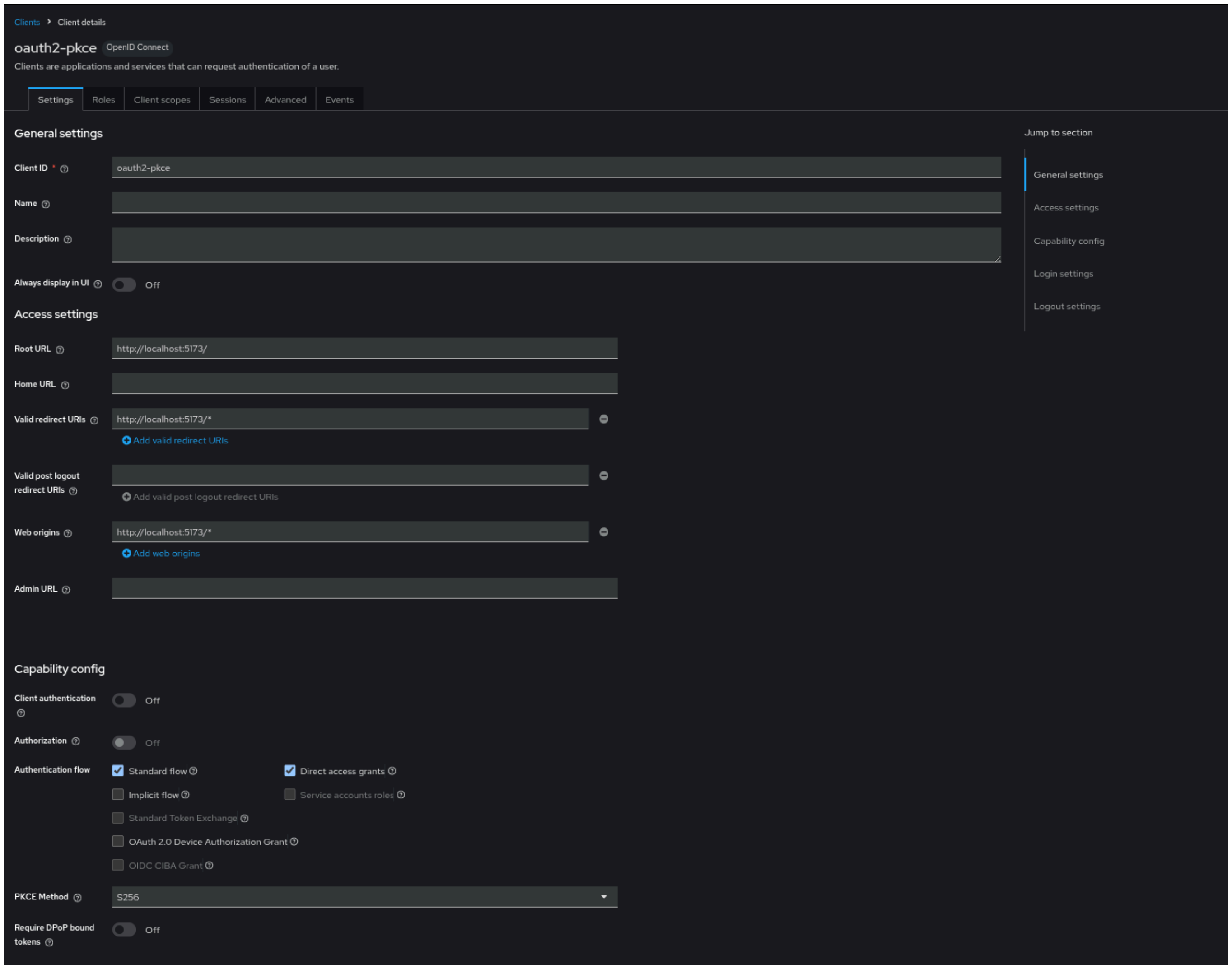
# Keycloak

## Client settings



Clients > Client details

oauth2-pkce OpenID Connect

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

General settings

Client ID *: oauth2-pkce

Name:

Description:

Always display in UI: Off

Access settings

Root URL: http://localhost:5173/

Home URL:

Valid redirect URIs: http://localhost:5173/*

Add valid redirect URIs

Valid post logout redirect URIs:

Add valid post logout redirect URIs

Web origins: http://localhost:5173/*

Add web origins

Admin URL:

Capability config

Client authentication: Off

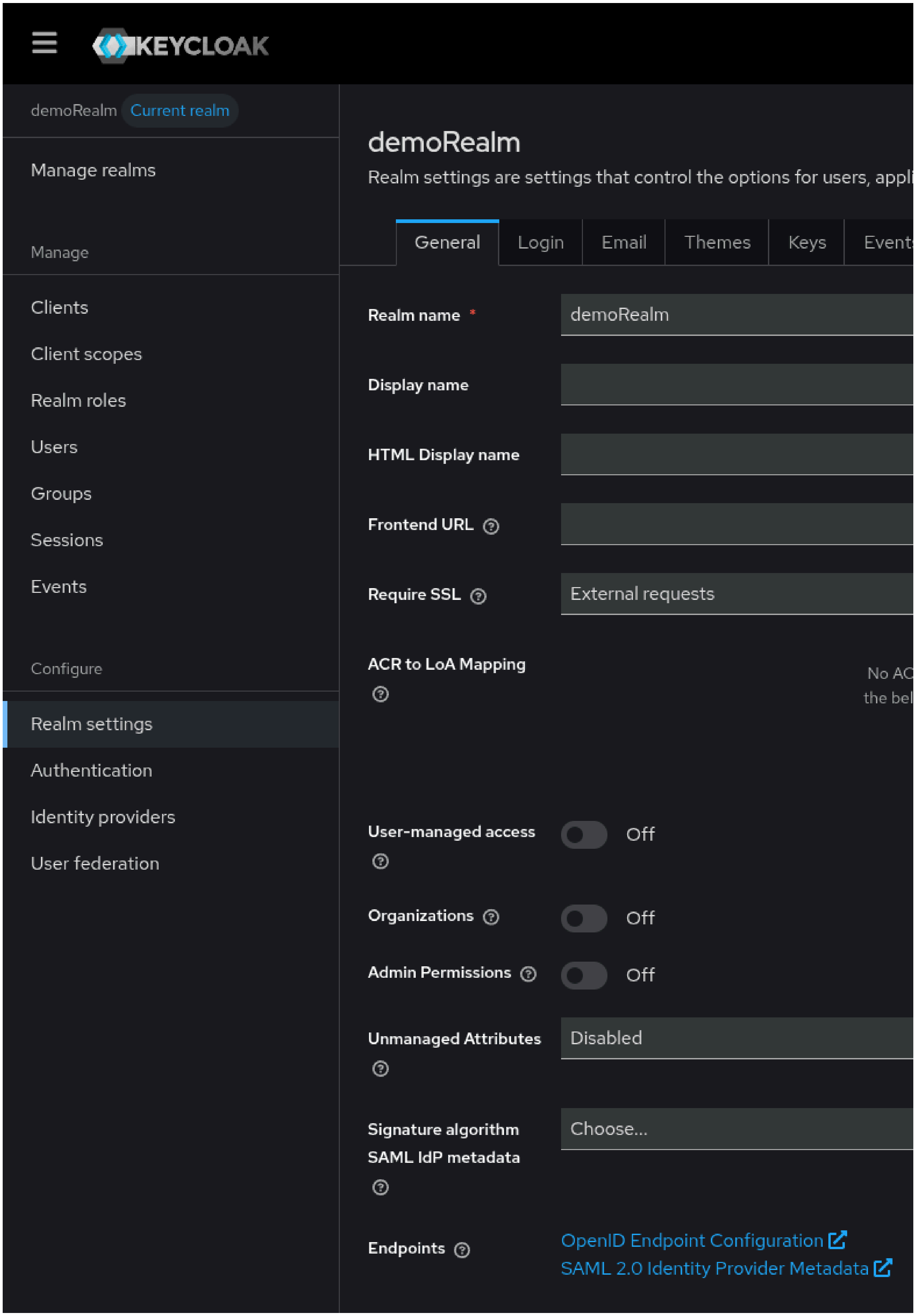
Authorization: Off

Authentication flow:
- [x] Standard flow
- [x] Direct access grants
- [ ] Implicit flow
- [ ] Service accounts roles
- [ ] Standard Token Exchange
- [ ] OAuth 2.0 Device Authorization Grant
- [ ] OIDC CIBA Grant

PKCE Method: S256

Require DPoP bound tokens: Off

- Client ID: ID of client
- Valid redirect URIs: your frontend App
- Web origins: your frontend App
- Standard flow & Direct access grants
- PKCE Method: S256



demoRealm

Realm settings are settings that control the options for users, appli

Realm name * demoRealm

Display name

HTML Display name

Frontend URL

Require SSL External requests

ACR to LoA Mapping

No AC
the bel

User-managed access Off

Organizations Off

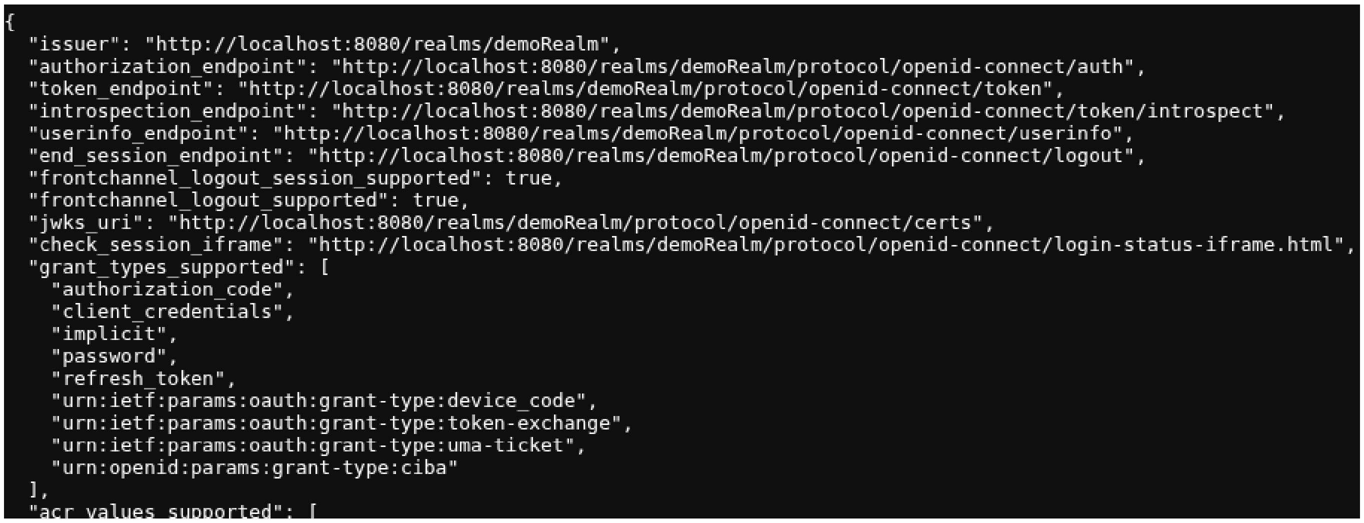
Admin Permissions Off

Unmanaged Attributes Disabled

Signature algorithm SAML IdP metadata Choose...

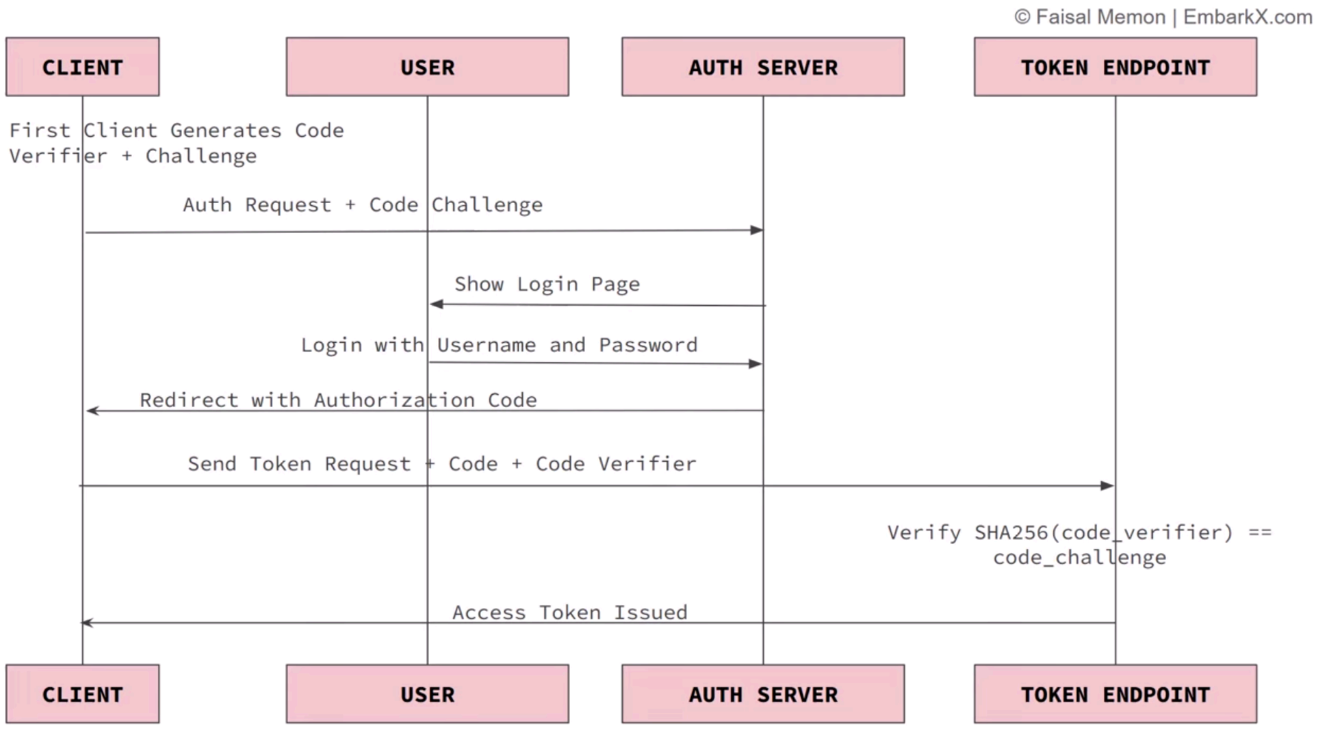
Endpoints OpenID Endpoint Configuration
SAML 2.0 Identity Provider Metadata

## OpenID Endpoint Configuration

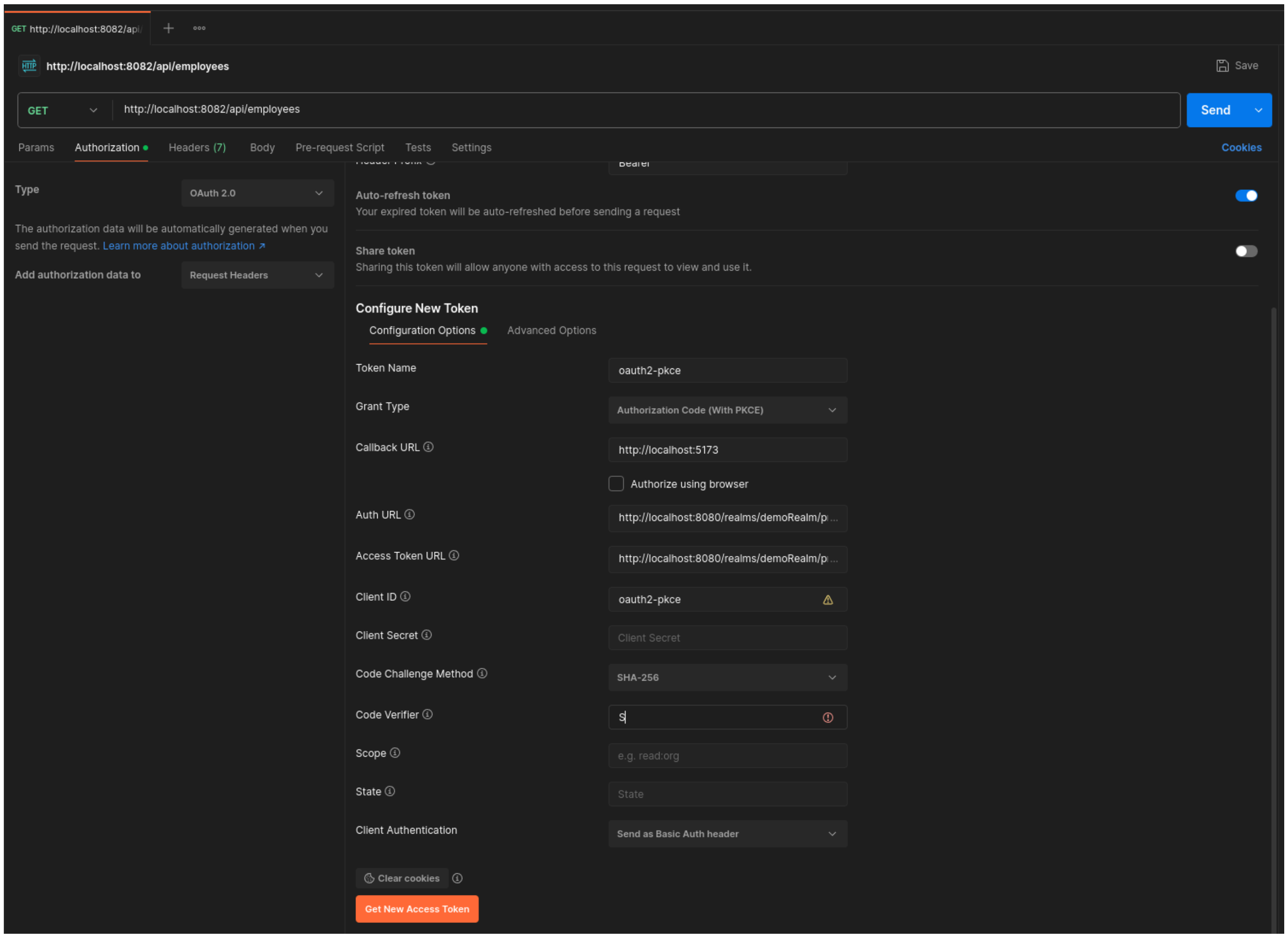


```
{
  "issuer": "http://localhost:8080/realms/demoRealm",
  "authorization_endpoint": "http://localhost:8080/realms/demoRealm/protocol/openid-connect/auth",
  "token_endpoint": "http://localhost:8080/realms/demoRealm/protocol/openid-connect/token",
  "introspection_endpoint": "http://localhost:8080/realms/demoRealm/protocol/openid-connect/token/introspect",
  "userinfo_endpoint": "http://localhost:8080/realms/demoRealm/protocol/openid-connect/userinfo",
  "end_session_endpoint": "http://localhost:8080/realms/demoRealm/protocol/openid-connect/logout",
  "frontchannel_logout_session_supported": true,
  "frontchannel_logout_supported": true,
  "jwks_uri": "http://localhost:8080/realms/demoRealm/protocol/openid-connect/certs",
  "check_session_iframe": "http://localhost:8080/realms/demoRealm/protocol/openid-connect/login-status-iframe.html",
  "grant_types_supported": [
    "authorization_code",
    "client_credentials",
    "implicit",
    "password",
    "refresh_token",
    "urn:ietf:params:oauth:grant-type:device_code",
    "urn:ietf:params:oauth:grant-type:token-exchange",
    "urn:ietf:params:oauth:grant-type:uma-ticket",
    "urn:openid:params:grant-type:ciba"
  ],
  "acr_values_supported": [
```

## PKCE - Flow


© Faisal Memon | EmbarkX.com
CLIENT
USER
AUTH SERVER
TOKEN ENDPOINT
First Client Generates Code Verifier + Challenge
Auth Request + Code Challenge
Show Login Page
Login with Username and Password
Redirect with Authorization Code
Send Token Request + Code + Code Verifier
Verify SHA256(code_verifier) == code_challenge
Access Token Issued
CLIENT
USER
AUTH SERVER
TOKEN ENDPOINT

## Postman


GET http://localhost:8082/api/

http://localhost:8082/api/employees

Save

GET http://localhost:8082/api/employees

Send

Params | Authorization ● | Headers (7) | Body | Pre-request Script | Tests | Settings | Cookies

Type: OAuth 2.0

The authorization data will be automatically generated when you send the request. Learn more about authorization ↗

Add authorization data to: Request Headers

Bearer

Auto-refresh token
Your expired token will be auto-refreshed before sending a request

Share token
Sharing this token will allow anyone with access to this request to view and use it.

Configure New Token

Configuration Options ● | Advanced Options

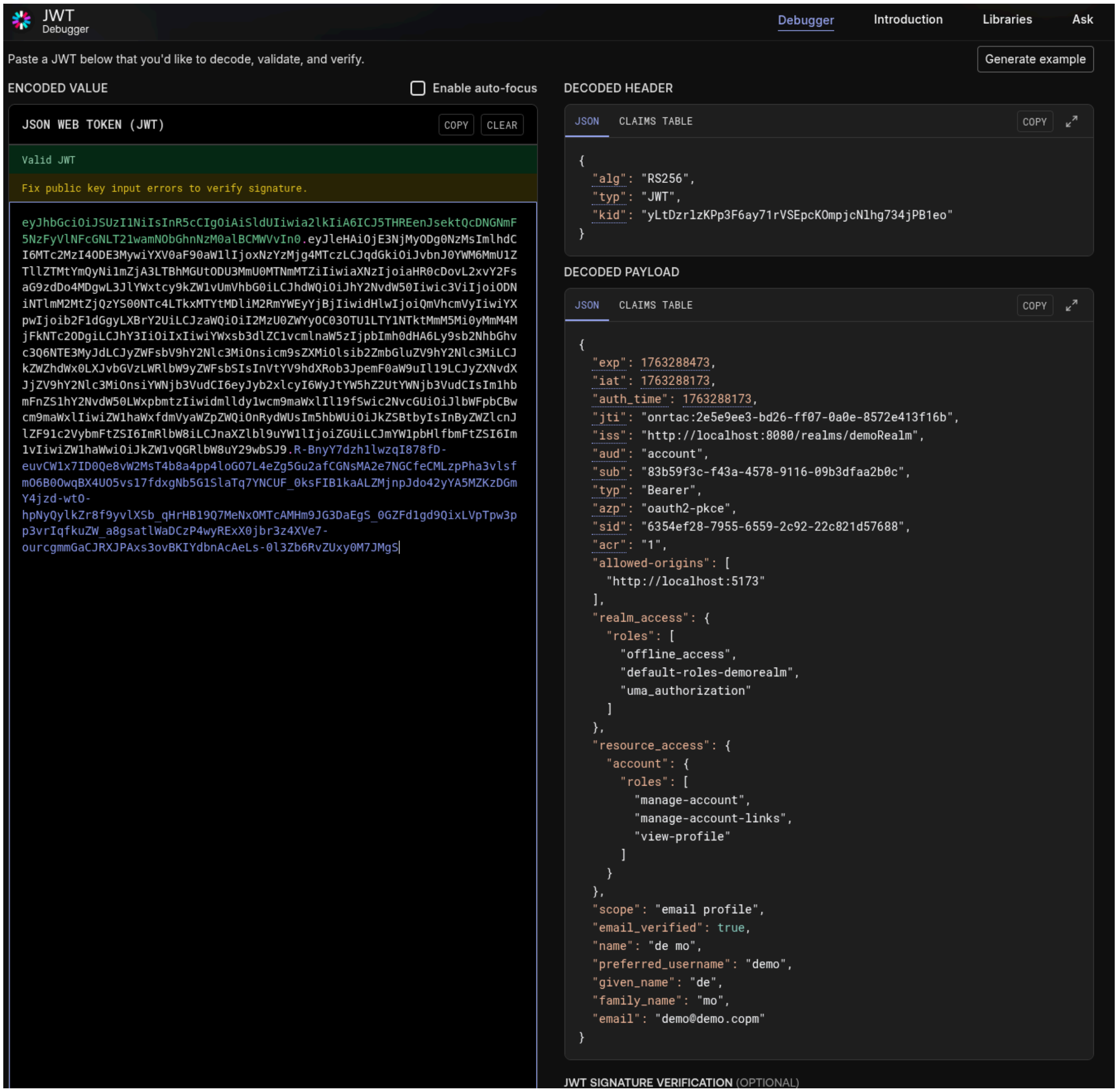
|  |  |
|---|---|
| Token Name | oauth2-pkce |
| Grant Type | Authorization Code (With PKCE) |
| Callback URL ⓘ | http://localhost:5173 |
|  | ☐ Authorize using browser |
| Auth URL ⓘ | http://localhost:8080/realms/demoRealm/p… |
| Access Token URL ⓘ | http://localhost:8080/realms/demoRealm/p… |
| Client ID ⓘ | oauth2-pkce |
| Client Secret ⓘ | Client Secret |
| Code Challenge Method ⓘ | SHA-256 |
| Code Verifier ⓘ | S |
| Scope ⓘ | e.g. read:org |
| State ⓘ | State |
| Client Authentication | Send as Basic Auth header |

Clear cookies ⓘ

Get New Access Token

- Here you need to set up the auth by adding the client and paths and then get the access token.
- After that you should be able to do requests to the java backend and get results back

## JWT.io



Paste a JWT below that you'd like to decode, validate, and verify.

ENCODED VALUE

JSON WEB TOKEN (JWT)

Valid JWT

Fix public key input errors to verify signature.

```
eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJ5THREenJsektQcDNGNmF
5NzFyVlNFcGNLT21wamNObGhnNzM0alBCMWVvIn0.eyJleHAiOjE3NjMyODg0NzMsImlhdC
I6MTc2MzI4ODE3MywiYXV0aF90aW1lIjoxNzYzMjg4MTczLCJqdGkiOiJvbnJ0YWM6MmU1Z
TllZTMtYmQyNi1mZjA3LTBhMGUtODU3MmU0MTNmMTZiIiwiaXNzIjoiaHR0cDovL2xvY2Fs
aG9zdDo4MDgwL3JlYWxtcy9kZW1vUmVhbG0iLCJhdWQiOiJhY2NvdW50Iiwic3ViIjoiODN
iNTlmM2MtZjQzYS00NTc4LTkxMTYtMDliM2RmYWEyYjBjIiwidHlwIjoiQmVhcmVyIiwiYX
pwIjoib2F1dGgyLXBrY2UiLCJzaWQiOiI2MzU0ZWYyOC03OTU1LTY1NTktMmM5Mi0yMmM4M
jFkNTc2ODgiLCJhY3IiOiIxIiwiYWxsb3dlZC1vcmlnaW5zIjpbImh0dHA6Ly9sb2NhbGhv
c3Q6NTE3MyJdLCJyZWFsbV9hY2Nlc3MiOnsicm9sZXMiOlsib2ZmbGluZV9hY2Nlc3MiLCJ
kZWZhdWx0LXJvbGVzLWRlbW9yZWFsbSIsInVtYV9hdXRob3JpemF0aW9uIl19LCJyZXNvdX
JjZV9hY2Nlc3MiOnsiYWNjb3VudCI6eyJyb2xlcyI6WyJtYW5hZ2UtYWNjb3VudCIsIm1hb
mFnZS1hY2NvdW50LWxpbmtzIiwidmlldy1wcm9maWxlIl19fSwic2NvcGUiOiJlbWFpbCBw
cm9maWxlIiwiZW1haWxfdmVyaWZpZWQiOnRydWUsIm5hbWUiOiJkZSBtbyIsInByZWZlcnJ
lZF91c2VybmFtZSI6ImRlbW8iLCJnaXZlbl9uYW1lIjoiZGUiLCJmYW1pbHlfbmFtZSI6Im
1vIiwiZW1haWwiOiJkZW1vQGRlbW8uY29wbSJ9.R-BnyY7dzh1lwzqI878fD-
euvCW1x7ID0Qe8vW2MsT4b8a4pp4loGO7L4eZg5Gu2afCGNsMA2e7NGCfeCMLzpPha3vlsf
mO6B0OwqBX4UO5vs17fdxgNb5G1SlaTq7YNCUF_0ksFIB1kaALZMjnpJdo42yYA5MZKzDGm
Y4jzd-wtO-
hpNyQylkZr8f9yvlXSb_qHrHB19Q7MeNxOMTcAMHm9JG3DaEgS_0GZFd1gd9QixLVpTpw3p
p3vrIqfkuZW_a8gsatlWaDCzP4wyRExX0jbr3z4XVe7-
ourcgmmGaCJRXJPAxs3ovBKIYdbnAcAeLs-0l3Zb6RvZUxy0M7JMgS
```

DECODED HEADER

```
{
  "alg": "RS256",
  "typ": "JWT",
  "kid": "yLtDzrlzKPp3F6ay71rVSEpcKOmpjcNlhg734jPB1eo"
}
```

DECODED PAYLOAD

```
{
  "exp": 1763288473,
  "iat": 1763288173,
  "auth_time": 1763288173,
  "jti": "onrtac:2e5e9ee3-bd26-ff07-0a0e-8572e413f16b",
  "iss": "http://localhost:8080/realms/demoRealm",
  "aud": "account",
  "sub": "83b59f3c-f43a-4578-9116-09b3dfaa2b0c",
  "typ": "Bearer",
  "azp": "oauth2-pkce",
  "sid": "6354ef28-7955-6559-2c92-22c821d57688",
  "acr": "1",
  "allowed-origins": [
    "http://localhost:5173"
  ],
  "realm_access": {
    "roles": [
      "offline_access",
      "default-roles-demorealm",
      "uma_authorization"
    ]
  },
  "resource_access": {
    "account": {
      "roles": [
        "manage-account",
        "manage-account-links",
        "view-profile"
      ]
    }
  },
  "scope": "email profile",
  "email_verified": true,
  "name": "de mo",
  "preferred_username": "demo",
  "given_name": "de",
  "family_name": "mo",
  "email": "demo@demo.copm"
}
```

JWT SIGNATURE VERIFICATION (OPTIONAL)

- Here you can check the contents of the jwt and what fields you can read out.