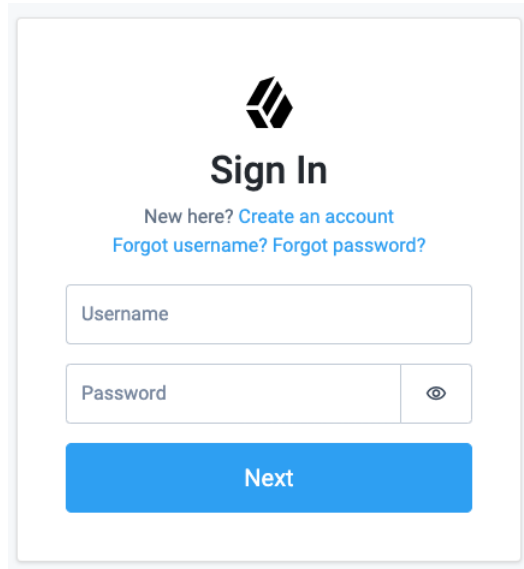


- b. In a new terminal, change to the the directory in which you saved the `platform_env` file, then start the Login UI:

```
docker run -t -i -p 8083:8080 --env-file=platform_env cbb85cd78931
Replacing env vars in JS

Setting AM URL as http://am.example.com:8080/am
Setting AM ADMIN URL as http://am.example.com:8080/am/ui-admin
Setting IDM REST URL as http://openidm.example.com:8080/openidm
Setting IDM ADMIN URL as http://openidm.example.com:8080/admin
...
```

Verify that the UI is running by going to <http://localhost:8083>:



- c. In a new terminal, change to the the directory in which you saved the `platform_env` file, then start the End User UI:

```
docker run -t -i -p 8888:8080 --env-file=platform_env b990cf769697
Replacing env vars in JS

Setting AM URL as http://am.example.com:8080/am
Setting AM ADMIN URL as http://am.example.com:8080/am/ui-admin
Setting IDM REST URL as http://openidm.example.com:8080/openidm
Setting IDM ADMIN URL as http://openidm.example.com:8080/admin
...
```

Verify that the UI is running by going to <http://localhost:8888>:

Chapter 5

Migrate From a Full Stack Deployment

Previous versions of the ForgeRock Identity Platform provided a sample deployment called the Full Stack Sample. This kind of deployment is not supported for new integrations between IDM and AM, and you should follow the steps outlined in this guide to set up a new deployment. There is no automated migration facility for an existing deployment based on the *full stack sample*, but you can follow the tips in this chapter help you move to a new integrated deployment.

Note

This is not an exhaustive list and you will have additional configuration changes to make, based on how you deployed the full stack sample.

Self-Service Configuration

All self-service configuration is now done in the AM UI, with the exception of the following elements:

- KBA
- Terms & Conditions
- Policies
- Privacy & Consent
- Email templates and services

You can therefore delete all the IDM `selfservice*` files from your configuration, apart from `selfservice.kba.json` and `selfservice.terms.json`. These two files are still used by the new AM authentication trees.

You will need to rebuild all existing self-service processes as AM authentication trees. Note that tree nodes are more modular than the legacy IDM self-service stages. It might take more than one node to replace a self-service stage.

In a platform configuration, you can optionally customize the IDM Admin UI to hide or remove all of the self-service functions that are no longer configured through IDM (such as registration and password reset).

Authentication

Integrated deployments now use a single `rsFilter` authentication module that allows authentication using AM bearer tokens. You must replace your project's `conf/authentication.json` file with the platform `authentication.json` file.

Configure OAuth clients and the IDM Provisioning Service in the AM UI.

UI Customization

The UI retrieves access tokens from AM. If you have customized your Self-Service UI for a full-stack deployment, it is recommended that you start a new customization, based on the Platform UI. Your existing UI customizations will not work in a new platform deployment.

Chapter 6

HSMs and ForgeRock Software

This part covers how you can use a Hardware Security Module (HSM) to protect ForgeRock software private and secret keys.

On Protecting Secrets

You must protect private keys and secret keys that servers use for cryptographic operations:

Operating System Protections

In many deployments, operating system protections are sufficient. Operating systems are always sufficient for public keys and keystores that do not require protection.

Isolate the server account on the operating system where it runs, and allow only that account to access the keys. If you store the keys with version control software, also control access to that.

This deployment option generally offers a better performance/cost ratio. You must take more care to prevent private and secret keys from being exposed, however.

HSM

For stronger protection, you can use an HSM.

An HSM is a secure device that holds the keys, and protects them from unauthorized access. With an HSM, no one gets direct access to the keys. If an attacker does manage to break into an HSM and theoretically get access to the keys, HSM are designed to make the tampering evident, so you can take action.

To put a private or secret key on an HSM, you have the HSM generate the key. The key never leaves the HSM. Instead, if you need the result of a cryptographic operation involving the key, you authenticate to the HSM and request the operation. The HSM performs the operation without ever exposing any private or secret keys. It only returns the result.

An HSM can be certified to comply with international standards, including FIPS-140 and Common Criteria. An HSM that is certified to comply with these standards can be part of your supported ForgeRock software solution.

ForgeRock software uses the HSM through standard PKCS#11 interfaces, and supports the use of compliant cryptographic algorithms.

An HSM generally offers higher security, but with a significant cost and impact on performance. Good HSMs and standards compliance come with their own monetary costs.

In terms of performance, each cryptographic operation incurs at minimum a round trip on a secure connection, compared with an operation in local memory for keys on the system. Even if the deployment may guarantee the same throughput, take the latency into account when deciding to deploy with HSMs.

Key Management Services

Key management services, such as Google Cloud KMS and others, offer similar advantages and tradeoffs as HSMs.

Latency for online key management services can be very high.

Performance

Throughput, the rate of operations completed, might or might not be impacted by your choice of protecting secrets.

Latency, how long individual operations take, will be impacted by your choice of protecting secrets.

Performance Example

For example, suppose you want a system that signs one million JWTs per second. As long as you can distribute the signing key across enough hardware, your system can sign one million JWTs a second. This is true even if each signing operation takes ten seconds. You simply need ten million systems, and the network hardware to use them in parallel. Throughput therefore depends mainly on how much you can spend.

Suppose, however, that each signing operation must take no longer than ten milliseconds. Furthermore, suppose you have an HSM that can perform a signing operation in one millisecond, but that the network round trip from the system to the HSM takes an average of eleven milliseconds. In this case, you cannot fix performance by buying a faster HSM, or by somehow speeding up the software. You must first reduce the network time if you want to meet your latency requirements.

Perhaps the same signing operation with a key stored on the operating system takes two milliseconds. Consider the options based on your cost and security requirements.

Performance also depends on the following:

- The impact of latency is greater when performing symmetric cryptographic operations (AES, HMAC) compared to public key cryptography (RSA, EC).
- For public key cryptography, only operations that use the private key must contact the HSM (signing, decryption).

Operations using the public key (verifying a signature, encryption) retrieve the public key from the HSM once, and then use it locally.

- Most public key cryptography uses hybrid encryption, where only a small operation must be done on the HSM, and the rest can be done locally.

For example, when decrypting a large message, typically, the HSM is only used to decrypt a per-message AES key that is then used to decrypt the rest of the message locally.

Similarly, for signing, the message to sign is first hashed in-memory using a secure hash algorithm, and only the short hash value is sent to the HSM to be signed.

In contrast, when using symmetric key algorithms directly with an HSM, the entire message must be streamed to and from the HSM.

How ForgeRock Services Interact with HSMs

ForgeRock services built with Java interact with HSMs through Java PKCS#11 providers.

The PKCS#11 standard defines a cryptographic token interface, a platform-independent API for accessing an HSM, for example. ForgeRock services support the use of the Sun PKCS11 provider.

The Sun PKCS11 provider does not implement every operation and algorithm supported by every HSM. For details on the Sun PKCS11 provider's capabilities, see the [JDK Providers Documentation](#), and the [JDK PKCS#11 Reference Guide](#).

How you configure the JVM to use your HSM depends on the Java environment and on your HSM. ForgeRock services do not implement or manage this configuration, but they do depend on it. Before configuring ForgeRock services to use your HSM, you must therefore configure the JVM's Sun PKCS11 provider to access your HSM. For details on how to configure the Java Sun PKCS11 provider with your HSM, see the documentation for your HSM, and the [JDK PKCS#11 Reference Guide](#).

Sun PKCS11 Provider Hints

The provider configuration depends on your provider.

Compare the following hints for ForgeRock software with the documentation for your HSM:

name = FooHsm

The name used to produce the Sun PKCS11 provider instance name for your HSM.

FooHsm is just an example.

CKA_TOKEN = false

Keys generated using the Java **keytool** command effectively have this set to true. They are permanent keys, to be shared across the deployment.

Set this to false to prevent temporary keys for RSA hybrid encryption used by some platform components from exhausting HSM storage.


```
CKA_PRIVATE = true
```

The key can only be accessed after authenticating to the HSM.

```
CKA_EXTRACTABLE = false
```

```
CKA_SENSITIVE = true
```

`CKA_EXTRACTABLE = false` means the key cannot be extracted *unless it is encrypted*.

`CKA_SENSITIVE = true` means do not let the secret key be extracted out of the HSM. Set this to true for long-term keys.

Only change these settings to back up keys to a different HSM, when you cannot use a proprietary solution. For example, you might change these settings if the HSMs are from different vendors.

```
CKA_ENCRYPT = true
```

The key can be used to encrypt data.

```
CKA_DECRYPT = true
```

The key can be used to decrypt data.

```
CKA_SIGN = true
```

The key can be used to sign data.

```
CKA_VERIFY = true
```

The key can be used to verify signatures.

```
CKA_WRAP = true
```

The key can be used to sign data.

```
CKA_UNWRAP = true
```

The key can be used to unwrap another key.

When Using Keytool

When using the Java **keytool** command to generate or access keys on the HSM, set the following options appropriately:

-alias *alias*

If key pair generation fails, use a new *alias* for the next try. This prevents the conflicts where the previous attempt to create a key was only a partial failure, leaving part of key pair using the previous alias.

-keystore none

Required setting.

-providername *providerName*

Required setting.

Prefix `SunPKCS11-` to the name in the configuration. If, in the Sun PKCS11 configuration, `name = FooHSM`, then the `providerName` is `SunPKCS11-FooHSM`.

-storetype pkcs11

Required setting.

If necessary, add the following settings:

-providerClass `sun.security.pkcs11.SunPKCS11`

Use this to install the provider dynamically.

-providerArg */path/to/config/file.conf*

Use this to install the provider dynamically.

The exact configuration depends on your HSM.

When Java Cannot Find Keys

When keys generated with one Java PKCS#11 provider are later accessed using the Sun PKCS11 provider, the providers may have different naming conventions.

Java's KeyStore abstraction requires that all private key objects have a corresponding Certificate object. SecretKey objects, such as AES or HMAC keys, are excluded from this requirement.

The Sun PKCS11 KeyStore provider loops through all defined private key entries in the HSM (class = private key), and tries to match them up with a corresponding certificate entry (class = certificate) by comparing the `CKA_ID` of the certificate entry to the `CKA_ID` of the private key entry. There may be multiple certificates using the same private key pair. The matching process can take several seconds at startup time if you have many keys.

If keys are not found, it is likely that the private key entry `CKA_ID` does not match the certificate entry `CKA_ID`.

HSM Features and ForgeRock Services

This part outlines how some ForgeRock platform features support HSMs.

JWT Encryption Algorithms

JWT Encryption Algorithm	Java Algorithm Equivalent	Supported by Sun PKCS11 Provider?
RSA1_5	RSA/ECB/PKCS1Padding ^a	Yes
RSA-OAEP	RSA/ECB/OAEPWithSHA-1AndMGF1Padding	No
RSA-OAEP-256	RSA/ECB/OAEPWithSHA-256AndMGF1Padding	No
ECDH-ES (+A128KW, and so on) ^b	ECDH (KeyAgreement vs. Cipher)	Yes
dir with A128GCM, A192GCM, A256GCM	AES/GCM/NoPadding	Yes
dir with A128CBC-HS256, and so on	AES/CBC/PKCS5Padding + HmacSHA256, and so on	No
A128KW, A192KW, A256KW	AESWrap	No

^aPKCS#1 version 1.5 padding has known vulnerabilities and should be avoided.

^bThe Sun PKCS11 implementation of ECDH KeyAgreement requires that the derived key be extractable. This is not a security issue, as the derived key is unique to each message, and therefore no more sensitive than the message it is protecting, due to the use of fresh ephemeral keys for each message (ECIES). To ensure that the derived keys are extractable, add the following to the PKCS11 configuration file:

```
attributes(*,CKO_PRIVATE_KEY,CKK_EC) = {
  CKA_SIGN = true
  CKA_DERIVE = true
  CKA_TOKEN = true
}

attributes(generate,CKO_SECRET_KEY,CKK_GENERIC_SECRET) = {
  CKA_SENSITIVE = false
  CKA_EXTRACTABLE = true
  CKA_TOKEN = false
}
```

This also ensures that EC keys generated with the **keytool** command are marked as allowing ECDH key derivation. The derived keys are also marked as **CKA_TOKEN = false**, which ensures that the derived keys are only created as session keys, and automatically deleted when the session ends to prevent filling up the HSM with temporary keys.

JWT Signing Algorithms

JWT Signing Algorithm	Java Algorithm Equivalent	Supported by Sun PKCS11 Provider?
HS256, HS384, HS512	HmacSHA256, HmacSHA384, HmacSHA512	Yes
RS256, RS384, RS512	SHA256WithRSA, SHA384WithRSA, SHA512WithRSA	Yes

JWT Signing Algorithm	Java Algorithm Equivalent	Supported by Sun PKCS11 Provider?
PS256, PS384, PS512	RSASSA-PSS or SHA256WithRSAAndMGF1, and so on	No
ES256, ES384, ES512	SHA256WithECDSA, and so on	Yes
EdDSA	Under development	No

Configure ForgeRock Services to Use an HSM

Once you have configured the Java environment to use your HSM through the Sun PKCS11 Provider, you can configure ForgeRock software to use the HSM:

- AM: See [Configuring Secrets, Certificates, and Keys](#).
- DS: See [PKCS#11 Hardware Security Module](#).
- IDM: See [Configuring IDM For a Hardware Security Module \(HSM\) Device](#).
- IG: See [Secrets, and HsmSecretStore](#).