Keycloak Client Documentation

Release 1.2.0

Akhil Lawrence

Jan 12, 2022

Features

1 Examples **3**

2 Implementation **5**

3 Core APIs **7**

4 Extensions **9**

4.1 Authentication 9
4.2 Authorization 10
4.3 Using Flask Extension 11
4.4 Using Starlette Extension 11
4.5 Using Django Extension 12
4.6 API Reference 12

Keycloak is an open source identity and access management (IAM) solution for the modern application and services. To know more about keycloak, please visit their official website. The focus of this library is to provide easy integration with keycloak server, so that the features like authentication, authorization etc can be used in a python applications very easily.

CHAPTER 1

Examples

https://github.com/keycloak-client/keycloak-client/tree/main/examples

CHAPTER 2

Implementation

This library consists of two sections

- Core APIs
- Extensions

CHAPTER 3

Core APIs

These consists of the core interactions with the keycloak server.

CHAPTER 4

Extensions

These consists of middleware implementations for standard frameworks like Flask, Django etc. Extensions are implemented using core APIs. While integrating keycloak with your app, either you can use the core APIs directly or you can use prebuilt extensions.

4.1 Authentication

Keycloak client provides two methods called *login* and *callback*, using which you can connect to the authentication endpoints of keycloak server and perform *openid* authentication easily.

The following snippet is an example written in Flask framework

```
#! -*- coding: utf-8 -*-
from flask import Flask, redirect, request, jsonify, session, Response
from keycloak import Client


api = Flask(__name__)
api.config['SECRET_KEY'] = 'EYxuFcNqGamVU78GgfupoO5N4z2xokA58XtL0ag'
kc = Client()


@api.route('/login', methods=['GET'])
def login():
    """ Initiate authentication """
    url, state = kc.login()
    session['state'] = state
    return redirect(url)


@api.route('/login/callback', methods=['GET'])
def login_callback():
    """ Authentication callback handler """
```

(continues on next page)

(continued from previous page)

```
    # validate state
    state = request.args.get('state', 'unknown')
    _state = session.pop('state', None)
    if state != _state:
        return Response('Invalid state', status=403)

    # retrieve tokens
    code = request.args.get('code')
    tokens = kc.callback(code)

    # retrieve userinfo
    access_token = tokens["access_token"]
    userinfo = kc.fetch_userinfo(access_token)
    session["user"] = userinfo

    # send userinfo to user
    return jsonify(userinfo)


if __name__ == '__main__':
    api.run(host='0.0.0.0')
```

4.2 Authorization

Authorization is performed with the help of *UMA (User Managed Access)*

4.2.1 Generating PAT

PAT (Protection API Token) is a special token with scope *uma_protection*. Keycloak provides a method called *pat* using with you can retrieve the PAT token.

```
#! -*- coding: utf-8 -*-
from keycloak import Client

kc = Client()
kc.pat()
```

4.2.2 Generating Permission Ticket

A permission ticket is a special type of token defined by the User-Managed Access (UMA) specification that provides an opaque structure whose form is determined by the authorization server. This structure represents the resources and/or scopes being requested by a client, the access context, as well as the policies that must be applied to a request for authorization data. See this for more details.

```
#! -*- coding: utf-8 -*-
from keycloak import Client

kc = Client()
pat = kc.pat()

```

(continues on next page)

(continued from previous page)

```
resources = [
     {
          "resource_id": "8762039c-cdfa-4ef9-9f70-45248863c4da",
          "resource_scopes": ["create", "read", "update", "delete]
     }
]
ticket = kc.find_ticket(resources, pat["access_token"]
```

4.3 Using Flask Extension

```
#! /usr/bin/env python
from flask import Flask

from keycloak.extensions.flask import AuthenticationMiddleware

app = Flask(__name__)
app.config["SECRET_KEY"] = "secret0123456789"


app.wsgi = AuthenticationMiddleware(
    app.wsgi,
    app.config,
    app.session_interface,
    callback_url="http://localhost:5000/kc/callback",
    redirect_uri="/howdy",
    logout_uri="/logout"
)


@app.route("/howdy")
def howdy():
    return "Howdy!"

@app.route("/logout")
def logout():
    return "User logged out!"


if __name__ == "__main__":
    app.run(debug=True)
```

4.4 Using Starlette Extension

```
#! /usr/bin/env python
import uvicorn

from starlette.applications import Starlette
from starlette.middleware.sessions import SessionMiddleware
from starlette.responses import PlainTextResponse

from keycloak.extensions.starlette import AuthenticationMiddleware
```

(continues on next page)

(continued from previous page)

```
app = Starlette()
app.debug = True
app.add_middleware(AuthenticationMiddleware, callback_url="http://localhost:8000/kc/
↪callback", redirect_uri="/howdy", logout_uri="/logout")
app.add_middleware(SessionMiddleware, secret_key="secret0123456789")


@app.route("/howdy")
def howdy(request):
    return PlainTextResponse("Howdy!")


@app.route("/logout")
def logout(request):
    return PlainTextResponse("User logged out!")


if __name__ == "__main__":
    uvicorn.run(app)
```

4.5 Using Django Extension

Please see the examples available in https://github.com/keycloak-client/keycloak-client/tree/main/examples

4.6 API Reference