FLOWX.AI

PLATFORM SETUP GUIDES / Access management / configuring-an-iam-solution

Contents

PLATFORM SETUP GUIDES / Access management / Configuring an IAM solution

Recommended Keycloak setup

To configure a minimal required Keycloak setup, follow these steps:

- Create a new realm
  - Define available roles and realm-level roles assigned to new users.
- Create/import user roles and groups
- Create new users
- Add clients
  - Configure access type, valid redirect URIs, and enable necessary flows.
- Add role mappers
- Add service accounts
  - Set up **admin**, **task management**, and **process engine** service accounts.

> ⓘ **INFO**
>
> Recommended keycloak version: **18.0.x**

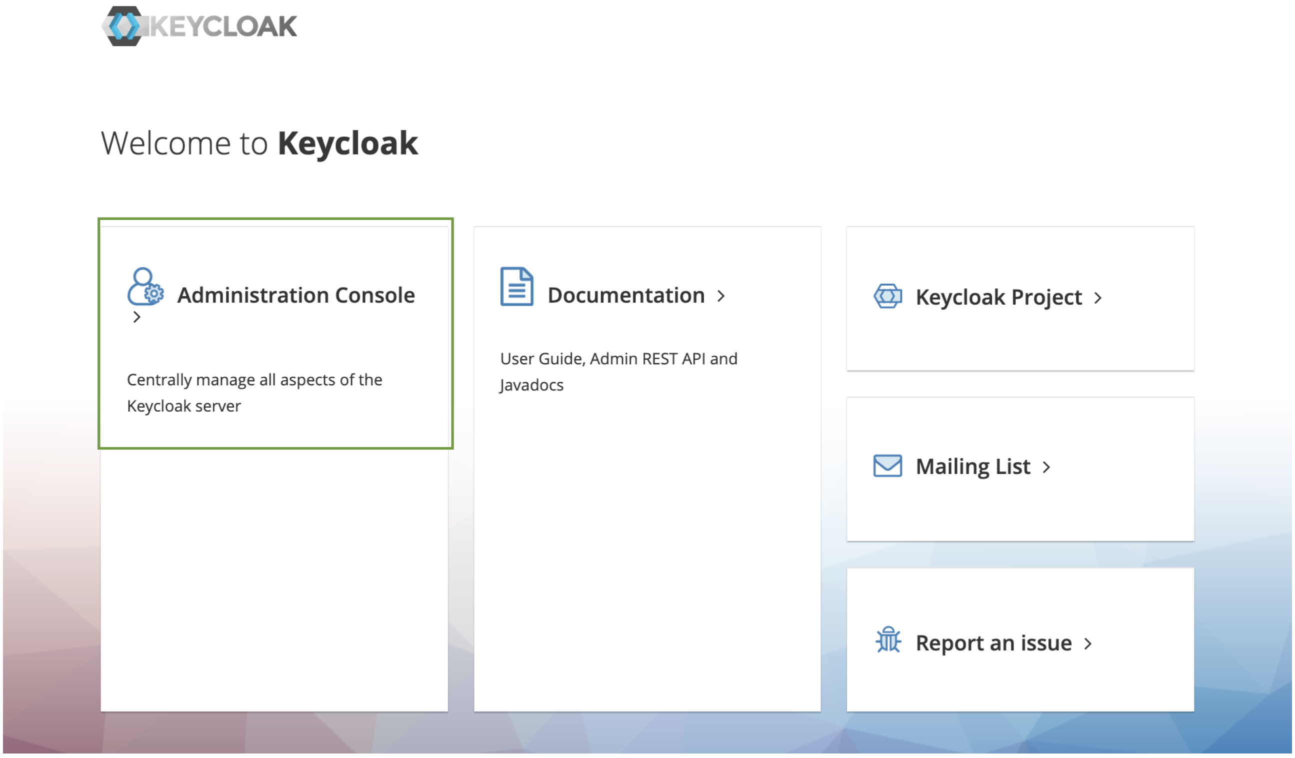
For more detailed information, refer to the official Keycloak documentation:

» Keycloak documentation

Creating a new realm

A realm is a space where you manage objects, including users, applications, roles, and groups. To create a new realm:

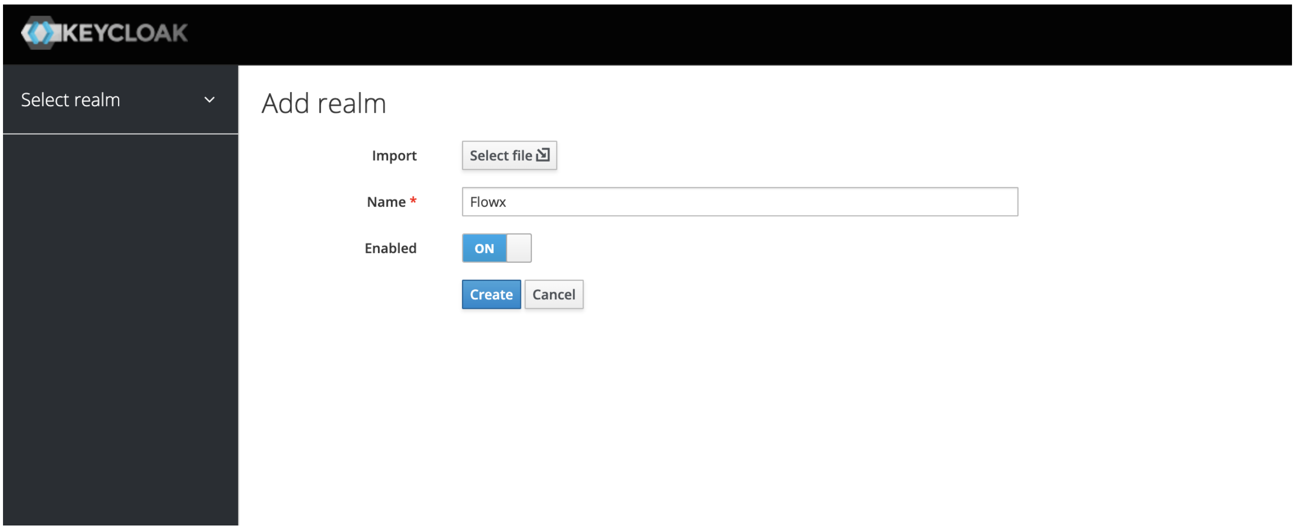
1. Log in to the **Keycloak Admin Console** using the appropriate URL for your environment (e.g., QA, development, production).



2. In the top left corner dropdown menu, click **Add Realm**.

> **INFO**
>
> If you are logged in to the master realm this dropdown menu lists all the realms created. The **Add Realm** page opens.

3. Enter a realm name and click Create.



Add realm

Import: Select file

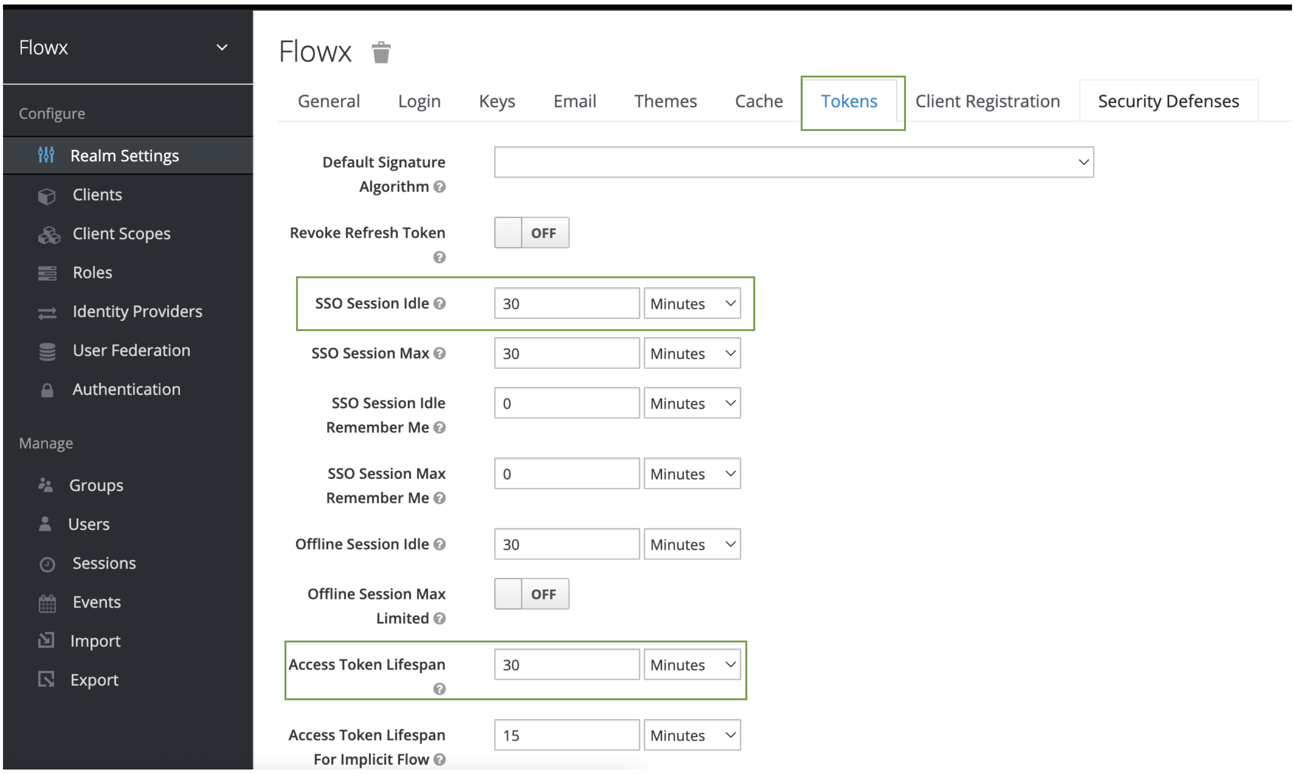
Name *: Flowx

Enabled: ON

Create Cancel

4. Configure the realm settings (**Realm Settings → Tokens**), such as SSO session idle and access token lifespan, according to your organization's needs:

- **SSO Session idle -** suggested: **30 Minutes**
- **Access Token Lifespan -** suggested: **30 Minutes**



Default Signature Algorithm

Revoke Refresh Token: OFF

SSO Session Idle: 30 Minutes

SSO Session Max: 30 Minutes

SSO Session Idle Remember Me: 0 Minutes

SSO Session Max Remember Me: 0 Minutes

Offline Session Idle: 30 Minutes

Offline Session Max Limited: OFF

Access Token Lifespan: 30 Minutes

Access Token Lifespan For Implicit Flow: 15 Minutes

Creating/importing user groups and roles

You can either create or import a user group into a realm. We prepared a script that helps you to import a **super admin group** provided with the necessary **default user roles**.

You can create or import user groups into a realm. If you choose to import, follow the provided script to import a **super admin group**(`SUPER_ADMIN_USERS`) with **default user roles**. After importing, add an admin user to the group and assign the necessary roles.

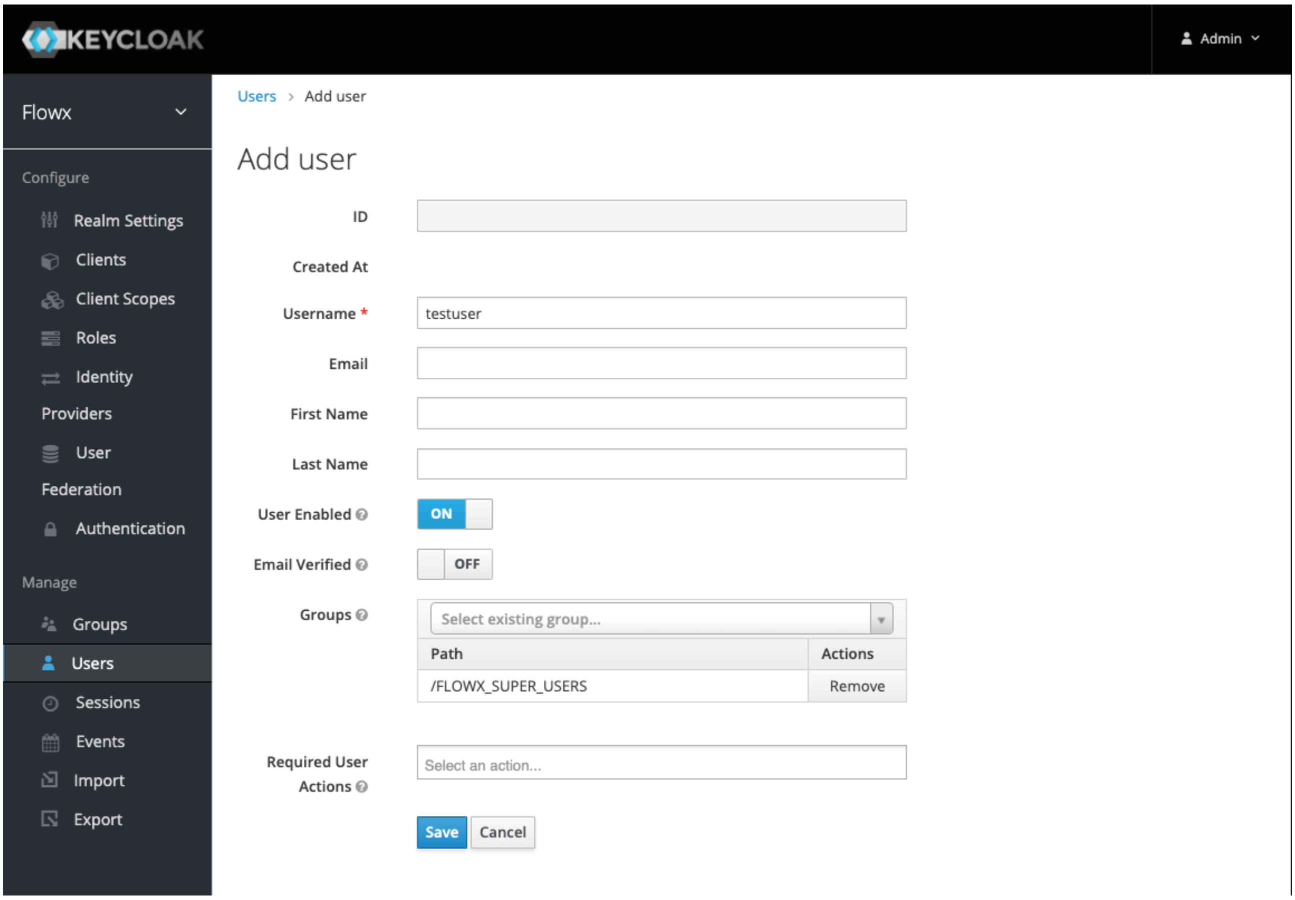
Make sure to validate the imported roles by checking the following section:

» Default roles

Creating new users

To create a new user in a realm and generate a temporary password:

1. In the left menu bar, click **Users** to open the user list page.
2. On the right side of the empty user list, click **Add User**.
3. Fill in the required fields, including the **username**, and ensure **Email Verified** is set to **ON**.
4. In the **Groups** field, choose a group from the dropdown menu, in our case: `FLOWX_SUPER_USERS`.



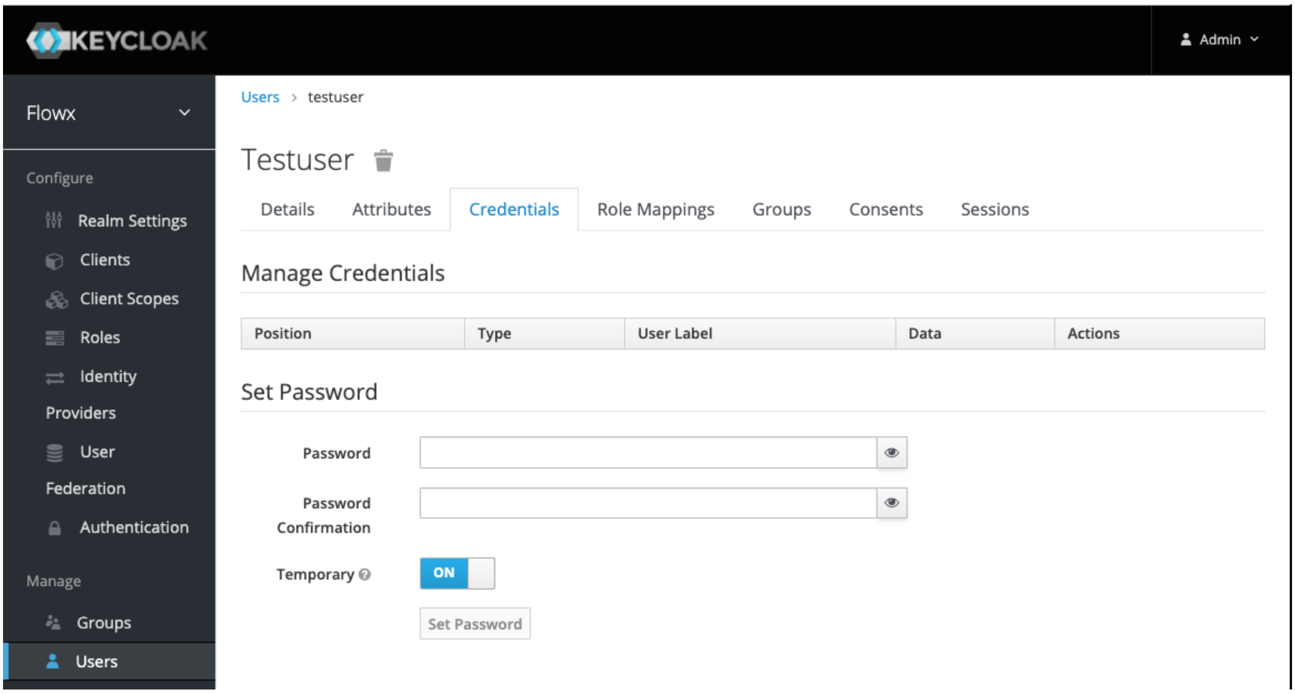
Add user

ID	
Created At	
Username *	testuser
Email	
First Name	
Last Name	
User Enabled	ON
Email Verified	OFF
Groups	Select existing group...

Path	Actions
/FLOWX_SUPER_USERS	Remove

Required User Actions: Select an action...

5. Save the user, go to the **Credentials** tab, and set a temporary password.



Testuser

Manage Credentials

Position	Type	User Label	Data	Actions

Set Password

Password

Password Confirmation

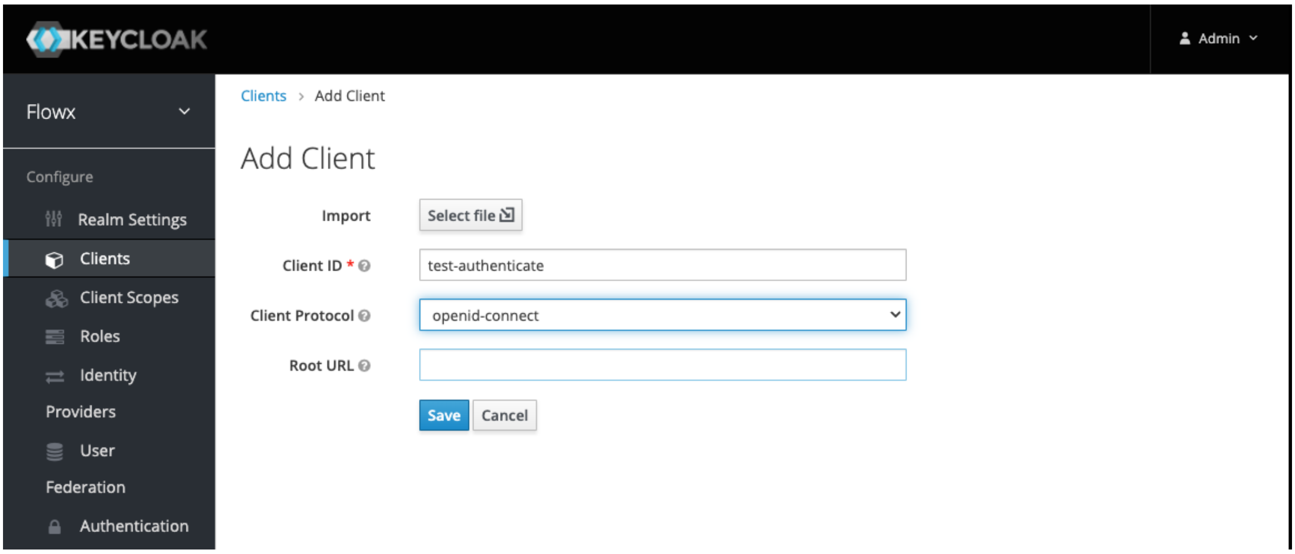
Temporary: ON

Set Password

Adding clients

Clients represent trusted browser apps and web services in a realm. To add clients:

1. Click **Clients** in the top left menu, then click **Create**.
2. Set a client ID as `{example}-authenticate`, which will be used for login, logout, and refresh token operations.
3. Set the **Client Protocol** type as `openid-connect`.



Add Client

Import: Select file

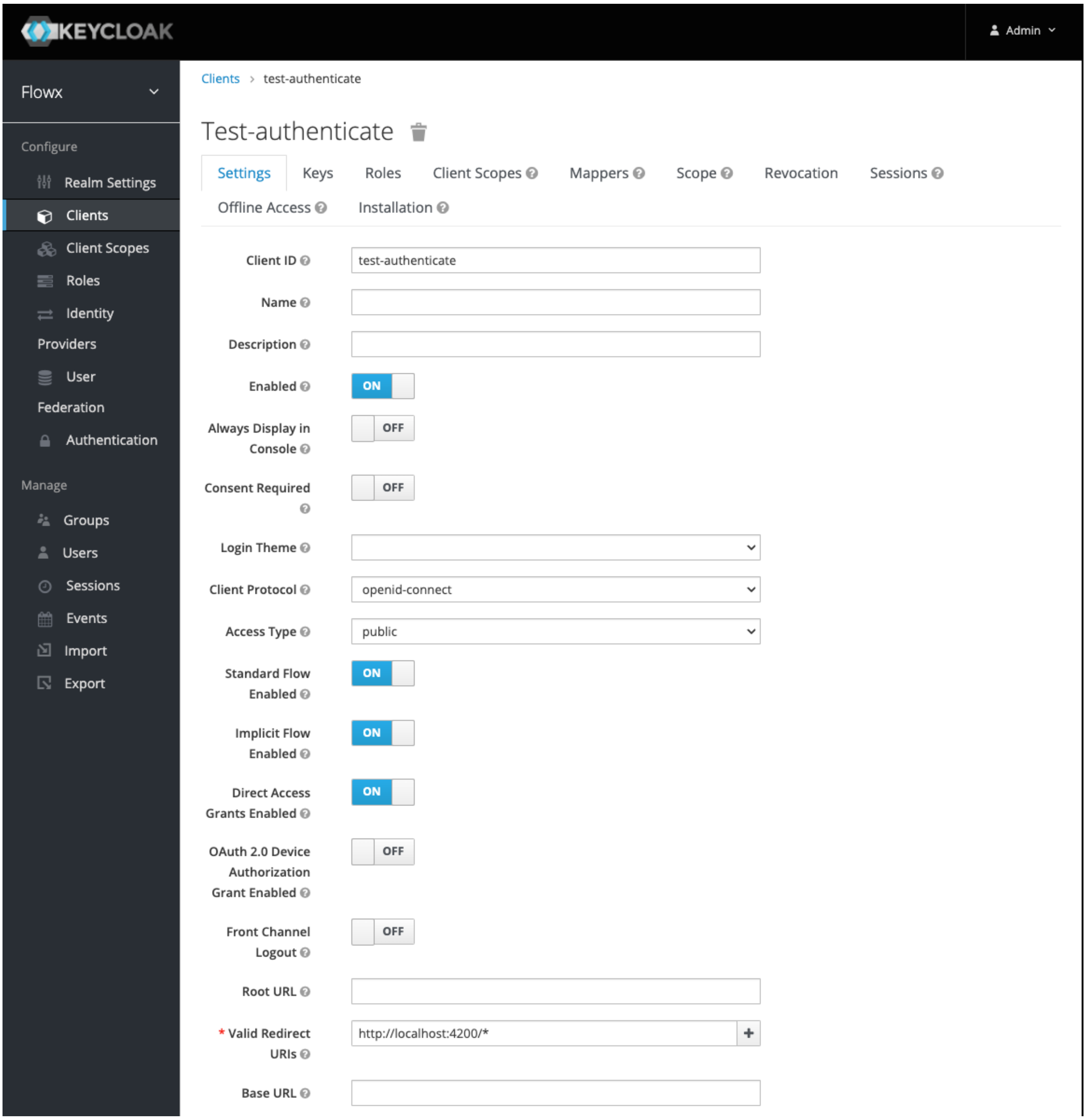
Client ID *: test-authenticate

Client Protocol: openid-connect

Root URL:

3. Open the newly created **client** and edit the following properties:

- Set **Access type** to **public** (this will not require a secret)
- Set **Valid redirect URIs**, specifying a valid URI pattern that a browser can redirect to after a successful login or logout, simple wildcards are allowed
- Enable **Direct Access Grants** and **Implicit Flow** by setting them to **ON**.
- Switch **Backchannel Logout Session Required** to **OFF**



Test-authenticate

Client ID	test-authenticate
Name	
Description	
Enabled	ON
Always Display in Console	OFF
Consent Required	OFF
Login Theme	
Client Protocol	openid-connect
Access Type	public
Standard Flow Enabled	ON
Implicit Flow Enabled	ON
Direct Access Grants Enabled	ON
OAuth 2.0 Device Authorization Grant Enabled	OFF
Front Channel Logout	OFF
Root URL	
* Valid Redirect URIs	http://localhost:4200/*
Base URL	

4. Add **mappers** to `{example}-authenticate` client.

> **INFO**

> Refer to the next section on how to add mappers and which mappers to clients.

Adding protocol mappers

Protocol mappers in Keycloak allow for the transformation of tokens and documents, enabling actions such as mapping user data into protocol claims or modifying requests between clients and the authentication server.

To enhance your clients, consider adding the following mappers:

- Group Membership mapper - `realm-groups`: This mapper can be utilized to map user groups to the authorization token.
- User Attribute mapper - `business filter mapper`: Use this mapper to map custom attributes, for example, mapping the businessFilters list, to the token claim.
- User Realm role - `realm-roles`: This mapper enables mapping a user's realm role to a token claim.

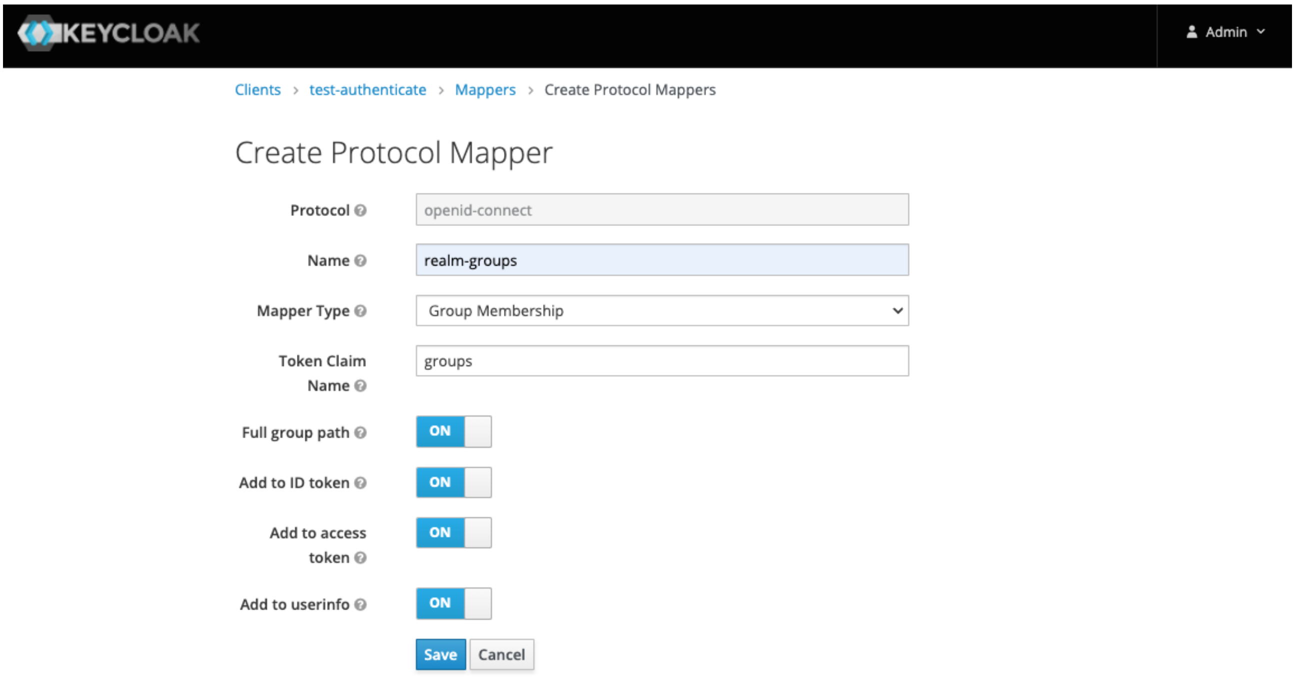
By incorporating these mappers, you can further customize and enrich the information contained within your tokens.

Group Membership mapper

To add a group membership mapper:

1. Navigate to **Clients** and select your desired client, in our case, `{example}-authenticate`

2. Go to the **Mappers** tab and click **Create** to create a new mapper.
3. Provide a descriptive **Name** for the mapper to easily identify its purpose.
4. Select **Group Membership** as the mapper type.
5. Set the token claim name for including groups in the token. In this case, set it as `groups`.


KEYCLOAK
Admin
Clients > test-authenticate > Mappers > Create Protocol Mappers

Create Protocol Mapper

Protocol	openid-connect
Name	realm-groups
Mapper Type	Group Membership
Token Claim Name	groups
Full group path	ON
Add to ID token	ON
Add to access token	ON
Add to userinfo	ON

By configuring the group membership mapper, you will be able to include the user's group information in the token for authorization purposes.

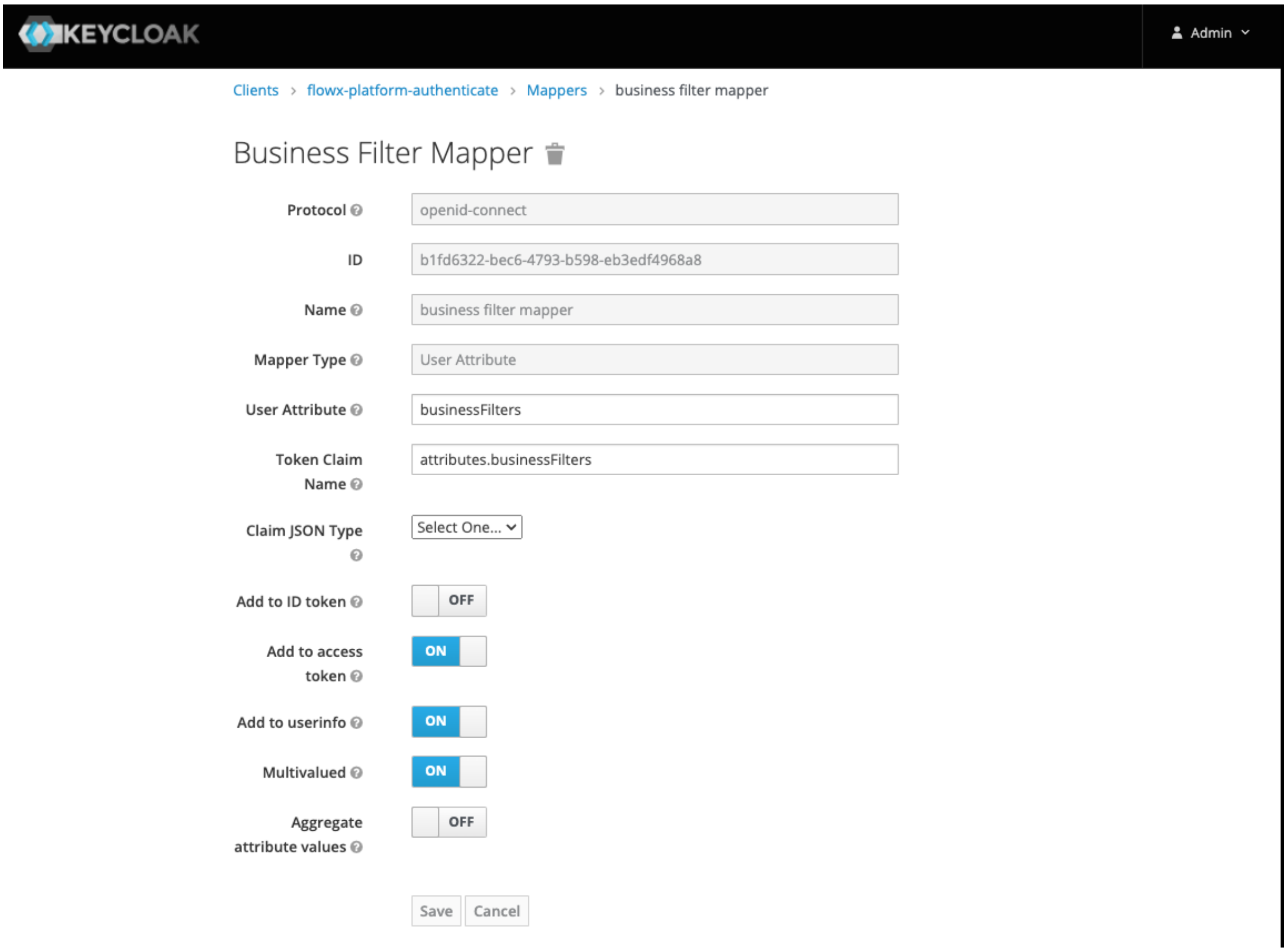
User Attribute mapper

To include custom attributes such as **business filters** in the token claim, you can add a user attribute mapper with the following settings:

1. Go to the desired client, `{example}-authenticate`, and navigate to the Mappers section.

2. Click on **Create** to create a new mapper.
3. Configure the following settings for the user attribute mapper:

- **Mapper Type**: User Attribute
- **User Attribute**: businessFilters
- **Token Claim Name**: attirubtes.businessFilters
- **Add to ID token**: OFF
- **Multivalued**: ON



Clients › flowx-platform-authenticate › Mappers › business filter mapper

Business Filter Mapper

Protocol	openid-connect
ID	b1fd6322-bec6-4793-b598-eb3edf4968a8
Name	business filter mapper
Mapper Type	User Attribute
User Attribute	businessFilters
Token Claim Name	attributes.businessFilters
Claim JSON Type	Select One...
Add to ID token	OFF
Add to access token	ON
Add to userinfo	ON
Multivalued	ON
Aggregate attribute values	OFF

By adding this user attribute mapper, the custom attribute "businessFilters" will be included in the token claim under the name "attributes.businessFilters". This will allow you to access and utilize the business filters information within your application.

You can find more information about business filters in the following section:

» Business filters

User realm role

Add **roles mapper** to `{example}-authenticate` client - so roles will be available on the OAuth user info response.

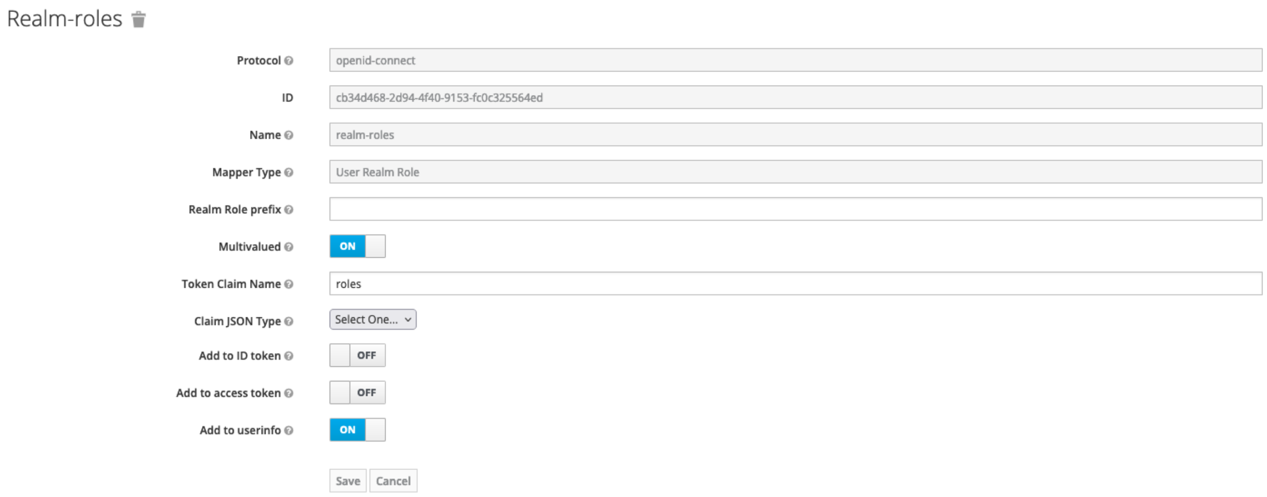
To add a roles mapper, follow these steps:

1. Go to the desired client, `{example}-authenticate`, and navigate to the Mappers section.
2. Click on **Create** to create a new mapper.
3. Configure the following settings for the user attribute mapper:

- **Mapper Type**: User Realm Role
- **Token Claim Name**: role
- **Add to userinfo**: ON

By adding this roles mapper, the assigned realm roles of the user will be available in the OAuth user info response under the claim name "roles". This allows you to access and utilize the user's realm roles within your application.

Please note that you can repeat these steps to add multiple roles mappers if you need to include multiple realm roles in the token claim.



Realm-roles

Protocol	openid-connect
ID	cb34d468-2d94-4f40-9153-fc0c325564ed
Name	realm-roles
Mapper Type	User Realm Role
Realm Role prefix	
Multivalued	ON
Token Claim Name	roles
Claim JSON Type	Select One...
Add to ID token	OFF
Add to access token	OFF
Add to userinfo	ON

Examples

Login

```
curl --location --request POST
'http://localhost:8080/realms/flowx/protocol/openid-
connect/token' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--data-urlencode 'grant_type=password' \
--data-urlencode 'username=admin@flowx.ai' \
--data-urlencode 'password=password' \
--data-urlencode 'client_id= example-authenticate'
```

Refresh token

```
curl --location --request POST
'http://localhost:8080/realms/flowx/protocol/openid-
connect/token' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--data-urlencode 'grant_type=refresh_token' \
--data-urlencode 'client_id= example-authenticate' \
--data-urlencode 'refresh_token=ACCESS_TOKEN'
```

User info

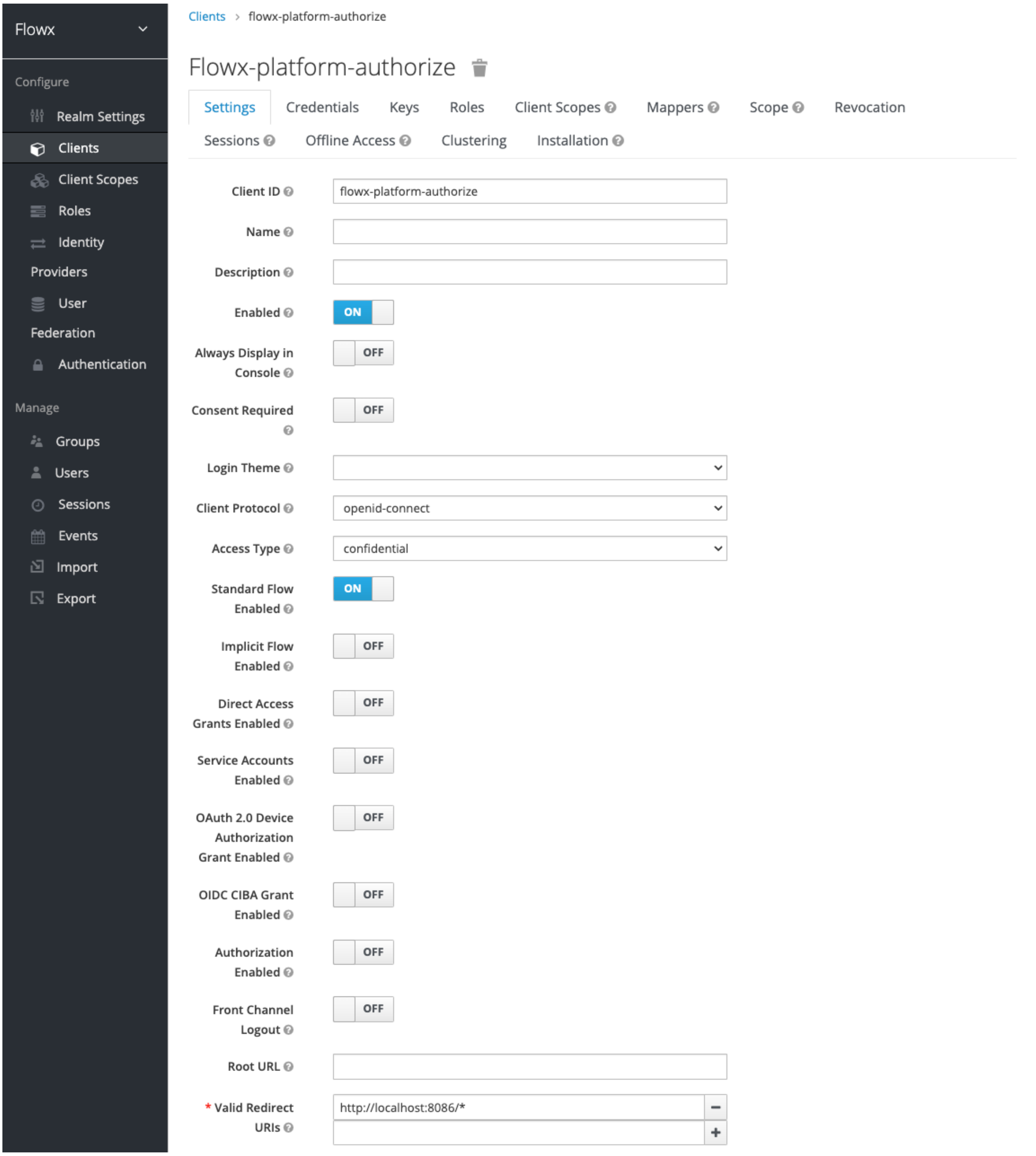
```
curl --location --request GET
'localhost:8080/realms/flowx/protocol/openid-
connect/userinfo' \
--header 'Authorization: Bearer ACCESS_TOKEN' \
```

Authorizing client

Add `{example}-platform-authorize` client - it will be used to authorize rest requests to microservices and Kafka

- set **Client Protocol** to **openid-connect**
- set **Access type** as **confidential**
- disable **Direct Access Grants Enabled** - OFF
- **Valid Redirect URIs** - mandatory
- disable **Backchannel Logout Session Required** - OFF

Once you have configured these settings, the `{example}-platform-authorize` client will be created and can be used to authorize REST requests to microservices and Kafka within your application.



Flowx-platform-authorize

Client ID	flowx-platform-authorize
Name	
Description	
Enabled	ON
Always Display in Console	OFF
Consent Required	OFF
Login Theme	
Client Protocol	openid-connect
Access Type	confidential
Standard Flow Enabled	ON
Implicit Flow Enabled	OFF
Direct Access Grants Enabled	OFF
Service Accounts Enabled	OFF
OAuth 2.0 Device Authorization Grant Enabled	OFF
OIDC CIBA Grant Enabled	OFF
Authorization Enabled	OFF
Front Channel Logout	OFF
Root URL	
* Valid Redirect URIs	http://localhost:8086/*

Minimal auth config for microservices

```
security:
  type: oauth2
  basic:
    enabled: false
  oauth2:
    base-server-url: http://localhost:8080
    realm: flowx
    client:
      access-token-uri: ${security.oauth2.base-server-
url}/realms/${security.oauth2.realm}/protocol/openid-connect/t
      client-id: example-authorize
      client-secret: CLIENT_SECRET
    resource:
      user-info-uri: ${security.oauth2.base-server-
url}/realms/${security.oauth2.realm}/protocol/openid-connect/u
```

Adding service accounts

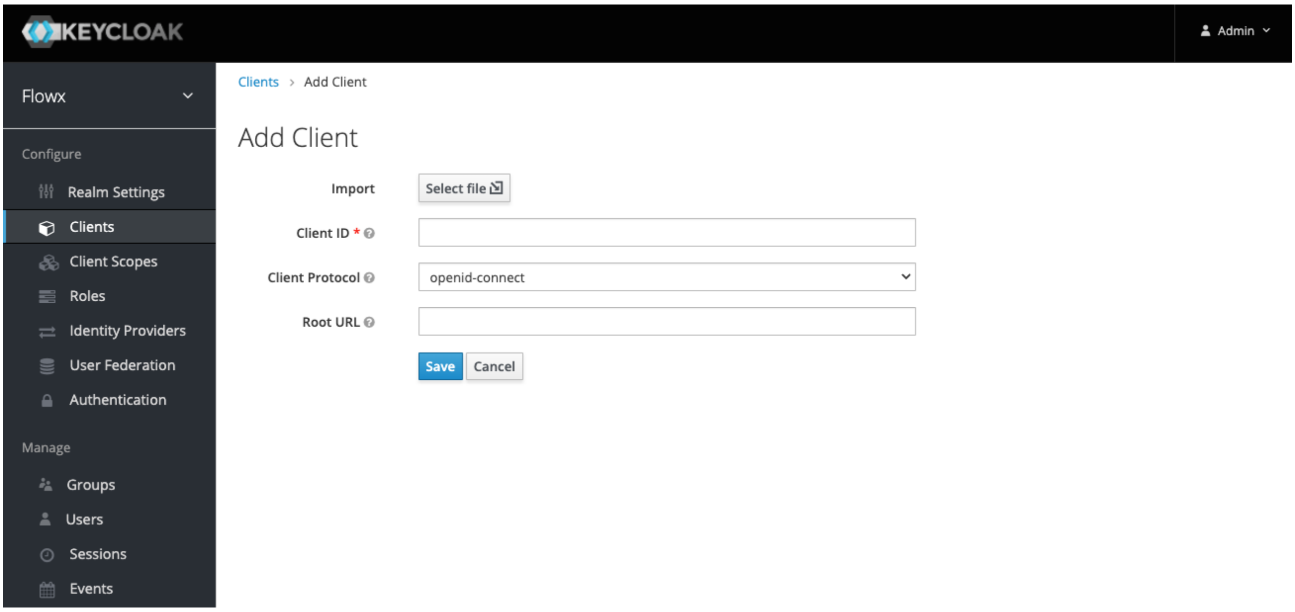
> INFO
>
> **What is a service account?**
>
> A service account is an account that grants direct access to the Keycloak API for a specific component.

Admin service account

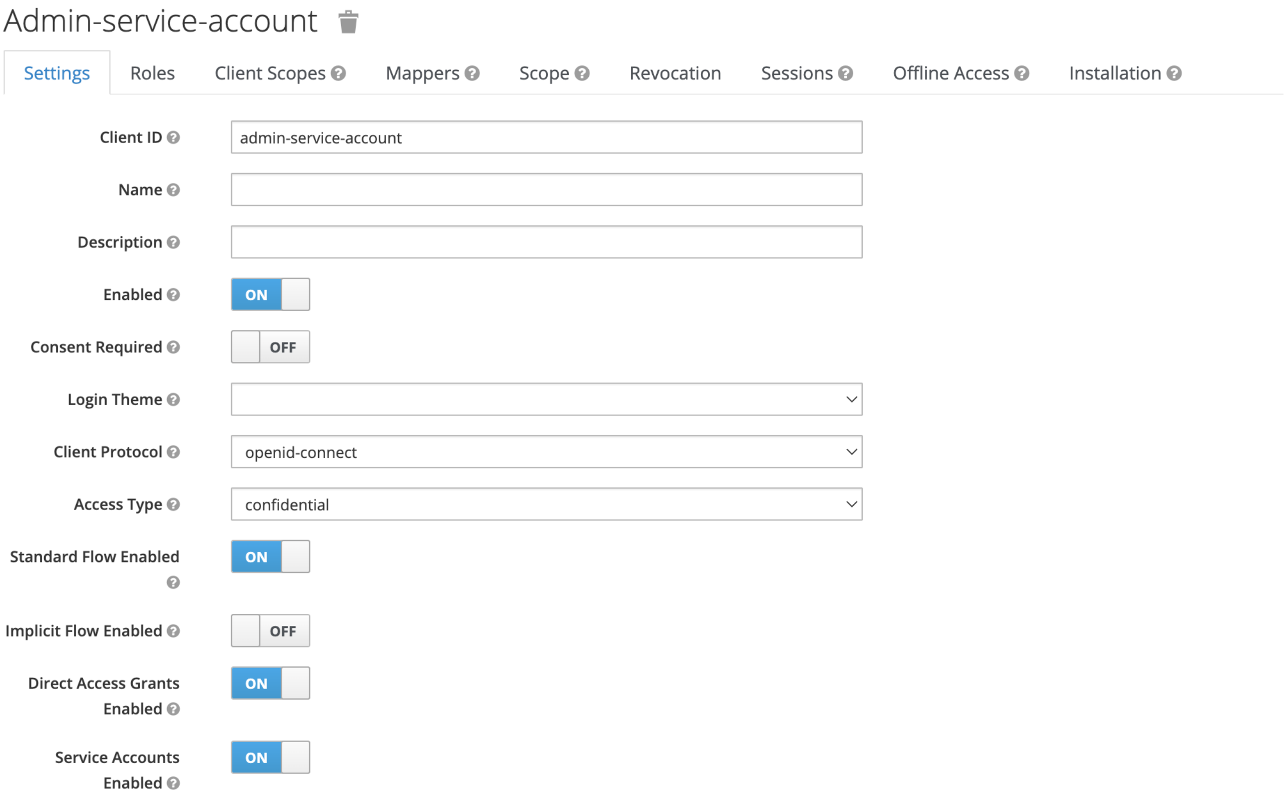
The admin microservice requires an admin service account to make direct calls to the Keycloak API.

Follow these steps to add an **admin service account**:

1. Add a new client by selecting **Clients** then click **Create**.



2. Next, set **Access type** as **confidential** and enable **Service Accounts**.



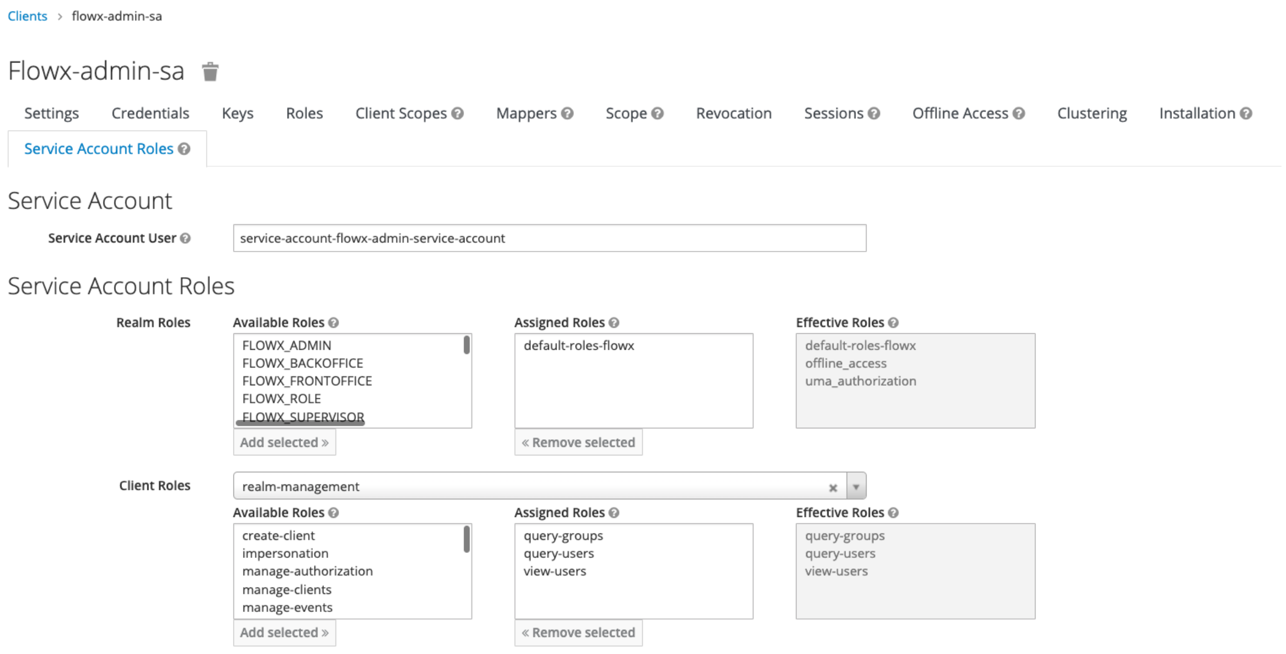
Admin-service-account

Client ID	admin-service-account
Name	
Description	
Enabled	ON
Consent Required	OFF
Login Theme	
Client Protocol	openid-connect
Access Type	confidential
Standard Flow Enabled	ON
Implicit Flow Enabled	OFF
Direct Access Grants Enabled	ON
Service Accounts Enabled	ON

3. Go to **Clients** → **realm-management** → **Roles** and add the following **service account client roles** under **realm-management**:

- **view-users**
- **query-groups**
- **query-users**

4. Assign the necessary **service account roles**:



Service Account

Service Account User: service-account-flowx-admin-service-account

Service Account Roles

Realm Roles

Available Roles	Assigned Roles	Effective Roles
FLOWX_ADMIN FLOWX_BACKOFFICE FLOWX_FRONTOFFICE FLOWX_ROLE FLOWX_SUPERVISOR	default-roles-flowx	default-roles-flowx offline_access uma_authorization

Client Roles: realm-management

Available Roles	Assigned Roles	Effective Roles
create-client impersonation manage-authorization manage-clients manage-events	query-groups query-users view-users	query-groups query-users view-users

In the provided example, the **admin service account** can have the following assigned roles, depending on the required access scopes:

- **manage-users**
- **query-users**
- **manage-realm**

> **INFO**
>
> The admin service account does not require mappers as it doesn't utilize roles. Service account roles include client roles from the `realm-management`.

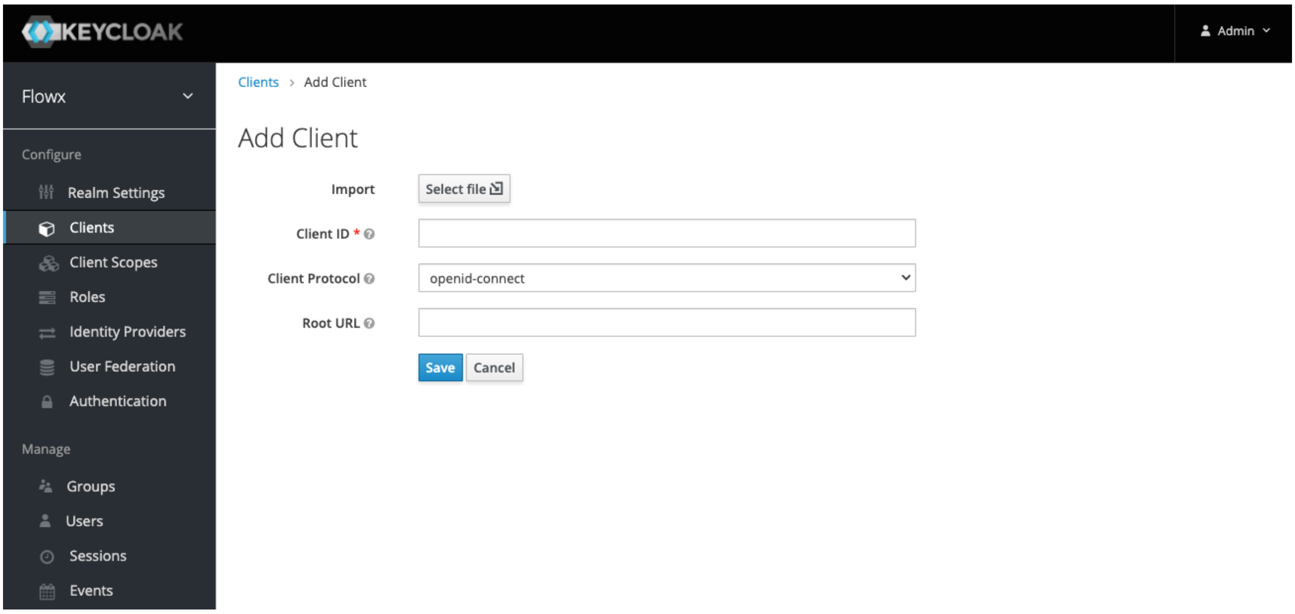
For detailed information, refer to the following section:

» Configuring access rights for admin

Task management service account

The task management microservice requires a service account to make direct calls to the Keycloak API. Follow these steps to add a task management service account:

1. Add a new client by selecting **Clients** then click **Create**.



Add Client

Import: Select file

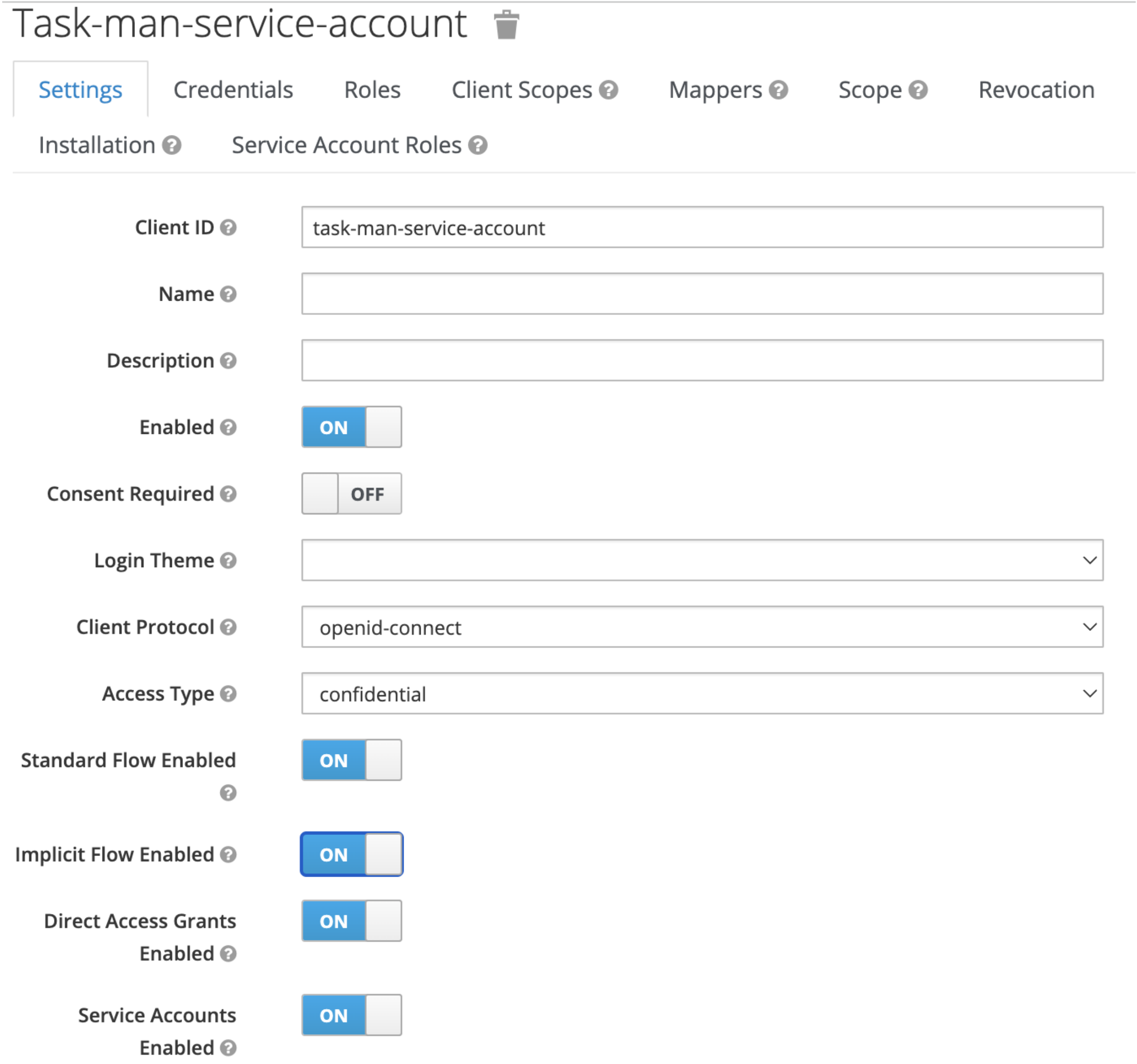
Client ID *

Client Protocol: openid-connect

Root URL

2. Next, set the following properties:

- **Access type** - confidential
- **Service Accounts Enabled** - ON

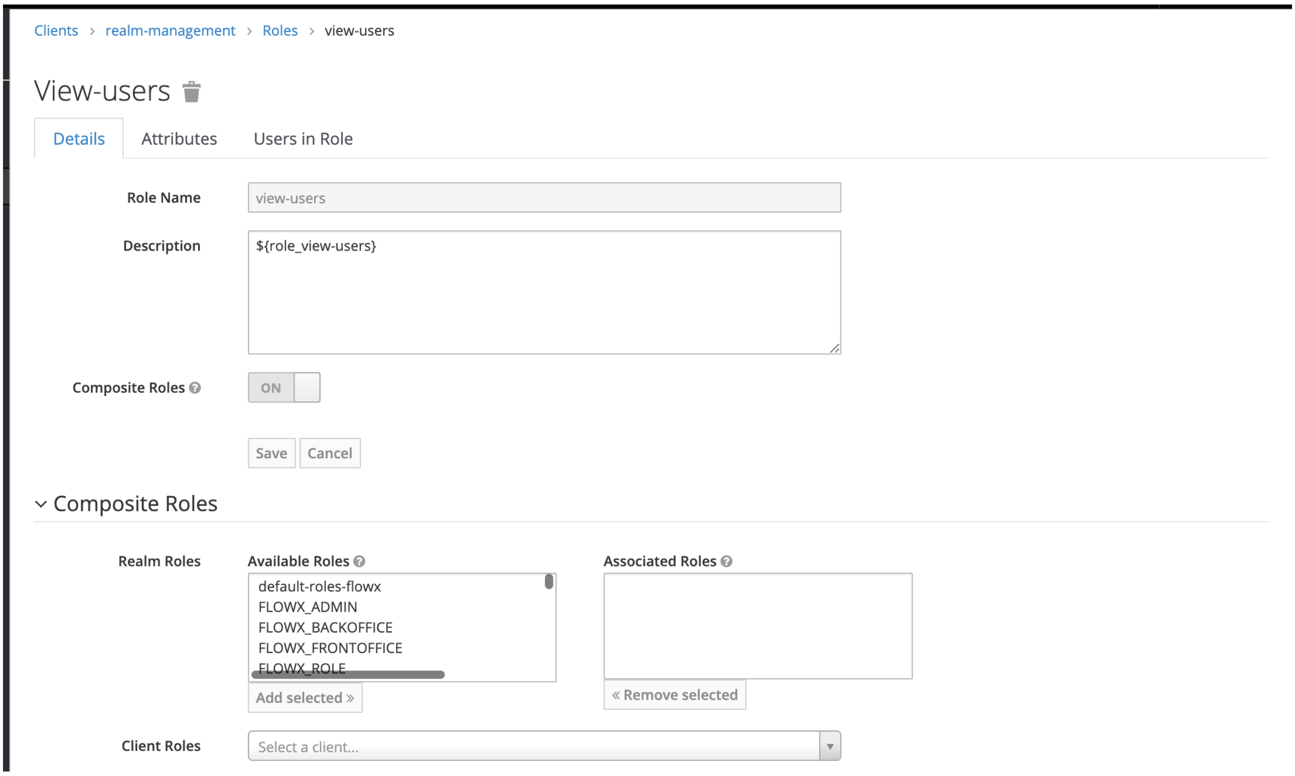


Task-man-service-account

Client ID	task-man-service-account
Name	
Description	
Enabled	ON
Consent Required	OFF
Login Theme	
Client Protocol	openid-connect
Access Type	confidential
Standard Flow Enabled	ON
Implicit Flow Enabled	ON
Direct Access Grants Enabled	ON
Service Accounts Enabled	ON

3. Go to **Clients** → **realm-management** → **Roles** and add the following **service account client roles**:

- **view-users**
- **query-groups**
- **query-users**



View-users

Role Name	view-users
Description	${role_view-users}
Composite Roles	ON

Composite Roles

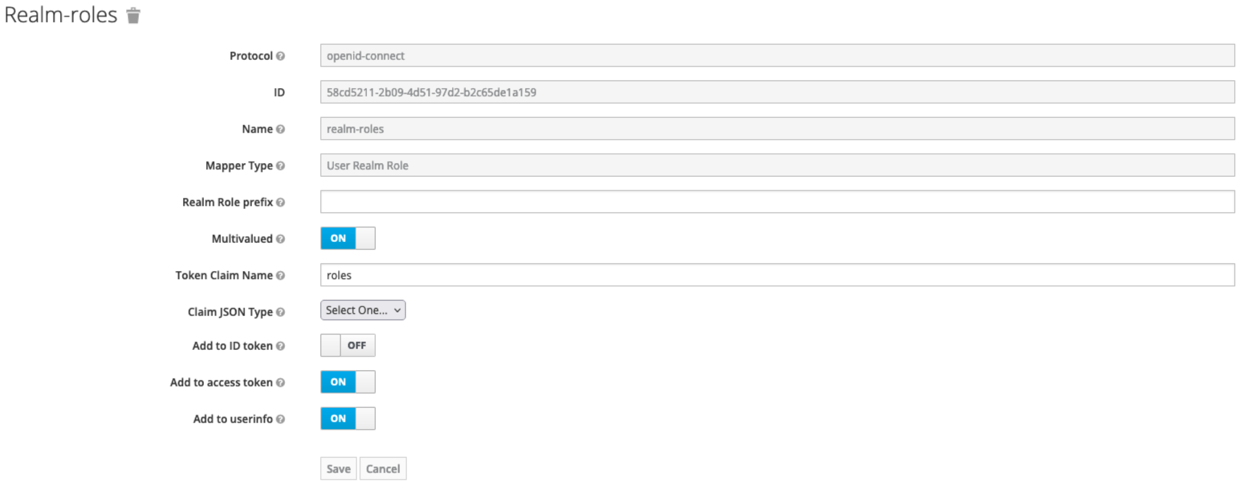
Realm Roles

Available Roles: default-roles-flowx, FLOWX_ADMIN, FLOWX_BACKOFFICE, FLOWX_FRONTOFFICE, FLOWX_ROLE

Associated Roles:

Client Roles: Select a client...

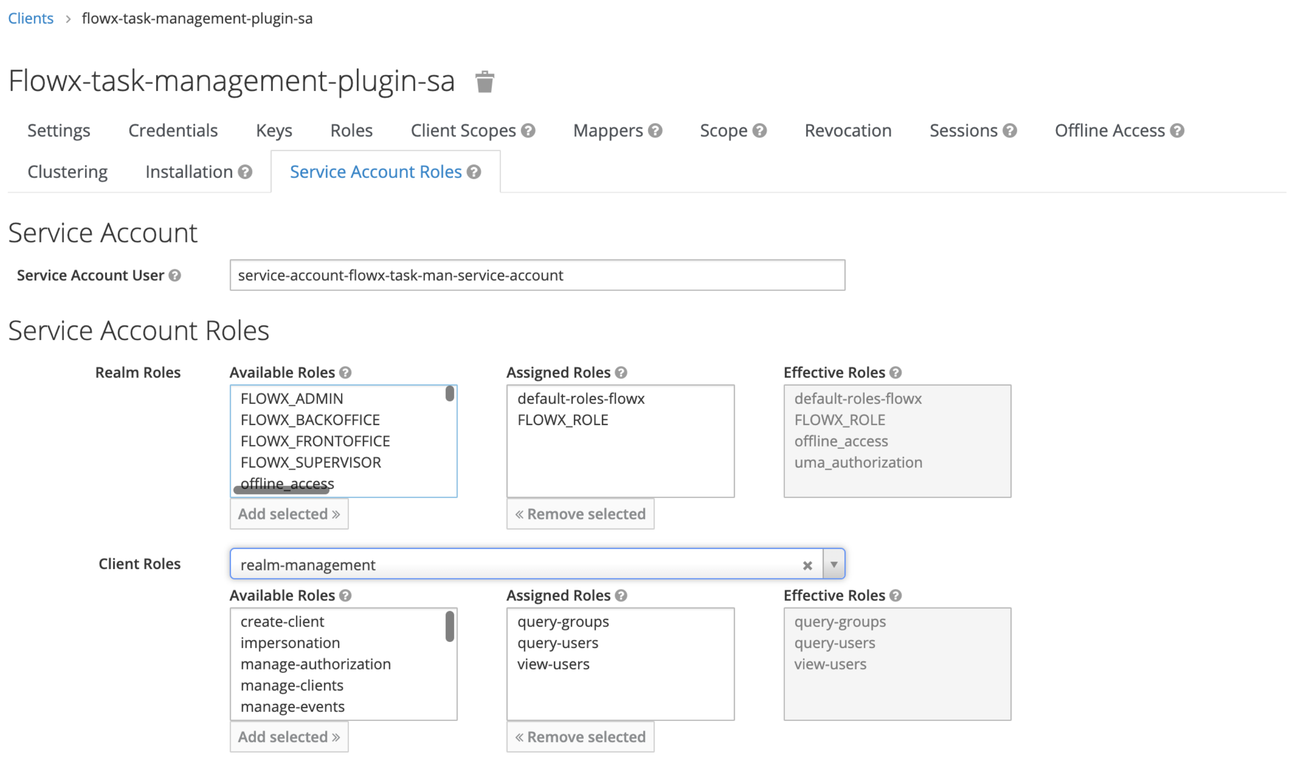
4. Configure a **realm roles mapper**:



Realm-roles

Protocol	openid-connect
ID	58cd5211-2b09-4d51-97d2-b2c65de1a159
Name	realm-roles
Mapper Type	User Realm Role
Realm Role prefix	
Multivalued	ON
Token Claim Name	roles
Claim JSON Type	Select One...
Add to ID token	OFF
Add to access token	ON
Add to userinfo	ON

5. Assign the necessary service account roles, including `FLOWX_ROLE`.



Flowx-task-management-plugin-sa

Service Account

Service Account User: service-account-flowx-task-man-service-account

Service Account Roles

Realm Roles

Available Roles: FLOWX_ADMIN, FLOWX_BACKOFFICE, FLOWX_FRONTOFFICE, FLOWX_SUPERVISOR, offline_access

Assigned Roles: default-roles-flowx, FLOWX_ROLE

Effective Roles: default-roles-flowx, FLOWX_ROLE, offline_access, uma_authorization

Client Roles: realm-management

Available Roles: create-client, impersonation, manage-authorization, manage-clients, manage-events

Assigned Roles: query-groups, query-users, view-users

Effective Roles: query-groups, query-users, view-users

In the provided example, the **task management service account** can have the following assigned roles, depending on the required access scopes:

- **view-users**
- **query-groups**
- **query-users**

For more information, check the following section:

» Configuring access rights for Task Management

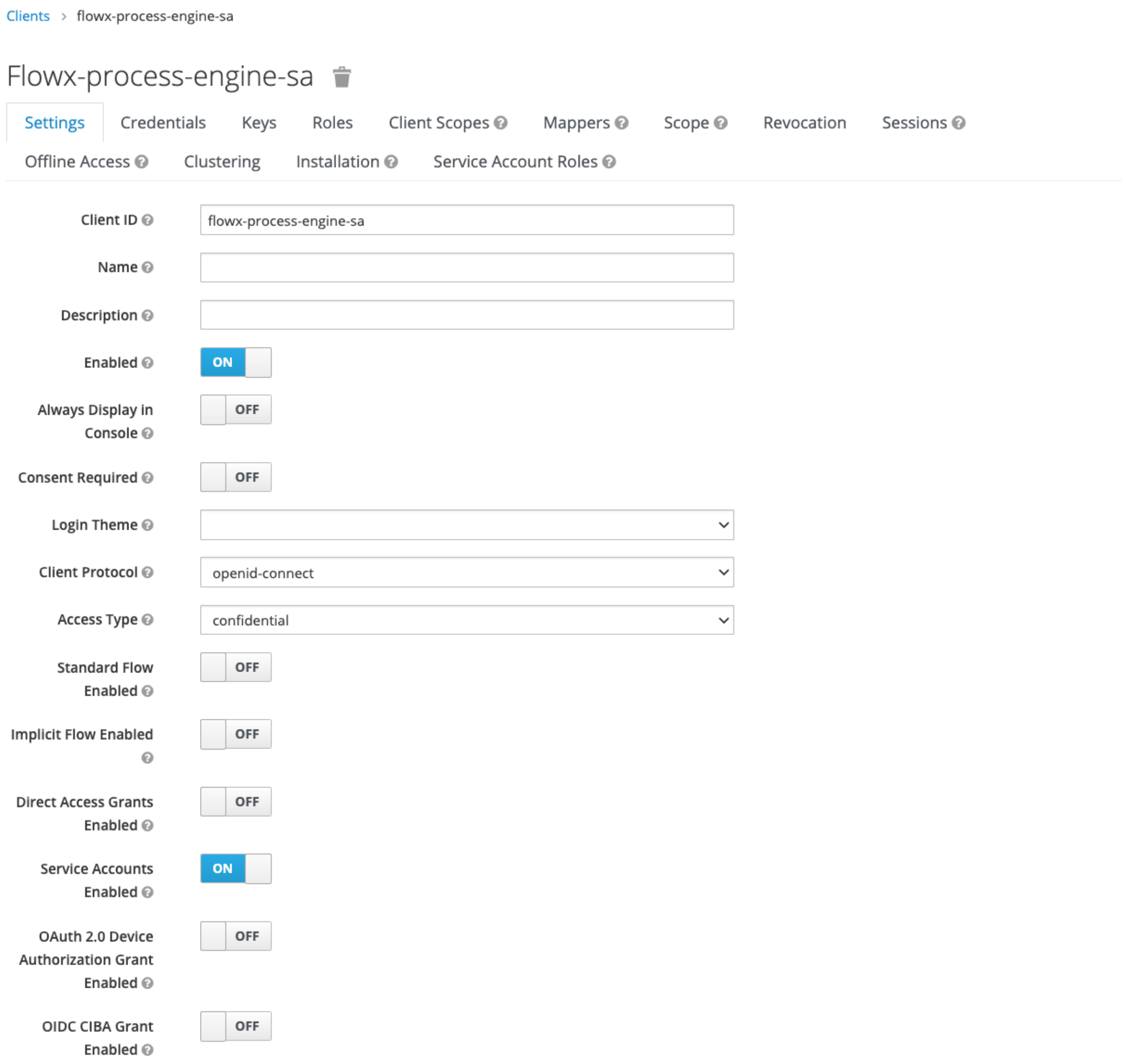
Process engine service account

The process engine requires a process engine service account to make direct calls to the Keycloak API.

> **INFO**
>
> This service account is needed so the use of Start Catch Event node is possible.

Follow these steps to add a **process engine service account**:

1. Add a new client by selecting **Clients** then click **Create**.



Client ID: flowx-process-engine-sa

Name:

Description:

Enabled: ON

Always Display in Console: OFF

Consent Required: OFF

Login Theme:

Client Protocol: openid-connect

Access Type: confidential

Standard Flow Enabled: OFF

Implicit Flow Enabled: OFF

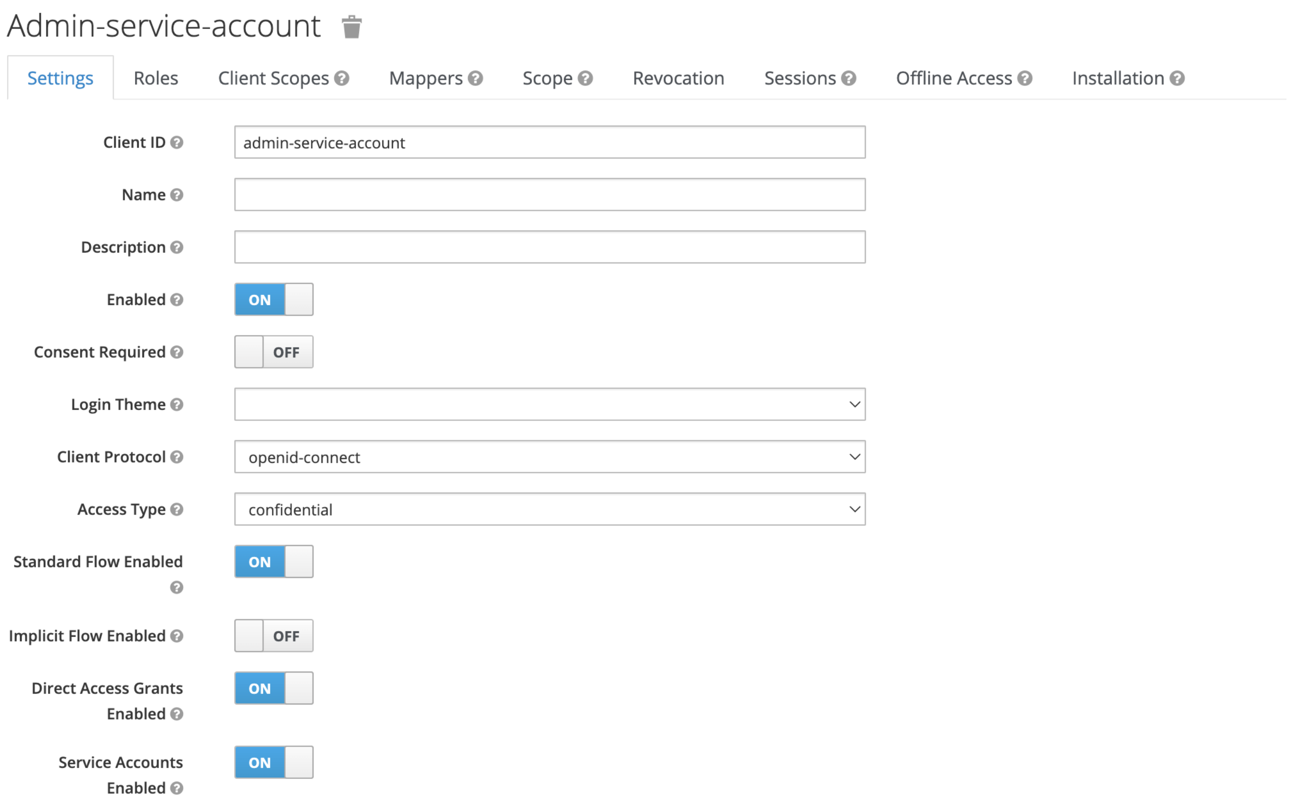
Direct Access Grants Enabled: OFF

Service Accounts Enabled: ON

OAuth 2.0 Device Authorization Grant Enabled: OFF

OIDC CIBA Grant Enabled: OFF

2. Next, set **Access type** as **confidential** and enable **Service Accounts**.



Admin-service-account

Client ID: admin-service-account

Name:

Description:

Enabled: ON

Consent Required: OFF

Login Theme:

Client Protocol: openid-connect

Access Type: confidential

Standard Flow Enabled: ON

Implicit Flow Enabled: OFF

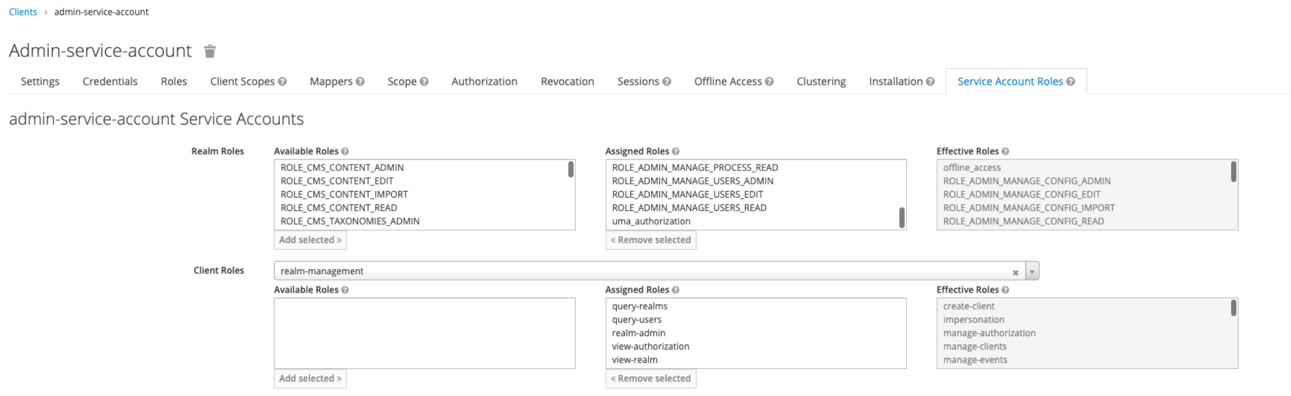
Direct Access Grants Enabled: ON

Service Accounts Enabled: ON

INFO

This service account does not require client roles.

3. Assign the necessary service account roles, including `FLOWX_ROLE`.



admin-service-account Service Accounts

Realm Roles

Available Roles
ROLE_CMS_CONTENT_ADMIN
ROLE_CMS_CONTENT_EDIT
ROLE_CMS_CONTENT_IMPORT
ROLE_CMS_CONTENT_READ
ROLE_CMS_TAXONOMIES_ADMIN

Assigned Roles
ROLE_ADMIN_MANAGE_PROCESS_READ
ROLE_ADMIN_MANAGE_USERS_ADMIN
ROLE_ADMIN_MANAGE_USERS_EDIT
ROLE_ADMIN_MANAGE_USERS_READ
uma_authorization

Effective Roles
offline_access
ROLE_ADMIN_MANAGE_CONFIG_ADMIN
ROLE_ADMIN_MANAGE_CONFIG_EDIT
ROLE_ADMIN_MANAGE_CONFIG_IMPORT
ROLE_ADMIN_MANAGE_CONFIG_READ

Client Roles: realm-management

Available Roles

Assigned Roles
query-realms
query-users
realm-admin
view-authorization
view-realm

Effective Roles
create-client
impersonation
manage-authorization
manage-clients
manage-events

Was this page helpful?