

Setting up Authentication in Design Room ONE with Keycloak

This document describes how to setup and use authentication and user management in Design Room ONE by means of Keycloak Integration. This functionality is provided as EXPERIMENTAL. It is included for users' evaluation and feedback and is not recommended for production use. Please see the list of [know limitations](#) in the end of the document.

Table of Contents

- Keycloak Server Installation..... 3**
 - Downloading Keycloak..... 3
 - Starting Keycloak Server..... 3
- Keycloak Server Configuration 4**
 - Creating a Realm Admin 4
 - Configuring Master Realm..... 4
 - Configuring Drone Realm 8
- Setting up Users and Roles..... 13**
 - Creating Users..... 13
 - Managing Access with Roles..... 16
 - Importing Users from Active Directory 18
- Configuring Design Room ONE Server 23**
 - Starting the Design Room ONE Server 24
 - Logging into the Design Room ONE Server with the New User 25
- Exporting Models with Design Room ONE Integration Plugin 25**
 - Prerequisites: 25
- Single Sign-On with Jazz Authorization Server via OpenID 31**
 - Prerequisites..... 31
 - Creating a New Identify Provider in Keycloak 31
 - Setting up JAS Configuration Security 33
 - Creating a Relying Party Application in JAS..... 34

Creating a New Identify Provider in Keycloak Continued..... 36
Logging in with JAS Authentication..... 37
Known Limitations..... 39

Keycloak Server Installation

The instructions below use the following “variables” that need to be replaced with their actual values

DRONE_HOST_NAME e.g. drone.mycomp.any

KEYCLOAK_HOST_NAME e.g. keycloakhost.mycomp.any

KEYCLOAK_INSTALL_DIR e.g. C:/Install/Keycloak

If Keycloak and Design Room ONE are installed on different machines, there should be no firewall blocking their communication.

Downloading Keycloak

Use the following link:

<https://downloads.jboss.org/keycloak/7.0.1/keycloak-7.0.1.zip>

Starting Keycloak Server

Unzip the downloaded file keycloak-7.0.1.zip into an installation directory of your choice. We will refer to this installation directory as **KEYCLOAK_INSTALL_DIR**.

Run the standalone version of the server in **KEYCLOAK_INSTALL_DIR/bin**

For Linux systems use the command: **standalone.sh -b 0.0.0.0**

For Windows: **standalone.bat -b 0.0.0.0**

This script uses **KEYCLOAK_INSTALL_DIR/standalone/configuration/standalone.xml** as properties input by default

```
<socket-binding-group name="standard-sockets" default-interface="public" port-  
offset="${jboss.socket.binding.port-offset:0}">  
  <socket-binding name="management-  
http" interface="management" port="${jboss.management.http.port:9990}"/>  
  <socket-binding name="management-  
https" interface="management" port="${jboss.management.https.port:9993}"/>  
  <socket-binding name="ajp" port="${jboss.ajp.port:8009}"/>  
  <socket-binding name="http" port="${jboss.http.port:8080}"/>  
  <socket-binding name="https" port="${jboss.https.port:8443}"/>  
  <socket-binding name="txn-recovery-environment" port="4712"/>  
  <socket-binding name="txn-status-manager" port="4713"/>  
  <outbound-socket-binding name="mail-smtp">  
    <remote-destination host="localhost" port="25"/>  
  </outbound-socket-binding>  
</socket-binding-group>
```

By default, Keycloak will use **8080** as an http port, **8443** as an https port and **localhost** as host. If you decided use custom **KEYCLOAK_HOST_NAME**, it should be specified as shown below.

```
<server name="default-server">  
  <http-listener name="default" socket-binding="http" redirect-  
socket="https" enable-http2="true"/>
```

```
<https-listener name="https" socket-binding="https" security-  
realm="ApplicationRealm" enable-http2="true"/>  
  <host name="default-host" alias="keycloak.mycomp.any">  
    <location name="/" handler="welcome-content"/>  
    <http-invoker security-realm="ApplicationRealm"/>  
  </host>  
</server>
```

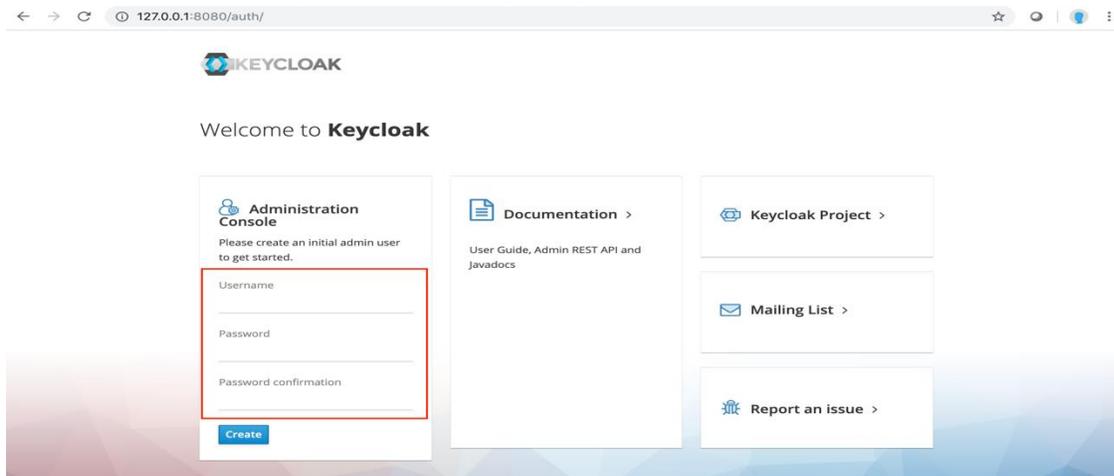
Keycloak Server Configuration

Creating a Realm Admin

This admin user can be thought of as a super admin with all access (realm creation, update, deletion, user creation, update, deletion, etc)

Open a browser and navigate to https://KEYCLOAK_HOST_NAME:8443/auth (or HTTP endpoint).

Note: the browser may show a warning if e.g. certificate does not match the host name

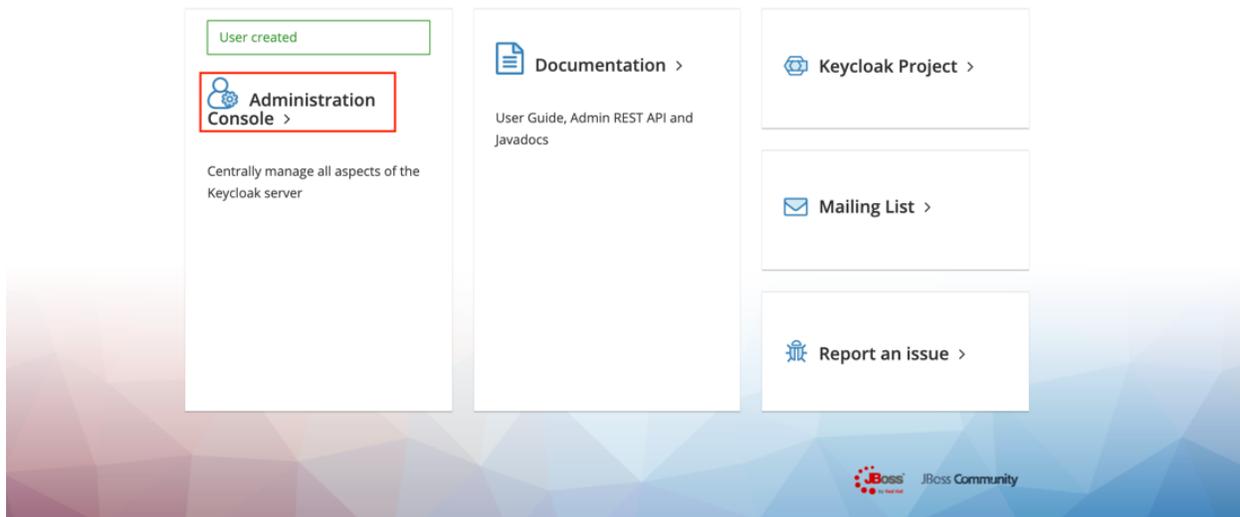


Specify the admin credentials and press create.

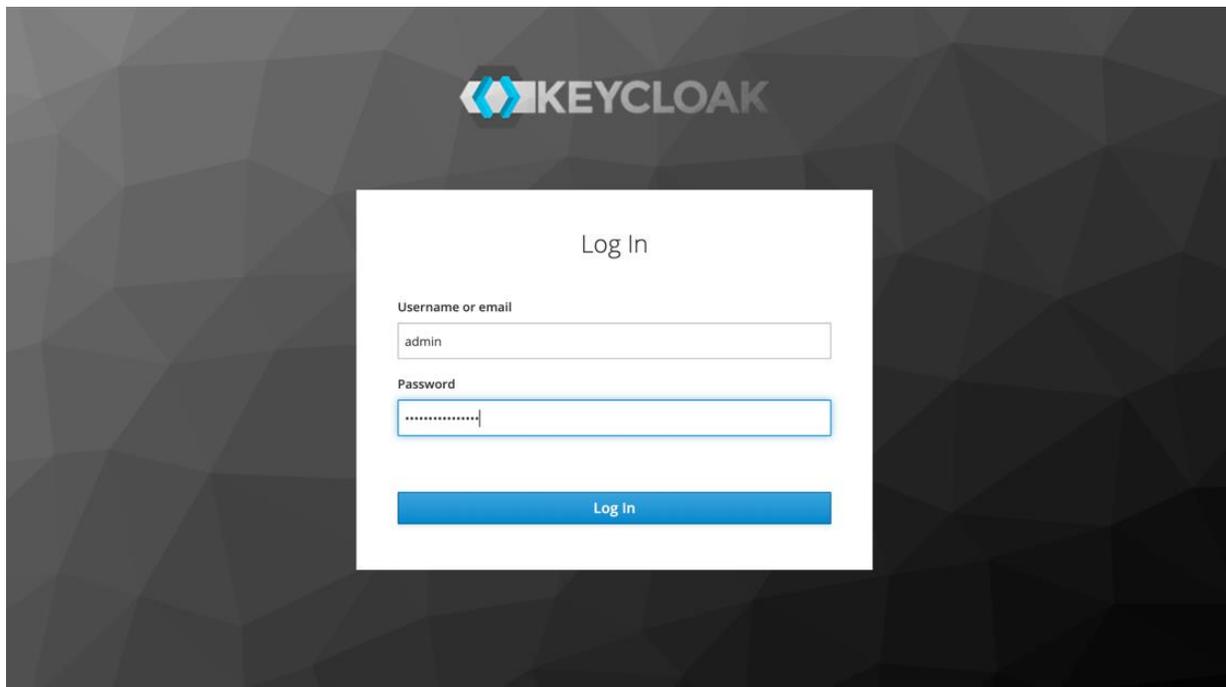
Configuring Master Realm

Navigate to the administration console and login by clicking on the **Administration Console** link as illustrated below

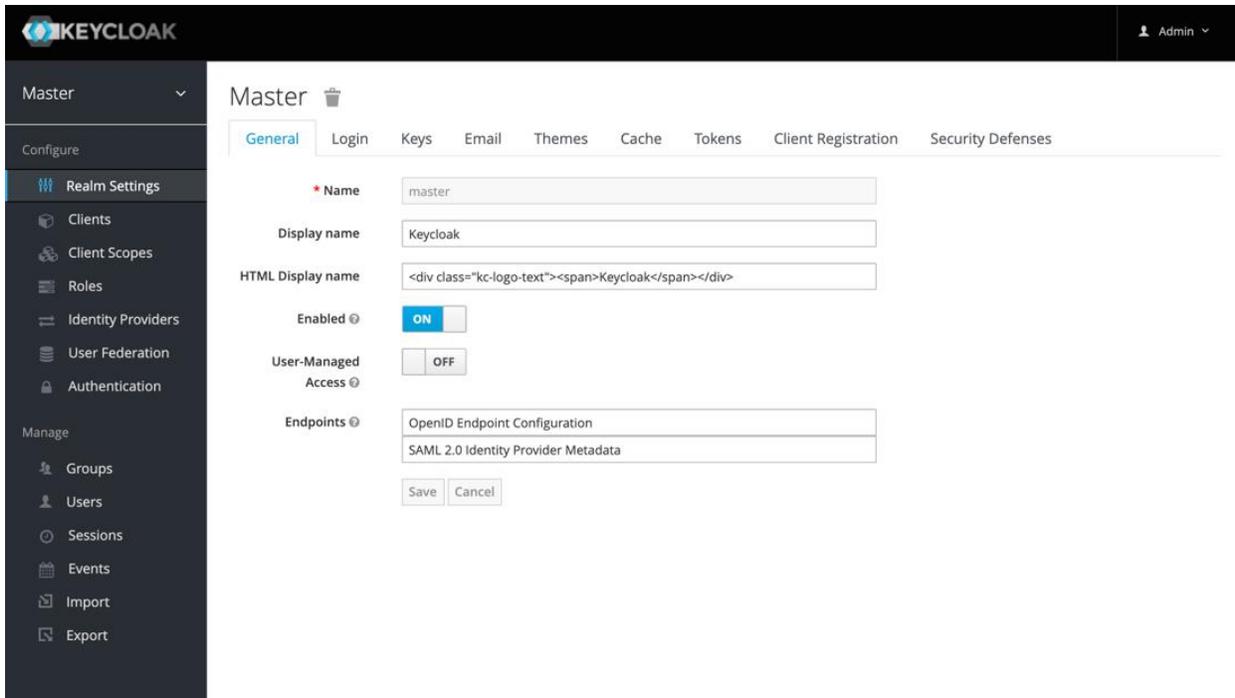
Welcome to **Keycloak**



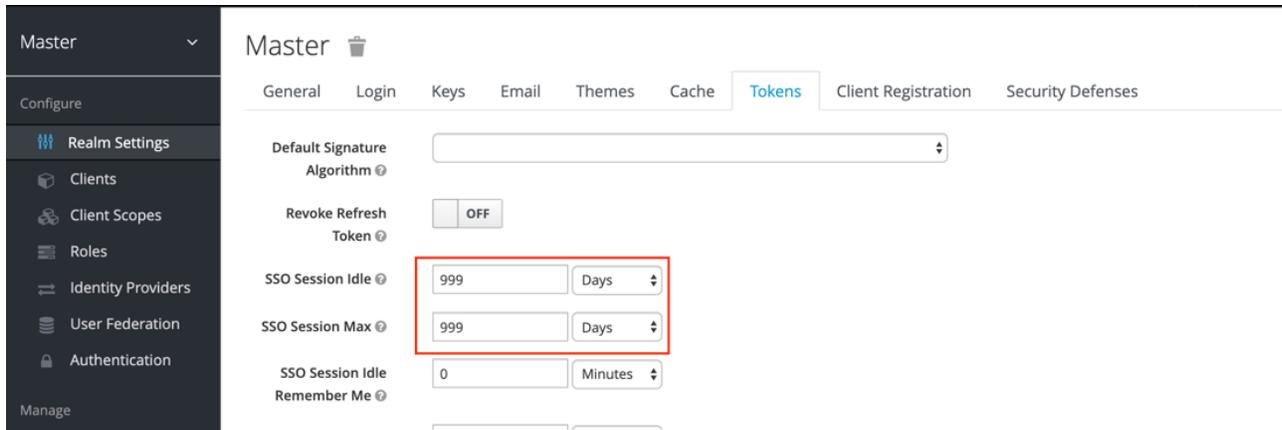
You will be directed to



After a successful login, you should a similar console:



1. Set SSO Session Idle and SSO Session Max to 999 days
 - a. Click on “Tokens” tab
 - b. Set SSO Session Idle and SSO Session Max values to 999 days.



2. Ensure **Access Type** is set to “**confidential**”
 - a. In the left menu, click on “Clients”, then click on “admin-cli” client
3. Activate **Service Accounts Enabled** switch
4. Press **Save**

Master > Clients > admin-cli

Admin-cli

Settings Roles Client Scopes Mappers Scope Revocation Sessions Offline Access Installation

Client ID admin-cli

Name \$(client_admin-cli)

Description

Enabled ON

Consent Required OFF

Login Theme

Client Protocol openid-connect

Access Type confidential

Standard Flow Enabled OFF

Implicit Flow Enabled OFF

Direct Access Grants Enabled ON

Service Accounts Enabled ON

5. Under the **Service Account Roles**
 - a. Add the admin role to the master service account

KEYCLOAK Admin

Master > Clients > admin-cli

Admin-cli

Settings Credentials Roles Client Scopes Mappers Scope Authorization Revocation Sessions

Offline Access Clustering Installation Service Account Roles

admin-cli Service Accounts

Realm Roles Available Roles Assigned Roles Effective Roles

admin offline_access uma_authorization

admin create-realm offline_access uma_authorization

Client Roles

6. Click on the **Settings** tab, scroll down and set token to 999 days under **Advanced Settings**
7. Click **Save**

Direct Access Grants Enabled OFF
 Service Accounts Enabled ON
 Authorization Enabled ON
 Root URL
 Base URL
 Admin URL

> Fine Grain OpenID Connect Configuration ?

> OpenID Connect Compatibility Modes ?

> Advanced Settings ?
 Access Token Lifespan Days

OAuth 2.0 Mutual TLS Certificate Bound Access Tokens Enabled OFF

> Authentication Flow Overrides ?

Save Cancel

Configuring Drone Realm

1. Import the realm data
 - a. Hover over **Master** in top left corner and click on **Add realm**

KEYCLOAK Admin Admin

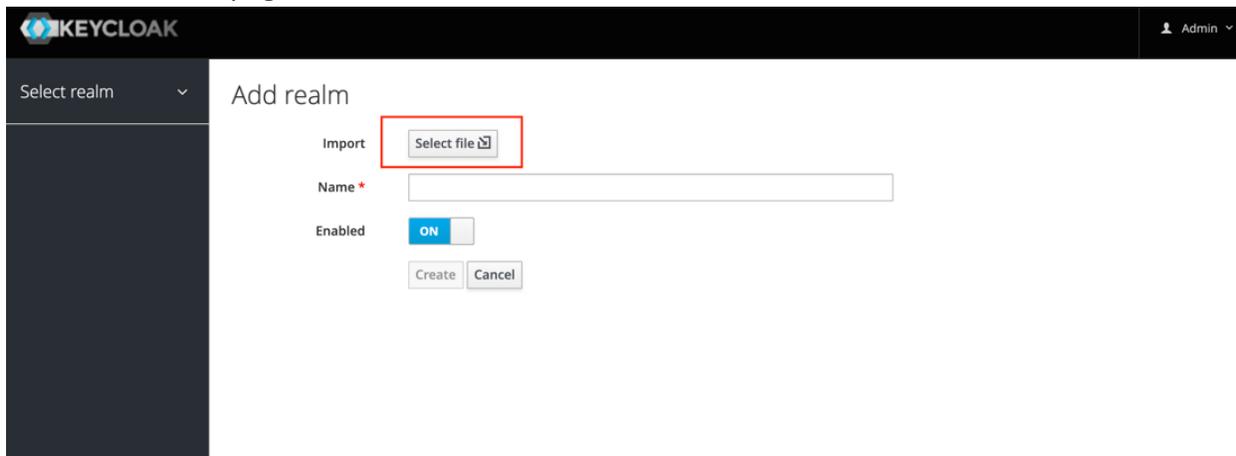
Master Add realm

Master General Login Keys Email Themes Cache Tokens Client Registration Security Defenses

* Name
 Display name
 HTML Display name
 Enabled ON
 User-Managed Access OFF
 Endpoints

Save Cancel

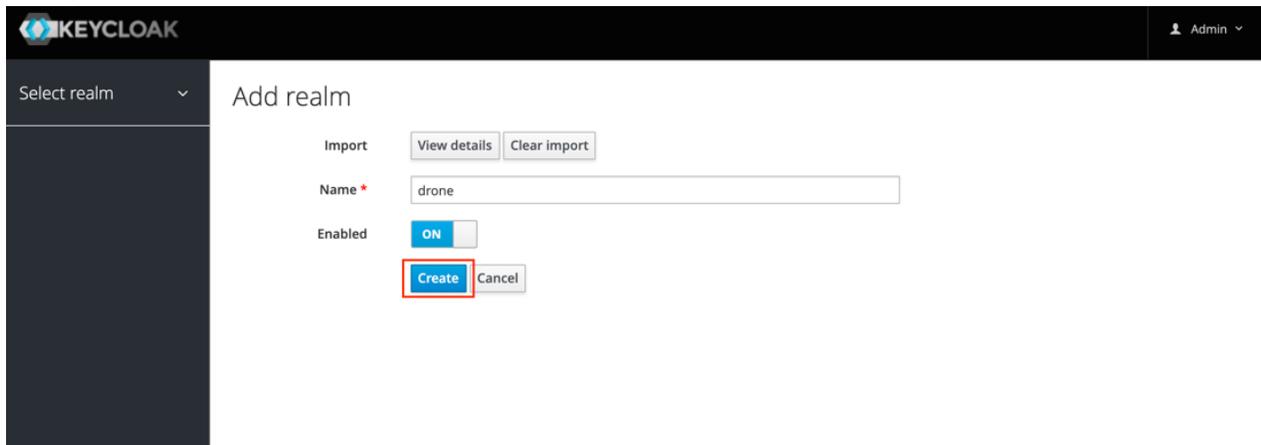
You should see a page similar to the one below.



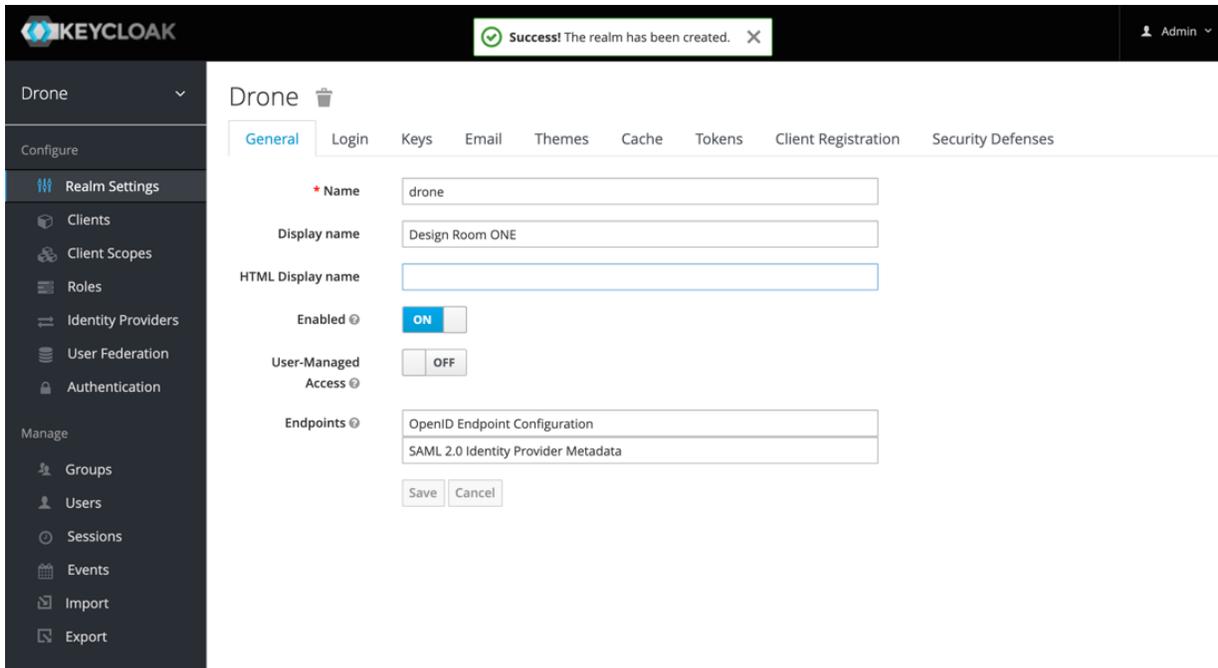
Click on **Select File** and point to **drone-realm-export.json** file. It can be found in machine where **Design Room ONE** is installed under following path:

DR_ONE_INSTALL_DIR/DR_Install/Resources/Keycloak/drone-realm-export.json

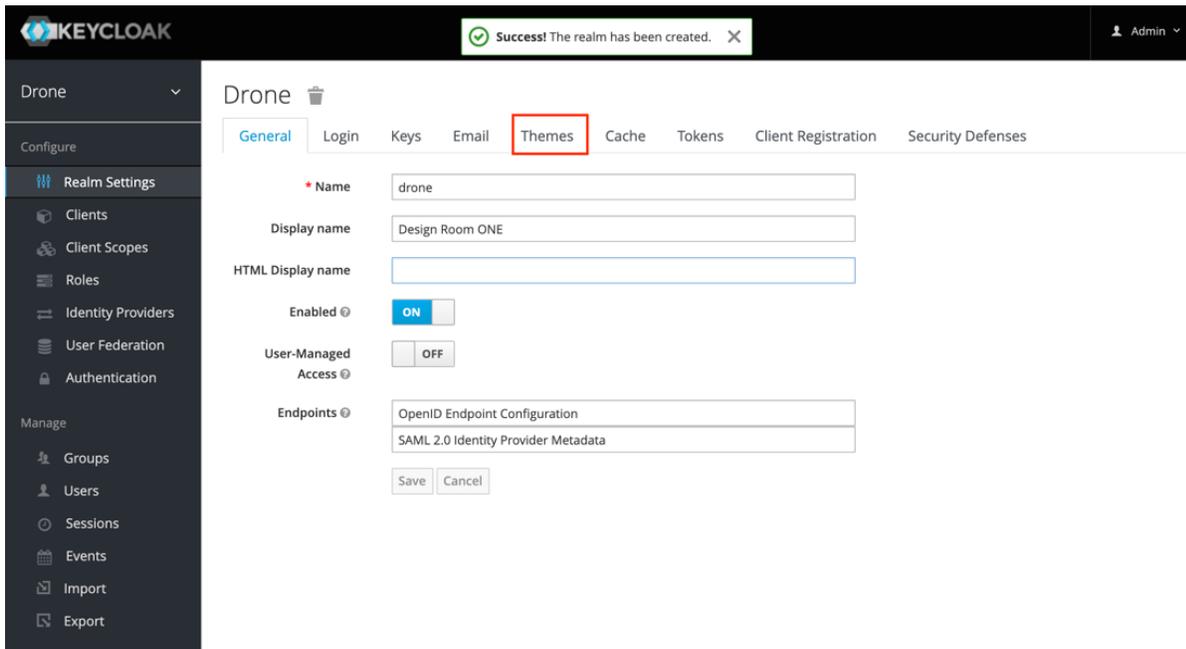
Then, click **Create**



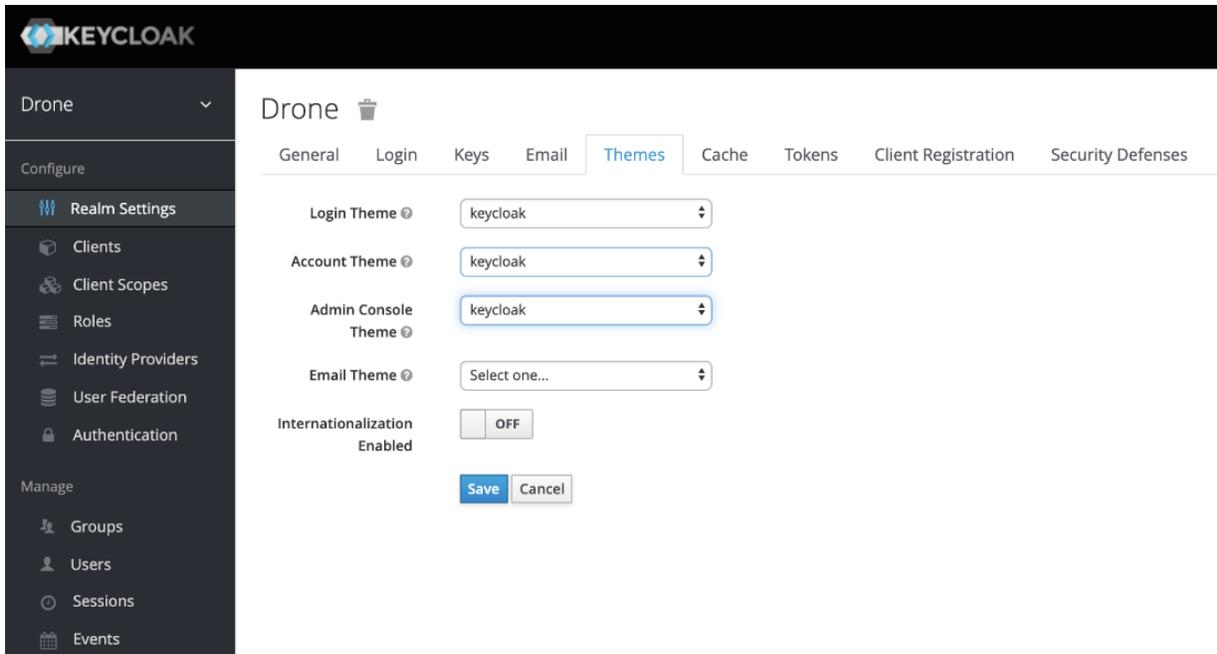
Your **Drone** realm should be created successfully as shown below:



2. Assign the proper theme
 - a. Click on the **Themes** tab

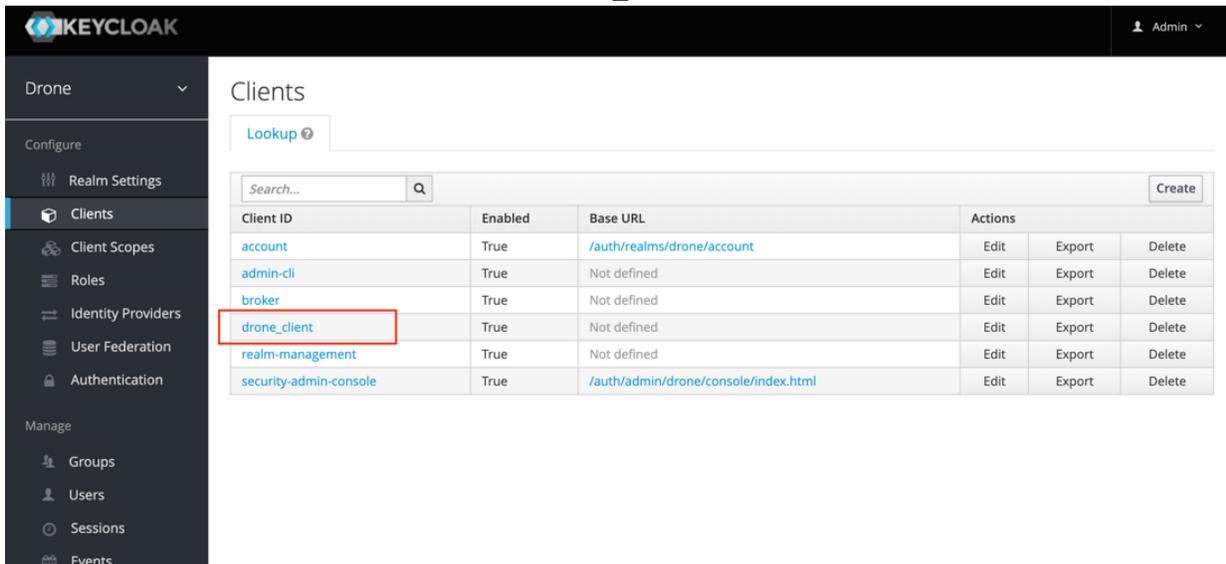


- b. Assign keycloak themes as shown below and **Save**.



3. Setup drone_client

- a. Under **Clients** click on the drone_client link



- b. Click on the **Settings** top menu, scroll down and add **DRONE_HOST_NAME** (or ip address) accessible by other machines for the server Keycloak as valid redirect URIs as in example below. It is recommended to use lowercase letters.

Direct Access Grants Enabled ?

Root URL ?

https://drone.mycomp.any:10101/dr*	-
	+

* Valid Redirect URIs ?

Base URL ?

Admin URL ?

	-
	+

Web Origins ?

> Fine Grain OpenID Connect Configuration ?

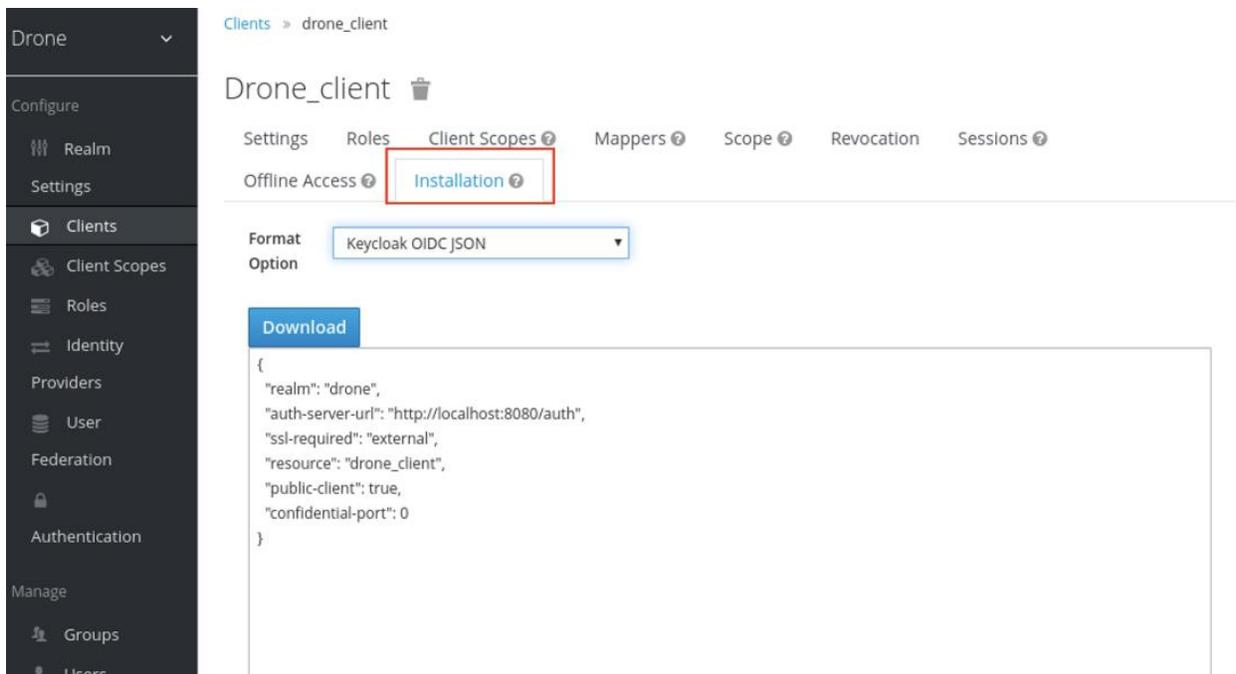
> OpenID Connect Compatibility Modes ?

> Advanced Settings ?

> Authentication Flow Overrides ?

Save	Cancel
------	--------

- c. Under **Installation** menu, download the config file. The file will be downloaded as keycloak.json. Do not change the name and move the file to Design Room ONE machine under **DR_ONE_INSTALL_DIR\OnPrem_Design_Room\config** folder



Important: After the keycloak.json file is moved, open it to ensure correct **KEYCLOAK_HOST_NAME** (or ip-address) is specified in **auth-server-url** attribute and the URL is accessible by machine where Design Room ONE server is installed and from user machines. It is recommended to use lowercase letters in the URL.

```

{
  "realm": "drone",
  "auth-server-url": "https://keycloak.mycomp.any:8443/auth",
  "ssl-required": "external",
  "resource": "drone_client",
  "public-client": true,
  "confidential-port": 0
}

```

Setting up Users and Roles

Creating Users

Before we create a user, you can notice that some roles were created by default in Keycloak notably the ones arrowed.

Keycloak Roles page showing a list of roles. The 'Roles' menu item in the sidebar is highlighted with a red arrow. The table below shows the roles and their descriptions:

Role Name	Composite	Description	Actions
drone_user	False	A user in DRONE must have this role to login	Edit Delete
offline_access	False	\$(role_offline-access)	Edit Delete
sample_all_access	False	This role gives read and write access to all designs.	Edit Delete
sample_partial_access	False	This role will give read access to all design starting with traffic	Edit Delete
uma_authorization	False	\$(role_uma_authorization)	Edit Delete

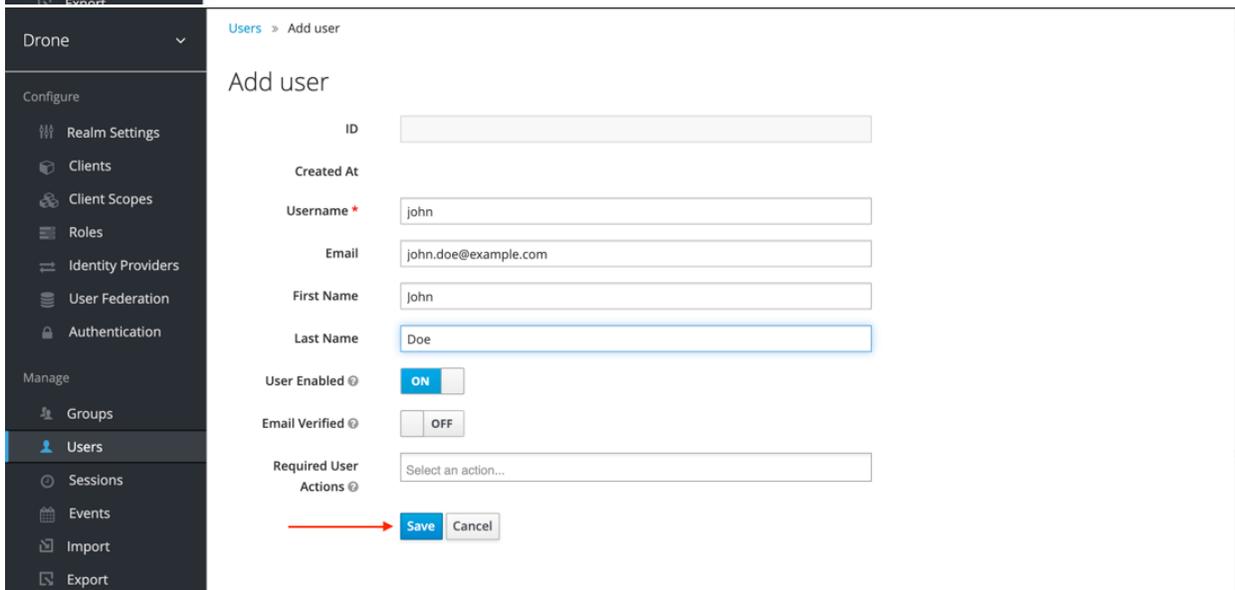
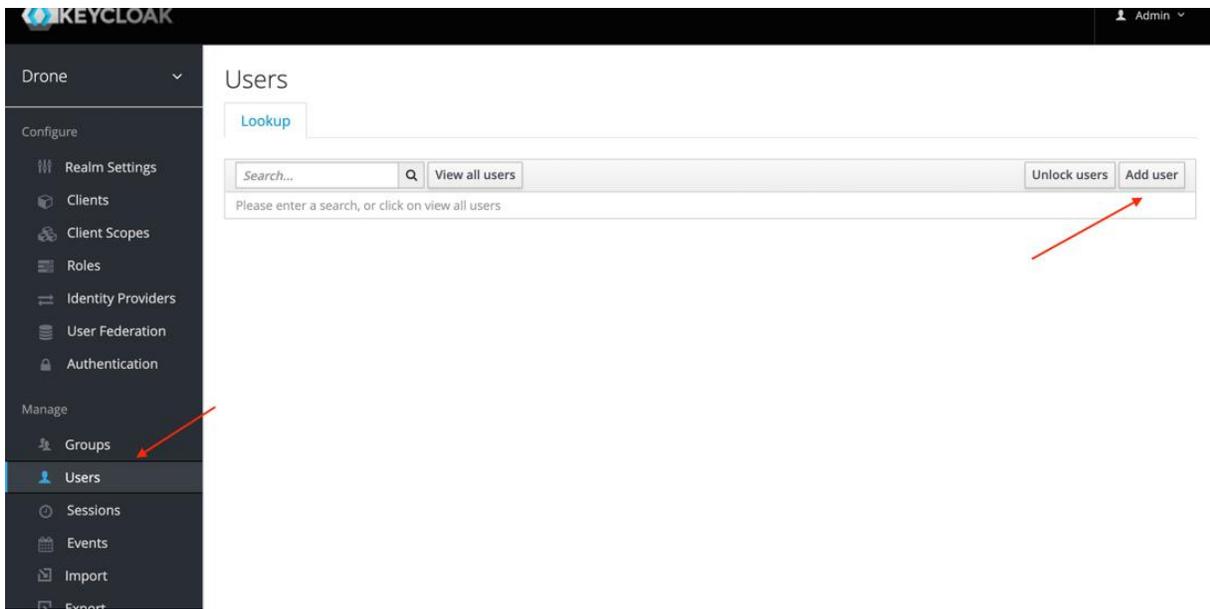
Note: **drone_user** is also a default role which means that every new user will by default inherit this role as shown below

Keycloak Roles page showing the 'Default Roles' configuration. The 'Default Roles' tab is selected. The 'Realm Default Roles' list includes the following roles:

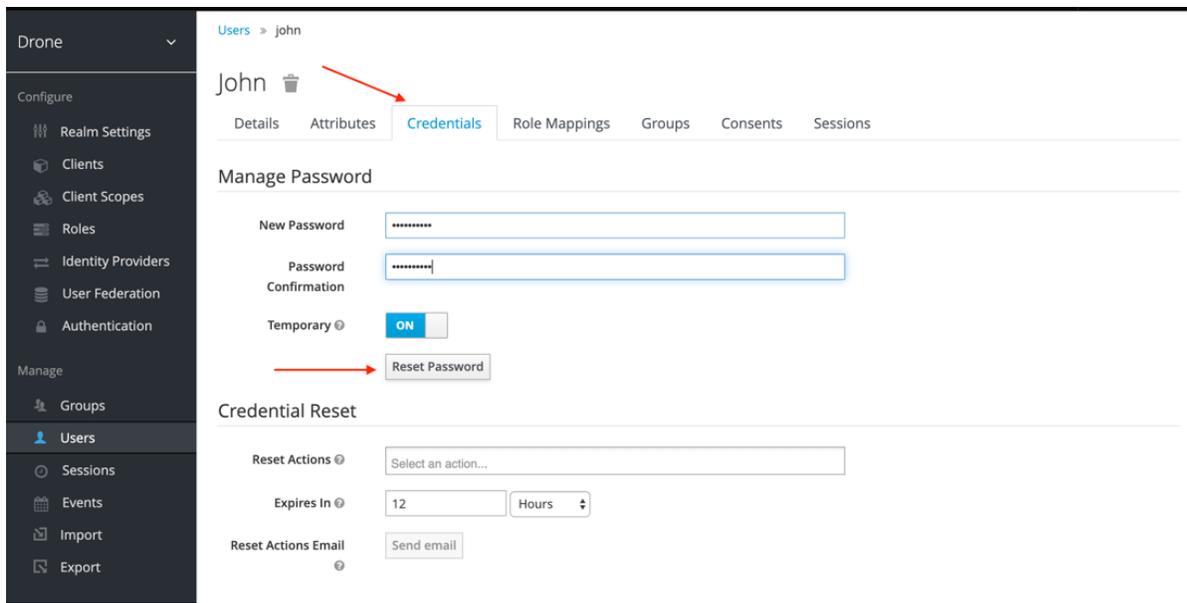
- drone_user
- offline_access
- uma_authorization

A red arrow points to 'drone_user' in the 'Realm Default Roles' list.

1. Create a user



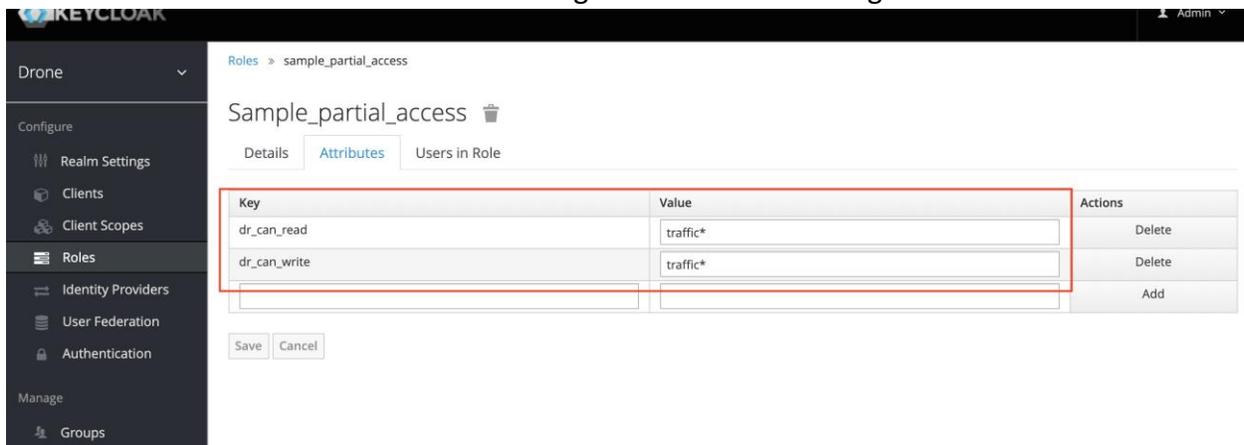
2. Setup the user password and click **Reset Password**



Managing Access with Roles

Access to designs in Design Room ONE is controlled with special attributes that can be specified for a role in Keycloak. The **dr_can_read** and **dr_can_write** attributes of roles give users respectively read and write access to specific designs. These attributes support wildcard as shown in the picture below.

Under **Roles > sample_partial_access > Attributes** you can see that the **dr_can_read** and **dr_can_write** attributes are both set to "traffic*" which means that users or groups with this role will be able to read and write to all designs with names starting with "traffic".



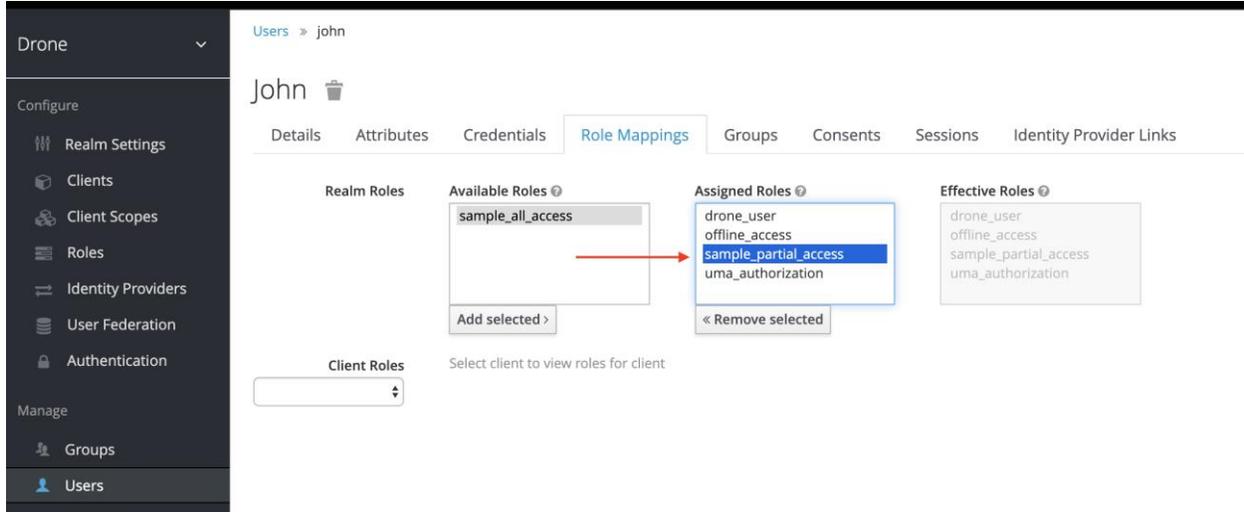
It is possible to specify several alternatives by using | character, e.g. the following value

traffic*|*_secretproject|*alice*

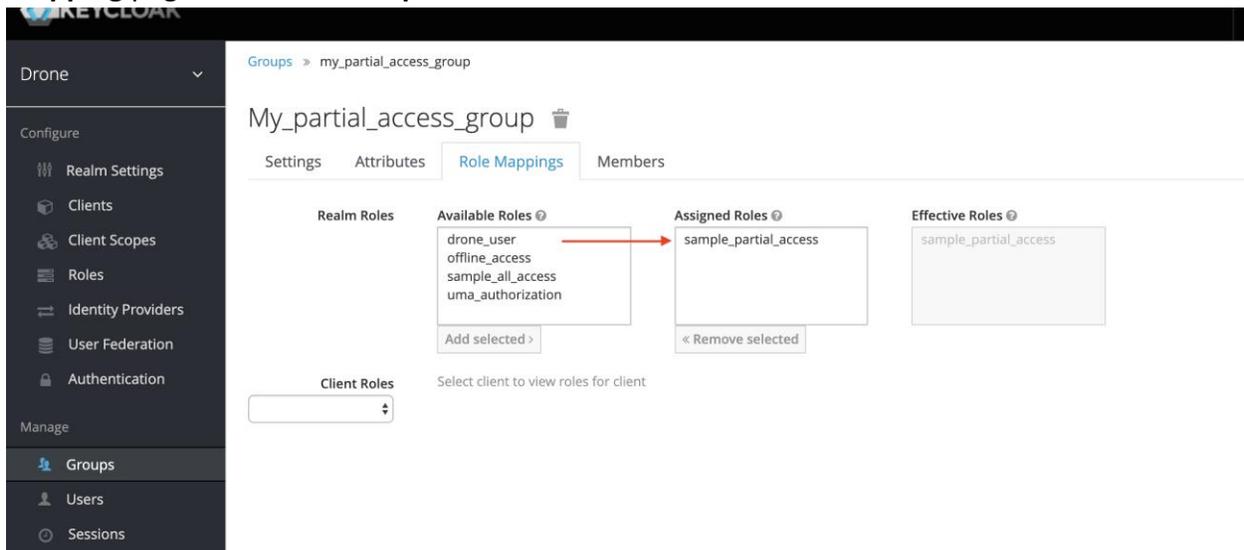
Will give read or write access (depending on the attribute it is specified in) to designs, which

name either starts with **traffic**, or ends with **_secretproject** or contains the word **alice**. Note that, design names are case sensitive.

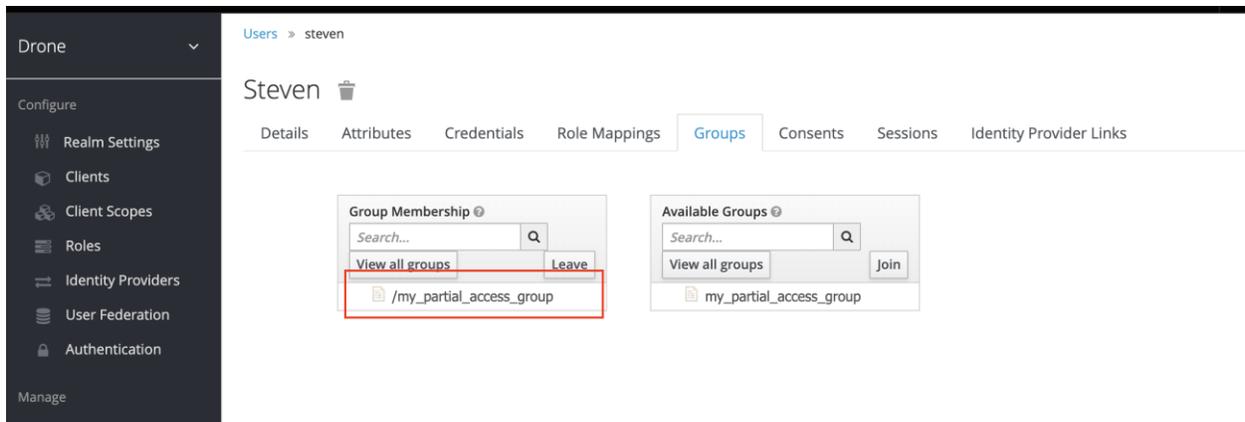
To explicitly assign a role to a user, **Role Mapping** page under the **Users** left menu can be used.



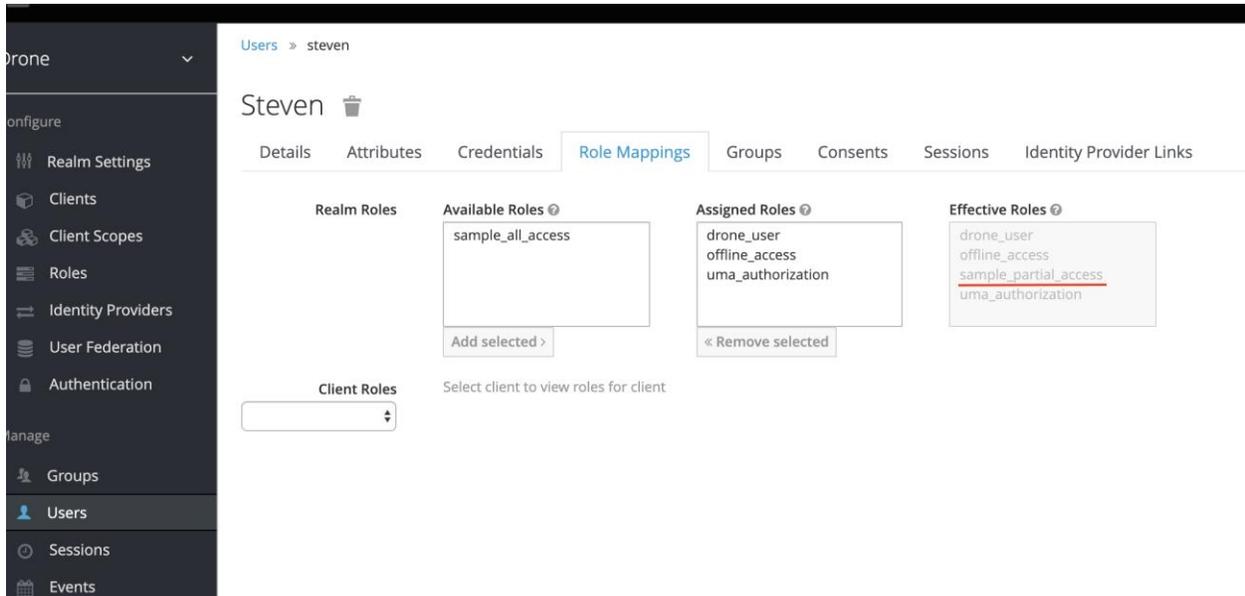
A user can implicitly get role from the groups they belong. To assign a role to a group **Role Mapping** page under the **Groups** left menu can be used.



Note: A user will inherit all the roles from the group the belong
In the example below user **Steven** belongs to **my_partial_access_group**



This means user **Steven** automatically inherits **sample_partial_access** role as it is assigned to the group.

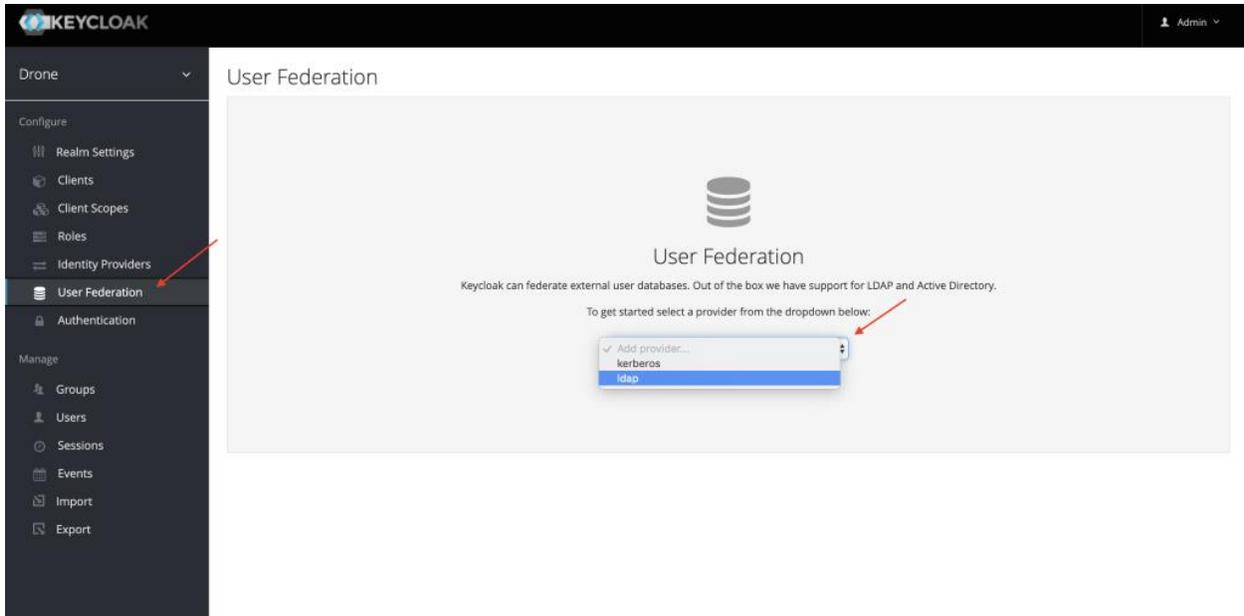


This results any **my_partial_access_group** member including user **Steven** being able to read and write all designs with names starting with “traffic” because he belongs to the group membership.

Importing Users from Active Directory

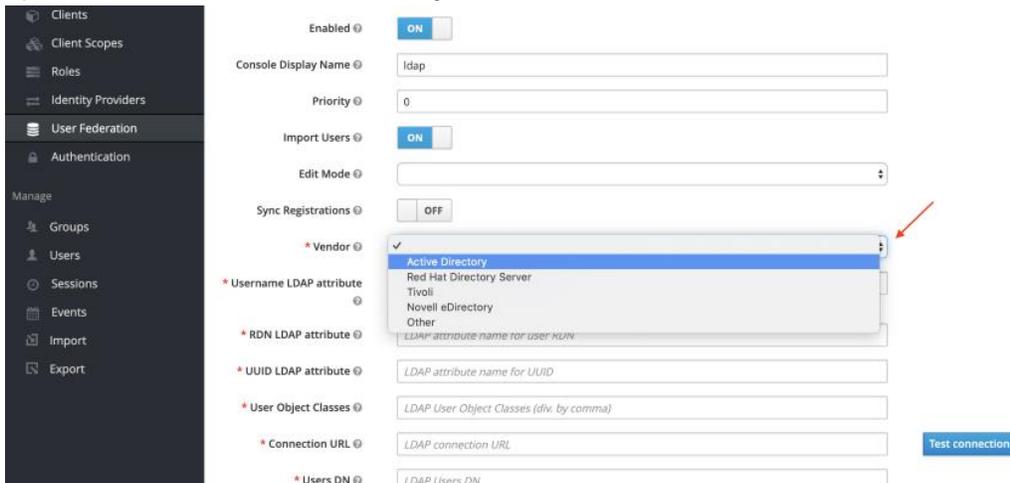
Note: The Active directory server reference is at the end of the file.

- 1) Ensure you are in the **Drone** realm and click on **User Federation** left menu item
Then select **Idap** from the dropdown



Note: We will go with basic active directory setup here. All the page options are configurable with tooltip information to allow for more advanced setups if one chooses to.

2) Set Vendor to Active Directory



You should see some fields pre-filled with default values as shown below:

* Vendor ?

* Username LDAP attribute ?

* RDN LDAP attribute ?

* UUID LDAP attribute ?

* User Object Classes ?

* Connection URL ?

* Users DN ?

* Bind Type ?

Test connection

2) Set the connection url and test the connection via the **Test connection** button as shown below:

User Federation

Authentication

Groups

Users

Sessions

Events

Import

Export

Import Users ON ✔ Success! LDAP connection successful. ✕

Edit Mode

Sync Registrations OFF

* Vendor ?

* Username LDAP attribute ?

* RDN LDAP attribute ?

* UUID LDAP attribute ?

* User Object Classes ?

* Connection URL ?

* Users DN ?

* Bind Type ?

Enable StartTLS ? OFF

Test connection ↗

3) Set the active directory users database here

* Users DN ?

* Bind Type ?

Enable StartTLS ? OFF

4) Set the Active Directory (AD) admin credentials and test authentication to the AD server as shown below:

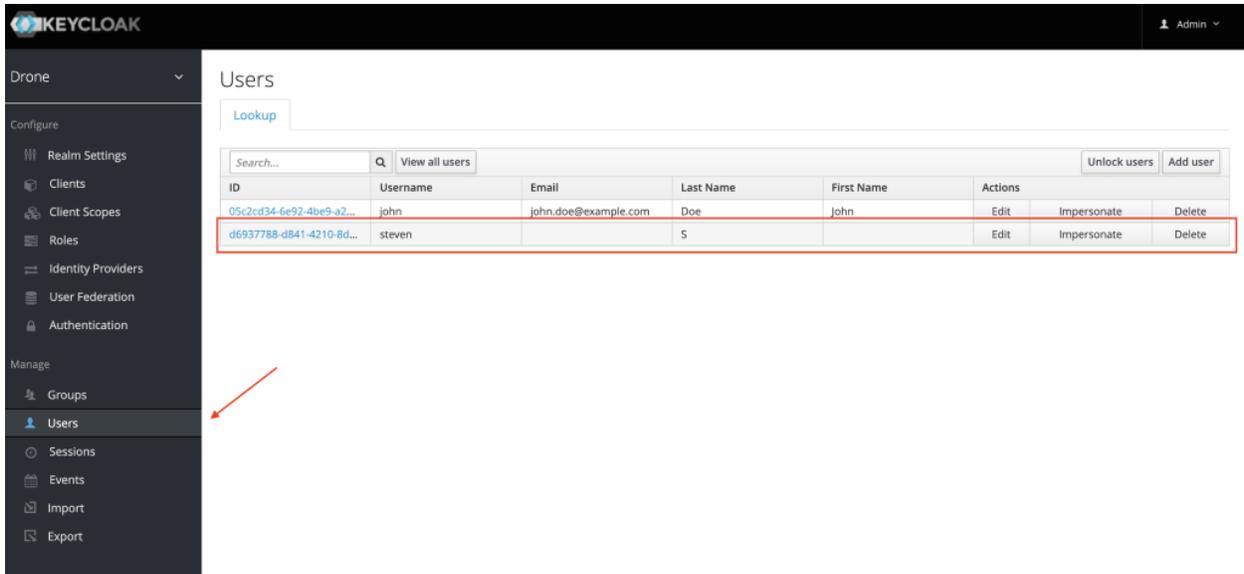
* Connection URL Success! LDAP authentication successful. X Test connection
 * Users DN
 * Bind Type
 Enable StartTLS
 * Bind DN Test authentication
 * Bind Credential
 Custom User LDAP Filter

5) Leave the rest as is and scroll down and click **Save**, then click on **Synchronize all users** to import all existing users from AD to Keycloak

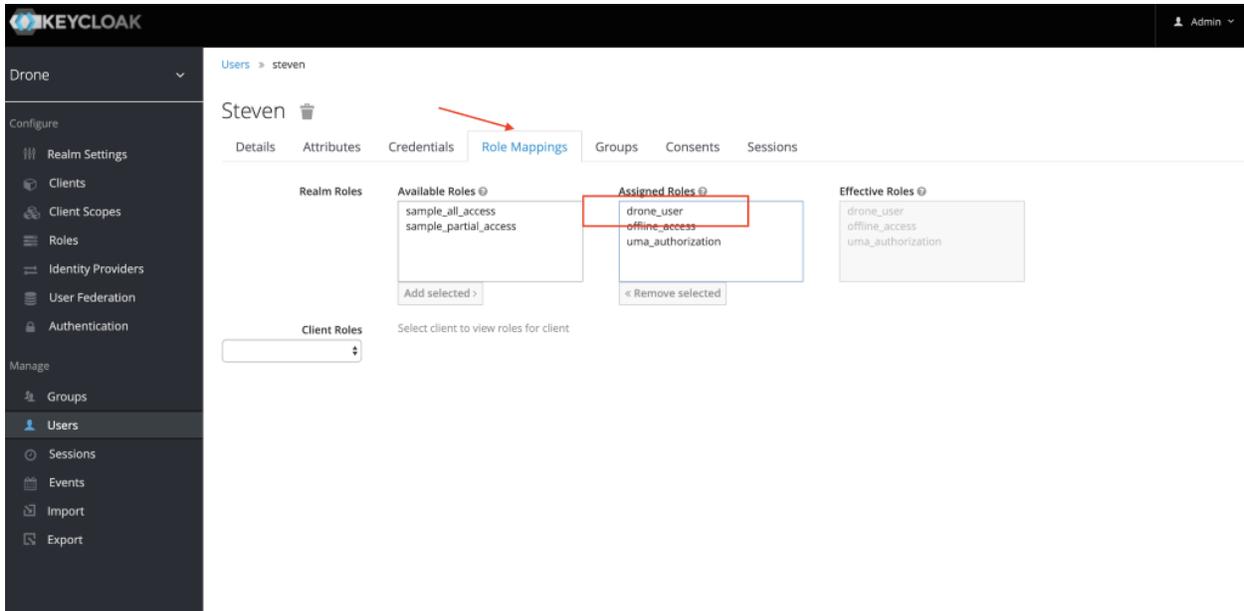
Connection Timeout
 Read Timeout
 Pagination
Kerberos Integration
 Allow Kerberos authentication
 Use Kerberos For Password Authentication
Sync Settings
 Batch Size
 Periodic Full Sync
 Periodic Changed Users Sync
Cache Settings
 Cache Policy
Save Cancel Synchronize changed users Synchronize all users remove imported Unlink users

In our case 1 user was imported from our active directory server.

6) If we now click on the Users menu, we see our imported user.



7) Click on the user and observe the role mappings. The user should inherit all default roles (including drone_user) to have access to the Design Room ONE server



8) You can now login into Design Room ONE with the newly created user.

LDAP Directory Information Tree data reference

```
#
# LDAPv3
# base <dc=hcl,dc=com> (default) with scope subtree
# filter: (objectclass=*)
# requesting: ALL
#
# hcl.com
dn: dc=hcl,dc=com
objectClass: top
objectClass: dcObject
objectClass: organization
o: hcl
dc: hcl

# admin, hcl.com
dn: cn=admin,dc=hcl,dc=com
objectClass: simpleSecurityObject
objectClass: organizationalRole
cn: admin
description: LDAP administrator

# users, hcl.com
dn: ou=users,dc=hcl,dc=com
objectClass: organizationalUnit
ou: users

# steven, users, hcl.com
dn: cn=steven,ou=users,dc=hcl,dc=com
cn: steven
sn: US
objectClass: organizationalPerson

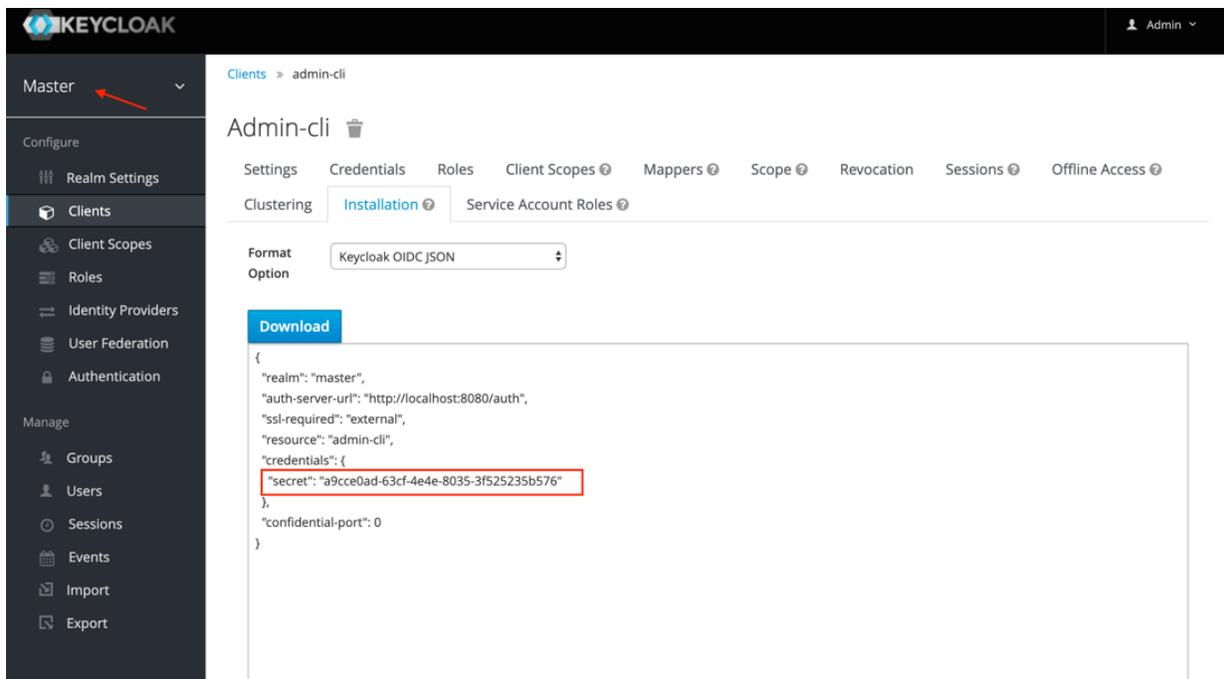
# search result
search: 2
result: 0 Success

# numResponses: 5
# numEntries: 4
```

Summary: Hcl.com(organization)->users(organization unit or OU)->steven(Member of OU)

Configuring Design Room ONE Server

Under the **Master Realm**> Clients> Admin-cli>Installation, copy secret key



Uncomment and paste the value for **kc_admin_secret** in your Design Room ONE server configuration file (DR_ONE_INSTALL_DIR > OnPrem_Design_Room > config > server-config.json) as shown below:

```
// Number of milliseconds to wait for an ongoing server request to complete before shutting
// down the server.
"dr_shutdown_timeout": 5000,

// Authentication (for accessing information stored in Design Room ONE)
// "none": Do not use any authentication. Everyone can access all designs.
// "jazz": Use Jazz authentication. User needs to be logged in to Jazz to access designs.
// "keycloak": Use Keycloak authentication. User needs to be logged via keycloak to access designs.
"dr_auth": "keycloak",

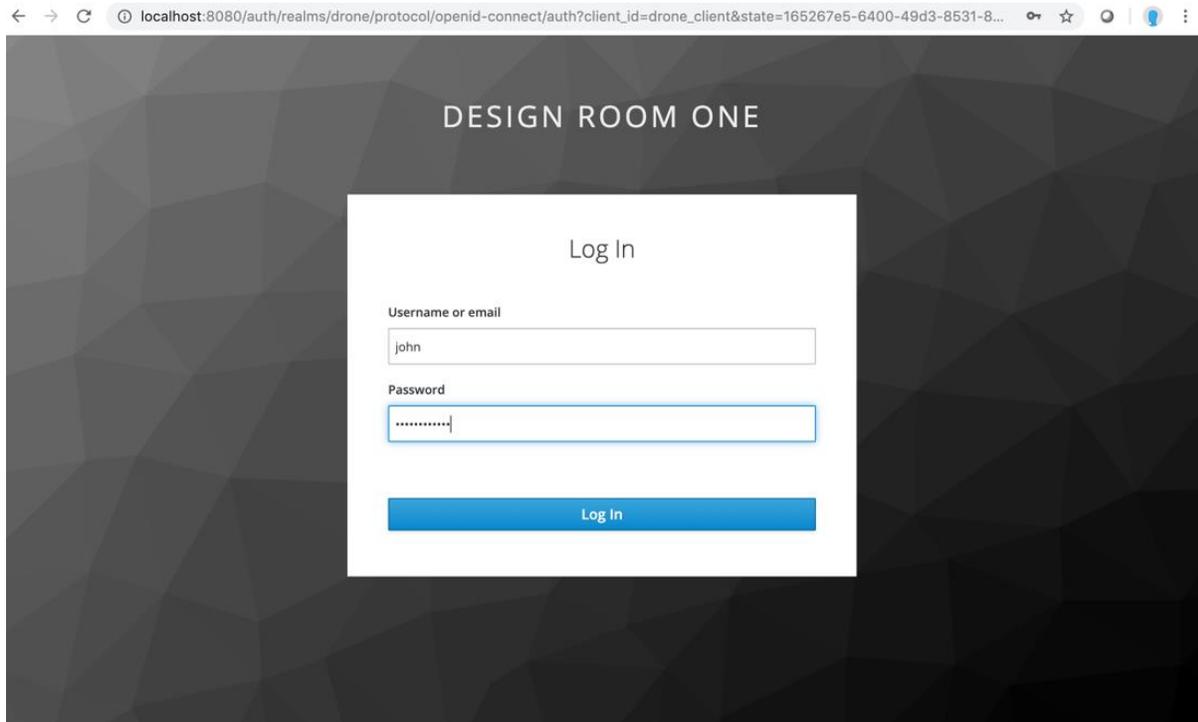
//this secret key MUST be set when using keycloak
"kc_admin_secret": "a9cce0ad-63cf-4e4e-8035-3f525235b576",
```

Ensure that you use **“keycloak”** as value for **“dr_auth”** to use keycloak for authentication. You have now successfully configured Design Room ONE.

Starting the Design Room ONE Server

Ensure Design Room ONE server is started. Navigate to your Design Room ONE installation and launch the dr-deploy.js script.

Logging into the Design Room ONE Server with the New User



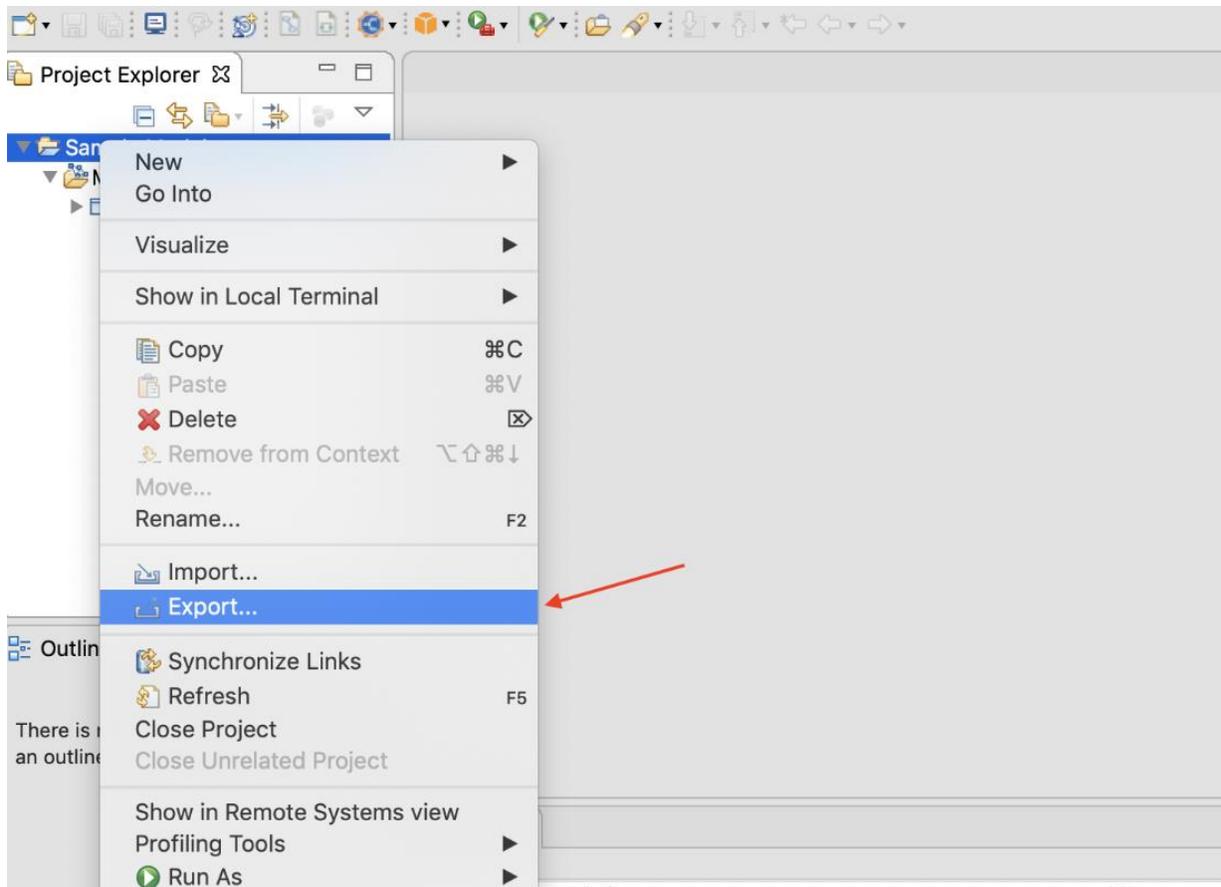
Exporting Models with Design Room ONE Integration Plugin

Below, we will highlight the steps to login in the eclipse client once Keycloak is enabled on the Design Room ONE server.

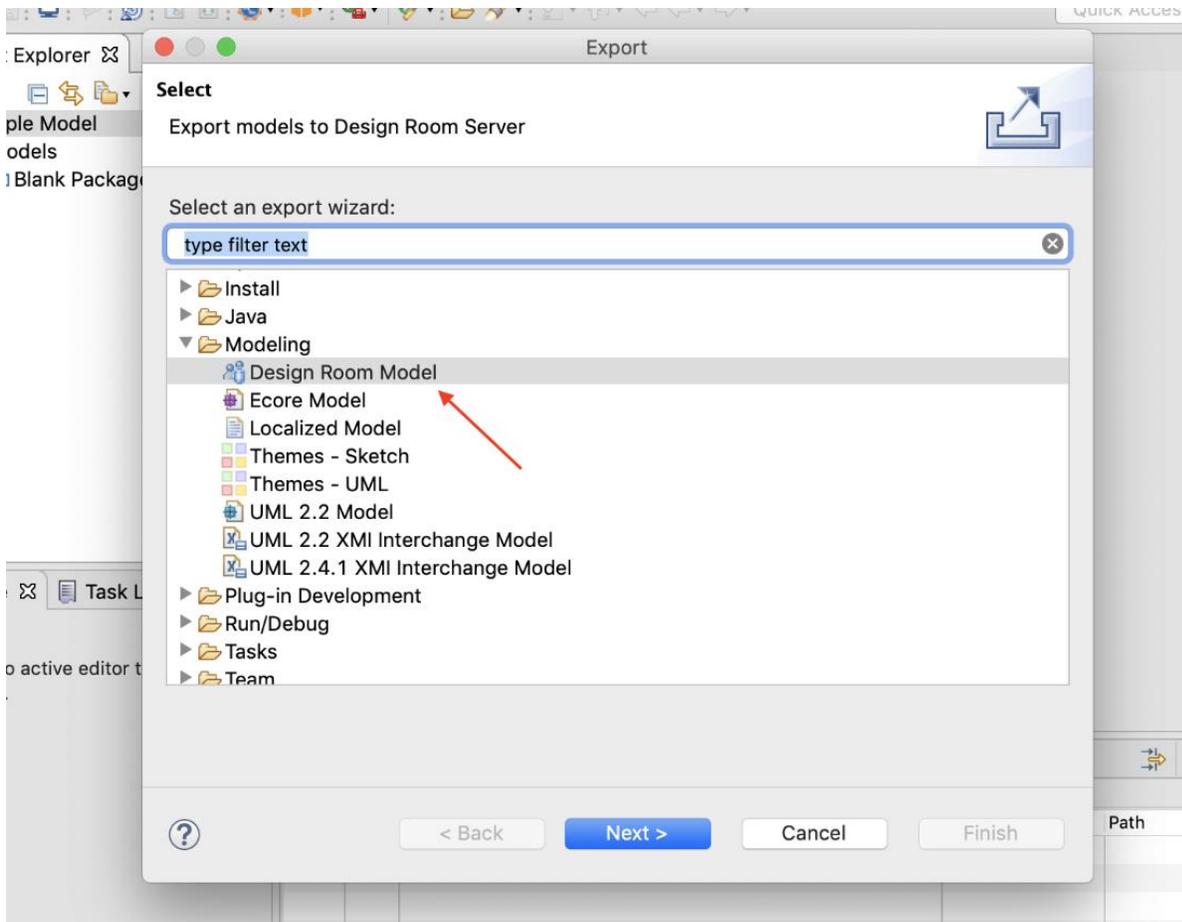
Prerequisites:

1. Install the Design Room ONE Integration feature in your modeling software by following the steps in the Design Room ONE installation document
2. Then ensure that the Design Room ONE server is started successfully.

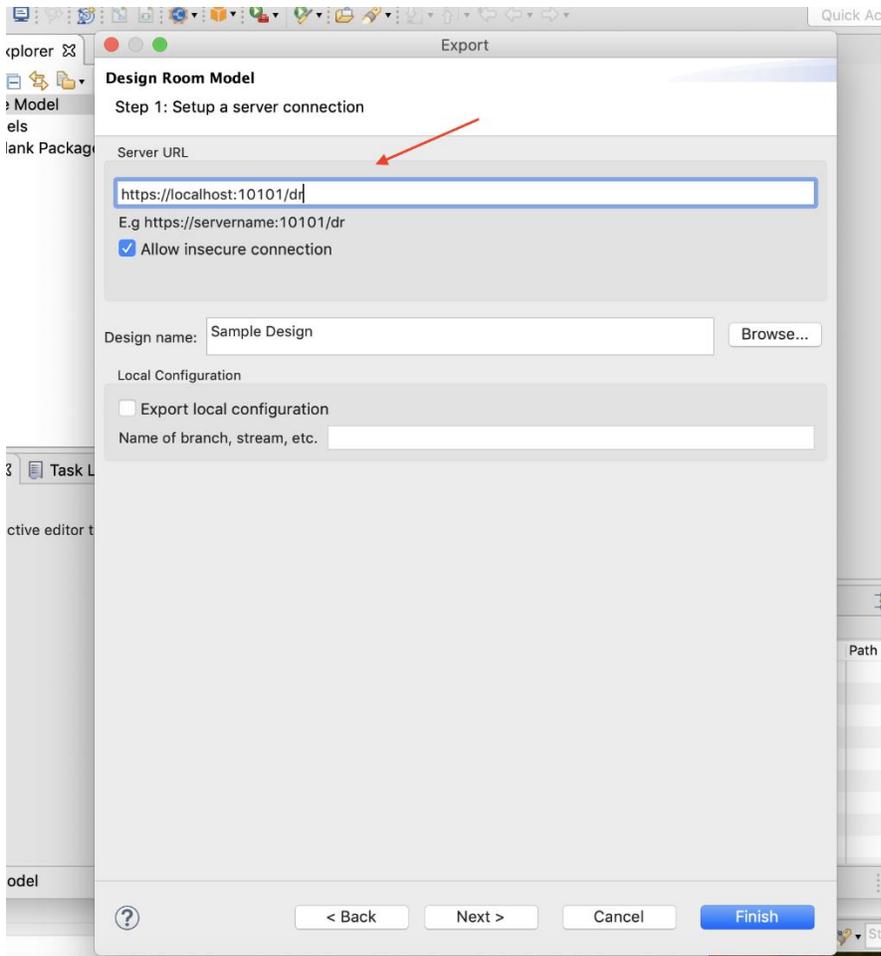
Right click on a project you want to export and select **Export**



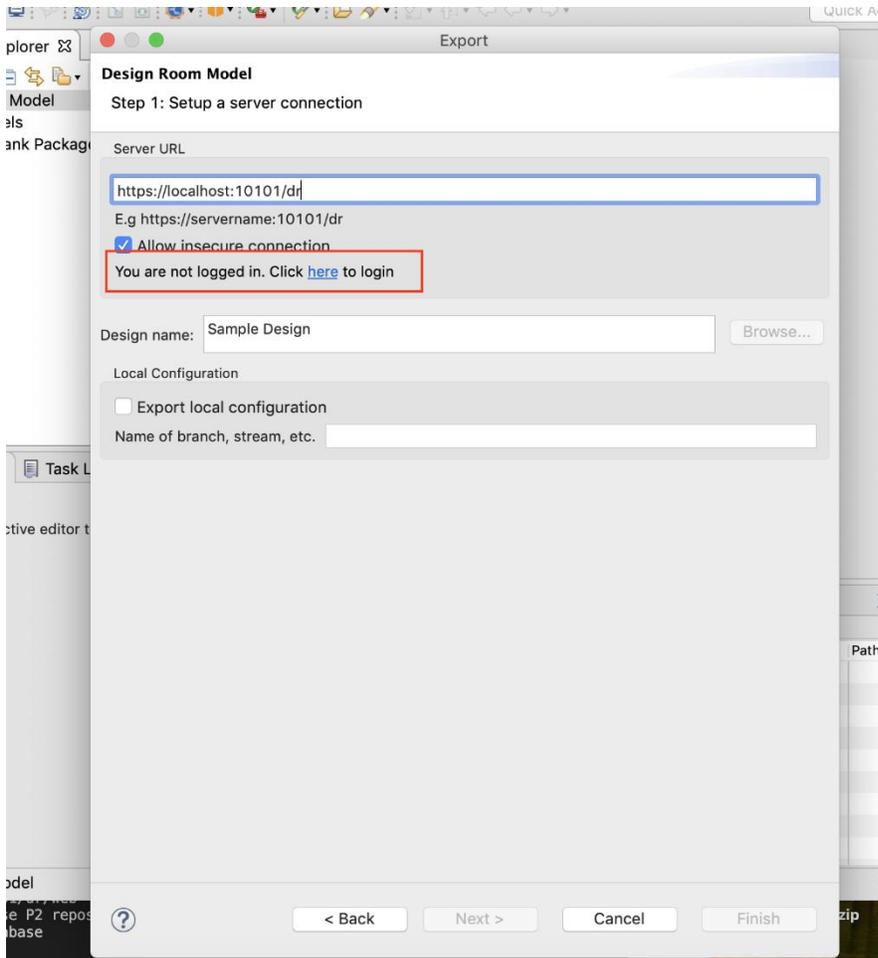
Under the **Modeling** folder, select **Design Room Model**



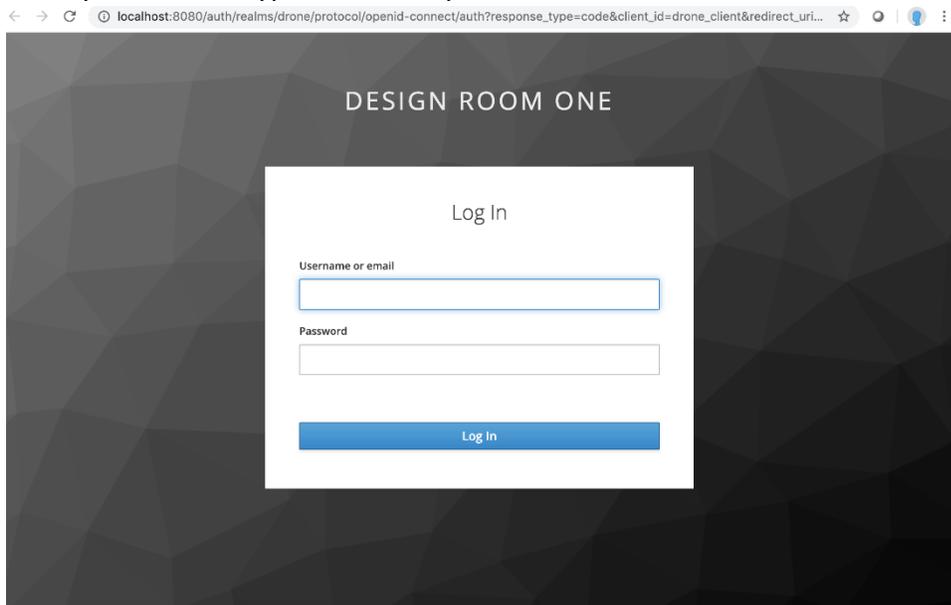
Then enter the server URL for your Design Room ONE server.



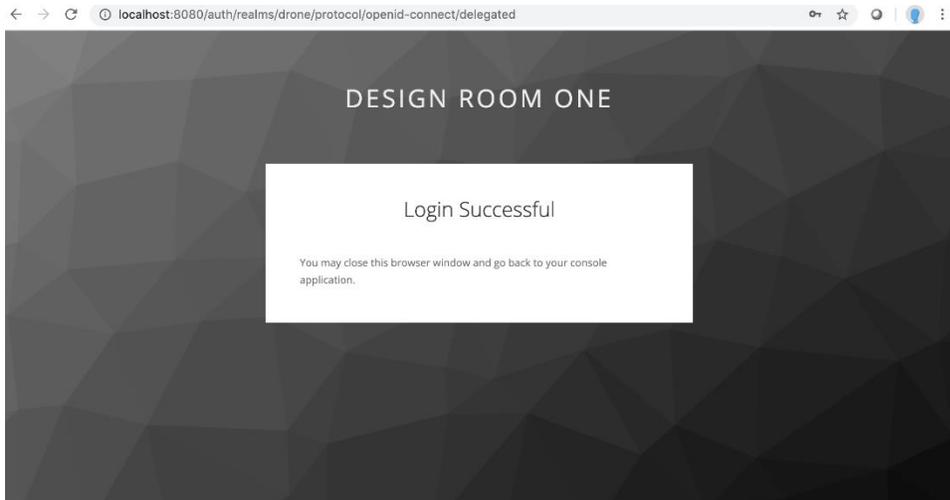
If authentication is enabled on the server and the URL is valid, you will see the message with a link to login.



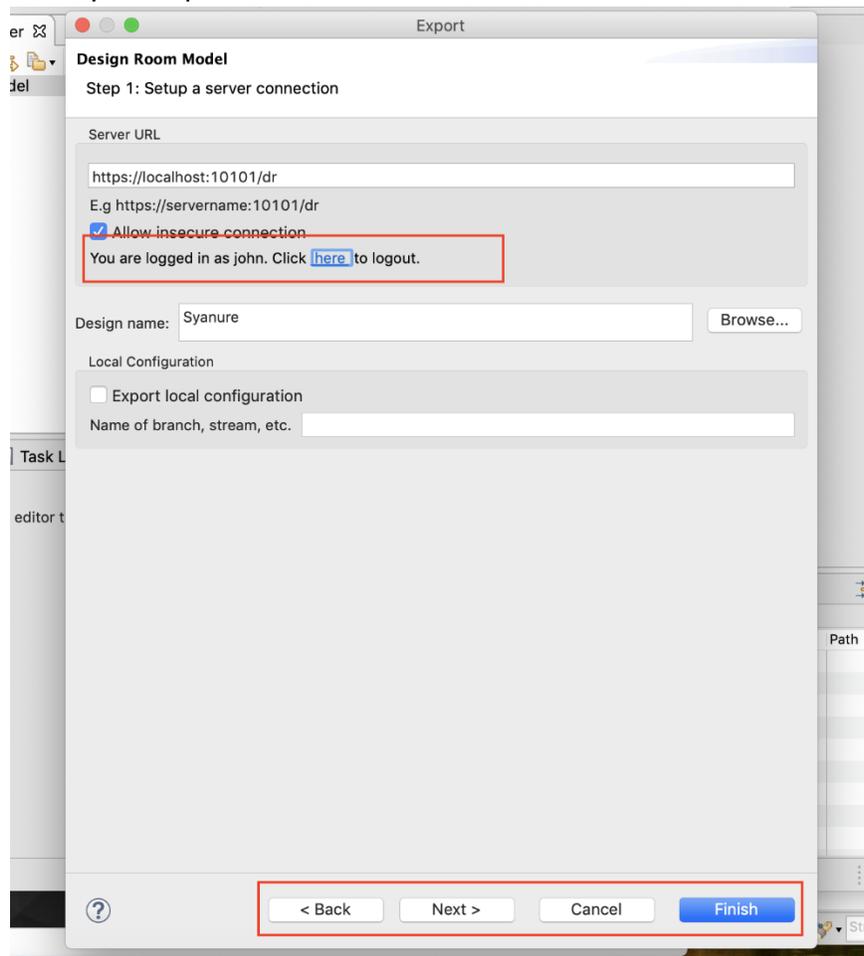
Once you click on hyperlink **here**, you will be redirected to the native browser to login



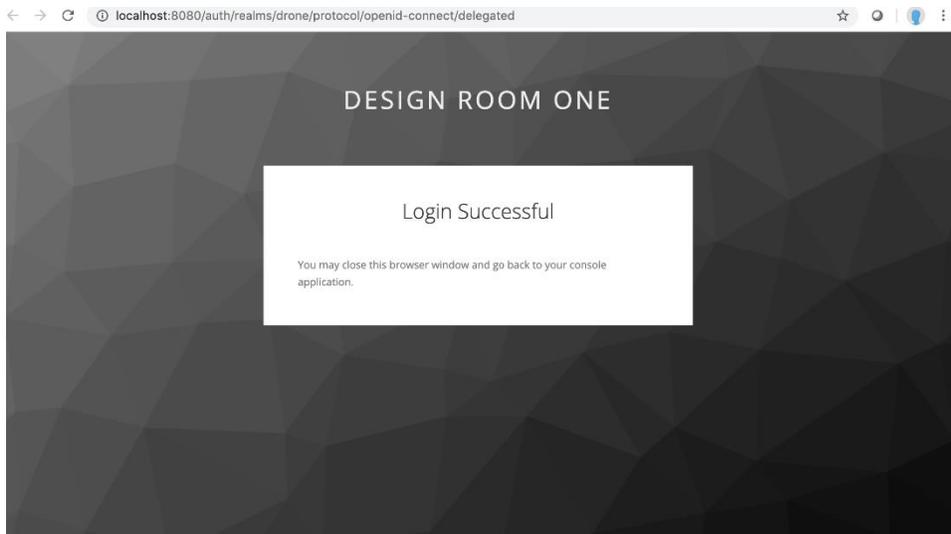
After a successful login, you will see the message



Once you go back to your modeling tool, you will see that you are logged in and can proceed to make your export



To logout, you can click on the **here** link again and the window below will open in your native browser.



Single Sign-On with Jazz Authorization Server via OpenID

Prerequisites

1. Ensure the [Jazz Authorization Server \(JAS\)](#) is installed
2. Ensure the Keycloak 7.0.1 Server installed

Creating a New Identify Provider in Keycloak

1. Login into Keycloak server as admin
2. Click on **DRONE** realm on the top left
3. Click on **Identity Providers** left menu open
4. Under the dropdown, select **OpenID Connect v1.0** as shown below

The screenshot shows the Keycloak administration interface. The left sidebar contains a navigation menu with 'Identity Providers' highlighted. The main content area is titled 'Identity Providers' and features a table with the following data:

Name	Provider	Enabled	Hidden	Link only	GUI order
JAServer	oidc	True	False	False	

A dropdown menu is open from the top right of the table, showing a list of provider options. 'OpenID Connect v1.0' is selected and highlighted with a red box. Other options include 'User-defined', 'SAML v2.0', and various social media providers like GitHub, Twitter, Facebook, etc.

5. Create the alias name of your choice and use the redirect URI below to proceed in the next steps

Drone

Identity Providers > Add identity provider

Add identity provider

Redirect URI

* Alias

Display Name

Enabled

Store Tokens

Stored Tokens Readable

Trust Email

Account Linking Only

Hide on Login Page

GUI order

First Login Flow

Post Login Flow

OpenID Connect Config

* Authorization URL

Pass login_hint

Pass current locale

* Token URL

Logout URL

Backchannel Logout

Disable User Info

User Info URL

* Client ID

* Client Secret

- You will notice that you are missing a few required fields such as “Authorization URL”, “Token URL”, “Client ID” and “Client Secret”. We will get these values from the jazz authorization server and complete the form later. Make note of the “Redirect URI” as we will need it in the next step.

Setting up JAS Configuration Security

If the JAS server is started with a self-signed certificate or no certificate at all, modify the **oauthProvider** element in the **appConfig.xml** file as shown below. Change the **httpsRequired** attribute to false.

```

<oauthProvider id="JazzOP"
  httpsRequired="false"
  autoAuthorize="true"
  customLoginURL="/jazzop/form/login"
  accessTokenLifetime="7201"
  authorizationGrantLifetime="604801">
  <autoAuthorizeClient>client01</autoAuthorizeClient>
    <databaseStore dataSourceRef="OAuthFvtDataSource" />
</oauthProvider>

```

Creating a Relying Party Application in JAS

1. Create a **body.json** file as shown below and make sure that you also add the above Keycloak redirect URI to the list of `redirect_uris`

```

{
  "token_endpoint_auth_method": "client_secret_basic",
  "scope": "openid profile email general",
  "grant_types": [
    "authorization_code",
    "client_credentials",
    "implicit",
    "refresh_token",
    "urn:ietf:params:oauth:grant-type:jwt-bearer"
  ],
  "response_types": [
    "code",
    "token",
    "id_token token"
  ],
  "application_type": "web",
  "subject_type": "public",
  "preauthorized_scope": "openid profile email general",
  "introspect_tokens": true,
  "trusted_uri_prefixes": [
    "https://keycloak.mycomp.any:*"
  ],
  "redirect_uris": [
    "http://keycloak.mycomp.any:8080/auth/realms/drone/broker/oidc/endpoint",
    "https://keycloak.mycomp.any:8443/auth/realms/drone/broker/oidc/endpoint"
  ]
}

```

Open command line and run the below command to create an application in the JAS server

```
curl --insecure --user admin:password --data @"./body.json"
```

http://jas.mycomp.any:9280/oidc/endpoint/jazzop/registration --header "Content-Type: application/json"

2. If successful, the JAS server will respond with a response as shown below

```
{
  "client_id_issued_at":1583359157,
  "registration_client_uri":"https://localhost:9643/oidc/endpoint/jazzop/registration/d5852705ab4f4204ae29812373537277",
  "client_secret_expires_at":0,
  "token_endpoint_auth_method":"client_secret_basic",
  "scope":"openid profile email general",
  "grant_types":[
    "authorization_code",
    "client_credentials",
    "implicit",
    "refresh_token",
    "urn:ietf:params:oauth:grant-type:jwt-bearer"
  ],
  "response_types":[
    "code",
    "token",
    "id_token token"
  ],
  "application_type":"web",
  "subject_type":"public",
  "preauthorized_scope":"openid profile email general",
  "introspect_tokens":true,
  "trusted_uri_prefixes":[
    "https://localhost:*/"
  ],
  "resource_ids":[]
},
  "client_id":"d5852705ab4f4204ae29812373537277",
  "client_secret":"p7Da1WpAxs0ujm80mFn9pN2HDVXXtDEWXJy3pIa792TNgfUyGbU7tyKXPGSd",
  "client_name":"d5852705ab4f4204ae29812373537277",
  "redirect_uris":[
    "http://localhost:8080/auth/realms/drone/broker/oidc/endpoint",
    "https://localhost:8443/auth/realms/drone/broker/oidc/endpoint"
  ],
  "allow_regex_redirects":false
}
```

3. Note that we now have a valid **client_id** and **client_secret** to complete the creation of our identity provider application in Keycloak. Please make a note of that **client_id** and **client_secret**.

4. Browsing to the public endpoint <http://jas.mycomp.any:9280/oidc/endpoint/jazzop/.well-known/openid-configuration> will provide the remaining Authorization and token URLs of the JAS server assuming the JAS http server is started locally and on port 9280. A sample response is shown below.

```

{
  "introspection_endpoint": "http://localhost:9280/oidc/endpoint/jazzop/introspect",
  "coverage_map_endpoint": "http://localhost:9280/oidc/endpoint/jazzop/coverage_map",
  "issuer": "http://localhost:9280/oidc/endpoint/jazzop",
  "authorization_endpoint": "http://localhost:9280/oidc/endpoint/jazzop/authorize",
  "token_endpoint": "http://localhost:9280/oidc/endpoint/jazzop/token",
  "jwks_uri": "http://localhost:9280/oidc/endpoint/jazzop/jwk",
  "response_types_supported": [
    "code",
    "token",
    "id_token token"
  ],
  "subject_types_supported": [
    "public"
  ],
  "id_token_signing_alg_values_supported": [
    "HS256"
  ],
  "userinfo_endpoint": "http://localhost:9280/oidc/endpoint/jazzop/userinfo",
  "registration_endpoint": "http://localhost:9280/oidc/endpoint/jazzop/registration",
  "scopes_supported": [
    "openid",
    "general",
    "profile",
    "email",
    "address",
    "phone"
  ],
  "claims_supported": [
    "sub",
    "groupIds",
    "name",
    "preferred_username",
    "picture",
    "locale",

```

5. We can now proceed back to Keycloak admin web page and finalize the identity provider form.

Creating a New Identify Provider in Keycloak Continued

Fill in the **Client ID**, **Client Secret** and other fields as shown in the picture below.

- Clients
- Client Scopes
- Roles
- Identity Providers
- User Federation
- Authentication
- Groups
- Users
- Sessions
- Events
- Import
- Export

Redirect URI

* Alias

Display Name

Enabled

Store Tokens

Stored Tokens Readable OFF

Trust Email OFF

Account Linking Only OFF

Hide on Login Page OFF

GUI order

First Login Flow

Post Login Flow

▼ OpenID Connect Config

* Authorization URL

Pass login_hint OFF

Pass current locale OFF

* Token URL

Logout URL

Backchannel Logout

Disable User Info OFF

User Info URL

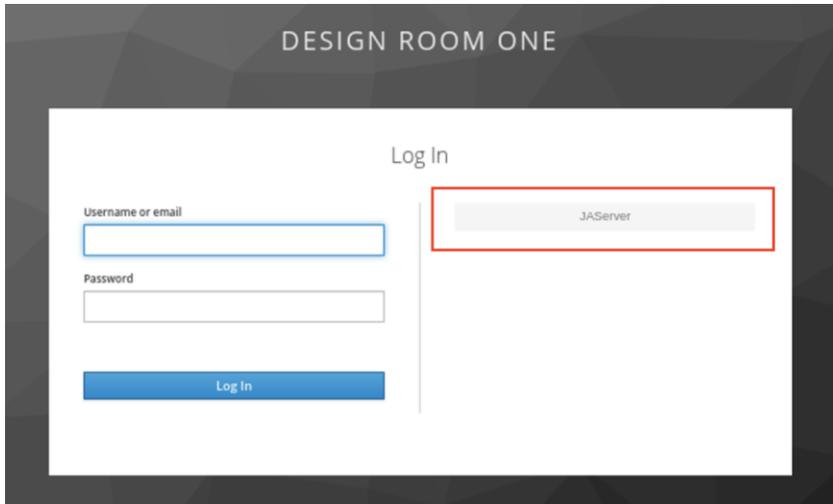
* Client ID

* Client Secret

After all required fields are completed, click the “Save” button.

Logging in with JAS Authentication

1. Ensure that Keycloak is enabled in your Design Room ONE server (i.e. server-config.json) file. Start your Design Room ONE server and navigate to some URL e.g. <https://drone.mycomp.any:10101/dr/web>
You should see the new identify provider option (JAServer) as shown here.



2. Click on JAServer option and you will be redirected to the JAS server to login



3. After a successful login you will be redirected back to the Keycloak server to add additional details for the user.

The image shows a web form titled "Update Account Information" within a dark grey header labeled "DESIGN ROOM ONE". The form is white and contains four input fields: "Username" (with "ADMIN" entered), "Email", "First name", and "Last name". A blue "Submit" button is located at the bottom of the form.

After a successful completion of the form, you will be directed back to the Design Room ONE page you navigated.

Known Limitations

1. Dockerized setup instructions to be provided in a later delivery
2. Automated export scenarios do not support authentication yet.
3. After clicking on **logout** link in the exporting wizard in a modeling tool a web browser with a message "Login Successful" appears, the message should say "Logout Successful"