

Documentation EN COURS

Table des matières

Rappel.....	1
Spring.....	9
Keycloak :	9
Interaction Keycloak Open API.....	22
Export/Import keycloak.....	23
Log Keycloak	28
Configuration Swagger/Postman/Keycloak (à destination des développeurs).....	31
L'interaction entre Keycloak et le client Swagger :	31
Traitement du Thème Keycloak	47
VAULT	59
Lancement de Keycloak au travers d'un Docker file :	59

Rappel

Keycloak est un outil de gestion des identités et d'authentification. Il est porté par Red Hat et est open-source. Cet outil est utilisé pour le SSO, il implémente les standards OAuth 2.0 avec les protocoles sous-jacents **OpenID** ou **SAML**.

OAuth 2.0 défini, pour son mécanisme, 4 acteurs :

Le Resources Owner : Ceci correspond le plus souvent à un **utilisateur final** Recolnat. Il s'agit d'une personne physique (Remarque : il peut s'agir aussi d'un service Web, authentifié avec les informations d'identification du client).

Le Resource Server : c'est une API (le plus souvent REST) chargée de servir les ressources. Dans le cadre Recolnat, ce sont les micro-services **collection-manager**, **authorisation** Comme Recolnat emploie Spring, ce sont des applications avec @RestController (ou @Controller avec @ResponseBody).

Le Client : c'est un logiciel qui doit accéder aux ressources via un ou plusieurs serveurs de ressources (**Resources Owner**) ; il le fait en son propre nom (avec les informations d'identification du client) ou au nom d'un utilisateur final pour lequel il dispose d'un jeton d'accès. Dans le cadre Recolnat, c'est l'**application front End** (en React) qui fait office de client.

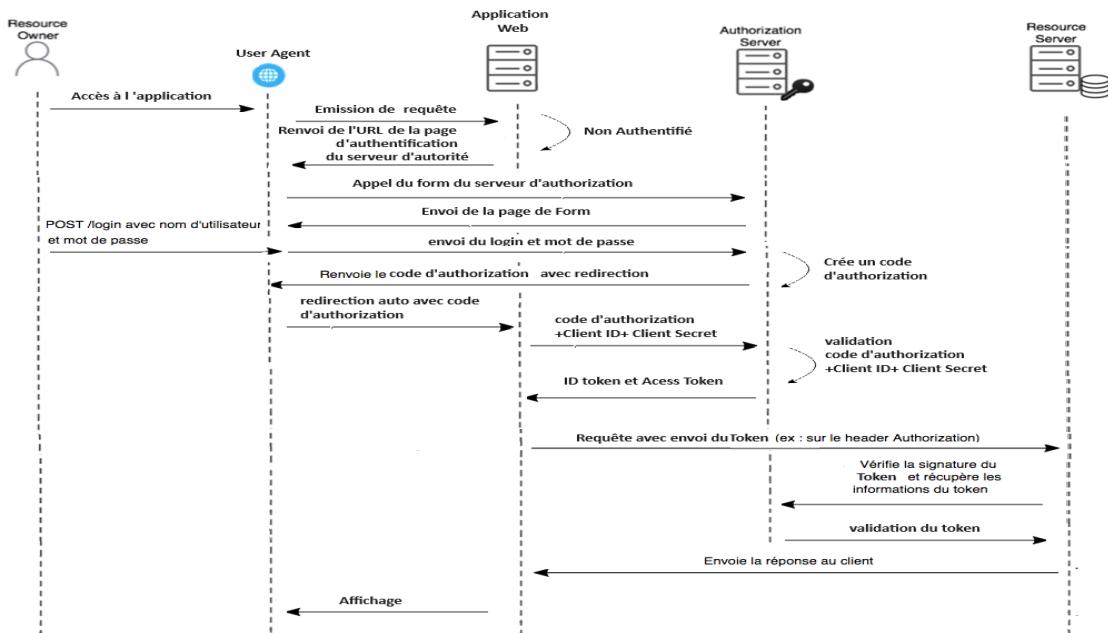
Authorization Server : c'est le serveur délivrant et certifiant les identités et la délégation d'identité (ce qu'un utilisateur final a autorisé à un client à faire en son nom). Dans le cadre Recolnat, c'est le serveur **Keycloak**.

Il existe un cinquième acteur « technique ».

Le User Agent : Dans le cas d'une application Web, le User Agent est le navigateur Web utilisé pour accéder à l'application. Pour une application mobile, le User Agent est le système d'exploitation du téléphone mobile. Dans les deux scénarios, le User Agent est un acteur que l'application ne contrôle pas. Cet acteur va faire l'interface technique entre le **Resources Owner** (l'utilisateur) et l'**Authorization Server**

Authorization Code Grant Type Flow (le plus souvent employé sur des architectures N tiers, avec Application Web classique. Remarque : Dans le cadre d'application monolithique classique. Le serveur applicatif fait office à la fois d'application web et de resource owner).

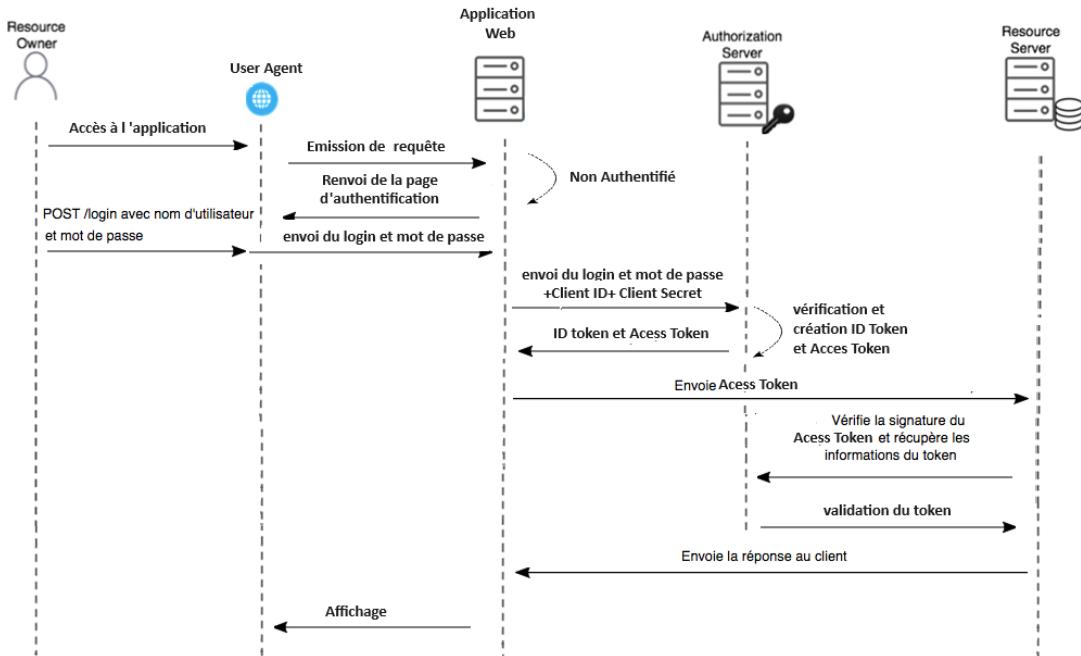
Ce mécanisme d'authentification, va appeler, au travers du user agent (browser), l'IHM d'authentification du serveur d'autorité (ici, Keycloak), puis l'utilisateur remplit le formulaire (login/mot de passe). Lorsque le serveur d'autorité va recevoir le formulaire, il va alors vérifier l'identité de l'utilisateur, puis lui attribué une session et retourné vers le user agent (browser) un code d'autorisation (le redirect est effectué via l'url transmise dans l'envoie du formulaire). Ensuite, au travers du user agent (browser), un appel est réalisé vers le serveur applicatif. Cet appel inclut le code d'autorisation. Le serveur applicatif va transmettre le code d'autorisation au serveur d'autorité qui va le validé et lui retourné un token. Lors de la suite des échanges entre la partie front et le serveur applicatif, le token est transmis dans le header des appels. Le serveur applicatif est en charge, par la suite, lors de chaque échange, de s'assurer de la validité du token, en le transmettant au serveur d'autorité.



Resource Owner Password Flow : (cadre où le client (application web) et le resource owner ont un fort lien de confiance. Permettant au resource owner de lui transmettre ses crédentails (login/pwd))

Ce mécanisme d'authentification, à la différence du mécanisme d'« authorization Code Grant Type Flow », ne s'appuie pas sur le formulaire d'authentification du serveur d'autorité, mais sur le formulaire d'authentification du serveur applicatif. (C'est pour cela que l'on indique une notion de lien de confiance)

Ce mécanisme d'authentification, va appeler, au travers du user agent (browser), l'IHM d'authentification du serveur applicatif, puis l'utilisateur remplit le formulaire (login/mot de passe). Lorsque le serveur applicatif va recevoir le formulaire, il peut éventuellement le validé, puis va alors transmettre l'identité de l'utilisateur au serveur d'autorité, qui va lui attribuer une session et renourné un token. Lors de la suite des échanges entre la partie front et le serveur applicatif, le token est transmis dans le header des appels. Le serveur applicatif est en charge, par la suite, lors de chaque échange, de s'assurer de la validité du token, en le transmettant au serveur d'autorité.



Ex d'appel curl :

```
$ curl --request POST \
--url 'http://localhost:8180/realm/recolnat-oauth2/protocol/openid-connect/token' \
--header 'content-type: application/x-www-form-urlencoded' \
--data username=respinsttest1@recolnat.com \
--data password=test \
--data grant_type=password \
--data client_id=recolnat-oidc \
--data client_secret=ee5jLU9Z27S1SNyusPI5o8ODmcjJ0Mw5 \
>
```

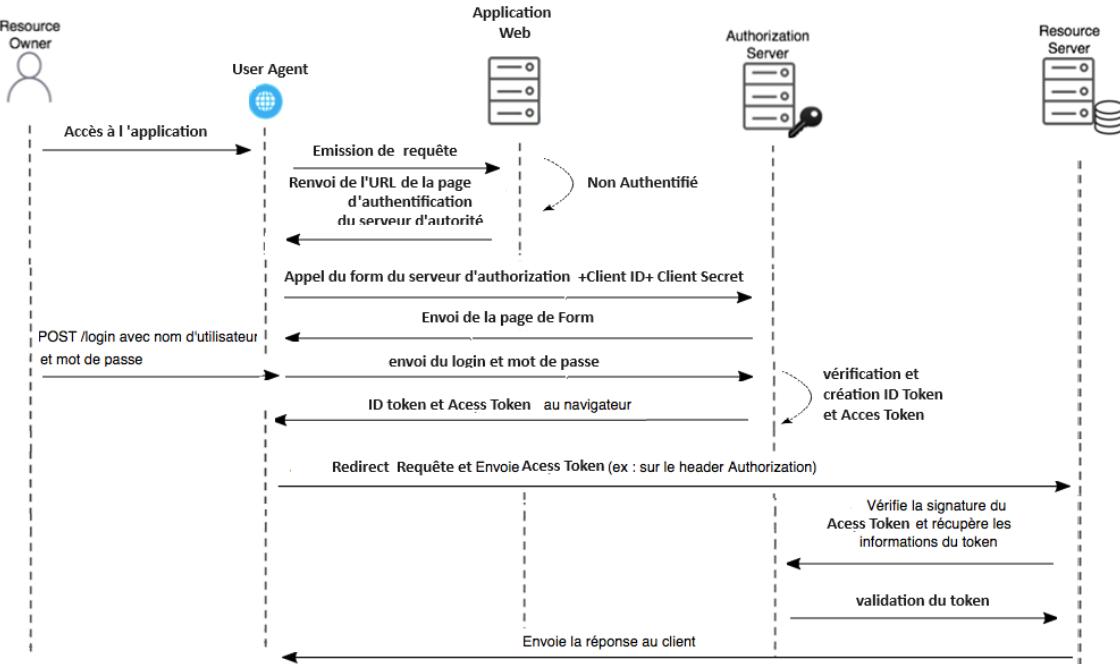
REPONSE :

% Total	% Received	% Xferd	Average Speed	Time	Time	Time	Current
Dload	Upload	Total	Spent	Left	Speed		
100	2660	100	2520	100	140	21418	1189
22542	{"access_token": "eyJhbGciOiJSUzI1NiIsInR5cClgOiAiSlldUliwia2IkliA6ICJFVGhGaTY1RGhsYmVOOFV1VTZGaHE1RFIEbzY2RHhvRGpyWHhvRVZIUTFvln0.eyJleHAiOjE2OTY3OTgzNjYsImhdCI6MTY5Njc2OTU2NiwianRpIjoiZjA1YzNiOGQtMDRhZC00ZTQxLWjkYEtYjnJzDVIZWRjZTMzIwiwiaXNzIjoiaHR0cDovL2xvY2FsaG9zdDo4MTgwL3JYWytcy9yZWNvbG5hdC1vYXV0aDlLCJhdWQiOjYzWNvbG5hdC1vaWRjliwic3ViljoiYjU3NjmZTUtMWM4Yi00MzhhLThiY2ltN2Q5YTg3N2FmNDZkliwidHlwjoiQmVhcmVylwiYXpwliocmVjb2xuYXQtb2IkYyIsInNlc3Npb25fc3RhdGUiOiI0DhiZTczS00YjklLTRIOtYtYTE1MC1mZmQ1MWQzOTQ5ZTgiLCJhbGxvd2VklW9yaWdpbnMiOlsiaHR0cDovL2xvY2FsaG9zdDozMDAwI0sInJIYWxtX2FjY2Vzcyl6eyJyb2xlcyI6WjBRE1JTl9JTINUSVRVEIPTiJdfSwicmVzb3VyY2VfYWNjZXNzIjp7ImFjY291bnQiOnsicm9sZXMiOlsibWFuYyWdlWFjY291bnQiLCJtYW5hZ2UtYWNjb3VudC1saW5rcylsInZpZXctcHJvZmlsZSJdfXOsInNjb3BljoiZW1haWwgchJvZmlsZSIslnNpZCI6ljU4OGJINzNILTRioWQtNGU5Ni1hMTUwLWZmZDUxZDM5NDIiOCislmluc3RpdHV0aW9uljoiMSIsInVpZCI6InJlc3BpbnN0dGVzdDEiLCJwcmVmZXJyZWRMYW5ndWFnZSI6IkZSIwiZmlyc3RuYW1ljoiiUmVzcG9uc2FibGUIcLJlbWFpbF92ZXJpZmlIzCI6dHJ1ZSwibWFpbCI6InJlc3BpbnN0dGVzdDFAcMvjb2xuYXQuY29tliwibmFtZSI6IlJlc3BvbnnHymxIIFJlc3BJbnN0VGVzdDEiLCJwcmVmZXJyZWRfdXNlcm5hb						

WUiOijyZXNwaW5zdHRlc3QxliwiZ2l2ZW5fbmFtZSI6IjJlc3BvbnNhYmxliwiZmFtaWx5X25hbWUiOijSZXNwSW5zdFRlc3QxliwiZ
W1haWwiOijyZXNwaW5zdHRlc3QxQHJIY29sbmF0LmNvbSISlmxhc3RuYW1IjoiUmVzcEluc3RUZXN0MSJ9.JmW2SydE1AZ4_w
OvvNZU9C1SuXjc41712rInINuy-
Gugdvi7RBjBErmoFnK7QOANoBkpQH_7j29mxwMAYDuFjPzgpqKJmc9uQpradUPi4Dowlwtt6iebwHJdoHNRdXp-SQkkL-
byDM_SKDJgVx2YWf2yw0K-onz5I56xrukwldXHAS9rwpnEFh9rHWcx5VUUzFp-
fUILUXU3kjxc2qiaAC1qbUGB48uQDJBH1MkV_ABu3dQH781TFIdtGJ0TCBX7sXvaKk5rbGrANzFB1pEaSdfs619KI5UMNZ7ZgT
wsXzFPBKoNO4qYbxGQy7K_XScn2blZt1rP27865rnFUKTrA","expires_in":28800,"refresh_expires_in":1800,"refresh_token":"
eyJhbGciOiJIUzI1NilsInR5cClgOiAiSlidUliwiia2IkliA6ICJjZGZjNWYzNi1hMWU5LTQ4MjgtYmZjMi0zYTEwMDgyYzg2NjifQ.eyJleH
AiOjE2OTY3NzEzNjYsImIhdCI6MTY5Njc2OTU2NiwanRpljoiMTA3MTJmNmEtNTg3Ni00ZjI1LWI3NzctMzdkYWY5NmE3OGUwl
iwiaXNzljoiaHR0cDovL2xvY2FsaG9zdDo4MTgwL3JYWyxtcy9yZWNvbG5hdC1vYXV0aDliLCJhdWQiOjodHRwOi8vbG9jYWxob3
N0OjgxODAvcmVhbG1zL3JY29sbmF0LW9hdXR0MilsInN1Yi6ImI1NzYyZmU1LTFjOGItNDM4YS04YmNiLTdkOWE4NzdhZjQ2Z
ClslnR5cCl6IjJlZnJlc2giLCJhenAiOijyZWNvbG5hdC1vaWRjliwic2Vzc2lvbl9zdGF0ZSI6IjU4OGJINzNILTRiOWQtNGU5Ni1hMTUw
LWZmZDUxZDM5NDIIOClslnNjb3BljoiZW1haWwgchJvZmlsZSlslnNpZCl6IjU4OGJINzNILTRiOWQtNGU5Ni1hMTUwLWZmZDU
xZDM5NDIIOCJ9.mx7ZN97wBhPfbaPQ8gVCC55OzhDh8-IXdy7vcu-YMac","token_type":"Bearer","not-before-
policy":0,"session_state":"588be73e-4b9d-4e96-a150-ffd51d3949e8","scope":"email profile"}

Implicit flow (emploi dans le cadre d'application de type SPA, (React, Angular). Le UserAgent et l'application Web forment un même bloc). Cadre RecolNat.

Ce mécanisme d'authentification est une version « simplifié » du mécanisme d'« authorization Code Grant Type Flow ». Il va appeler, au travers du user agent (browser), l'IHM d'authentification du serveur d'autorité (ici, Keycloak), puis l'utilisateur remplit le formulaire (login/mot de passe). Lorsque le serveur d'autorité va recevoir le formulaire, il va alors vérifier l'identité de l'utilisateur, puis lui attribué une session et retourné, directement, vers le user agent (browser) le token, sans passer par l'étape intermédiaire du code d'autorisation (le redirect est effectué via l'url transmise dans l'envoi du formulaire). Ensuite, au travers du user agent (browser), un appel est réalisé vers le serveur applicatif. Cet appel inclut le token qui est transmis dans le header de l'appel. Lors de la suite des échanges entre la partie front et le serveur applicatif, le token sera toujours transmis dans le header des appels. Le serveur applicatif est en charge, par la suite, lors de chaque échange, de s'assurer de la validité du token, en le transmettant au serveur d'autorité.



Client Credentials Flow (Dans ce cadre, il n'y a pas de notion de resources Owner (d'utilisateur physique))

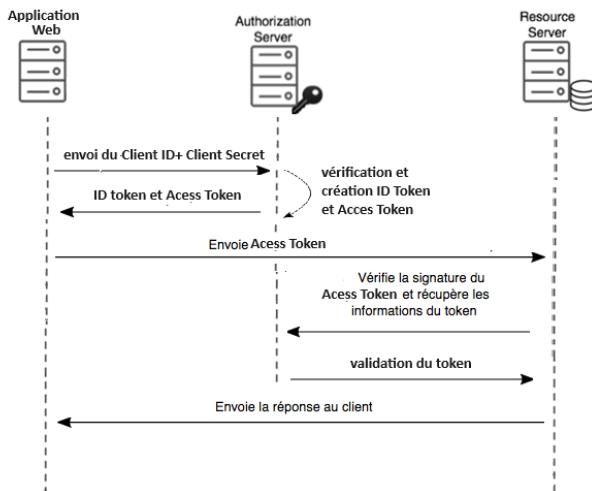
Ce mécanisme d'authentification ne s'applique que pour l'interaction de machine à machine. L'exemple le plus simple est l'emploi de Postman, en vous mettant en client credentials.

The screenshot shows the Postman interface with the 'Authorization' tab selected. The configuration includes:

- Grant type: Client Credentials
- Access Token URL: https://recolnat-test.mnnh.fr/auth/realms/recolnat-oauth2/protocol/openid-connect/token
- Client ID: ITjmYLGADlbOj0haHbj0PKHD1x1CDInC
- Client Secret: (redacted)
- Scope: e.g. read:org
- Client Authentication: Send as Basic Auth header

Vous allez demander, au travers d'une URL (comprenant un client ID et un client secret), un token. Un token vous est alors transmis par le serveur d'autorité, que vous exploitez lors de la suite des échanges avec le serveur applicatif. Le serveur applicatif est en charge, par la suite, lors de chaque échange, de s'assurer de la validité du token, en le transmettant au serveur d'autorité. Attention, ceci implique trois choses, premièrement que le client Keycloak soit configuré pour accepter les client credentials. Deuxièmement, que l'url de retour soit aussi configurée sur le client Keycloak (exemple : dans le cadre de Postman, c'est http://localhost :3000). Et troisièmement, que le token résultant est « anonyme », celui-ci ne contiendra que l'information comme quoi, le système appelant est « authentifié », mais sans plus d'informations.

Remarque : ne confondez pas avec l'interaction entre micro-services. Un micro-service, reçoit une requête de la partie front, il transmet le token au serveur d'autorité afin de s'assurer de la validité du token, puis exécute son traitement. Si le traitement implique un appel à un autre micro-service, il envoie sa requête au second micro-service en lui transmettant dans le header, le token. Et le second micro-service s'assurera aussi de la validité du token. Mais le token sera un token avec un utilisateur authentifié.



Ex d'appel curl :

```
$ curl --request POST \
--url 'http://localhost:8180/realm/recolnat-oauth2/protocol/openid-connect/token' \
--header 'content-type: application/x-www-form-urlencoded' \
--data grant_type=client_credentials \
--data client_id=collection-manager-oidc \
--data client_secret=ee5jLU9Z27S1SNyusPI5o8ODmcjJ0Mw5 \
>
```

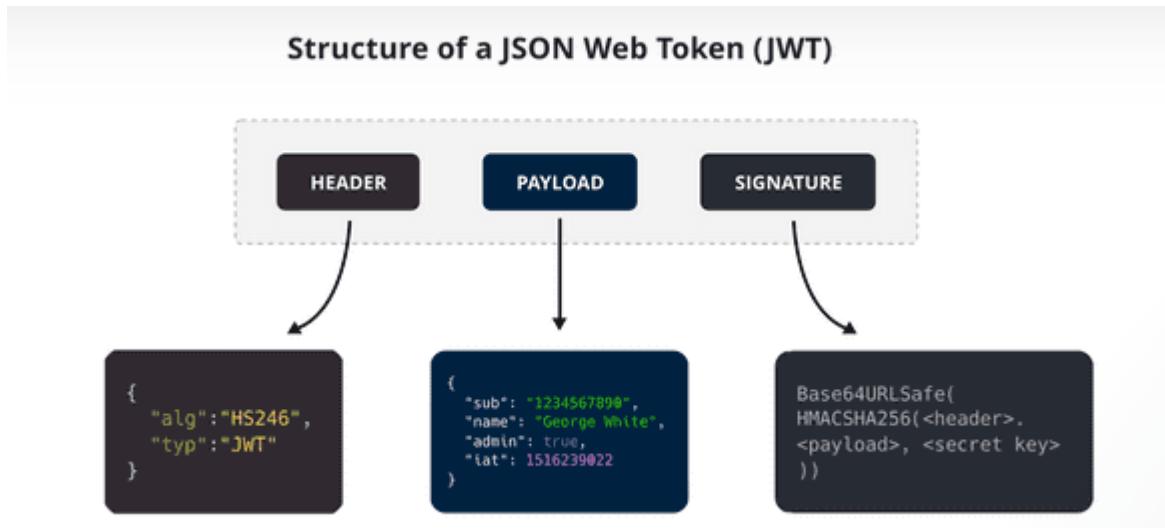
REPONSE

% Total	% Received	% Xferd	Average Speed	Time	Time	Time	Current
Dload	Upload	Total	Spent	Left	Speed		
100	1550	100	1440	100	110	6871	524
7416	"access_token": "eyJhbGciOiJSUzI1NiIsInR5cIgOiAiSldeUliwia2IkliA6ICJFVGhGaTY1RGhsYmVOOFV1VTZGaHE1RFIEbzY2RHhvRGpyWHhvRVZIUTFvln0.eyJleHAiOjE2OTY3OTg3OTksImhdCI6MTY5Njc2OTk5OSwanRpIjoiNzQwNzc0MDEtZTQ1My00N2M0LTk3OGItNTA1YjliMzI3YTE1liwiaXNzljoiaHR0cDovL2xvY2FsaG9zdDo4MTgwL3JYWxtcy9yZWNvbG5hdC1vYXV0aDliLCJhdWQiOjyZWNvbG5hdC1vaWRjliwic3ViljoiMDUxMjA2NDYtZjVmMi00NjNhLWEwOWItNTljZmMzYTIxZGNjliwidHlwIjoiQmVhcmVylowiYXpwlijoiY29sbGvjdGlvbi1tYW5hZ2VvLW9pZGMiLCJyZWFSbV9hY2Nlc3MiOnsicm9sZXMiOlsizGVmYXVsdc1yb2xlcY1yZWNvbG5hdC1vYXV0aDliLCJvZmZsaW5IX2FjY2VzcyIsInVtYV9hdXRob3JpemF0aW9ull19LCJyZXNvdXjzV9hY2Nlc3MiOnsiYWNjb3VudCl6eyJyb2xlcyl6WyJtYW5hZ2UtYWNVjb3VudCIsIm1hbmfnsZs1hY2NvdW50LWxpbtzliwidmldy1wcm9maWxlll19fSwic2NvcGUIoJlbWFpbCBwcm9maWxliwiZW1haWxfdmVyaWZpZWQiOmZhbHNILCjbGllbnRJZC16lmNvbGxIY3Rpb24tbWFuYVwdlc1vaWRjliwiY2xpZW50SG9zdCl6ljE3Mi4yOC4wLjEiLCJwcmVmZXJyZWRfdXNlcm5hbWUiOjzXJ2aWNlWFjY291bnQtY29sbGVjdGlvbi1tYW5hZ2VvLW9pZGMiLCJjbGllbnRBZGRyZXNzljoimTcylj4LjAuMSJ9.VHaZAd84ro5wmrn382AZQRfYQ8g9dXrfYwZcbyh_bElKuBOZif84-wDX_hv12PK2mPZs7fkKyD-Vl-dJM9Lju6HA4qTc5PsAznn-RbAYHOkyF3ySvftlg_i_gvST4qk5usTMly3nJDKwZZSxOX0dfEjr9hplqlernN3MMXjxv_GzpLWb_U3iL41AKmD1UX8S2IQ2gUyN7rjcegMxNj9nuVQ7KSRVKXIw6TM8dPS0V7roSFenjpcz4LwbxzZ8PpH33yDCqW1TzmqUsKtkAnaSDsUewEEJmBc9Us30Ra9gWuXR-HX8yQm3sRWphPfRKDPxsDKLU_LvOFdoP3OvaLu84A", "expires_in": 28800, "refresh_expires_in": 0, "token_type": "Bearer", "not-before-policy": 0, "scope": "email profile"}						

Difference OAUTH2 et OIDC

TODO

JWT Token :



Rappel un token JWT (Token ID) est constitué de 3 parties : un header, un payload/body (contenant des claims) et une signature.

Le header : contient les informations pour les opérations de cryptage. Le plus souvent l'algorithme de hachage et cryptage (HMAC... SHA256, RSA...) et le type du JWT

Le payload : contient les déclarations sur l'entité (utilisateur) et des attributs supplémentaires (Claims).

La signature : employé pour vérifier que l'expéditeur du JWT est bien celui qu'il dit être et pour garantir que le message n'a pas été modifié. La création de la signature se fait par encodage en base 64 du header et du payload, puis de la concaténation des deux encoding avec un point séparateur et le tout est encrypté avec l'algo défini dans le header et une clé secrète.

Attention, dans le cadre d'un JWT classique, le header et le payload reste en claire. La signature n'a qu'un but de non altération du token. Il existe d'autres type de JWT que l'on abordera brièvement, car dans le cadre de Recolnat, l'emploi du token n'est qu'à des fins d'authentification et non d'autorisation, ni d'accès aux infos/profils de l'utilisateur.

Les Claims :

Les claims réservés (par défaut présent dans JWT)

- iss (issuer : émetteur) : émetteur du JWT
- sub (subject) : Object du JWT (l'utilisateur)
- aud (audience):Destinataire du JWT
- exp (délai d'expiration) : délai après lequel le JWT expire
- nbf (pas avant time) : heure avant laquelle le JWT ne sera accepté pour être traiter
- iat (heure d'émission) : heure d'émission du JWT (permet de définir l'âge du JWT)
- jti (identifiant du JWT) : identifiant unique du JWT (unicité d'emploi du JWT)

Liste des claims génériques publiques officiel reconnus par la IANA : <https://www.iana.org/assignments/jwt/jwt.xhtml>

On peut aussi ajouter des claims personnalisés, ce point sera vu dans la partie de Keycloak, sur la partie [Les scopes client keycloak](#) (pour remarque : Les claims personnalisés, spécifiques à une application sont dits privés)

Remarque : Dans Keycloak, vous pouvez, en tant qu'administrateur sur un realm déjà existant, voir/tester les émission de token pour un client donné (pour un client Keycloak ayant à minima un « Authentication flow » ou l'authentification d'actif) :

KEYCLOAK

Clients > Client details

collection-manager-oidc (OpenID Connect)

Enabled Action

Client scopes

Manage

Clients

Realms

Users

Groups

Sessions

Events

Configure

Realm settings

Authentication

Identity providers

User federation

Scope parameter: openid Select scope parameters

Users: respcoltest1

Search protocol mappers: respinsttest2

audience	roles	Token mapper	0
realm roles	roles	Token mapper	40
given name	profile	Token mapper	0
email	email	Token mapper	0

Generated ID token

Generated user info

KEYCLOAK

Clients > Client details

collection-manager-oidc (OpenID Connect)

Enabled Action

Client scopes

Manage

Clients

Realms

Users

Groups

Sessions

Events

Configure

Realm settings

Authentication

Identity providers

User federation

Scope parameter: openid Select scope parameters

Users: respcoltest1

```
exp: 1696916064,
"iat": 1696887264,
"jti": "9ff72446-5388-49a4-b9ff-10f6f766d70",
"iss": "http://localhost:8089/realms/recolnat-oauth2",
"aud": "recolnat-oidc",
"sub": "82e20227-b0d7-46b4-b44d-2257d86f67b1",
"typ": "Bearer",
"azp": "collection-manager-oidc",
"session_state": "34eb43d8-ld87-4la0-8202-56f60e9c065c",
"resource_access": {
  "account": {
    "roles": [
      "manage-account",
      "manage-account-links",
      "view-profile"
    ]
  }
},
"scope": "openid email profile",
"sid": "34eb43d8-ld87-4la0-8202-56f60e9c065c",
"email_verified": true,
```

Effective protocol mappers

Effective role scope mappings

Generated access token

Generated ID token

Generated user info

KEYCLOAK

Clients > Client details

collection-manager-oidc (OpenID Connect)

Enabled Action

Client scopes

Manage

Clients

Realms

Users

Groups

Sessions

Events

Configure

Realm settings

Authentication

Identity providers

User federation

Scope parameter: openid Select scope parameters

Users: respcoltest1

```
{
  "exp": 1696916064,
  "iat": 1696887264,
  "auth_time": 0,
  "jti": "9dcae70-8e0b-4dca-bc74-48640159381c",
  "iss": "http://localhost:8089/realms/recolnat-oauth2",
  "aud": "collection-manager-oidc",
  "sub": "82e20227-b0d7-46b4-b44d-2257d86f67b1",
  "typ": "ID",
  "azp": "collection-manager-oidc",
  "session_state": "c29d3ae1-753d-4d37-9501-9f8a321cd867",
  "sid": "c29d3ae1-753d-4d37-9501-9f8a321cd867",
  "email_verified": true,
  "name": "RespCollTest1 Responsible of collection Test1",
  "preferred_username": "respcoltest1",
  "given_name": "RespCollTest1",
  "family_name": "Responsible of collection Test1",
  "email": "respcoltest1@recolnat.com"
}
```

Effective protocol mappers

Effective role scope mappings

Generated access token

Generated ID token

Generated user info

The screenshot shows the Keycloak administration interface. On the left, a sidebar menu is visible with various options like Manage, Clients, Client scopes, Realm roles, Users, Groups, Sessions, Events, Configure, Realm settings, Authentication, Identity providers, and User federation. The main content area has a header stating 'This page allows you to see all protocol mappers and role scope mappings'. It includes search fields for 'Scope parameter' and 'Users', both set to 'respcoltest1'. Below this, there is a JSON representation of a user object:

```
{
  "sub": "62ee20227-b0d7-46b4-b44d-2257d66f67bf",
  "email_verified": true,
  "name": "RespCollTest1 Responsible of collection Test1",
  "preferred_username": "respcoltest1",
  "given_name": "RespCollTest1",
  "family_name": "Responsible of collection Test1",
  "email": "respcoltest1@recolnat.com"
}
```

On the right side, there are several links: 'Effective protocol mappers', 'Effective role scope mappings', 'Generated access token', 'Generated ID token', and 'Generated user info' (which is highlighted with a yellow box).

Spring :

Dans la librairie de sécurité Spring (employé par Recolnat), il correspondra à ce que l'on appelle le principal dans le contexte de sécurité.

Keycloak :

Keycloak

Comme Keycloak implémente les standards OAuth 2.0, que pour les différents services, on délègue l'identification par Keycloak (serveur d'autorité). Pour remarque, d'autres identity provider, qui sont proposés par keycloak, existent comme Google, GitHub, Facebook ou encore Twitter.

Vous trouverez les différentes configurations, par défaut, proposé par Keycloak, ici : <https://github.com/keycloak/keycloak-quickstarts>.

Vous trouverez aussi le repository git Keycloak, ici : <https://quay.io/repository/keycloak/keycloak?tab=tags&tag=latest>

Installation Locale :

Actuellement, dans les différents projets Recolnat

TODO

Realm :

Le realm recolnat-oauth2

Un realm recolnat-oauth2 gère un ensemble de clients, d'utilisateurs, d'informations d'identification, de rôles et de groupes. Un client ou un utilisateur appartient à un realm et s'y connecte. Un Keycloak peut avoir plusieurs realms mais les realms sont isolés les uns des autres et ne peuvent gérer et authentifier que les utilisateurs ou clients qu'ils contrôlent.

Les clients (à distinguer des utilisateurs) définis pour le realm recolnat-oauth2

Client ID	Enabled	Base URL	Actions
account	True	http://localhost:8080/realm/recolnat-oauth2/account/	Edit Export Delete
admin-console	True	http://localhost:8080/realm/recolnat-oauth2/console/	Edit Export Delete
admin-oidc	True	Not defined	Edit Export Delete
authorization-oidc	True	Not defined	Edit Export Delete
broker	True	Not defined	Edit Export Delete
collection-manager-oidc	True	Not defined	Edit Export Delete
datavene-oidc	True	Not defined	Edit Export Delete
realm-management	True	Not defined	Edit Export Delete
recruit-oidc	True	Not defined	Edit Export Delete
security-admin-console	True	http://localhost:8080/admin/recolnat-oauth2/console/	Edit Export Delete

Les clients sont des entités qui peuvent demander à Keycloak d'authentifier un utilisateur. Le plus souvent, les clients sont des applications et des services qui souhaitent utiliser Keycloak pour se sécuriser et fournir une solution d'authentification unique. Les clients peuvent également être des entités qui souhaitent simplement demander des informations d'identité ou un jeton d'accès afin de pouvoir appeler en toute sécurité d'autres services sur le réseau sécurisé par Keycloak.

Chaque client Keycloak va être associé à un micro-service (non requis). Par exemple, dans le cadre de Recolnat, vous avez le micro-service « Authorisation » et « Collection-Manager », vous allez avoir une déclaration de client associé. (Avec l'extension « -oidc »). Les clients peuvent être configurés différemment au sein de Keycloak.

Comme vous pouvez le voir en dessous, les clients « Authorisation-oidc » et « Collection-Manager-oidc » sont actifs mais n'ont que le service account d'actif. C'est-à-dire qu'à partir de ces clients vous ne pourrez avoir que des jetons d'accès

Les jetons d'accès sont des jetons opaques conformes au framework OAuth 2.0. Ils ne contiennent que des informations d'autorisation (ce que le service peut faire), mais aucunes informations d'identité (d'un utilisateur). Les jetons d'accès sont dits opaques de par leur format (qui peut être une simple chaîne) et ne sont pas exploitable. A noter que dans le cadre de Keycloak, ils ont un format JWT. Attention faites bien la différence entre un accès Token et un ID token

Comme on l'a indiqué précédemment, Chaque client détient un ID unique, un statut actif et un service account actif (qui permet au Keycloak d'authentifier un service et de permettre l'obtention d'un acces token (Client credentials Grant) pour ce client).

Exemple d'appel pour un token d'accès :

Format :

```
curl --request POST \
--url 'https://[yourDomain]/oauth/token' \
--header 'content-type: application/x-www-form-urlencoded' \
--data grant_type=client_credentials \
--data client_id=YOUR_CLIENT_ID \
--data client_secret=YOUR_CLIENT_SECRET \
--data audience=YOUR_API_IDENTIFIER_OPTIONAL
```

```
curl --request POST \
--url 'http://localhost:8089/realm/recolnat-oauth2/protocol/openid-connect/token' \
--header 'content-type: application/x-www-form-urlencoded' \
--data grant_type=client_credentials \
--data client_id=collection-manager-oidc \
--data client_secret=ee5jLU9Z27S1SNyusPl5o8ODmcj0Mw5 \
```

```
$ curl --request POST /c/workspace3/recolnat-backend-stack (R2ARC-21)
--url 'http://localhost:8089/realm/recolnat-oauth2/protocol/openid-connect/token' \
--header 'content-type: application/x-www-form-urlencoded' \
--data grant_type=client_credentials \
--data client_id=collection-manager-oidc \
--data client_secret=ee5jLU9Z27S1SNyusPl5o8ODmcj0Mw5 \
> % Total  % Received % Xferd Average Speed Time Time Current
>          Dload Upload Total Spent Left Speed
100 1564 100 1454 100 110 19177 1450 - - - - - 20833["access_token": "eyJhbGciOiJSUHlwiazIiAiAGICFVGhATY1RhGhYnVOOFV1VTZgHHE1RFbxZyRihvRGpW
HhvRV2UTFvIno_eyJtEHAIoJE20TQNjcmM5Imh1G6TYSNjQz0D1MywianRpIjoiYW0SYzhNTUOGmWS00M231LkhyAtODQwYjg3MzfKNDawTiwiAxNzIjoiHR0CoDwL2xvY2FsasSz2D04M0gSL331Ywtxcybz2WvbGShd1vYXVOd1i1
CJhdw10i1yzWNvbGshd1cvawj1iwi1c3Vi1i1o10M2ZT1yMntNzgOY500MGjhLw1hDktMwf=M0DUne30Dkz1i1v1hWz1jotOmVhcmV1i1w1Yxw1jotY29sbvjdglvh1t1yWl9pZ0h1Cjhy31i01x1i1w1cm/hbG1FYWNzX2NzIj1p7InJvb
Gvz1jpb1mrLzmf1b1Qtcms9Z2Xtcmjb2xuvXQtbz1f1dggy1iw1bZmb1lqz1v9h2V1c3m1Lc1b1nfFXV0a9yaxphdGlvj1dfSwim1zb3vY2FyNy1zNzIj1p7Imf1qz1b1nQ0ns1cm9zZm1s1b1fw1d1LwFjY21b1nq1Lc1Yw5h2z1utYw1j1
3vudC1saw5rcy1is1npZ2xctch1Vm1s2z5dfXosInj3b31ljo1Zwh1hwgch1Vzm1s2z1s1m1v1y1s1x3z1cm1maW1k1pmYnxzZsw1Y2xpzW5Q0i1j1jb2xsZwNoaw9uL1hbmFnZxtb2lkYris1mSaavUdhev3Q101x1x1z1u1jEJM4x1i1w1ch1T2
mVycmVx3VzzXjUy11Tjoi2cVydmljZs1h1y2Ndw50LwVbGx1Y3Rp24t1bwFuYw1l1vaRj1w1Y2xpzW5Q0i1j1jb2xsZwNoaw9uL1hbmFnZxtb2lkYris1mSaavUdhev3Q101x1x1z1u1jEJM4x1i1w1ch1T2
l1oeGTfd0C...hzF2ef0H1LV1fxh4nqU-g1i8t_y_ObA7frerP_81tEf2121jB1Qb8wAkwcYm,Vy7wJuEdEz5_zo_1f1Nz1Bq6TTBXG-Y5b4q1m1gV1kuN9hnOE_wiktGFZo-UDhgbkMby1670nSpfri5z21jls9DREUEzoIpav0E7zwXLBcDLe
pxCd2-xYHrhcuaZs1gt14q0NNiG8ndBjkkz1gSyi-0Br-Pe6vL9W9w0Az9nT2ZPfb1ygrDdWACg", "refresh_expires_in": 28800, "refresh_expires_in": "28800", "token_type": "Bearer", "not-before-policy": "0", "scope": "email profile"}
```



The screenshot shows a JSON web token (JWT) analysis interface. On the left, a large yellow box contains the raw JWT string:

```
"eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSl0UIi
wia2IkIiA6ICJFVGhGaTY1RGhsYmV00FV1VTZGa
HE1RF1EbzY2RHhvRGpyWHhvRVZIUTFvIn0..eyJl
eHAiOjE20TY0NjcwMDMsImIhdCI6MTY5NjQzODI
wMywianRpIjoiYWQ5YzhjNTUtOGMwMS00M2JilW
JmYjAtODQwYjg3MzFkNDAwIiwiXNzIjoiaHR0c
DovL2xvY2FsaG9zdDo4MDg5L3J1YWxtcy9yZWNv
bG5hdC1vYXV0aDIiLCJhdWQiOjJyZWNvbG5hdC1
vaWRjIiwic3ViIjoiOWN1ZTJiYmItNzg0YS00MG
JhLWJkNDktMWFmMDU2NmE3ODkzIiwidHlwIjoiQ
mVhcmVyiwiYXpwljoiY29sbGVjdGlvbi1tYWh5h
Z2VylW9pZGMiLCJhY3IiOixIiwiitmVhbG1fYW
NjZXNzIjp7InJvbGVzIjp7IiZmR1bHQtcms9sz
XMcmtmVjb2xuYXQtb2F1dGgyIiwb2ZmbGluZV9h
Y2Nlc3MiLCJ1bWFfYXV0aG9yaXphdGlvbiJdfs
wicmVzb3VyY2VfYWnnjZXNzIjp7ImFjY291bnQi
Onsicm9sZXMiOlsibWFuYWd1LWFjY291bnQiLCJ
tYW5hZZ2utYWNjb3VudCisaW5rcyIsInZpZCtcH
JvZmlsZSJdfX0sInNjb3BlIjoiZWlhaWwgchJy
ZmlsZSIisImVtYWlsX3ZlcmlmaWVkiJpmYWxzZSw
iY2xpZW50SWQiOijb2xsZWN0aW9uLW1hbFnZX
Itb2lkYyIsImNsawWVudEhvc3QiOixNzIuMjEuM
C4xIiwcHJlZmVycmVkX3VzXJuYW1Iijoic2Vy
dmljZS1hY2NvdW50LWNvbGx1Y3Rpb24tbWFuYWh
lc1ivaWRjIiwiY2xpZW50QWRkcmVzcIy6IjE3Mi
4yMS4wLjEif0.BJ12q-
KxaD30b0F8HqMRENtkDqbptEPc3KCXm1Cx8YUV
2PP3JKnSiHcUjSWQtnD6gy2xzJlsHloecGTfd0
Cg_hzfk2eF0HGjLVlfXhj4nqU-
gli8t_yE_0bATfreP_8HtEf2I21JB1Qrb8wAkW
cYmLvY7JuEoEZn5zo_1fN1ZiBq6TTBXG-
Yb5qEmIgV5Iku8N9hn0ZE_WiktTGFZ0-
UDh6ybKfMby167onSpHriSzJ1Js9VD0REUEz0I
pay0E7zWX18cDLepxCdz-
xyRHrcuqXZ9tgI4qONNkG8ndBjkzIqGsyi-
0BrPe6vL9W9w5Az9NTz2FrnbN-ygRdWACg
```

The right side of the interface is divided into three sections: HEADER, PAYLOAD, and VERIFY SIGNATURE.

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "RS256",
  "typ": "JWT",
  "kid": "EThFi65DhlbeNB0UuU6Fhq5DYDo66Dxo0jrXxaEVHQ1o"
}
```

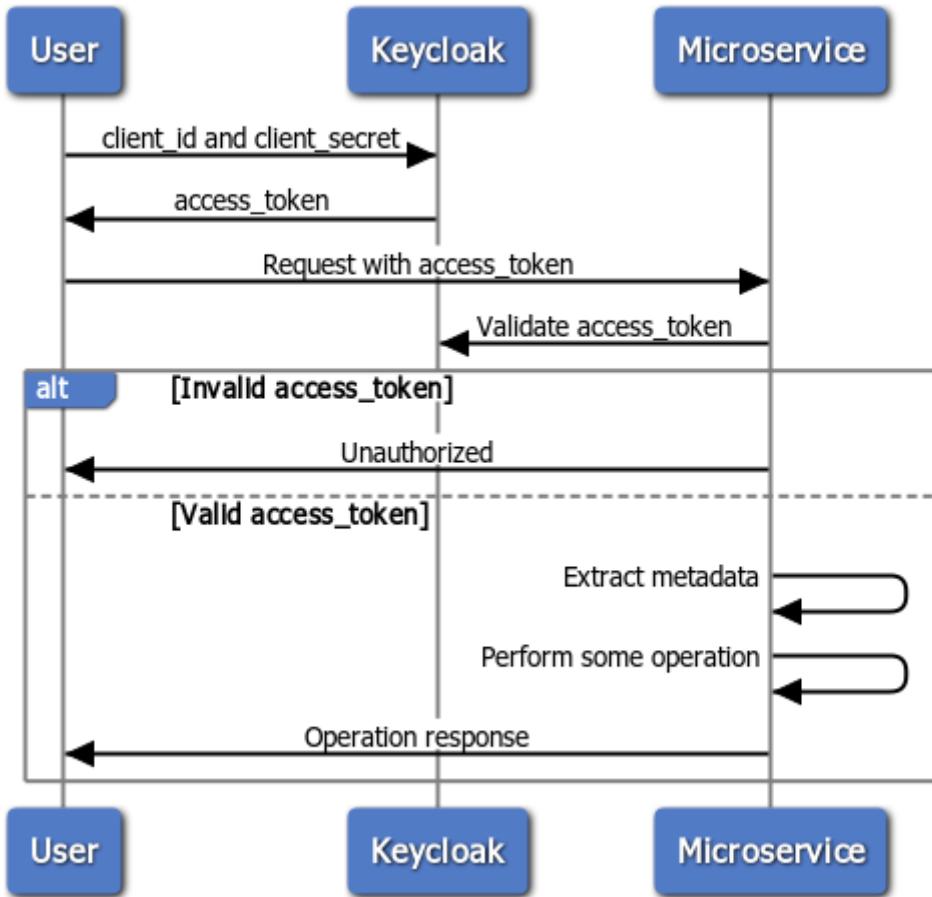
PAYOUT: DATA

```
{
  "exp": 1696407883,
  "iat": 1696438283,
  "jti": "ad9c8c55-8c81-43bb-bfb8-848b8731d400",
  "iss": "http://localhost:8889/realms/recolnat-oauth2",
  "aud": "recolnat-oidc",
  "sub": "9c5e2bbb-784a-48ba-bd49-1af8066a7893",
  "typ": "Bearer",
  "azp": "collection-manager-oidc",
  "acr": "1",
  "realm_access": {
    "roles": [
      "default-roles-recolnat-oauth2",
      "offline_access",
      "uma_authorization"
    ]
  },
  "resource_access": {
    "account": {
      "roles": [
        "manage-account",
        "manage-account-links",
        "view-profile"
      ]
    }
  },
  "scope": "email profile",
  "email_verified": false,
  "clientid": "collection-manager-oidc",
  "clientHost": "172.21.8.1",
  "preferred_username": "service-account-collection-manager-oidc",
  "clientAddress": "172.21.8.1"
}
```

VERIFY SIGNATURE

```
RSASHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  {
    ...
  }
)
```

```
curl --request POST \
--url 'https://{yourDomain}/oauth/token' \
--header 'content-type: application/x-www-form-urlencoded' \
--data username=admintest12@recolnat.com \
--data password=test \
--data grant_type=password \
--data client_id=YOUR_CLIENT_ID \
--data client_secret=YOUR_CLIENT_SECRET \
```



Utilisateur - Les utilisateurs sont des entités capables de se connecter à votre système

Client - Les clients sont des entités qui peuvent demander à Keycloak d'authentifier un utilisateur. Le plus souvent, les clients sont des applications et des services qui souhaitent utiliser Keycloak pour se sécuriser et fournir une solution d'authentification unique. Les clients peuvent également être des entités qui souhaitent simplement demander des informations d'identité ou un jeton d'accès afin de pouvoir invoquer en toute sécurité d'autres services sur le réseau sécurisé par Keycloak.

[Les scopes client keycloak \(todo refaire l intro\)](#)

Les scopes clientOAuth2/ Keycloak, sont à distinguer en partie des claims définis dans un token JWT. Les scopes sont déclarés dans Keycloak, au niveau des clients Keycloak. Les scopes permettent de limiter l'accès d'une application aux données de l'utilisateur. Plutôt que d'accorder un accès complet au compte d'un utilisateur, il est utile de donner aux applications, un moyen de demander une portée plus limitée de ce qu'elles sont autorisées à faire au nom d'un utilisateur. Certaines applications n'emploient Oauth que pour identifier l'utilisateur. D'autres applications peuvent avoir besoin de connaître des informations plus sensibles telle que l'anniversaire de l'utilisateur, ou elles peuvent avoir besoin de connaître des informations.

Exemple :

Création d'un Client Scope

1. Cliquez sur l'ajout d'un scope client

Name	Assigned type	Protocol	Display	Description
acr	Default	OpenID Connect	–	OpenID Connect scope for add acr (authentication context class reference) to the token
address	Optional	OpenID Connect	–	OpenID Connect built-in scope: address
email	Default	OpenID Connect	–	OpenID Connect built-in scope: email
microprofile-jwt	Optional	OpenID Connect	–	Microprofile - JWT built-in scope
offline_access	Optional	OpenID Connect	–	OpenID Connect built-in scope: offline_access

2. Ajout d'un client scope avec le nom Test. De type « Optional » (transmis dans le Token, si le scope est précisé dans la demande (sinon « Défaut », ajout par défaut dans le Token, ou « none », jamais inclus dans le Token)

Create client scope

Name *	<input type="text" value="Test"/>
Description	<input type="text" value="client scope de test"/>
Type	<input type="text" value="Optional"/>
Protocol	<input type="text" value="OpenID Connect"/>
Display on consent screen	<input checked="" type="checkbox"/> On
Consent screen text	<input type="text"/>
Include in token scope	<input checked="" type="checkbox"/> On
Display Order	<input type="text"/>

Save Cancel

3. Ajout d'un mapper (mapping dans le Token du scope)

The screenshot shows the Keycloak administration interface. On the left, a sidebar menu includes 'Manage', 'Clients', 'Client scopes' (which is selected), 'Realm roles', 'Users', 'Groups', 'Sessions', 'Events', 'Configure', 'Realm settings', 'Authentication', 'Identity providers', and 'User federation'. The main content area is titled 'Client scopes > Client scope details' for 'Test (openid-connect)'. Below this, there are tabs for 'Settings', 'Mappers' (which is selected and highlighted in yellow), and 'Scope'. A large 'No mappers' message is centered, with a note: 'If you want to add mappers, please click the button below to add some predefined mappers or to configure a new mapper.' Two buttons are present: 'Add predefined mapper' and 'Configure a new mapper'.

4. Choisir le type de mapper

This screenshot shows a modal dialog titled 'Configure a new mapper'. The dialog lists various mapping types. The 'User Attribute' option is highlighted with a yellow background. Other listed options include 'Claims parameter Token', 'Claims parameter with value ID Token', 'Group Membership', 'Hardcoded claim', 'Hardcoded Role', 'Pairwise subject identifier', 'Role Name Mapper', 'User Address', 'User Client Role', 'User Property', 'User Realm Role', 'User Session Note', and 'User's full name'. Each option has a brief description below it.

Liste des mappers possibles :

Name	Description
Allowed Web Origins	Adds all allowed web origins to the 'allowed-origins' claim in the token
Audience	Add specified audience to the audience (aud) field of token
Audience Resolve	Adds all client_ids of "allowed" clients to the audience field of the token. Allowed client means the client for which user has at least one client role
Authentication Context Class Reference (ACR)	Maps the achieved LoA (Level of Authentication) to the 'acr' claim of the token
Claims parameter Token	Claims specified by Claims parameter are put into tokens.
Claims parameter with value ID Token	Claims specified by Claims parameter with value are put into an ID token
Group Membership	Map user group membership
Hardcoded claim	Hardcode a claim into the token.
Hardcoded Role	Hardcode a role into the access token.
Pairwise subject identifier	Calculates a pairwise subject identifier using a salted sha-256 hash. See OpenID Connect specification for more info about pairwise subject identifiers.
Role Name Mapper	Map an assigned role to a new name or position in the token.
User Address	Maps user address attributes (street, locality, region, postal_code, and country) to the OpenID Connect 'address' claim.
User Attribute	Map a custom user attribute to a token claim.
User Client Role	Map a user client role to a token claim.
User Property	Map a built in user property (email, firstName, lastName) to a token claim.
User Realm Role	Map a user realm role to a token claim.
User Session Note	Map a custom user session note to a token claim.

User's full name	Maps the user's first and last name to the OpenID Connect 'name' claim. Format is <first> + ' ' + <last>
------------------	--

The screenshot shows the Keycloak Admin Console interface. The left sidebar is titled 'Client scopes' and lists various options like Manage, Clients, Client scopes, Realm roles, Users, Groups, Sessions, Events, Configure, Realm settings, Authentication, Identity providers, and User federation. The current page is 'Mapper details' under 'Client scope details' for 'recolnat-oauth2'. The 'Add mapper' section is active, showing a 'User Attribute' type mapper named 'TestMapper'. The configuration includes 'User Attribute' set to 'attributeUser', 'Token Claim Name' set to 'claimTest', and 'Claim JSON Type' set to 'String'. Under 'Add to token' options, 'Add to ID token' is 'On', while 'Add to access token' and 'Add to userinfo' are both 'Off'. Other settings like 'Multivalued' and 'Aggregate attribute values' are also shown. At the bottom are 'Save' and 'Cancel' buttons.

5. Résultat du nouveau scope créé

The screenshot shows the 'Client scopes' list in the Keycloak Admin Console. The left sidebar is identical to the previous screenshot. The main area displays a table of client scopes. The columns are 'Name', 'Assigned type', 'Protocol', 'Display order', and 'Description'. The scopes listed are: 'acr' (Default, OpenID Connect), 'address' (Optional, OpenID Connect), 'email' (Default, OpenID Connect), 'microprofile-jwt' (Optional, OpenID Connect), 'offline_access' (Optional, OpenID Connect), 'phone' (Optional, OpenID Connect), 'profile' (Default, OpenID Connect), 'role_list' (Default, SAML), 'roles' (Default, OpenID Connect), 'Test' (Optional, OpenID Connect), and 'web Origins' (Default, OpenID Connect). The 'Test' scope is highlighted in yellow. The table has a header row and 11 data rows. A 'Create client scope' button is visible at the top of the table.

6. Ajout du scope créé dans le client Keycloak ciblé (attention, ici, on va tester avec un appel de type password, donc il faut que le client ait cette fonctionnalité)

7. Ajout du scope client précédemment créé

Name	Assigned type	Description
address	Optional	OpenID Connect built-in scope: address
email	Default	OpenID Connect built-in scope: email
profile	Default	OpenID Connect built-in scope: profile
roles	Default	OpenID Connect scope for add user roles to the access token
web-origins	Default	OpenID Connect scope for add allowed web origins to the access token

Name	Protocol	Description
test	OpenID Connect	client scope de test
acr	OpenID Connect	OpenID Connect scope for add acr (authentication context class reference) to the token
roles	OpenID Connect	OpenID Connect scope for add user roles to the access token

8. On définit le scope en optional (c'est-à-dire que si le service demandant un token, ne précise pas le scope dans sa demande, alors la valeur (claims) qui va être défini dans l'utilisateur ne sera pas transmise).

Add client scopes to recolnat-oidc

Name	Protocol	Description
Test	OpenID Connect	client scope de test
acr	OpenID Connect	OpenID Connect scope for add acr (authentication context class reference) to the token

Default
Optional

Add Cancel

9. Résultat de l'ajout dans le client

Clients > Client details
recolnat-oidc (OpenID Connect)

Name	Assigned type	Description
recolnat-oidc-dedicated	none	Dedicated scope and mappers for this client
address	Optional	OpenID Connect built-in scope: address
email	Default	OpenID Connect built-in scope: email
microprofile-jwt	Optional	Microprofile - JWT built-in scope
offline_access	Optional	OpenID Connect built-in scope: offline_access
phone	Optional	OpenID Connect built-in scope: phone
profile	Default	OpenID Connect built-in scope: profile
roles	Default	OpenID Connect scope for add user roles to the access tokens
Test	Optional	client scope de test
web Origins	Default	OpenID Connect scope for add allowed web origins to the access token

10. Ajout de l'attribut dans un utilisateur et de la valeur pour l'utilisateur.

Users > User details
respinsttest1

Key	Value
institution	1
preferredLanguage	FR
attributeUser	exemple

Add an attribute

Save Revert

11. Test d'appel de Token

12. Emploi de JWT.io, pour voir le contenu du token

Maintenant, on a défini ce scope client comme Optional, donc si on supprime le scope, dans la demande de token. Il ne sera pas transmis.

Maintenant si vous définissez le scope client comme une valeur par défaut (au niveau du client)

Keycloak Admin UI - Client Details for 'recolnat-oauth2' (OpenID Connect)

Clients are applications and services that can request authentication of a user.

Enabled Action

Name	Assigned type	Description
Assigned client scope	none	Dedicated scope and maps for this client
email	Optional	OpenID Connect built-in scope:email
microprofile-jwt	Optional	Microprofile - JWT built-in scope
offline_access	Optional	OpenID Connect built-in scope:offline_access
profile	Default	OpenID Connect built-in scope:profile
roles	Default	OpenID Connect scope for add user roles to the access token
test	Default	client scope of test
web Origins	Default	OpenID Connect scope for add allowed web origins to the access token

Et relancez le test.

```

        "sat": "1696798888",
        "jti": "d3767cd8-5d5a-4471-ad8c-ed77265bcccd",
        "iss": "http://localhost:8180/realm/recolnat-oauth2",
        "aud": "recolnat-oauth2",
        "sub": "b3762fe5-1cb8-438a-8bcb-7d9e877af46d",
        "typ": "Bearer",
        "azp": "recolnat-oauth2",
        "session_state": "b3b94e62-73d4-4829-83db-c1e078cedcfe",
        "allowed_origins": [
            "http://localhost:3800"
        ],
        "realm_access": {
            "roles": [
                "ADMIN_INSTITUTION"
            ]
        },
        "resource_access": {
            "account": {
                "roles": [
                    "manage-account",
                    "manage-account-links",
                    "view-profile"
                ]
            }
        },
        "scope": "Test_email_profile",
        "sid": "b3b94e62-73d4-4829-83db-c1e078cedcfe",
        "preferredLanguage": "FR",
        "firstname": "Responsable",
        "email_verified": true,
        "email": "respinstest@recolnat.com",
        "client_id": "respinstest",
        "preferred_username": "respinstest1",
        "given_name": "Responsable",
        "last_name": "Respinstest1",
        "institution": "I",
        "uid": "respinstest1",
        "name": "Responsable Respinstest1",
        "family_name": "Respinstest1",
        "email": "respinstest1@recolnat.com"
    }

    VERIFY SIGNATURE
    RSASHA256(
        base64UrlEncode(header) + "." +
        base64UrlEncode(payload),
        {
            "e": "AQAB",
            "kty": "RSA",
            "n": "tN1e05PEc4ZTV3GFcq"
        }
    )

```

Maintenant testons les cas implicite et client credentials

Pour le cas du clients credentials, nous allons prendre l'un des client Keycloak qui ne déclare qu'un service account.

1. Commençons par supprimer le scope du Client précédent

Name	Assigned type	Description
Assigned client scope	none	Dedicated scope and mappers for this client
recolnat-oauth2-dedicated	Optional	OpenID Connect built-in scope: address
address	Default	OpenID Connect built-in scope: email
email	Optional	OpenID Connect built-in scope: offline_access
microprofile-jwt	Optional	Microprofile - JWT built-in scope
offline_access	Optional	OpenID Connect built-in scope: offline_access
phone	Optional	OpenID Connect built-in scope: phone
profile	Default	OpenID Connect built-in scope: profile
roles	Default	OpenID Connect scope for add user roles to the access token
Test	Default	client scope de test
web Origins	Default	OpenID Connect scope for add allowed web origins to the access token

2. Allez sur un client qui ne définit qu'un service account

3. Maintenant, procédez comme dans le cas précédent, pour l'ajout du client scope (en default)

4. Effectué le test (pour rappel, il n'y a pas de notion de resources Owner dans ce type scénario)

(Todo ajout des écrans postman)

Par défaut, Keycloak défini les scopes suivants par défaut pour tout nouveau client : profile, email, roles et web-origins

Interaction Keycloak Open API

TODO

Export/Import keycloak

L'export keycloak au travers de l IHM est partielle (elle n'inclut pas les utilisateurs)

The screenshot shows the 'Partial Export' section of the Keycloak interface. On the left, there's a sidebar with navigation links for 'Configure' (Realm Settings, Clients, Client Scopes, Roles, Identity Providers, User Federation, Authentication) and 'Manage' (Groups, Users, Sessions, Events, Import, Export). The 'Export' link is highlighted with a blue border. The main area has two toggle switches: 'Export groups and roles' (ON) and 'Export clients' (ON). A large blue 'Export' button is at the bottom.

Remarques :

Liste d'informations suite à tentative de lancement de d export au travers du container Keycloak

Accès au container en root (sinon user Keycloak qui n'a pas beaucoup de droits au sein de l'instance).

```
PS C:\Users\ppelle_ext> docker exec -ti -u 0 keycloak bash
```

```
[root@keycloak /]# /opt/keycloak/bin/kc.sh --help
Keycloak - Open Source Identity and Access Management

Find more information at: https://www.keycloak.org/docs/latest

Usage:
kc.sh [OPTIONS] [COMMAND]

Use this command-line tool to manage your Keycloak cluster.
Make sure the command is available on your "PATH" or prefix it with "./" (e.g.:
"./kc.sh") to execute from the current folder.

Options:
--cf, --config-file <file>
      Set the path to a configuration file. By default, configuration properties are
      read from the "keycloak.conf" file in the "conf" directory.
-h, --help
      This help message.
-v, --verbose
      Print out error details when running this command.
-V, --version
      Show version information

Commands:
build          Creates a new and optimized server image.
start          Start the server.
start-dev       Start the server in development mode.
export         Export data from realms to a file or directory.
import         Import data from a directory or a file.
show-config    Print out the current configuration.
tools          Utilities for use and interaction with the server.
completion     Generate bash/zsh completion script for kc.sh.

Examples:
Start the server in development mode for local development or testing:
$ kc.sh start-dev

Building an optimized server runtime:
$ kc.sh build <OPTIONS>

Start the server in production mode:
$ kc.sh start <OPTIONS>

Enable auto-completion to bash/zsh:
$ source <(kc.sh tools completion)

Please, take a look at the documentation for more details before deploying in
production.

Use "kc.sh start --help" for the available options when starting the server.
Use "kc.sh <command> --help" for more information about other commands.
```

Remarque : Dans le cadre sans spécification de base de données. C'est la base H2 qui est employée. Vous la trouverez ici

```
[root@keycloak /]# cd opt/keycloak/data/
[root@keycloak data]# ls
h2  tmp
```

Autre point, l'image de l'OS employé dans la version 17 est du Red Hat 8.5 avec du microdnf.

```
[root@keycloak /]# cd etc
[root@keycloak etc]# ls
GREP_COLORS      bashrc      dnf      group      inputrc      krb5.conf.d      localtime      nsswitch.conf.bak      pe      rc0.d      redhat-release      shadow      swuid      xattr.conf
X11             bindnssport.blacklist      environment      gshadow      issue      ld.so.cache      machine-id      opendns      popt.d      rc1.d      resolv.conf      shadow      syconfig      xdg
aliases        chattr      config      crypt      crypttab      exports      host.conf      java      libnsswitch.conf      ld.so.conf.d      net      openssh      profile      rc2.d      resolv.conf      shadow      systemd-logins      xinetd.d
alternatives   crypt      crypttab      filesystems      hostname      jvm      libaudit.conf      netconfig      passwd      profile.d      rc3.d      rpc      skel      system-release      yum.repos.d
alternatives.conf      crypt      crypttab      hosts      jvm-common      libaudit      networks      pkcs11      protocols      rc4.d      rpsvc      ssl      system-release-cpe
authconfig      crypt      crypttab      init      jvmbase      libaudit      networks      pki      rc5.d      rc6.d      services      subsystem      terminalinfo
authconfig.d      crypt      crypttab      init.d      jvmbase      libaudit      networks      pki      rc6.d      rc7.d      services      subsystem      vcconsole.conf
[root@keycloak etc]# cat os-release
NAME="Red Hat Enterprise Linux"
VERSION="8.5 (Ootpa)"
ID="rhel"
ID_LIKE="fedora"
PRETTY_NAME="Red Hat Enterprise Linux 8.5 (Ootpa)"
PLATFORM_ID="platform:el8"
PRETTY_NAME="Red Hat Enterprise Linux 8.5 (Ootpa)"
ANSI_COLOR="0;31"
REDHAT_CDN="http://mirror.centos.org/centos/8/BaseOS/x86_64/os/"
HOME_URL="https://www.redhat.com/"
DOCUMENTATION_URL="https://access.redhat.com/documentation/red_hat_enterprise_linux/8/"
BUG_REPORT_URL="https://bugzilla.redhat.com/"

REDHAT_BUGZILLA_PRODUCT="Red Hat Enterprise Linux 8"
REDHAT_SUPPORT_PRODUCT="Red Hat Enterprise Linux"
REDHAT_SUPPORT_PRODUCT_VERSION="8.5"
```

La plupart des fonctionnalités sont désactivées (Ex : pas de commande ps, pour voir les processus). Pour installer le ps, si cela vous est nécessaire, vous devez faire la commande suivante : microdnf install procps

```
[root@keycloak ~]# microdnf install procps
(microdnf:802): librsm-WARNING **: 13:42:24.701: Found 0 entitlement certificates
(microdnf:802): librsm-WARNING **: 13:42:24.713: Found 0 entitlement certificates
Downloading metadata...
Downloading metadata...
Downloading metadata...
Package
Installing:
  procps-ng-3.3.15-13.el8.x86_64
Transaction Summary:
  Installing: 1 packages
  Reinstalling: 0 packages
  Upgrading: 0 packages
  Obsoleting: 0 packages
  Removing: 0 packages
  Downgrading: 0 packages
Downloading packages...
Running transaction test...
Installing: procps-ng;3.3.15-13.el8;x86_64;ubi-8-baseos
Complete.
```

```
[root@keycloak ~]# ps aux | grep java
keycloak 1 2.8 5.2 586338 419368 ? Ssl 13:01 1:13 java -Xms64m -Xmx128m -XX:MaxMetaspaceSize=96M -XX:MaxMetaspaceSize=256m -Djava.net.preferIPv4Stack=true -Dkc.home.dir=/opt/keycloak/bin/.. -Djboss.server.conf
ig.dir=/opt/keycloak/bin/..conf -Djava.util.logging.manager=org.jboss.logmanager.LogManager -Dquarkus.log-max-startup-records=10000 -cp /opt/keycloak/bin/..lib/quarkus-run.jar:./quarkus.bootstrap.runner.QuarkusEntryPoint
t=profileDev start-dev --auto-build
root 870 0.0 0.0 16396 1156 pts/0 S+ 13:49 0:00 grep --color=auto java
```

Evitez de faire un « microdnf update », Keycloak ne semble pas appréciée.

Autre remarque : la documentation actuelle, indique ceci.

To export a realm, you can use the `export` command. Your Keycloak server instance must not be started when invoking this command.

Ce point pose différents problèmes, car vous ne pouvez pas arrêter l'instance Keycloak dans le container, sans arrêter celui-ci. (Le pid est à 1. Killer le process, fait tomber le container). Dans tous les cas, ce n'est pas une solution car si on arrête l'instance Keycloak dans le k8s, il y a des chances que le kubelet le relance.

Vous ne pouvez pas non plus, lancer la commande d'export en interne. La commande passe de temps à autre, mais indique principalement que le container est passé à un état peu stable. La plupart du temps, on obtient ceci (Ce point est dû au fait que Keycloak n'a pas été arrêté) :

```
[root@keycloak ~]# /opt/keycloak/bin/kc.sh export --dir ./home --realm Recolnat-oauth2 --users realm_file
2023-10-06 14:14:29,224 INFO [org.keycloak.quarkus.runtime.hostname.DefaultHostnameProvider] (main) Hostname settings: FrontEnd: <request>, Strict HTTPS: false, Path: <request>, Strict BackChannel: false, Admin: <request>2023-10-06 14:14:29,846 WARN [org.infinispan.PERSISTENCE] (keycloak-cache-init) ISPN000554: jboss-marshalling is deprecated and planned for removal
2023-10-06 14:14:29,864 WARN [org.infinispan.CONFIG] (keycloak-cache-init) ISPN000569: Unable to persist Infinispan internal caches as no global state enabled
2023-10-06 14:14:29,891 INFO [org.infinispan.CONTAINER] (keycloak-cache-init) ISPN000556: Starting user marshaller 'org.infinispan.jboss.marshalling.core.JBossUserMarshaller'
2023-10-06 14:14:30,124 INFO [org.infinispan.CONTAINER] (keycloak-cache-init) ISPN000128: Infinispan version: Infinispan 'Triskadekaphobia' 13.0.5.Final
2023-10-06 14:14:30,555 INFO [org.keycloak.exportimport.dir.DirExportProvider] (main) Exporting into directory ./home
2023-10-06 14:14:30,861 INFO [org.keycloak.connections.Infinispan.DefaultInfinispanConnectionProviderFactory] (main) Node name: node_189137, Site name: null
2023-10-06 14:14:31,297 INFO [org.keycloak.services] (main) KC-SERVICES0034: Export of realm 'Recolnat-oauth2' requested.
2023-10-06 14:14:31,558 ERROR [org.keycloak.quarkus.runtime.cli.ExecutionExceptionHandler] (main) ERROR: Failed to start server in (import_export) mode
2023-10-06 14:14:31,558 ERROR [org.keycloak.quarkus.runtime.cli.ExecutionExceptionHandler] (main) For more details run the same command passing the '--verbose' option. Also you can use '--help' to see the details about the usage of the particular command.
```

L'hypothèse d'avoir un container à l'arrêt et de le redémarrer (start) en incluant une commande Shell spécifiant un traitement d'export, n'est pas possible au travers de Docker.

Import help :

```
[root@keycloak ~]# /opt/keycloak/bin/kc.sh import --help
Import data from a directory or a file.

Usage:

kc.sh import [OPTIONS]

Import data from a directory or a file.

Options:

--dir <path>      Set the path to a directory where files will be created with the exported data.
--file <path>      Set the path to a file that will be created with the exported data.
-h, --help          This help message.
--override <false> Set if existing data should be skipped or overridden. Default: true.
--realm <realm>    Set the name of the realm to export
```

Export help :

```
[root@keycloak home]# /opt/keycloak/bin/kc.sh export --help
Export data from realms to a file or directory.

Usage:
kc.sh export [OPTIONS]

Export data from realms to a file or directory.

Options:

--dir <path>      Set the path to a directory where files will be created with the exported data.
--file <path>      Set the path to a file that will be created with the exported data.
-h, --help          This help message.
--realm <realm>    Set the name of the realm to export
--users <strategy> Set how users should be exported. Possible values are: skip, realm_file,
                     same_file, different_files. Default: different_files.
--users-per-file <number>
                     Set the number of users per file. It's used only if --users=different_files.
```

Remarque :

- Les deux commandes ont des paramètres qui ne sont pas nécessaires de préciser car ils ont des valeurs par défaut.
- Il y a différentes possibilités d'export et d'import. Et il est conseillé, dans le cadre, d'un certains nombres de client, de **ne pas employer un fichier unique**. Il est indiqué que l'emploi d'un répertoire est préférable. (Le nombre d'utilisateurs par fichier, par défaut, est de 50)
- Le container (comme la version standalone) ne semble pas être prévu pour être arrêté. (Ce point apparaît à partir de la version 17 où une bascule technique a été faite d'une version Keycloak WildFly (JBoss) vers du quarkus/graalVm).
- Remarque : si vous faites un « kc build », faites derrière un « start - -optimized », pour accélérer le démarrage de l'application Keycloak.

Structure générale des commandes CLI d'import et d'export Keycloak :

Les export et import Keycloak sont sous un format JSON.

L'import et l'export via CLI ne s'applique que dans le cadre ou il y a des utilisateurs définis dans une base de données. De ce fait, Keycloak doit avoir une série d'informations pour pouvoir se connecter à la base (vendor) auquel il a été associé.

Il y a différentes possibilités pour fournir ces informations a Keycloak :

- Soit en ligne de commande, soit dans un fichier de configuration (le format sera alors du type : « --key_défini_pour_keycloak_avec_underscore = valeur ») avec, dans certains cas des abréviations de key « abbreviation=value »
- Soit comme variables d'environnement (pour **ConfigMap**), soit défini dans la configuration du keystore: (le format sera alors du type : « --KC_key_défini_pour_keycloak_avec_underscore = valeur »)

Exemple :

Source	Format
Command line parameters	--db-url=cliValue
Environment variable	KC_DB_URL=envVarValue
Configuration file	db-url=confFileValue
Java KeyStore file	kc.db-url=keystoreValue

Pour avoir la liste des variables de configuration, pour l'ensemble de la configuration Keycloak et de ces providers : <https://www.keycloak.org/server/all-config> . Exemple avec export et import.

Export

	Value
▼ <code>dir</code> Set the path to a directory where files will be created with the exported data. CLI: <code>--dir</code> Env: <code>KC_DIR</code>	
▼ <code>realm</code> Set the name of the realm to export. If not set, all realms are going to be exported. CLI: <code>--realm</code> Env: <code>KC_REALM</code>	
▼ <code>users</code> Set how users should be exported. CLI: <code>--users</code> Env: <code>KC_USERS</code>	<code>skip</code> , <code>realm_file</code> , <code>same_file</code> , <code>different_files</code> (default)
▼ <code>users-per-file</code> Set the number of users per file. It is used only if <code>users</code> is set to <code>different_files</code> . Increasing this number leads to exponentially increasing export times. CLI: <code>--users-per-file</code> Env: <code>KC_USERS_PER_FILE</code>	<code>50</code> (default)

Import

	Value
▼ <code>file</code> Set the path to a file that will be read. CLI: <code>--file</code> Env: <code>KC_FILE</code>	
▼ <code>override</code> Set if existing data should be overwritten. If set to false, data will be ignored. CLI: <code>--override</code> Env: <code>KC_OVERRIDE</code>	<code>true</code> (default), <code>false</code>

Détails sur les commandes d'import et d'export dans le cadre simple d'un Keycloak installé sur une machine physique :

Export de Json (comprenant les realms, roles, client (peut-être optionnel), users, scope)

- Il y a 4 configurations possibles d'exportations :

- Soit exporter dans différents fichiers json (cela s'applique lorsque l'on exporte un grand nombre de users Keycloak). Vous aurez un fichier json pour le master realm, un fichier pour les users du master realm (admin(s) Keycloak), un fichier par realm applicatif et N fichiers contenant un certain nombre d'utilisateurs par realm applicatif (fichier de users dont vous spécifiez le nb max de users par fichier : variable `users-per-file`). Ceci implique de définir un répertoire d'export.

```
bin/kc.[sh|bat] export --dir <dir> --users different_files --
users-per-file 100
```

Vous pouvez ajouter la spécification d'un realm applicatif spécifique

```
bin/kc.[sh|bat] export --dir <dir> --users different_files --
users-per-file 100 --realm le_nom_du_real_a_exporter
```

- Sois-vous définissez de faire le choix de deux fichiers par realm. Un pour le realm et un pour les users Keycloak. Vous aurez un fichier json pour le master realm, un fichier pour les users du master realm

(admin(s) Keycloak), un fichier par realm applicatif et 1 fichier contenant tous les utilisateurs par realm applicatif.

```
bin/kc.[sh|bat] export --dir <dir> --users same_file
```

Vous pouvez ajouter la spécification d'un realm applicatif spécifique

```
bin/kc.[sh|bat] export --dir <dir> --users same_file --realm le_nom_du_real_a_exporter
```

- Sois-vous définissez de faire le choix d'exporter les datas du realm et ses utilisateurs dans un même fichier. Vous aurez un fichier json pour le master realm et les users du master realm (admin(s) Keycloak), un fichier par realm applicatif et tous les utilisateurs par realm applicatif. (Définissez un directory dans le cadre de plusieurs realms)

```
bin/kc.[sh|bat] export --dir <dir> --users realm_file
```

- Dans le cadre où vous n'auriez qu'un fichier de realm, définissez le nom du fichier au lieu de définir un répertoire. (Notamment dans le cadre où vous spécifiez un realm défini)

```
bin/kc.[sh|bat] export --file <file_name> --users same_file
```

Vous pouvez ajouter la spécification d'un realm applicatif spécifique

```
bin/kc.[sh|bat] export --file <file_name> --users same_file --realm le_nom_du_real_a_exporter
```

- Sois-vous ne voulez pas exporter les users, mais uniquement un realm (ou les realms)

```
bin/kc.[sh|bat] export [--dir|--file] <path> --users skip --realm le_nom_du_real_a_exporter
```

Dans le cadre de docker (docker compose) et k8s. Comme indiqué, vous ne pouvez faire un export que dans le cadre, où le Keycloak fonctionne (ceci était encore possible dans la version Jboss mais plus depuis la version 17)

Il faut :

1. Créer une image de l'instance de container Keycloak en cours

Log Keycloak :

Pour définir les traces de log sur Keycloak :

Allez, en tant qu'administrateur, sur la partie « Realm Setting », dans l'onglet « events ». Le premier sous onglet de cette page, permet de définir la gestion des logs soit sous forme de log soit par mailing.

The screenshot shows the Keycloak administration interface for the realm 'recolnat-oauth2'. The left sidebar is dark-themed and lists various management sections like Manage, Clients, Client scopes, etc. The 'Realm settings' section is currently selected. The main content area has a light background. At the top, there's a navigation bar with tabs: General, Login, Email, Themes, Keys, Events (which is highlighted in yellow), Localization, Security defenses, Sessions, Tokens, Client policies, and User registration. Below the tabs, there are two sub-sections: 'Event listeners' and 'User events settings'. Under 'Event listeners', there's a dropdown menu showing 'jboss-logging' and 'email' as options. On the right side of the header, there are buttons for 'Enabled' (which is set to 'Enabled') and 'Action'. The overall layout is clean and modern, typical of the Keycloak web interface.

Le second écran, permet de définir les évènements utilisateurs à prendre en compte dans les logs. La partie « Save Events » va permettre de sauvegarder les logs en base sur une période à définir. Le bouton « Add saved types » va permettre de définir les évènements que vous voulez voir dans vos logs. (Les trois points verticaux sur la droite, permet de supprimer un évènement des logs)

This screenshot shows the 'User events settings' page within the 'Events' section of the Keycloak administration interface. The left sidebar remains the same as the previous screenshot. The main content area has a light background. At the top, it shows the realm name 'recolnat-oauth2' and a 'Learn more' link. Below that is a navigation bar with tabs: General, Login, Email, Themes, Keys, Events (highlighted in blue), Localization, Security defenses, Sessions, Tokens, Client policies, and User registration. Under 'Events', the 'User events settings' tab is selected. On the left, there's a 'Save events' toggle switch set to 'Off'. Below it are 'Save' and 'Revert' buttons. Further down are 'Clear user events' and 'Add user events' buttons. The main right-hand area contains a table titled 'Event saved type' with columns for 'Event saved type' and 'Description'. The table lists various event types such as 'Send reset password', 'Update consent error', 'Grant consent', etc. A search bar at the top of the table allows filtering by event type. The bottom right corner of the table shows a page number '1 - 20' and navigation arrows. The overall design is consistent with the previous screenshot, maintaining a professional and user-friendly look.

Le troisième sous onglet permet de sauvegarder les évènements associés aux actions de l'administrateur (ceux-ci doivent être spécifiquement initialisé. Non actif par défaut)

Vous retrouverez les événements dans l'onglet suivant :

Time	User	Event type	IP address	Client
October 9, 2023 at 11:29 AM	No user details	CLIENT_LOGIN_ERROR	172.28.0.1	recolnat-oidc
October 8, 2023 at 11:22 PM	No user details	CLIENT_LOGIN_ERROR	172.28.0.1	collection-manager-oidc
October 8, 2023 at 10:02 PM	No user details	LOGIN_ERROR	172.28.0.1	recolnat-oidc
October 8, 2023 at 10:01 PM	No user details	LOGIN_ERROR	172.28.0.1	recolnat-oidc
October 8, 2023 at 10:01 PM	No user details	LOGIN_ERROR	172.28.0.1	recolnat-oidc
October 8, 2023 at 10:00 PM	No user details	LOGIN_ERROR	172.28.0.1	collection-manager-oidc
October 8, 2023 at 9:51 PM	No user details	LOGIN_ERROR	172.28.0.1	recolnat-oidc
October 8, 2023 at 9:39 PM	No user details	LOGIN_ERROR	172.28.0.1	recolnat-oidc
October 8, 2023 at 9:39 PM	No user details	LOGIN_ERROR	172.28.0.1	recolnat-oidc
October 8, 2023 at 5:21 PM	No user details	LOGIN_ERROR	172.28.0.1	recolnat-oidc
October 8, 2023 at 5:21 PM	No user details	LOGIN_ERROR	172.28.0.1	recolnat-oidc
October 8, 2023 at 5:19 PM	No user details	LOGIN_ERROR	172.28.0.1	recolnat-oidc

Configuration Swagger/Postman/Keycloak (à destination des développeurs)

L'interaction entre Keycloak et le client Swagger :

Swagger implique différentes briques techniques (SpringDoc, OpenApi /OpenApi-generator (Swagger), Spring Boot/Secu, Keycloak). Ceux-ci doivent être configurés ensemble pour pouvoir communiquer.

Rappel :

OpenApi /OpenApi-generator (Swagger) :

OpenApi est une version normalisée de la spécification Swagger. Il permet de présenter les API REST suivant une approche indépendante de la langue. Vous pouvez générer des représentations OpenApi pour la représentation et l'exécution de vos services. C'est-à-dire comme pour SOAP, vous pouvez générer la partie code client, du « contrat » de vos services, mais éventuellement aussi la partie serveur. La démarche, dans le cadre du projet Recolnat a été de produire la partie serveur à partie d'un fichier de déclaration YAML (emploi possible de format JSON). Le plus souvent le développeur déclare une série d'annotation côté serveur (dans le code Java), ce qui produira un fichier JSON, lors du runtime, qui sera exploité par la partie javascript de OpenApi et produira une page définissant les services (et leurs emploi).

The screenshot shows the Swagger UI interface with the following sections:

- Retrieve collections**:
 - GET /v1/collections Retrieve all collections
 - GET /v1/institutions/{institutionId}/collections Retrieve collections by institution id
 - GET /v1/collections/refresh Collection
- Specimen update Media**:
 - PUT /v1/collections/{collectionId}/specimens/{specimenId}/medias/{mediaId} Specimen Update Media
 - PATCH /v1/collections/{collectionId}/specimens/{specimenId}/medias Specimen Media
 - PATCH /v1/collections/{collectionId}/specimens/{specimenId}/medias/review Specimen Media
 - PATCH /v1/collections/{collectionId}/specimens/{specimenId}/medias/draft Specimen Media
- Specimen integration as draft**:
 - POST /v1/collections/{collectionId}/specimens/draft Specimen integration as draft
- Specimen integration**:
 - POST /v1/specimens/batch Specimen
 - POST /v1/collections/{collectionId}/specimens Specimen integration
 - POST /v1/collections/{collectionId}/specimens/{specimenId}/duplicate/{collectionIdTarget} Specimen integration
 - POST /v1/collections/{collectionId}/specimens/bulk Bulk Specimen integration

Vous pouvez, à partir du Json résultant et l'emploi d'un plugin, produire la partie cliente (pour différents types de clients, comme Angular, React...). [Exemple de plugins :

<https://www.npmjs.com/package/react-openapi-client> , <https://github.com/anttiviljami/react-openapi-client> , ...]

Dans le cadre de Recolnat, les développeurs ne font pas de déclarations annotées, au sein du code, mais emploie un fichier YAML, pour effectuer les déclarations de services. Et ensuite, via l'emploi d'un plugin Maven « openapi-generator-maven-plugin », vont produire les interfaces, avec les annotations OpenApi/Swagger, qui seront à implémenter. (Comme dans le cas précédent, à partir de ces annotations, au runtime, un fichier JSON sera produit et exploité par la partie javascript de OpenApi, pour effectuer l'affichage).

Pour OpenApi, vous trouverez **la syntaxe à employer**, pour écrire le fichier YAML à cette adresse :
<https://spec.openapis.org/oas/latest.html>

Vous la retrouverez dans le GitHub, du projet OpenApi : <https://github.com/OAI/OpenAPI-Specification/blob/main/versions/3.1.0.md> (attention au Markdown de la version)

Vous pouvez, par ailleurs, regarder la documentation d'origine de Swagger, mais attention assurez-vous d'être sur la bonne version lorsque vous naviguez sur le site.

<https://swagger.io/docs/specification/about/>

Vous pouvez aussi employer l'éditeur Swagger, pour valider la syntaxe de votre fichier YAML.

Attention, il y a, actuellement, deux versions. La version courante : <https://editor.swagger.io> , la nouvelle version : <https://editor-next.swagger.io/>

Remarque :

- Vous pouvez trouver différents exemples de fichiers YAML, ici :
<https://github.com/OpenAPITools/openapi-generator/blob/master/samples/client/petstore/java/webclient-jakarta/api/openapi.yaml>
- Divers outils complémentaires sont présents pour OpenApi, vous les trouverez, ici :
<https://openapi.tools/>

La gestion des droits avec Swagger et Keycloak

Vous pouvez définir une gestion d'authentification avec Swagger et Keycloak et selon les différents mécanismes Oauth.

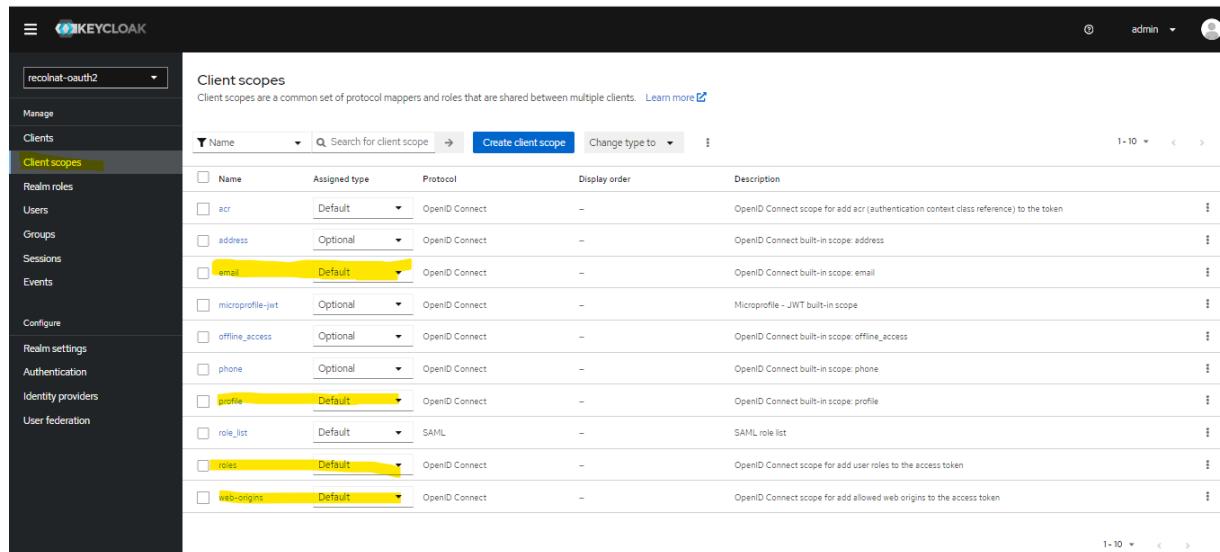
Exemple avec *Client Credentials Flow* :

Simple déclaration (directe) dans un YAML Swagger

```
securitySchemes:  
  authorisation-oidc:  
    type: oauth2  
    description: This API uses OAuth 2 with the clientCredentials grant flow.  
    flows:  
      clientCredentials:  
        tokenUrl: http://localhost:8089/realm/recolnat-oauth2/protocol/openid-connect/token  
        scopes:  
          email: email info  
          profile: user profile  
          roles: user roles  
          web-origin: cors web  
  
security:  
  - authorisation-oidc
```

Ici, pour l'exemple, on déclare directement, dans Swagger, la partie sécurisée (sans surcharge coté java. À ne jamais faire). Celle-ci comprend dans la déclaration, le type de sécurité (**oauth2**) et le type de flow (**clientCredentials**). L'url de demande de token [**tokenUrl**] (selon l'api défini du serveur. Ici, emploi de l'api Keycloak avec protocole OIDC). La liste des scopes définit dans Keycloak, qui sont définis par défaut. Chaque token d'accès OAuth peut être balisé avec plusieurs scopes. Les scopes sont des droits d'accès qui contrôlent si les informations d'identification fournies par un utilisateur permettent d'effectuer l'appel nécessaire au serveur de ressources. Ils n'accordent aucune autorisation supplémentaire au client, à l'exception de celles dont il dispose déjà.

Vous retrouvez la liste des scopes pour l'ensemble des clients, au niveau Keycloak, dans le menu « Client scopes ».



The screenshot shows the Keycloak administration interface. The left sidebar is dark with white text, showing navigation links like 'Manage', 'Clients', 'Client scopes', 'Realm roles', etc. The 'Client scopes' link is highlighted in blue. The main content area has a light gray header with 'Client scopes' and a sub-header: 'Client scopes are a common set of protocol mappers and roles that are shared between multiple clients.' Below this is a search bar and a 'Create client scope' button. A table lists 12 client scopes:

Name	Assigned type	Protocol	Display order	Description
acr	Default	OpenID Connect	-	OpenID Connect scope for add acr (authentication context class reference) to the token
address	Optional	OpenID Connect	-	OpenID Connect built-in scope address
email	Default	OpenID Connect	-	OpenID Connect built-in scope email
microprofile-jwt	Optional	OpenID Connect	-	Microprofile - JWT built-in scope
offline_access	Optional	OpenID Connect	-	OpenID Connect built-in scope offline_access
phone	Optional	OpenID Connect	-	OpenID Connect built-in scope phone
profile	Default	OpenID Connect	-	OpenID Connect built-in scope profile
role_list	Default	SAML	-	SAML role list
roles	Default	OpenID Connect	-	OpenID Connect scope for add user roles to the access token
web Origins	Default	OpenID Connect	-	OpenID Connect scope for add allowed web origins to the access token

Comme vous pouvez le voir, on spécifie dans la déclaration Swagger, un id qui sera employé pour spécifier les méthodes (ou l'ensemble des méthodes qui seront concernées par la sécurité). On peut définir le nom du client Keycloak, comme marqueur, mais ce n'est pas requis.

securitySchemes:
authorisation-oidc:

On retrouvera les Scopes, **par défaut**, qui sont définis sur ce client.

Name	Assigned type	Description
email	Default	OpenID Connect built-in scope: email
profile	Default	OpenID Connect built-in scope: profile
roles	Default	OpenID Connect scope for add user roles to the access token
web-origins	Default	OpenID Connect scope for add allowed web origins to the access token

Dans le cas du client credentials, le client Keycloak ‘cible’ doit être configuré avec « service account roles » et le « client authentication » actif.

Allows you to authenticate this client to Keycloak and retrieve access token dedicated to this client. In terms of OAuth2 specification, this enables support of 'Client Credentials Grant' for this client.

[Client authentication](#) [Service accounts roles](#)

General settings

Client ID: **authorisation-oidc**

Name:

Description:

Always display in UI: **Off**

Access settings

Root URL: **http://localhost:8080/swagger-ui/index.html**

Home URL:

Admin URL: **http://localhost:8080/swagger-ui/index.html**

Capability config

Client authentication: **Service accounts**

Authorization: **Off**

Authentication flow:

- Standard flow
- Implicit flow
- Service accounts
- OAuth 2.0 Device Authorization Grant
- OIDC CIBA Grant

Clients > Client details

authorisation-oidc OpenID Connect

Clients are applications and services that can request authentication of a user.

Settings Keys Credentials Roles Client scopes Service accounts roles Sessions Advanced

General Settings

Client ID *

Name

Description

Always display in UI Off

Access settings

Root URL

Home URL

Admin URL

Capability config

Client authentication On

Authorization Off

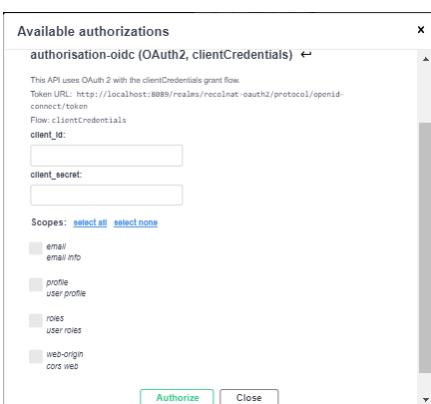
Authentication flow Standard flow Direct access grants
 Implicit flow Service accounts roles
 OAuth 2.0 Device Authorization Grant
 OIDC CIBA Grant

Login settings

Login theme Choose

Vous devez aussi définir dans le client Keycloak, l'url de retour lorsque « le client Swagger » se sera authentifié (re direct effectué, par le Keycloak et « interprété » par le browser). Via le Root Url

Lorsque vous allez vous authentifier, via Swagger. Vous aurez le popup suivant qui va s'ouvrir :



Il vous suffira de définir le client-id et le client secret :

Le client id est égal au client cible défini dans Keycloak : authorisation-oidc

Le client secret est égal au credentials du client cible défini dans Keycloak

Exemple avec [Client implict Flow](#) :

Coté Swagger, la déclaration est définie comme suit (vous retrouverez la liste des différents types de déclarations ici : <https://swagger.io/docs/specification/authentication/oauth2/>)

```

tags:
- name: Recolnat collection management
  description: api to manage tech hot subjects
security:
- collection-oidc: [] # <--- Use the same name as specified in securitySchemes*, here without scope
paths:
/v1/specimens:
  get:
    operationId: retrieveAllSpecimen
    tags:
      - specimens

components:
  securitySchemes:
    collection-oidc: # <--- Arbitrary name for the security scheme. Used to refer to it from elsewhere.
      type: openIdConnect
      openIdConnectUrl: http://sample.com/realms/recolnat-oauth2/.well-known/openid-configuration
schemas:
  CollectionIntegrationRequest:

```

Notez que l'url déclarée n'est pas celle qui sera employée. (On ne définit pas l'url en dure dans la déclaration Swagger, pour de simples raisons de sécurité)

Vous devez ensuite faire une surcharge java pour traiter l'url et les échanges.

```

@Configuration
@OpenAPIDefinition
@Slf4j
public class SwaggerOpenIdConfig {

    private static final String OPEN_ID_SCHEME_NAME = "collection-oide";
    private static final String OPENID_CONFIG_FORMAT = "%s/.well-known/openid-configuration";
    private static final String OPENID_CONNECT_CONFIG_FORMAT = "%s/protocol/openid-connect";

    @Bean
    OpenAPI customOpenApi(KeycloakProperties keycloakProperties) {
        log.trace(keycloakProperties.toString());
        OpenAPI openAPI = new OpenAPI();
        return openAPI
            .components(new Components()
                .addSecuritySchemes(OPEN_ID_SCHEME_NAME, createOpenIdSchemeOAuth(keycloakProperties)))
            .addSecurityItem(new SecurityRequirement().addList(OPEN_ID_SCHEME_NAME));
    }

    private SecurityScheme createOpenIdSchemeOAuth(KeycloakProperties properties) {
        String connectUrl = String.format(OPENID_CONNECT_CONFIG_FORMAT, properties.getAuthServerUrlOpenApi());

        return new SecurityScheme()
            .type(SecurityScheme.Type.OAUTH2).description("Oauth2 Flow")
            .flows(new OAuthFlows().implicit(
                new OAuthFlow().authorizationUrl(connectUrl.concat("/auth"))
                    .tokenUrl(connectUrl.concat("/token")).scopes(new Scopes())));
    }
}

```

Application.yml

```

springdoc:
  show-actuator: true
  show-oauth2-endpoints: true
  swagger-ui:
    csrf.enabled: false
    oauth:
      appName: ^project.name^
      clientId: ${AUTH_CLIENT_ID:recolnat-oide}
      clientSecret: ${CLIENT_SECRET:eaCcFXA78r1eYmmDnZPh8jv42e7wwabU}
keycloak:
  auth-server-url-OpenApi: ${AUTH_SERVER:http://localhost:8089/auth/realm/recolnat-oauth2}

```

Note : ne confondez pas avec la déclaration yml suivante (qui est l'interaction entre le micro-service et le serveur d'autorité Keycloak, pour la validation des token) :

```

security:
  oauth2:
    resourceServer:
      opaquetoken:
        introspection-uri:
${AUTH_SERVER_INTRO:http://localhost:8089/auth/realm/recolnat-oauth2/protocol/openid-connect/token/introspect}
        client-id: ${CLIENT_ID:collection-manager-oide}
        client-secret: ${CLIENT_SECRET:ee5jLU9Z27s1SNyusPI5o80DmcjJ0Mw5}

```

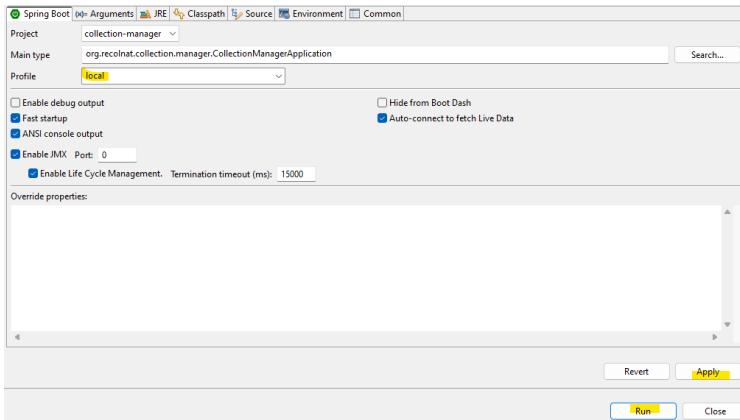
Partie docker /docker compose :

```
keycloak:
  container_name: keycloak_collection
  image: quay.io/keycloak/keycloak:23.0.1
  #build:
  #  context: ./keycloak
  command: ['start-dev','--import-realm']  # -V start --import-realm
  depends_on:
    postgres:
      condition: service_healthy
  volumes:
    - type: bind
      source: ./src/test/resources/keycloak/recolnat-oauth2-realm.json
      target: /opt/keycloak/data/import/recolnat-oauth2-realm.json
      bind:
        create_host_path: true
    - type: bind
      source: ./src/test/resources/keycloak/recolnat-login-
theme.jar
      target: /opt/keycloak/providers/keycloak-recolnat-login-theme.jar
      read_only: true
  environment:
    KC_DB: postgres
    # KEYCLOAK_IMPORT: /opt/test/recolnat-oauth2-realm.json -
Dkeycloak.profile.feature.upload_scripts=enabled
    #JAVA_OPTS_APPEND: -Dkeycloak.import=keycloak/data/recolnat-oauth2-
realm.json
    KC_DB_URL_DATABASE: keycloak
    KC_DB_URL: jdbc:postgresql://postgres:5432/keycloak
    KC_DB_USERNAME: adminkc
    KC_DB_PASSWORD: adminkc
    KC_HEALTH_ENABLED: true
    KC_HTTP_ENABLED: true
    KC_METRICS_ENABLED: true
    KC_LOG_LEVEL: WARN
    KC_HTTP_RELATIVE_PATH: auth
    # KC_HOSTNAME: keycloak
    # KC_HOSTNAME_PORT: 8180
    #KC_HOSTNAME_URL: http://keycloak.com.au:8180
    #KC_PROXY: reencrypt
    KEYCLOAK_ADMIN: admin
    KEYCLOAK_ADMIN_PASSWORD: admin
    IMP_AUTHORIZATION_OPENAPI: http://localhost:8081/*
    IMP_COLLECTIONMANAGER_OPENAPI: http://localhost:8080/*
    IMP_EXPLORER_OPENAPI: http://localhost:8082/*
    IMP_FRONT: http://localhost:3000/*
    SECRET_EXPLORER: C2Mdx8W8Fq2MdcPZbHD8wST7b1SWYpgp
    SECRET_COLLECTION_MANAGER: ee5jLU9Z27S1SNyusPI5o8ODmcjJ0Mw5
    SECRET_AUTHORIZATION: eaCcFXA78rleYmmDrnZPh8jv42e7wwabU
  ports:
    - "8089:8080"
    - "9990:9990" # debug port
  networks:
    - manager
```

A partir du micro-service collection-manager, lancez, via un PowerShell, la commande suivante :

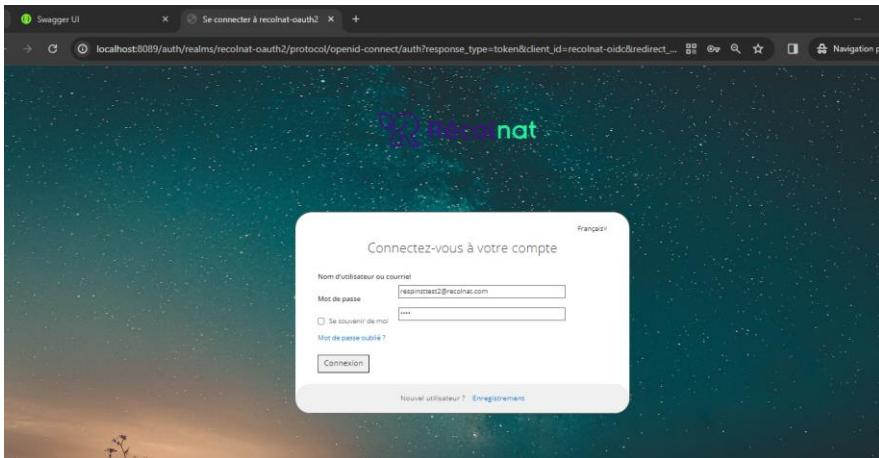
```
PS C:\workspace6\collection-manager> docker compose up -d
[+] Running 5/5
  ✓ Network collection-manager_manager  Created
  ✓ Container elastic_collection       Started
  ✓ Container postgres_collection     Healthy
  ✓ Container kibana_collection      Started
  ✓ Container keycloak_collection    Started
PS C:\workspace6\collection-manager>
```

Puis lancez le micro-service collection-manager. (Ex : via Eclipse) :



Puis allez sur <http://localhost:8080/swagger-ui/index.html#/> et cliquez sur « authorize »

Dans le formulaire du login/mot de passe, mettez respinsttest2@recolnat.com et test



Un redirect est de nouveau établi vers le swagger, indiquant que vous avez un token, vous pourrez cliquer sur logout pour vous déconnecter, par la suite.

Notez que l'url d'appel de la page d'authentification est la suivante :

http://localhost:8089/auth/realms/recolnat-oauth2/protocol/openid-connect/auth?response_type=token&client_id=recolnat-oidc&redirect_uri=http%3A%2F%2Flocalhost%3A8080%2Fswagger-ui%2Foauth2-redirect.html&state=V2VkJEZlYiAxNCAYMDI0IDEzOjA1OjEzIEdNVCswMTAwIChoZXVyZSBr3JtYWxIIGTigJFdXJvcGUgY2VudHjhGUp

C'est-à-dire qu'elle part de la configuration suivante :

```
keycloak:
  auth-server-url-OpenApi: ${AUTH_SERVER:http://localhost:8089/auth/realms/recolnat-oauth2}
```

Puis elle est complétée par la surcharge Java. Les paramètres transmis sont ceux déclarés dans la partie suivante :

```
clientId: ${AUTH_CLIENT_ID:recolnat-oidc}

clientSecret: ${CLIENT_SECRET:eaCcFXA78r1eYmmDnZPh8jv42e7wwabU}
```

Et l'url de redirection est construit par swagger.

Coté keycloak :

Client ID * recolnat-oidc

Name

Description

Always display in UI Off

Access settings

Root URL

Home URL

Valid redirect URIs

Valid post logout redirect URIs

Web origins

Admin URL

Capability config

Client authentication Off

Authorization Off

Authentication flow

- Standard flow Direct access grants
- Implicit flow Service accounts roles
- OAuth 2.0 Device Authorization Grant
- OIDC CIBA Grant

Capability config

Client authentication Off

Authorization Off

Authentication flow

- Standard flow Direct access grants
- Implicit flow Service accounts roles
- OAuth 2.0 Device Authorization Grant
- OIDC CIBA Grant

Login settings

Login theme

Consent required Off

Display client on screen Off

Client consent screen text

Logout settings

Front channel logout Off

Backchannel logout URL

Backchannel logout session required On

Backchannel logout revoke offline sessions Off

Partie implicit avec Postman :

Grant Type : Implicit

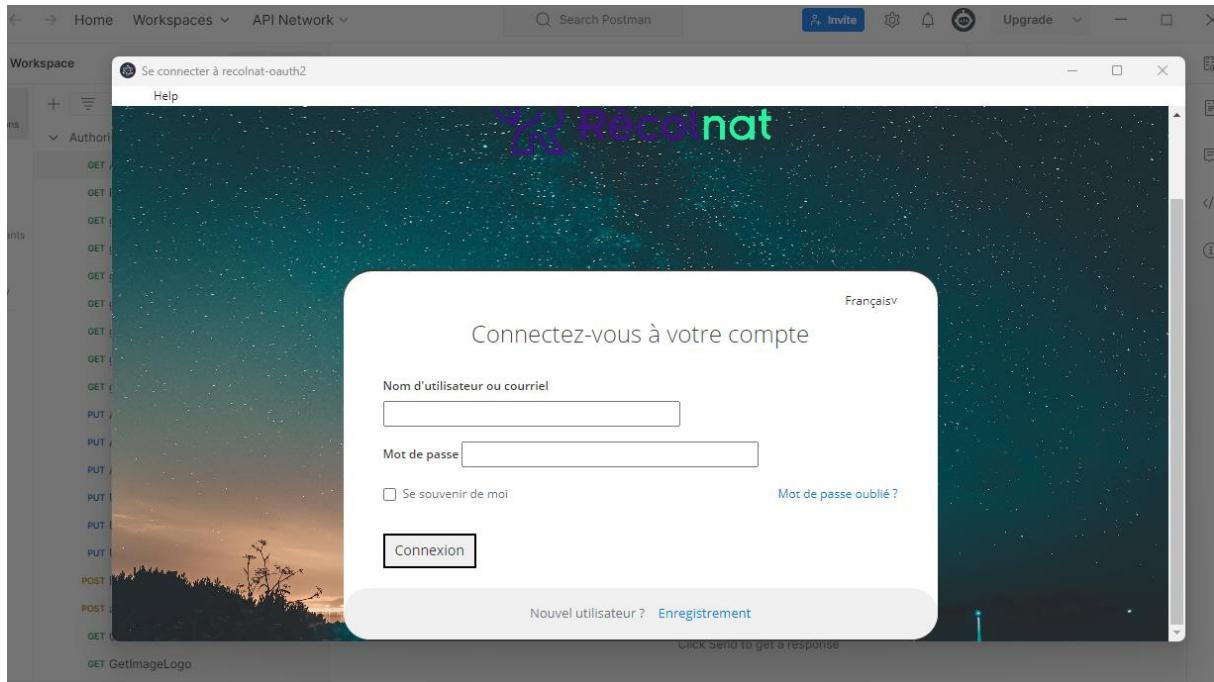
url callback : <http://localhost:3000/callback>

Auth URL : {{authUrl}}/auth/realms/recolnat-oauth2/protocol/openid-connect/auth Avec authUr comme variable d'environnement correspondant à l'uri de keycloak : <http://localhost:8089>

Ou <http://localhost:8089/auth/realms/recolnat-oauth2/protocol/openid-connect/auth>

The screenshot shows the Postman interface with the 'Authorization' tab selected. A warning message at the top right of the main panel says: "Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about [variables](#)." Below this, the 'Current Token' section is visible, showing fields for 'Token Name' (set to 'Token') and 'Header Prefix' (set to 'Bearer'). A note indicates that the token has expired. The 'Auto-refresh Token' toggle is turned on. In the 'Share Token' section, the toggle is turned off. Below these, the 'Configure New Token' section is expanded, showing fields for 'Token Name' (placeholder 'Enter a token name...'), 'Grant type' (set to 'Implicit'), 'Callback URL' (set to 'http://localhost:3000/callback'), and 'Auth URL' (set to 'http://localhost:8089/auth/realms/recolnat-...'). Other fields include 'Client ID' (set to 'recolnat-oidc' with a warning icon), 'Scope' (set to 'e.g. read:org'), 'State' (set to 'State'), and 'Client Authentication' (set to 'Send as Basic Auth header'). At the bottom left of the configuration panel, there are buttons for 'Clear cookies' and 'Get New Access Token'.

Vous allez avoir ceci lorsque vous allez cliquer sur « get new Access Token »



Mettez respinsttest2@recolnat.com et test, et vous aurez un token

Token Name	Access Token
eyJhbGciOiJSUzI1NiIsInR5cClgOiAiSldeUliwia2IkliA6ICJFV0hG0tY1RGhsYmVOOFV1VTZGaHE1RFIEbzY2RHhvRGpyWHhvRVZlUTfVln0.	eyJleHAIoJE3MDc5MjzNTlslmhdC16MTcwNzkyMTQ1MiwYXV0aF90aW1lljoxNzA3OTIxNDUyLCjqdGkiOii3Yzg3NjQxNl0zMTijLTQ1NmMtOTE3OS01MDlwODMwMTBkN2YiLCJpc3MioijodHRwOi8vbG9jYWxob3N0OjgwODkvXXVaC9yZWfsbxMvcmVjb2xuYXQtb2F1dGgyliwiYXVkljoimVjb2xuYXQtb2lkYylslnN1Yi6ljg2NzRiYmJlTg2Y
eyJleHAIoJE3MDc5MjzNTlslmhdC16MTcwNzkyMTQ1MiwYXV0aF90aW1lljoxNzA3OTIxNDUyLCjqdGkiOii3Yzg3NjQxNl0zMTijLTQ1NmMtOTE3OS01MDlwODMwMTBkN2YiLCJpc3MioijodHRwOi8vbG9jYWxob3N0OjgwODkvXXVaC9yZWfsbxMvcmVjb2xuYXQtb2F1dGgyliwiYXVkljoimVjb2xuYXQtb2lkYylslnN1Yi6ljg2NzRiYmJlTg2Y	WUtNGZmZi1hYTlyLWJINTk2NzViODBiOSlslnR5cCl6ikJlYXJlcilsmF6cCl6inJlY29sbmF0LW9pZGMiLCJzZXNzaW9uX3NOYXRlljoiNTBjODdiZDytYzFhmC00NW1LTlmZmEtN2ZhZDVhMThjZTVkliwiYXxs b3dlZC1vcmlnaW5zljpbmh0dHA6Ly9sb2NhbGhvc3Q6ODA4MCIsImh0dHA6Ly9sb2NhbGhvc3Q6ODA4MilsImh0dHA6Ly9sb2NhbGhvc3Q6ODA4MSlsImh0dHBzOi8vb2F1dGgucHNObW4uaW8iLCJodHRwOi8vbG9jYWxob3N0OjMwMDAiXSwicmVhbG1fYWNjZXNzljp7InJ

Remarque : vous pouvez cocher la case « Authorize using browser » (attention, il faut que votre browser, par défaut, soit configurer pour un redirect sur postman)

Callback URL [\(i\)](#)

<https://oauth.postman.io/v1/callback>

Authorize using browser

Néanmoins si vous faites un premier essai, vous allez voir que les choses ne semblent pas correctement se passer.
(Aucune action après votre login/mot de passe.)

La raison en est simple, lorsque vous avez cliquez sur la case « Authorize using browser », ceci a eu pour effet de redéfinir l'url de redirection. (<https://oauth.pstmn.io/v1/callback>)

Il vous faut alors définir cette url de redirection, comme cela a été fait pour l'url précédemment employé, dans le client keycloak acceptant les appels de flow implicit (<http://localhost:3000/callback>)

The screenshots show the configuration of a client named "recolnat-oauth2" in the Keycloak administration interface.

General settings:

- Client ID: recolnat-oauth2
- Name: (empty)
- Description: (empty)
- Always display in UI: Off

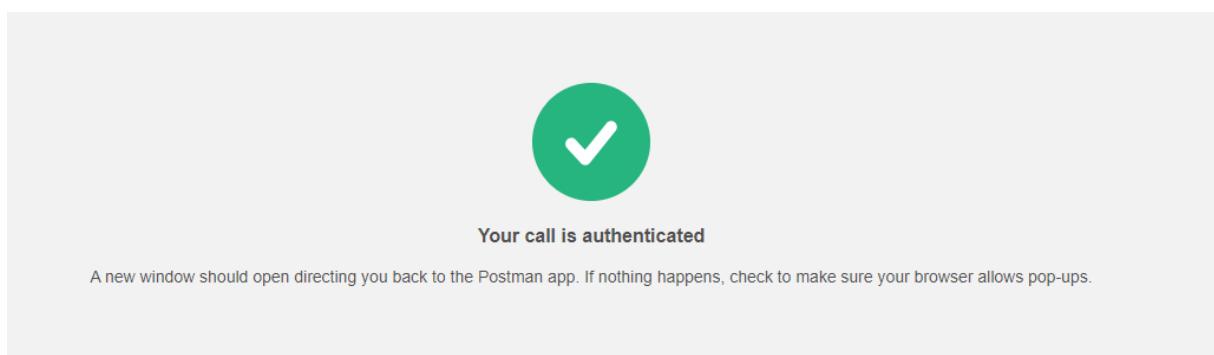
Access settings:

- Root URL: (empty)
- Home URL: (empty)

Capability config:

- Client authentication: Off
- Authorization: Off
- Authentication flow:
 - Standard flow: checked
 - Implicit flow: checked
 - Direct access grants: checked
 - Service accounts roles: unchecked
 - OAuth 2.0 Device Authorization Grant: unchecked
 - OIDC CIBA Grant: unchecked
- Login settings:
 - Login theme: Choose...
 - Consent required: Off
 - Display client on: Off

Remarque si vous n'avez pas configurer votre browser, vous aurez ceci. (voir ["Authorize using browser" fails to complete authentication · Issue #8717 · postmanlabs/postman-app-support · GitHub](#))



Remarques informatives :

Comme vous pouvez le voir, Recolnat dispose de différents clients : recolnat-oidc qui permet de réaliser les appels implicit. Et les autres clients Recolnat, n'ont que pour rôle que la validation des tokens, à chaque fois qu'un micro service reçoit une demande, impliquant une authentification

Client ID	Name	Type	Description	Home URL
account	#{client_account}	OpenID Connect	—	http://localhost:8089/auth/realms/recolnat-oauth2/account/
account-console	#{client_account-console}	OpenID Connect	—	http://localhost:8089/auth/realms/recolnat-oauth2/account/
admin-cli	#{client_admin-cli}	OpenID Connect	—	—
authorisation-oidc	—	OpenID Connect	client keycloak oidc authorisation	—
broker	#{client_broker}	OpenID Connect	—	—
collection-manager-oidc	—	OpenID Connect	client keycloak oidc collection manager	—
explorer-oidc	explorer-oidc	OpenID Connect	client keycloak oidc explorer	—
realm-management	#{client_realm-management}	OpenID Connect	—	—
recolnat-oidc	—	OpenID Connect	—	—
security-admin-console	#{client_security-admin-console}	OpenID Connect	—	http://localhost:8089/auth/admin/recolnat-oauth2/console/

Pour les clients (au sens Keycloak), vous retrouverez les secrets dans la table client. (Ne vous amusez à modifier cette colonne pour changer un secret d'un client, ce n'est pas référencé qu'à cette endroit)

	ABC client_id	123 not_before	public_client	ABC secret
1	master-realm	0	[]	[NULL]
2	account	0	[v]	[NULL]
3	account-console	0	[v]	[NULL]
4	broker	0	[]	[NULL]
5	security-admin-console	0	[v]	[NULL]
6	admin-cli	0	[v]	[NULL]
7	recolnat-oauth2-realm	0	[]	[NULL]
8	account	0	[]	48f0fcfc4-5fed-4635-bbec-169f120c4bd4 /
9	admin-cli	0	[v]	d339fd60-2b46-48d4-a560-419db53fe91 /
10	recolnat-oidc	0	[v]	a76b0064-4e0e-4ee6-952e-7e2c04f1356k /
11	authorisation-oidc	0	[]	eaCcfXA78r1eYmmDnZPh8jv42e7wwab /
12	collection-manager-oidc	0	[]	ee5jLU9Z27S1SNyusPl5e8ODmcj0Mw5 /
13	broker	0	[]	9e5jlb42-f3b2-42a3-934a-387cd90ce1a /
14	realm-management	0	[]	fdd570f0-4f20-484f-a76d-151361e7c2f1 /
15	security-admin-console	0	[v]	ec72fcad-17f9-4731-9eab-98ac5a2c99d4 /
16	account-console	0	[v]	[NULL]

Pour les utilisateurs, vous trouverez les password des utilisateurs vous les trouverez dans la table credentials. (Celles-ci sont encryptées en base avec un salt.)

	123 created_date	ABC user_label	ABC secret_data	abc cr
1	1502397814447 [NULL]		{"value": "ZFzU2zi2NxVks5HvjolbPekdq ("hash	("value": "aLFmlKY+HT8BXkgJkbeT0Ta+ERGizZWQdfk
2	1502397814447 [NULL]		{"value": "ZFzU2zi2NxVks5HvjolbPekdq ("hash	,"value": "imHhoU4KMxK9gsBR9l+t4PVf ("hash
3	1697108526226 [NULL]		{"value": "imHhoU4KMxK9gsBR9l+t4PVf ("hash	,"value": "aLFmlKY+HT8BXkgJkbeT0Ta+ERGizZWQdfk
4	1697109212640 My password		{"value": "aLFmlKY+HT8BXkgJkbeT0Ta+ERGizZWQdfk ("hash	,"value": "RxXOcs5ovtgY6v8n/ToK0Mg3 ("hash
5	1697109233661 My password		{"value": "aLFmlKY+HT8BXkgJkbeT0Ta+ERGizZWQdfk ("hash	

The screenshot shows the 'collection-manager-oidc' client configuration page in Keycloak. The top navigation bar includes tabs for Settings, Keys, Credentials, Roles, Client scopes, Service accounts roles, Sessions, and Advanced. The 'General Settings' section contains fields for Client ID (set to 'collection-manager-oidc'), Name, and Description. A toggle switch for 'Always display in UI' is set to Off. The 'Access settings' section includes fields for Root URL, Home URL, and Admin URL. The 'Capability config' section shows Client authentication as On and Authorization as Off. Under Authentication flow, 'Service accounts roles' is checked. The 'Login settings' section allows selecting a Login theme. A sidebar on the right provides links to General Settings, Access settings, Capability config, Login settings, and Logout settings.

Traitement du Thème Keycloak

Il y a deux façons de faire, mais attention ce n'est pas si simple, vous allez trouver différents exemples, où les personnes semblent se contenter de déclarer un fichier de properties, cela semble aussi aller dans ce sens dans la documentation. Mais en réalité, c'est trompeur. Beaucoup des exemples que vous allez voir, sont liés à d'anciennes versions (souvent Jboss/WildFly), cela ne fonctionne plus de la même façon sur la version Quarkus. De plus, Keycloak est en pleine mutation vers du reactV18, ce qui a pour effet d'engendrer de sérieuses évolutions entre les versions, même sur des versions mineures. (Ex : Le jar que nous allons employer dans le premier exemple, contient, dans la version 22.0.3, les fichiers que nous allons utiliser dans le deuxième exemple et qui sont dans un autre jar, dans la version 22.0.4). Second point, vous allez voir que l'on reprend et dupliquons l'intégralité de certains jars, ceci est lié au fait que dans les exemples que vous allez trouver sur le web, les développeurs ont réécrits, par exemple dans le cadre du login, le fichier login.ftl, en supprimant les points qui ne les intéressaient pas. Dans notre cas, nous n'avons besoin que de changer la CSS et les images, mais si vous ne fournissez pas un fichier Freemarker, dans votre thème customisé, vous allez avoir ce type d'erreur, ci-dessous. Et vous pourrez toujours spécifiez un quelconque parent (comme défini dans la [doc](#)), cela ne changera rien, si vous ne spécifiez pas, à minima, le fichier Freemarker « de lancement » (exemple pour la page d'authentification, le fichier login.ftl) , vous aurez cette erreur.(Ce point est évoqué, dans l'exemple, dans la [doc](#), mais sans être vraiment souligné).

```

at org.jboss.threads.DelegatingRunnable.run(DelegatingRunnable.java:29)
at org.jboss.threads.ThreadLocalResettingRunnable.run(ThreadLocalResettingRunnable.java:29)
at io.netty.util.concurrent.FastThreadLocalRunnable.run(FastThreadLocalRunnable.java:30)
at java.base/java.lang.Thread.run(Thread.java:833)
Caused by: freemarker.template.TemplateNotFoundException: Template not found for name "index.ftl".
The name was interpreted by this TemplateLoader: org.keycloak.theme.freemarker.DefaultFreeMarkerProvider$ThemeTemplateLoader@34d69a2a.
    at freemarker.template.Configuration.getTemplate(Configuration.java:2957)
    at freemarker.template.Configuration.getTemplate(Configuration.java:2777)
    at org.keycloak.theme.freemarker.DefaultFreeMarkerProvider.getTemplate(DefaultFreeMarkerProvider.java:66)
    at org.keycloak.theme.freemarker.DefaultFreeMarkerProvider.processTemplate(DefaultFreeMarkerProvider.java:45)
    ... 42 more

```

Remarque : si vous avez l'intention de modifier un des fichiers Freemarker, l'aide est ici :
<https://freemarker.apache.org/docs/index.html>

Sur Keycloak, vous pouvez customiser les 5 parties suivantes :

Account — Account management (c'est la partie auquel peut accéder un utilisateur pour mettre à jour son profil)

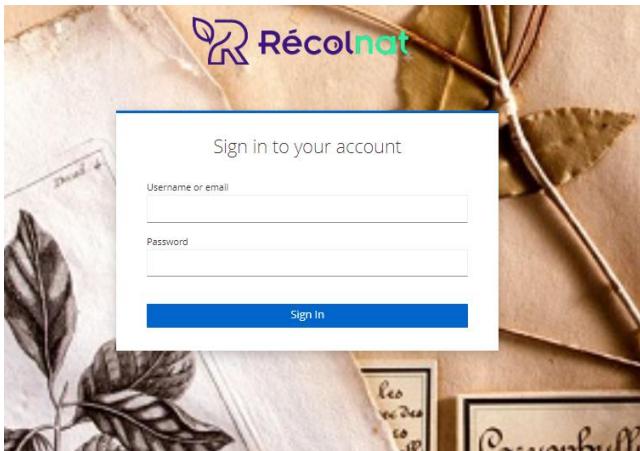
[Remarque : vous trouverez des exemples de customisation de l'account fourni par Keycloak sur <https://github.com/keycloak/keycloak-quickstarts/tree/latest/extension/extend-account-console> et <https://github.com/keycloak/keycloak/tree/main/examples/themes>]

Admin — Admin console (c'est la console de l'administrateur, elle n'est customisé le plus souvent que pour changer le logo)

[Vous trouverez un exemple ici : <https://github.com/keycloak/keycloak-quickstarts/tree/latest/extension/extend-admin-console>]

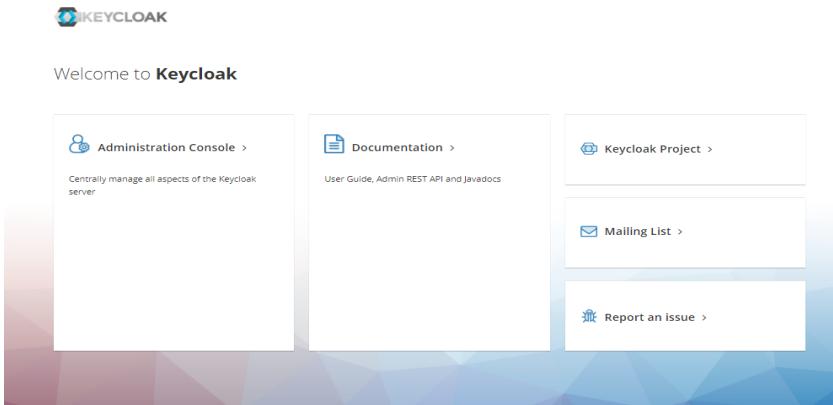
Email — Emails (c'est la simple partie des gestions de mails)

Login — Login forms (c'est la partie, si vous associez votre customisation au realm Master, qui sera employée pour s'authentifier sur l'application Keycloak et qui est aussi employée lorsque l'utilisateur s'authentifie sur un client applicatif, dans le cadre de Authorization Code Grant Type Flow ou de l'Implicit flow.



[Vous trouverez des exemples ici <https://github.com/alxrodav/keycloak-theme> et d'autres là <https://github.com/topics/keycloak-theme>, <https://www.baeldung.com/keycloak-custom-login-page> (noter que beaucoup sont anciens et mais que pour une part d'entre eux, vous allez retrouver un contenu similaire à ce que l'on va produire)]

Welcome — Welcome page (c'est la page d'accueil, celle que vous avez ouvert la première fois que vous êtes allé sur Keycloak)



[Vous trouverez un exemple de custom ici : <https://www.baeldung.com/spring-keycloak-custom-themes>]

Nous allons commencer par surcharger le login. Nous allons employer le format Jar puis nous ferons l'autre démarche qui est un dépôt des fichiers directement dans un répertoire « themes » de Keycloak. (Remarque : Pour tester votre customisation, la seconde est à préférer, par contre, pour le packaging, la première est plutôt conseillée)

1. Packager sous forme de Jar

Commencez par chargez le produit Keycloak sous sa forme de zip :
<https://www.keycloak.org/downloads>



Downloads 22.0.4

For a list of community maintained extensions check out the [Extensions](#) page.

Server

Keycloak	Distribution powered by Quarkus	 ZIP (sha1) TAR.GZ (sha1)
Container image	For Docker, Podman, Kubernetes and OpenShift	 Quay
Operator	For Kubernetes and OpenShift	 OperatorHub

Quickstarts

Quickstarts distribution	 GitHub ZIP
--------------------------	--

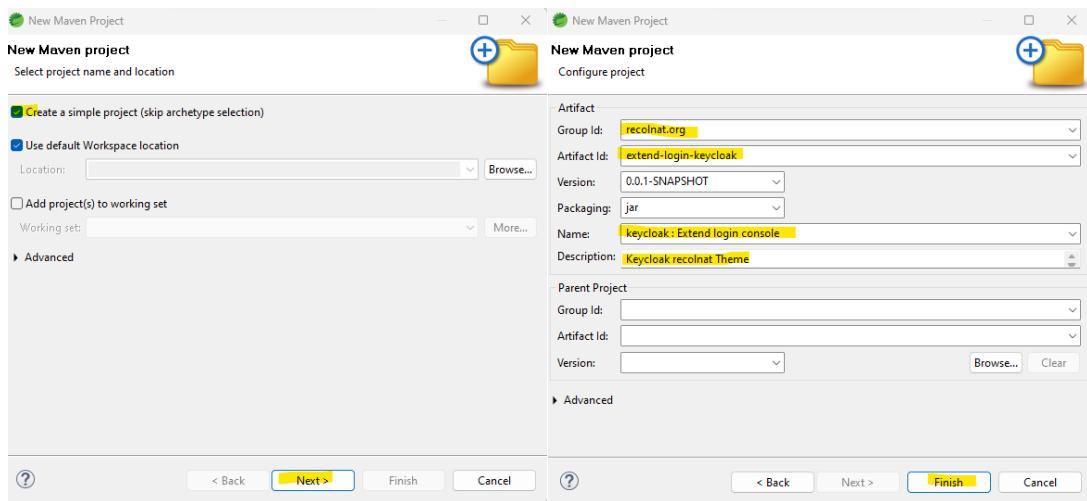
Ensuite dézipper le ou vous voulez, il va vous servir pour faire vos tests localement. Et, comme vous allez le voir, il faut réemployer le contenu d'une des librairies Keycloak, pour produire le Jar.

Lancez-le une fois via un PowerShell -> ./kc start-dev

```
PS C:\keycloak-22.0.4\bin> ./kc start-dev
Updating the configuration and installing your custom providers, if any. Please wait.
2023-10-13 04:39:48,780 INFO [io.quarkus.deployment.QuarkusAugmentor] (main) Quarkus augmentation completed in 7952ms
2023-10-13 04:39:51,173 INFO [org.keycloak.quarkus.runtime.hostname.DefaultHostnameProvider] (main) Hostname settings: Base URL: <unset>, Hostname: <request>, Strict HTTPS: false, Path: <request>, Strict BackChannel: false, Admin URL: <unset>, Admin: <request>, Port: -1, Proxied: false
2023-10-13 04:39:55,548 WARN [io.quarkus.agroal.runtime.DataSources] (main) Datasource <default> enables XA but transaction recovery is not enabled. Please enable transaction recovery by setting quarkus.transaction-manager.enable-recovery=true, otherwise data may be lost if the application is terminated abruptly
2023-10-13 04:39:54,735 WARN [org.infinispan.PERSISTENCE] (keycloak-cache-init) ISPN000554: jboss-marshalling is deprecated and planned for removal
2023-10-13 04:39:54,857 WARN [org.infinispan.CONFIG] (keycloak-cache-init) ISPN000569: Unable to persist Infinispan internal caches as no global state enabled
2023-10-13 04:39:54,983 INFO [org.infinispan.CONTAINER] (keycloak-cache-init) ISPN000556: Starting user marshaller 'org.infinispan.jboss.marshalling.core.JBossUserMarshaller'
2023-10-13 04:39:55,199 INFO [org.keycloak.broker.provider.AbstractIdentityProviderMapper] (main) Registering class org.keycloak.broker.provider.mappersync.ConfigSyncEventListener
2023-10-13 04:39:57,655 INFO [org.keycloak.quarkus.runtime.storage.legacy.liquibase.QuarkusJpaUpdaterProvider] (main) Initializing database schema. Using changelog META-INF/jpa-changelog-master.xml
2023-10-13 04:39:59,696 INFO [org.keycloak.connections.infinispan.DefaultInfinispanConnectionProviderFactory] (main) Node name: node_819005, Site name: null
2023-10-13 04:39:59,892 INFO [org.keycloak.services] (main) KC-SERVICES0050: Initializing master realm
2023-10-13 04:40:02,388 INFO [io.quarkus] (main) Keycloak 22.0.4 on JVM (powered by Quarkus 3.2.6.Final) started in 13.058s. Listening on: http://0.0.0.0:8080
2023-10-13 04:40:02,388 INFO [io.quarkus] (main) Profile dev activated.
2023-10-13 04:40:02,388 INFO [io.quarkus] (main) Installed features: [agroal, cdi, hibernate-orm, jdbc-h2, jdbc-mariadb, jdbc-mssql, jdbc-mysql, jdbc-oracle, jdbc-postgresql, keycloak, logging-gelf, micrometer, narayana-jta, reactive-routes, resteasy, resteasy-jackson, smallrye-context-propagation, smallrye-health, vertx]
2023-10-13 04:40:02,395 WARN [org.keycloak.quarkus.runtime.KeycloakMain] (main) Running the server in development mode. DO NOT use this configuration in production.
```

Puis allez sur un browser et faites <http://localhost:8080> et définissez un login /password d'administrateur puis connectez-vous pour validation et déconnectez-vous, ensuite.

Maintenant créez, dans un Eclipse, un simple projet Maven (File->New->Maven Project)



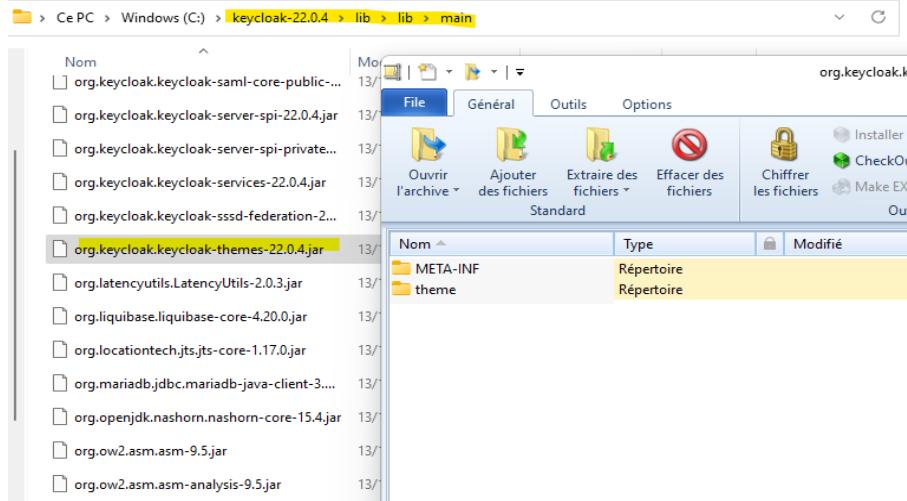
Ajoutez ensuite les properties et nom final suivant, dans le pom (le nom du jar n'a qu'une importance très relative) :

```

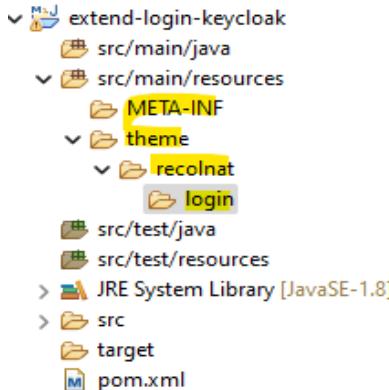
https://maven.apache.org/xsd/maven-4.0.0.xsd (xsi:schemaLocation)
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  <modelVersion>4.0.0</modelVersion>
  <groupId>recolnat.org</groupId>
  <artifactId>extend-login-keycloak</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>keycloak : Extend login console</name>
  <description>Keycloak recolnat Theme</description>
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
  </properties>
  <build>
    <finalName>keycloak-recolnat-login-theme</finalName>
  </build>
</project>

```

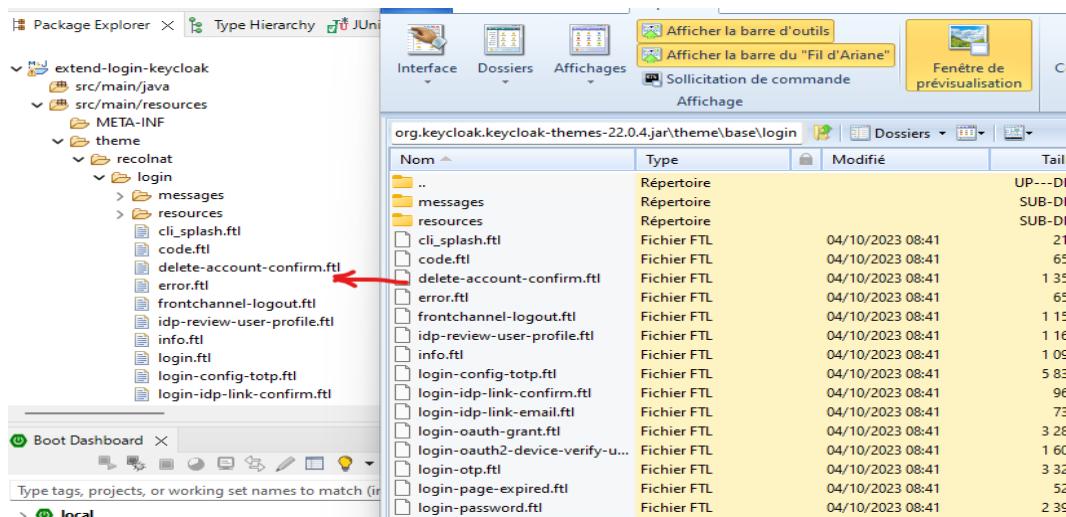
Maintenant, allez dans le répertoire lib/lib/main du Keycloak que vous venez de charger et d'installer. Et ouvrez, avec un outil de zip, le jar org.keycloak.keycloak-themes-22.0.4.jar.



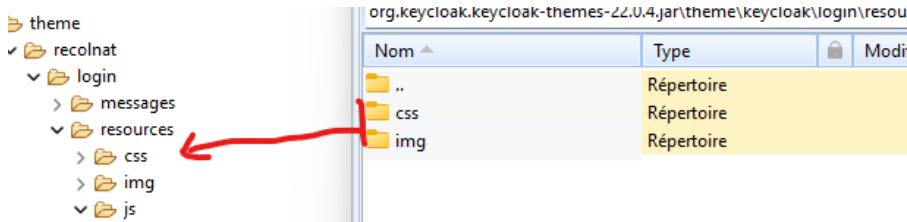
Allez sur votre projet, dans Eclipse, et créez, dans « src/main/resources », l'arborescence suivante (clic droit sur le répertoire « resources » ->new->Folder) :



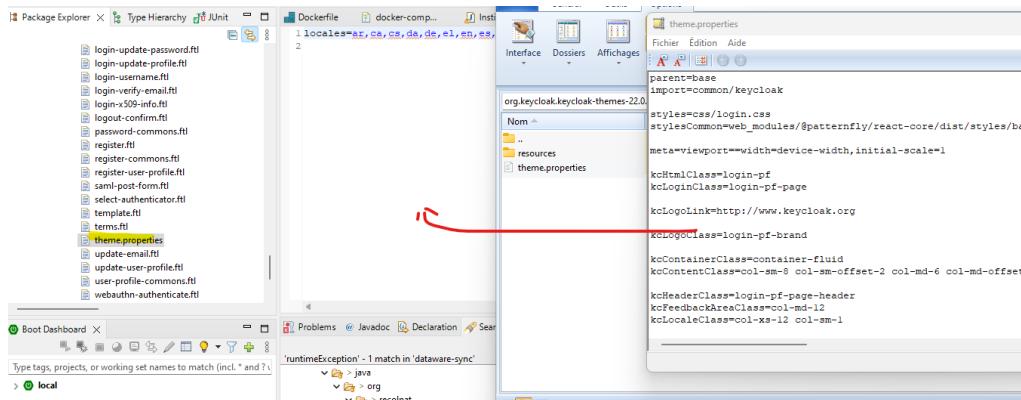
Ensuite copiez le contenu du zip suivant : « org.keycloak.keycloak-themes-22.0.4.jar\theme\base\login » dans le répertoire login. (Par un simple drag-drop)



Ensuite allez sur « org.keycloak.keycloak-themes-22.0.4.jar\theme\keycloak\login\resources » dans le zip et faites un drag-drop des répertoires présents dans le zip, dans le répertoire « theme/recolnat/login/resources »



Maintenant allez, dans le zip, dans « org.keycloak.keycloak-themes-22.0.4.jar\theme\keycloak\login » et copiez le contenu de theme.properties (exception des 2 premières lignes) dans le theme.properties présent dans votre projet (et conserver la ligne ,déjà présente, définissant les locales.)



```

1 locales=ax,ca,cs,da,de,el,en,es,fa,fr,fi,hu,it,ja,lt,nl,no,pl,pt-BR,ru,sk,sv,tr,zh-CN
2
3 styles=css/login.css
4 stylesCommon=web_modules/@patternfly/react-core/dist/styles/base.css web_modules/@patternfly/react-core/dist/
5
6 meta=viewport==width=device-width,initial-scale=1
7
8 kcHtmlClass=login-pf
9 kcLoginClass=login-pf-page
10
11 kcLogoLink=http://www.keycloak.org
12
13 kcLogoClass=login-pf-brand
14
15 kcContainerClass=container-fluid
16 kcContentClass=col-sm-8 col-sm-offset-2 col-md-6 col-md-offset-3 col-lg-6 col-lg-offset-3
17

```

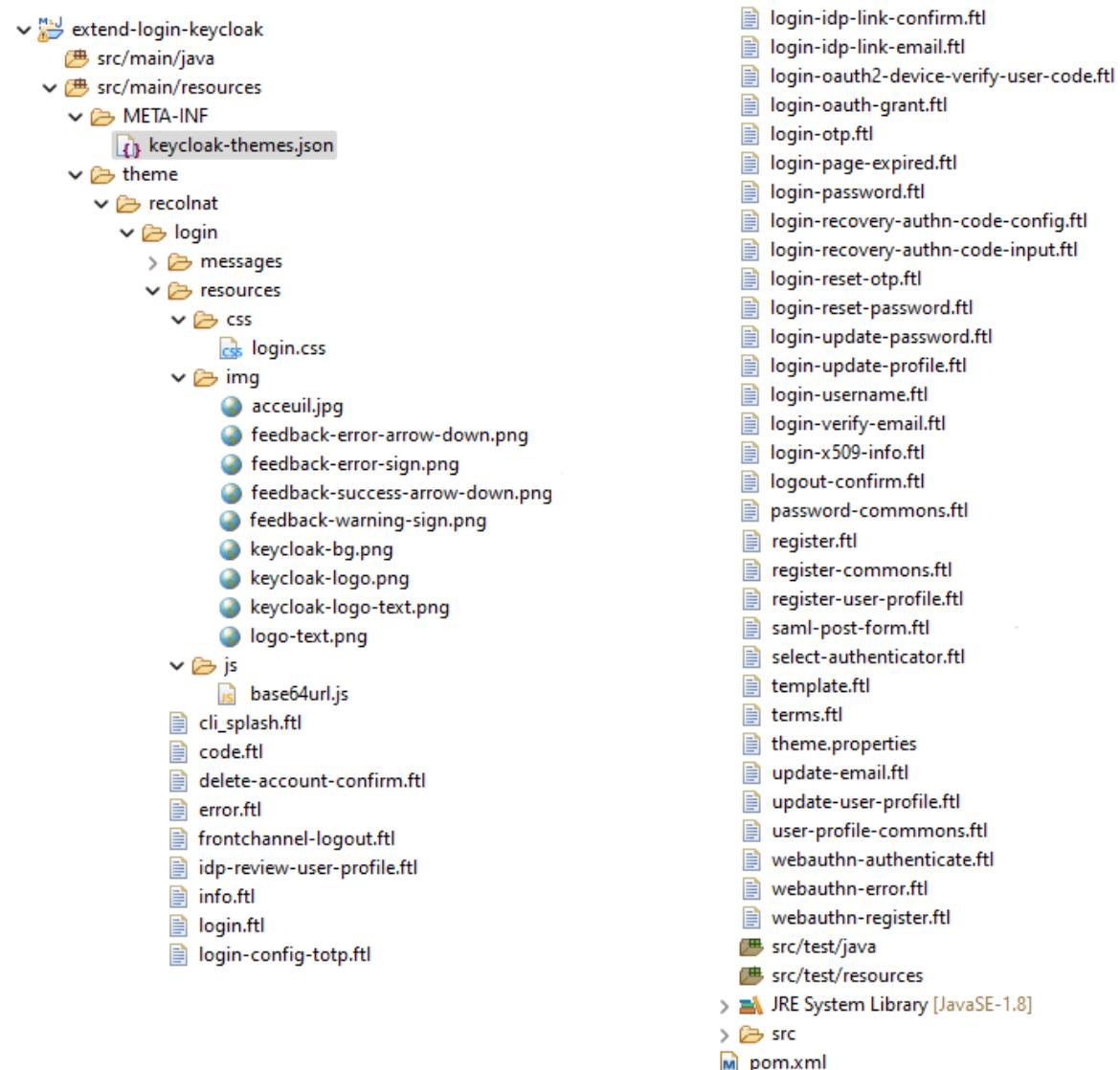
Maintenant, dans le répertoire META-INF, vous ajoutez un fichier keycloak-themes.json. Ce fichier va être employé, par Keycloak, pour identifier le(s) partie(s) que vous voulez customiser.

Le fichier Json à la structure suivante :

```
{
  "themes": [
    {
      "name" : "recolnat",
      "types": [ "login" ]
    }
  ]
}
```

Le nom correspond au nom de custom, le type correspond à l'une des 5 parties, que vous voulez customiser (Vous pouvez, déjà voir, du fait que c'est un tableau de types, que le jar peut contenir la customisation de différentes parties.). Nous le verrons plus loin, mais lorsque votre jar sera « intégré » au Keycloak, il va vous permettre de choisir le thème, sur la ou les cibles que vous aurez spécifiées dans ce fichier json.

Maintenant, vous avez la structure complète, qui devrait avoir l'arborescence suivante (Pour la version 22.0.4), pour customiser le login. (Remarque : dans l'impression écran, des images ont déjà été customisées pour Recolnat)



Vous pouvez dorénavant modifier les différents fichiers, images et CSS pour customiser le login.

Ensuite, faites un « mvn clean install » du projet, cela va produire un jar que vous allez placer dans le répertoire « **providers** », de votre Keycloak locale.

Ce PC > Windows (C:) > keycloak-22.0.4 > providers				Rechercher dans
Nom	Modifié le	Type	Taille	
keycloak-recolnat-login-theme.jar	13/10/2023 08:21	Fichier JAR	402 Ko	
README	13/10/2023 04:27	Fichier source Mar...	1 Ko	

Maintenant, vous pouvez lancer le keycloak :

Soit, vous lancez le start-dev (le keycloak fera un build par défaut, au démarrage)

```
PS C:\keycloak-22.0.4\bin> ./kc start-dev
```

Soit, vous lancez le start (mais vous devrez faire au préalable un build)

```
PS C:\keycloak-22.0.4\bin> ./kc build
Updating the configuration and installing your custom providers, if any. Please wait.
2023-10-13 13:59:20,313 INFO  [io.quarkus.deployment.QuarkusAugmentor] (main) Quarkus augmentation completed in 6910ms
Server configuration updated and persisted. Run the following command to review the configuration:

  kc.bat show-config
```

Ensuite, la première fois, allez dans le « **Realm Setting** » et rechercher votre Thème, puis sauvegarder le, puis déconnectez-vous et rafraîchissez la page, puis tenter de vous reconnecter. Les modifications que vous avez réalisées devrait apparaître dans la page de login.

The screenshot shows the Keycloak Admin UI for the 'master' realm. At the top, there's a navigation bar with tabs for General, Login, Email, Themes, Keys, Events, Localization, Security defenses, Sessions, Tokens, and Client po. The 'Themes' tab is currently selected. Below the tabs, there are four input fields for different themes:

- Login theme: A dropdown menu with 'recolnat' selected.
- Account theme: A dropdown menu with 'Select a theme'.
- Admin theme: A dropdown menu with 'recolnat'.
- Email theme: A dropdown menu with 'Select a theme'.

At the top right of the screen, there are buttons for 'Enabled' (which is checked), 'Action', and a dropdown menu.

Maintenant, comme vous pouvez le voir, ce n'est pas simple dans le cadre de bug et/ou de vous garantir de vos modifications, en effectuant, à chaque fois une compilation sur Eclipse puis de faire un déploiement.

Il y a une solution plus simple, c'est la seconde solution pour modifier l'une des cinq parties de Keycloak.

Remarque : Si vous regardez dans le jar « **org.keycloak.keycloak-themes-22.0.4.jar** », vous voyez les trois évolutions de la partie IHM (base, Keycloak et Keycloak. V2 (correspondant à la dernière version))

org.keycloak.keycloak-themes-22.0.4.jar\theme						
Nom	Type	Modifié	Taille	Ratio	Encapsulé	
..	Répertoire		UP---DIR		UP---DIR	
base	Répertoire		SUB-DIR		SUB-DIR	
keycloak.v2	Répertoire		SUB-DIR		SUB-DIR	
keycloak	Répertoire		SUB-DIR		SUB-DIR	

org.keycloak.keycloak-themes-22.0.4.jar\theme\base						
Nom	Type	Modifié				
..	Répertoire					
account	Répertoire					
admin	Répertoire					
email	Répertoire					
login	Répertoire					

Dans les versions précédentes, vous aviez l'intégralité des 5 parties, pour chaque version. Ce n'est plus le cas. Vous allez avoir, pour la version 22.0.4, deux jars supplémentaires contenant les versions Keycloak. V2 et V3 (pour account)

org.keycloak.keycloak-account-ui-22.0.4.jar	13/10/2023 04:27	Fichier JAR	4 299 Ko
org.keycloak.keycloak-admin-ui-22.0.4.jar	13/10/2023 04:27	Fichier JAR	9 129 Ko

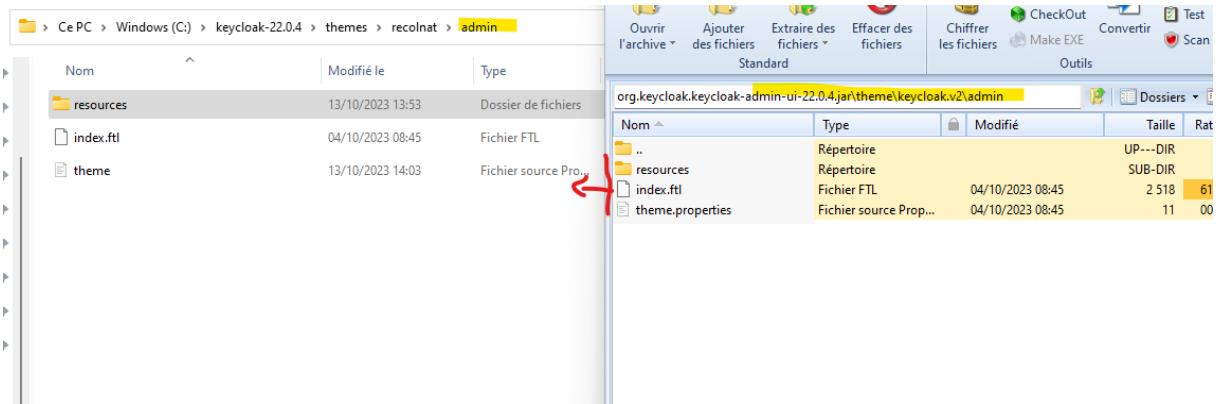
Nous allons employer le jar admin-ui, pour traiter notre partie admin.

Remarque : dans la cadre de la customisation, de la partie admin, il y a une [demande](#) qui avait été faite pour la customisation par un fichier de [properties](#). Le problème est que le fichier à lui seul n'est pas suffisant. Il faut réintégrer les fichiers, images... dans notre partie customisation. Sinon, Keycloak ne comprend pas. C'est ce que nous allons faire.

2. Customisation à partir d'une arborescence définie dans le répertoire « themes », de keycloak.
A la différence de la version Jar (où Keycloak emploi un Json pour définir la cible de la customisation). Dans le cadre du répertoire « themes », Keyclock s'appuie sur la structure arborescente du « projet » de customisation.
Dans le répertoire « themes », créer d'abord un répertoire avec le nom de votre custom (recolnat), puis un répertoire avec le nom de la partie cible (admin).

Ce PC > Windows (C:) > keycloak-22.0.4 > themes > recolnat			
	Nom	Modifié le	Type
	admin	13/10/2023 13:53	Dossier de fichiers

Ensuite, à partir du répertoire « \lib\lib\main » de votre Keycloak local, ouvrez le jar « org.keycloak.keycloak-admin-ui-22.0.4.jar » et copiez coller le contenu de « org.keycloak.keycloak-admin-ui-22.0.4.jar/theme/keycloak.v2/admin », dans le répertoire « admin », que vous venez de créer.



Maintenant, on va simplement employer, les champs properties qui ont été ajouté (suite à la [demande](#), et accepter par l'équipe keycloak)

```
Details
Sample theme.properties

# This is the logo in upper lefthand corner.
# It must be a relative path.
# Defaults to Keycloak logo.
logo=/public//retro-keycloak-logo.png

# This is the link followed when clicking on the logo.
# It can be any valid URL, including an external site.
# By default, this will go to the home page of the selected realm.
logourl=http://www.keycloak.org

# This is the icon for the admin console.
# It must be a relative path.
favicon=/public//favicon.ico

# This is the mime type for the favicon.
# Default value is 'image/svg+xml'
faviconType=image/x-icon
```

On va donc avoir ceci :

```
parent=base

# This is the logo in upper lefthand corner.
# It must be a relative path.
# Defaults to Keycloak logo.
logo=/logo-txt.png

# This is the link followed when clicking on the logo.
# It can be any valid URL, including an external site.
# By default, this will go to the home page of the selected realm.
logourl=https://www.mnhn.fr/fr/reseau-national-des-collections-naturalistes-recolnat

# This is the icon for the admin console.
# It must be a relative path.
favicon=/favicon.ico

# This is the mime type for the favicon.
# Default value is 'image/svg+xml'
faviconType=image/x-icon

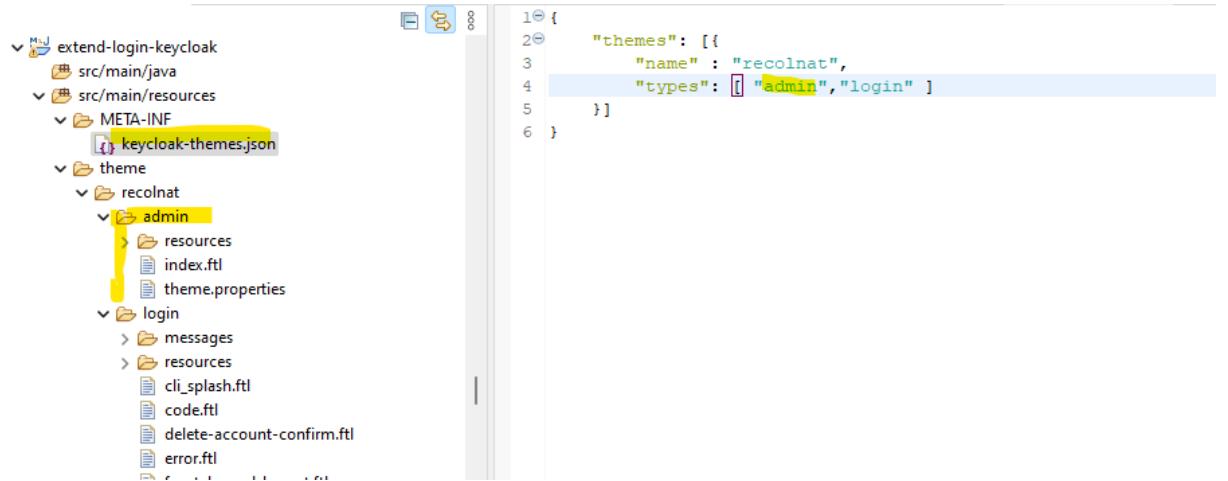
# title associated with the icon on the tab
title=Recolnat
```

On ajoute notre image dans la partie resources (celle que l'on a défini dans le theme.properties) :

Nom	Modifié le	Type	Taille
assets	13/10/2023 13:37	Dossier de fichiers	
locales	13/10/2023 13:37	Dossier de fichiers	
discovery-load-indicator	04/10/2023 08:45	Microsoft Edge H...	1 Ko
favicon	04/10/2023 08:45	Microsoft Edge H...	7 Ko
icon	04/10/2023 08:45	Microsoft Edge H...	11 Ko
img_avatar	04/10/2023 08:45	Microsoft Edge H...	2 Ko
logo	04/10/2023 08:45	Microsoft Edge H...	22 Ko
logo-txt	28/09/2023 18:26	Fichier PNG	10 Ko
robots	04/10/2023 08:45	Document texte	1 Ko

Maintenant, vous pouvez faire un start le Keycloak, via PowerShell. Soit par « ./kc start-dev » soit par « ./kc start » ou soit par « ./kc start --spi-theme-static-max-age=-1 --spi-theme-cache-themes=false --spi-theme-cache-templates=false » (si vous voulez désactiver les caches).

Maintenant, une fois que vous vous êtes assuré que cela fonctionne, vous pouvez passer à l'étape d'insertion dans un jar ou dans le jar précédent. Comme dans l'exemple du dessous :



Remarques :

1. Dans le cadre où vous voudriez customiser la page d'accueil, vous pouvez passer cette commande :
./kc start --spi-theme-welcome-theme=le répertoire custom, dans le répertoire « themes »
3. Vous trouverez des infos complémentaires, ici :
https://www.keycloak.org/docs/latest/server_development/#_themes

VAULT

Vault est un coffre-fort permettant un contrôle d'accès à des objets sensibles tel que des tokens, mots de passe et certificats... nous allons l'employer, ici pour protéger les clients secrets de Keycloak.

<https://github.com/hashicorp/docker-vault/blob/main/0.X/docker-entrypoint.sh>

Vous trouverez dans le même répertoire, un zip contenant, une implémentation de Keycloak avec Vault

TODO

Lancement de Keycloak au travers d'un Docker file :

Crée un répertoire Keycloak, à la racine de votre projet. Puis ajoutez un fichier Dockerfile et le fichier d'export Keycloak fourni.



Ensuite, ajoutez les lignes suivantes dans le Dockerfile : (ceci charge l'image Keycloak de Docker Hub, puis ajoute le fichier d'export Keycloak. Lance la commande d'import des datas de configuration Keycloak, lance le build Keycloak et enfin l'entry-point lance le « Keycloak container »)

```
FROM quay.io/keycloak/keycloak:22.0.4
ADD recolnat-oauth2-realm.json /opt/keycloak/
RUN /opt/keycloak/bin/kc.sh import --file /opt/keycloak/recolnat-oauth2-realm.json
RUN /opt/keycloak/bin/kc.sh build

ENTRYPOINT ["/opt/keycloak/bin/kc.sh", "start-dev"]
```

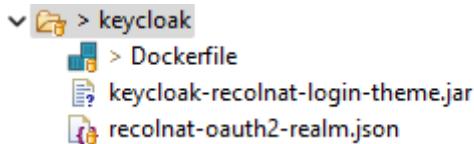
Dans le cadre de l'évolution, impliquant l'externalisation de certaines Datas du fichier d'export Keycloak.

Vous devrez ajouter des variables d'environnements dans le Dockerfile, comme suit :

```
FROM quay.io/keycloak/keycloak:22.0.4
ADD recolnat-oauth2-realm.json /opt/keycloak/
ENV IMP_AUTHORISATION_OPENAPI=http://localhost:8081/*
ENV IMP_COLLMANAGER_OPENAPI=http://localhost:8080/*
ENV IMP_DATAWARE_OPENAPI=http://localhost:8085/*
ENV IMP_FRONT=http://localhost:3000/*
ENV SECRET_DATAWARE=N1LSVQr5JPsi1OVQl51GKOnAkxR51z7O
ENV SECRET_COLLECTION_MANAGER=ee5jLU9Z27S1SNyusPI5o80DmcjJ0Mw5
ENV SECRET_AUTHORIZATION=eaCcFXA78r1eYmmDnZPh8jv42e7wwabU
RUN /opt/keycloak/bin/kc.sh import --file /opt/keycloak/recolnat-oauth2-realm.json
```

```
RUN /opt/keycloak/bin/kc.sh build
ENTRYPOINT ["/opt/keycloak/bin/kc.sh", "start-dev"]
```

Dans le cadre de l'ajout des thèmes Keycloak : ajoutez le jar fourni



Et modifiez le Dockerfile, comme suit :

```
FROM quay.io/keycloak/keycloak:22.0.4
ADD recolnat-oauth2-realm.json /opt/keycloak/
COPY keycloak-recolnat-login-theme.jar /opt/keycloak/providers/
ENV IMP_AUTHORISATION_OPENAPI=http://localhost:8081/*
ENV IMP_COLLMANAGER_OPENAPI=http://localhost:8080/*
ENV IMP_DATAWARE_OPENAPI=http://localhost:8085/*
ENV IMP_FRONT=http://localhost:3000/*
ENV SECRET_DATAWARE=N1LSVQr5JPsI1OVQ151GKOnAkxR51z7O
ENV SECRET_COLLECTION_MANAGER=ee5jLU9Z27S1SNyusPI5o8ODmcjJ0Mw5
ENV SECRET_AUTHORIZATION=eaCcFXA78r1eYmmDnZPh8jv42e7wwabU
RUN /opt/keycloak/bin/kc.sh import --file /opt/keycloak/recolnat-oauth2-
realm.json
RUN /opt/keycloak/bin/kc.sh build

ENTRYPOINT ["/opt/keycloak/bin/kc.sh", "start-dev"]
```

Pour lancer le Dockerfile de Keycloak, faites les commandes suivantes :

```
$ docker build . -t monTagName
```

Puis effectuez le lancement du container :

```
$ docker run monTagName
```

Remarque :

Vous pouvez ajouter un mapping Keycloak avec une base PostgreSQL.

Exemple d'un Dockerfile de base pour PostgreSQL :

```
FROM postgres:16-alpine
ADD init_postgres_instance.sql /docker-entrypoint-initdb.d/
```

Dans cet exemple, on charge une image PostgreSQL, à partir de Docker Hub et on ajoute l'import d'un fichier d initialisation des bases (avec login et mot de passe). Dans le cadre de PostgreSQL, les fichiers déposés dans le répertoire « docker-entrypoint-initdb.d », vont être pris en compte lors de

l'initialisation du container. (Pour plus d'infos, voir : https://hub.docker.com/_/postgres, dans la partie « Initialization script »)

Le type de fichiers SQL transmis, peut-être de la nature suivante :

```
CREATE USER authdb WITH PASSWORD 'authdb' CREATEDB;
CREATE DATABASE authdb
WITH
    OWNER = authdb
    ENCODING = 'UTF8'
    LC_COLLATE = 'en_US.utf8'
    LC_CTYPE = 'en_US.utf8'
    TABLESPACE = pg_default
    CONNECTION LIMIT = -1;

CREATE USER itv WITH PASSWORD 'itv' CREATEDB;
CREATE DATABASE itv
WITH
    OWNER = itv
    ENCODING = 'UTF8'
    LC_COLLATE = 'en_US.utf8'
    LC_CTYPE = 'en_US.utf8'
    TABLESPACE = pg_default
    CONNECTION LIMIT = -1;

CREATE USER syncdb WITH PASSWORD 'syncdb' CREATEDB;
CREATE DATABASE syncdb
WITH
    OWNER = syncdb
    ENCODING = 'UTF8'
    LC_COLLATE = 'en_US.utf8'
    LC_CTYPE = 'en_US.utf8'
    TABLESPACE = pg_default
    CONNECTION LIMIT = -1;

CREATE USER adminkc WITH PASSWORD 'adminkc' CREATEDB;
CREATE DATABASE keycloak
WITH
    OWNER = adminkc
    ENCODING = 'UTF8'
    LC_COLLATE = 'en_US.utf8'
    LC_CTYPE = 'en_US.utf8'
    TABLESPACE = pg_default
    CONNECTION LIMIT = -1;
```

Vous pouvez aussi passer par un script, au lieu d'un fichier Sql

Exemple :

```
#!/bin/bash
set -e
set -u

function create_user_and_database() {
    local database=$1
    echo " Creating user and database '$database'"
    psql -v ON_ERROR_STOP=1 --username "$POSTGRES_USER" <<-EOSQL
        CREATE USER $database;
        CREATE DATABASE $database;
        GRANT ALL PRIVILEGES ON DATABASE $database TO $database;
EOSQL
}

if [ -n "$POSTGRES_MULTIPLE_DATABASES" ]; then
    echo "Multiple database creation requested: $POSTGRES_MULTIPLE_DATABASES"
    for db in $(echo $POSTGRES_MULTIPLE_DATABASES | tr ',' ' '); do
        create_user_and_database $db
    done
    echo "Multiple databases created"
fi
```

Par contre dans ce cadre, vous devez modifier votre Dockerfile, comme suit :

```
FROM postgres:16-alpine

#soit
# see https://hub.docker.com/_/postgres for documentation
COPY create-multiple-postgresql-databases.sh /docker-entrypoint-initdb.d/
# attention comme dans le cadre windows et container linux, le caractères de fin de lignes est
différent, notamment pour les script shell
RUN dos2unix /docker-entrypoint-initdb.d/create-multiple-postgresql-databases.sh
ENV POSTGRES_USER=datawh
ENV POSTGRES_PASSWORD=datawh
ENV POSTGRES_MULTIPLE_DATABASES='authdb,itv, syncdb, keycloak'

#soit
#COPY init_postgres_instance.sql /docker-entrypoint-initdb.d/
```

Ceci implique la liste des noms des data bases que vous voulez créer (avec les même login et mot de passe). Mais aussi l'emploi de l'utilitaire linux dos2unix, qui change le caractère de fin de ligne de chaque ligne du script (car les scripts sont créés sur du Windows, mais la cible est un container PostgreSQL/linux. Voir https://fr.wikipedia.org/wiki/Fin_de_ligne)

Rem : voir doc : <https://www.baeldung.com/spring-boot-data-sql-and-schema-sql>

Remarques diverses

Configuration de la partie Swagger/Open API sur la sécurité

La configuration de la sécurité sur Open Api, comprend deux parties :

Une partie est destiné à la génération de la partie cliente (les plugins client de Swagger inclut, la plupart une génération de code de la partie sécurité). Donc la déclaration de la partie sécurité doit être défini dans le fichier YAML de déclaration swagger. Pour la partie génération Server

Exemple de déclaration en mode password dans le fichier Swagger (oauth 2)

```
openapi: 3.0.3
info:
  title: Recolnat
  description: |-  
    Le réseau national des collections naturalistes  
    est une infrastructure de recherche française dont l'objectif est  
    de produire un corpus de données basé sur les collections d'histoire naturelle de France,  
    permettant de valoriser les recherches au service de l'étude de la géo- et de la biodiversité actuelle et passée.  
  Liens utiles:  
  - [RECOLNAT] (https://fr.wikipedia.org/wiki/RECOLNAT)  
version: "1.0"
termsOfService: http://swagger.io/terms/
contact:
  email: apiteam@mnhn.fr
license:
  name: Apache 2.0
  url: http://www.apache.org/licenses/LICENSE-2.0.html
externalDocs:
  description: Find out more about RECOLNAT
  url: https://fr.wikipedia.org/wiki/RECOLNAT
servers:
  - url: https://petstore3.swagger.io/api/v3
tags:
  - name: Recolnat collection management
    description: api to manage tech hot subjects
security:
  - authorisation-oildc []:
    # <--- Use the same name as specified in securitySchemes*, here without scope
components:
  securitySchemes:
    authorisation-oildc: # <--- Arbitrary name for the security scheme. Used to refer to it from elsewhere.
      type: oauth2
      description: This API uses OAuth 2 with the clientCredentials grant flow.
      flows:
        password:
```

```

    tokenUrl: http://localhost:8089/realm/recolnat-oauth2/protocol/openid-connect/token
    scopes:
      email: email info
      profile: user profile
      roles: user roles
      web-origin: cors web

```

Exemple de déclaration en mode implicit dans le fichier Swagger (oauth2)

```

securitySchemes:
  explorer-oide: # <-- Arbitrary name for the security scheme. Used to refer to it from elsewhere.
    type: oauth2
    description: This API uses OAuth 2 with the implicit grant flow. For more information, see
    https://developers.getbase.com/docs/rest/articles/oauth2/requests
  flows:
    implicit:
      authorizationUrl: http://sample.com/realm/recolnat-oauth2/authorize
      scopes:
        email: email info
        profile: user profile
        roles: user roles
        web-origin: cors web

```

Exemple de déclaration en mode ALL dans le fichier Swagger (OIDC)

```

info:
  title: Recolnat
  description: |-  

    Le réseau national des collections naturalistes  

    est une infrastructure de recherche française dont l'objectif est  

    de produire un corpus de données basé sur les collections d'histoire naturelle de France,  

    permettant de valoriser les recherches au service de l'étude de la géo- et de la biodiversité actuelle et passée.  

  Liens utiles:  

  - [RECOLNAT] (https://fr.wikipedia.org/wiki/RECOLNAT)
  version: "1.0"
  termsOfService: http://swagger.io/terms/
  contact:
    email: apiteam@mnhn.fr
  license:
    name: Apache 2.0
    url: http://www.apache.org/licenses/LICENSE-2.0.html
  externalDocs:
    description: Find out more about RECOLNAT
    url: https://fr.wikipedia.org/wiki/RECOLNAT
  servers:
    - url: https://petstore3.swagger.io/api/v3
  tags:
    - name: Recolnat collection management
      description: api to manage tech hot subjects
  security:
    - collection-oide: [] # <-- Use the same name as specified in securitySchemes*, here without scope
  components:
    securitySchemes:
      collection-oide: # <-- Arbitrary name for the security schema. Used to refer to it from elsewhere.
        type: openidConnect
        openidConnectUrl: http://sample.com/realm/recolnat-oauth2/.well-known/openid-configuration

```

Coté java :

```

@Configuration
@OpenAPIDefinition
@Sif4j
public class SwaggerOpenIdConfig {

    /**
     * define in swagger
     */
    private static final String OPEN_ID_SCHEME_NAME = "collection-oide";
    private static final String OPENID_CONFIG_FORMAT = "%s/realm/%s/.well-known/openid-configuration";

    @Bean
    OpenAPI customOpenApi(KeycloakProperties keycloakProperties) {
        log.trace(keycloakProperties.toString());
        return new OpenAPI()
            .components(new Components()
                .addSecuritySchemes(OPEN_ID_SCHEME_NAME, createOpenIdScheme(keycloakProperties)))
            .addSecurityItem(new SecurityRequirement().addList(OPEN_ID_SCHEME_NAME));
    }

    /**
     * affiche la liste des openid-configuration definis dans keycloak
     * @param properties
     * @return
     */
}

```

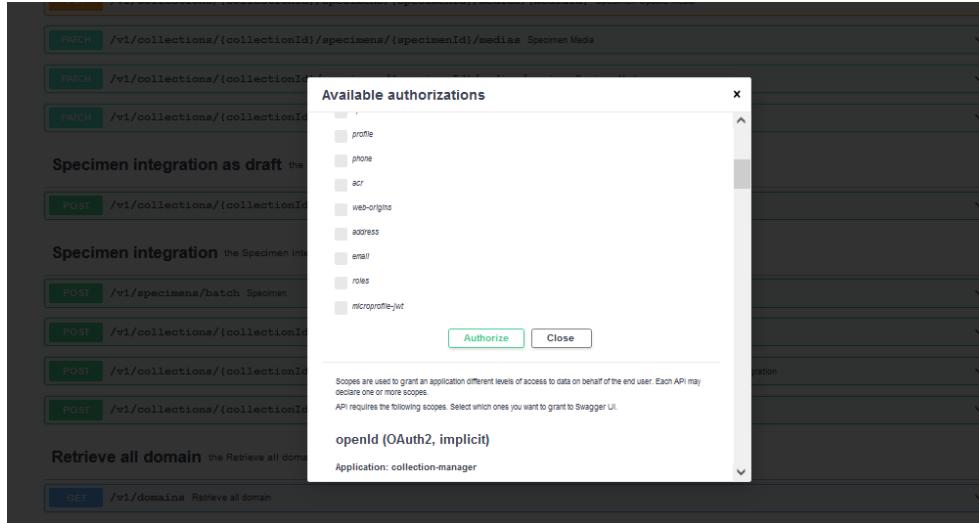
```

    */
    private SecurityScheme createOpenIdScheme(KeycloakProperties properties) {
        String connectUrl = String.format(OPENID_CONFIG_FORMAT, properties.getAuthServerUrlOpenApi(),
        properties.getRealmOpenApi());

        return new SecurityScheme()
            .type(SecurityScheme.Type.OPENIDCONNECT)
            .openIdConnectUrl(connectUrl);
    }
}

```

Vous allez avoir dans ce cadre un popup qui va vous afficher tous les modes possibles de connections/interactions avec le serveur d'authority (Keycloak). Selon votre configuration, un unique ou plusieurs d'entre eux peuvent être employés pour définir une authentification



Application	Scopes
collection-implicit	offline_access, openid, profile, phone, acr, web-orgs, address, email, roles, microprofile-jwt
collection-manager	offline_access, openid, profile, phone, acr, web-orgs, address, email, roles, microprofile-jwt

Scopes are used to grant an application different levels of access to data on behalf of the end user. Each API may declare one or more scopes.
API requires the following scopes. Select which ones you want to grant to Swagger UI.

openid (OAuth2, password)

Application: collection-manager

OpenID Connect URL: <http://localhost:8089/realm/recolnat-oauth2/.well-known/openid-configuration>
Token URL: <http://localhost:8089/realm/recolnat-oauth2/protocol/openid-connect/token>
Flow: client_credentials

client_id:

client_secret:

Scopes: [select all](#) [select none](#)

- offline_access
- openid
- profile
- phone
- acr
- web Origins
- address
- email
- roles
- microprofile-jwt

openid (OAuth2, urn:ietf:params:oauth:grant-type:device_code)

Application: collection-manager

OpenID Connect URL: <http://localhost:8089/realm/recolnat-oauth2/.well-known/openid-configuration>
Token URL: <http://localhost:8089/realm/recolnat-oauth2/protocol/openid-connect/token>
Flow: urn:ietf:params:oauth:grant-type:device_code

Scopes: [select all](#) [select none](#)

- offline_access
- openid
- profile
- phone
- acr
- web Origins
- address
- email
- roles
- microprofile-jwt

openid (OAuth2, urn:openid:params:grant-type:ciba)

Application: collection-manager

OpenID Connect URL: <http://localhost:8089/realm/recolnat-oauth2/.well-known/openid-configuration>
Flow: urn:openid:params:grant-type:ciba

Scopes: [select all](#) [select none](#)

- offline_access
- openid
- profile
- phone
- acr
- web Origins
- address
- email
- roles
- microprofile-jwt

Scopes are used to grant an application different levels of access to data on behalf of the end user. Each API may declare one or more scopes.
API requires the following scopes. Select which ones you want to grant to Swagger UI.

openid (OAuth2, urn:ietf:params:oauth:grant-type:device_code)

Application: collection-manager

OpenID Connect URL: <http://localhost:8089/realm/recolnat-oauth2/.well-known/openid-configuration>
Token URL: <http://localhost:8089/realm/recolnat-oauth2/protocol/openid-connect/token>
Flow: urn:ietf:params:oauth:grant-type:device_code

Scopes: [select all](#) [select none](#)

- offline_access
- openid
- profile
- phone
- acr
- web Origins
- address
- email
- roles
- microprofile-jwt

openid (OAuth2, urn:openid:params:grant-type:ciba)

Application: collection-manager

OpenID Connect URL: <http://localhost:8089/realm/recolnat-oauth2/.well-known/openid-configuration>
Flow: urn:openid:params:grant-type:ciba

Scopes: [select all](#) [select none](#)

- offline_access
- openid
- profile
- phone
- acr
- web Origins
- address
- email
- roles
- microprofile-jwt

Remarque sur la configuration des attributs :

Pour modifier les attributs associés aux utilisateurs d'un realm donnée :

Clients

Clients are applications and services that can request authentication of a user. [Learn more](#)

[Clients list](#) [Initial access token](#) [Client registration](#)

[Create client](#) [Import client](#)

Client ID	Name	Type	Description
account	`\${client_account}`	OpenID Connect	—
account-console	`\${client_account-console}`	OpenID Connect	—
admin-cli	`\${client_admin-cli}`	OpenID Connect	—
authorisation-oidc	—	OpenID Connect	—
broker	`\${client_broker}`	OpenID Connect	—
collection-manager-oidc	—	OpenID Connect	—
dataware-oidc	—	OpenID Connect	—
realm-management	`\${client_realm-management}`	OpenID Connect	—
recolnat-test-oidc	recolnat-test-oidc	OpenID Connect	—
security-admin-console	`\${client_security-admin-console}`	OpenID Connect	—

Clients > Client details

recolnat-test-oidc OpenID Connect

Clients are applications and services that can request authentication of a user.

[Settings](#) [Roles](#) [Client scopes](#) [Sessions](#) [Advanced](#)

[Setup](#) [Evaluate](#)

[Add client scope](#) [Change type to](#)

Assigned client scope	Assigned type	Description
recolnat-test-oidc-dedicated	none	Dedicated scope and mappers for this client
address	Optional	OpenID Connect built-in scope: address
email	Default	OpenID Connect built-in scope: email
microprofile-jwt	Optional	Micropattern - JWT built-in scope
offline_access	Optional	OpenID Connect built-in scope: offline_access
phone	Optional	OpenID Connect built-in scope: phone
profile	Default	OpenID Connect built-in scope: profile
roles	Default	OpenID Connect scope for add user roles to the access token
web-origins	Default	OpenID Connect scope for add allowed web origins to the access token

Clients > Client details > Dedicated scopes

recolnat-test-oidc

This is a client scope which includes the dedicated mappers and scope

[Mappers](#) [Scope](#)

[Add mapper](#)

Name	Category	Type	Priority
family_name	Token mapper	User Property	0
Institution	Token mapper	User Attribute	0
Cost Center	Token mapper	User Attribute	0
Address	Token mapper	User Attribute	0
Client Host	Token mapper	User Session Note	0
entry_identifiers	Token mapper	User Attribute	0
username	Token mapper	User Attribute	0
full_name	Token mapper	User's full name	0
Client ID	Token mapper	User Session Note	0
Client IP Address	Token mapper	User Session Note	0
Person Type	Token mapper	User Attribute	0
role-met-ask-in73053	Token mapper	User Attribute	0
email	Token mapper	User Property	0
Preferred Language	Token mapper	User Attribute	0
given_name	Token mapper	User Property	0

The screenshot shows the 'Mapper details' page for a client scope named 'collections'. The 'Mapper type' is set to 'User Attribute'. The 'Name' field contains 'collections'. The 'User Attribute' field also contains 'collections'. The 'Token Claim Name' field contains 'collections'. The 'Claim JSON Type' is set to 'String'. Under the 'Add to' section, 'Add to ID token' is off, 'Add to access token' is on, 'Add to userinfo' is on, and 'Multivalued' is on. The 'Aggregate attribute values' option is off. At the bottom, there are 'Save' and 'Cancel' buttons.

Remarque : Pour les idle timeouts

Pour les délais d'inactivité, il existe une fenêtre de temps de deux minutes pendant laquelle la session est active. Par exemple, lorsque le délai d'expiration est défini sur 30 minutes, il faudra 32 minutes avant l'expiration de la session.

Ce paramètre concerne uniquement les clients OIDC. Si un utilisateur est inactif pendant une durée supérieure à ce délai, la session utilisateur est invalidée. Cette valeur de délai d'attente est réinitialisée lorsque les clients demandent une authentification ou envoient une demande de jeton d'actualisation. Keycloak ajoute une fenêtre de temps au délai d'inactivité avant que l'invalidation de session ne prenne effet. Voir la note plus loin dans cette section.

Cette action est nécessaire pour certains scénarios dans les environnements de cluster et entre centres de données dans lesquels le jeton est actualisé sur un nœud de cluster peu de temps avant l'expiration et les autres nœuds de cluster considèrent à tort la session comme expirée car ils n'ont pas encore reçu le message concernant un actualisation réussie à partir du nœud d'actualisation.

When the SSO session expires you cannot refresh your token unless you specify in the scope parameter the value offline_access.

<https://www.janua.fr/offline-sessions-and-offline-tokens-within-keycloak/>

https://access.redhat.com/documentation/fr-fr/red_hat_single_sign-on/7.0/html/server_administration_guide/user_session_management