Python Keycloak Client Documentation

Release 0.2.4-dev

Peter Slump

Jan 16, 2022

Contents

The Python Keycloak Client is a set of API clients written in Python to communicate with the different API's which are exposed by Keycloak.

CHAPTER 1

Installation

```
$ pip install python-keycloak-client
```

1.1 Async

```
$ pip install python-keycloak-client[aio]
```

CHAPTER 2

Preparation

Make sure you have created a REALM and Client in Keycloak.

CHAPTER 3

Usage

Everything starts with an instance of `keycloak.realm.KeycloakRealm`

```
from keycloak.realm import KeycloakRealm


realm = KeycloakRealm(server_url='https://example.com', realm_name='my_realm')
```

```
from keycloak.aio.realm import KeycloakRealm


async def main(loop=None):
    realm_params = dict(
        server_url='https://example.com',
        realm_name='my_realm',
        loop=loop
    )
    async with KeycloakRealm(**realm_params) as realm:
        # do something
        print(realm.realm_name)

if __name__ == '__main__':
    import asyncio

    loop = asyncio.get_event_loop()
    loop.run_until_complete(main(loop))
```

3.1 OpenID Connect

The OpenID Connect entry point can be retrieved from the realm object.

```
from keycloak.realm import KeycloakRealm


realm = KeycloakRealm(server_url='https://example.com', realm_name='my_realm')

oidc_client = realm.open_id_connect(client_id='my-client',
                                    client_secret='very-secret-client-secret')
```

3.1.1 Async

```
from keycloak.aio.realm import KeycloakRealm


async def main(loop=None):
    realm_params = dict(
        server_url='https://example.com',
        realm_name='my_realm',
        loop=loop
    )
    async with KeycloakRealm(**realm_params) as realm:
        oidc_client = await realm.open_id_connect(
            client_id='my-client',
            client_secret='very-secret-client-secret'
        )
        # do something


if __name__ == '__main__':
    import asyncio

    loop = asyncio.get_event_loop()
    loop.run_until_complete(main(loop))
```

3.2 Authz (Authorization services)

The Authz client can be retrieved from the realm object.

```
from keycloak.realm import KeycloakRealm


realm = KeycloakRealm(server_url='https://example.com', realm_name='my_realm')

authz_client = realm.authz(client_id='my-client')
```

3.2.1 Async

```
from keycloak.aio.realm import KeycloakRealm


async def main(loop=None):
    realm_params = dict(
```

(continues on next page)

(continued from previous page)

```
        server_url='https://example.com',
        realm_name='my_realm',
        loop=loop
    )
    async with KeycloakRealm(**realm_params) as realm:
        authz_client = await realm.authz(client_id='my-client')
        # do something


if __name__ == '__main__':
    import asyncio

    loop = asyncio.get_event_loop()
    loop.run_until_complete(main(loop))
```

KeycloakAuthz.**entitlement**(*token*)

Client applications can use a specific endpoint to obtain a special security token called a requesting party token (RPT). This token consists of all the entitlements (or permissions) for a user as a result of the evaluation of the permissions and authorization policies associated with the resources being requested. With an RPT, client applications can gain access to protected resources at the resource server.

http://www.keycloak.org/docs/latest/authorization_services/index .html#_service_entitlement_api

Return type dict

3.3 Admin API

Manage Realms, Clients, Roles, Users etc.

http://www.keycloak.org/docs-api/3.4/rest-api/index.html

The admin API client get be retrieved from the realm object.

```
from keycloak.realm import KeycloakRealm


realm = KeycloakRealm(server_url='https://example.com', realm_name='my_realm')

admin_client = realm.admin
```

3.3.1 Async

```
from keycloak.aio.realm import KeycloakRealm


async def main(loop=None):
    realm_params = dict(
        server_url='https://example.com',
        realm_name='my_realm',
        loop=loop
    )
    async with KeycloakRealm(**realm_params) as realm:
        admin_client = realm.admin
```

(continues on next page)

(continued from previous page)

```
        # do something


if __name__ == '__main__':
    import asyncio

    loop = asyncio.get_event_loop()
    loop.run_until_complete(main(loop))
```

3.3.2 Realms

Currently there is no actual functionality available for Realm management. However this endpoint is the entrypoint for all other clients.

```
realm = realm.admin.realms.by_name('realm-name')
```

3.3.3 Clients

Manage clients

```
clients = realm.admin.realms.by_name('realm-name').clients
```

The following methods can be accessed:

`Clients.all()`

3.3.4 Roles

Manage client roles

```
roles = realm.admin.realms.by_name('realm-name').clients.by_id('#client id').roles
```

The following methods are available:

Actions on a specific role

```
role = realm.admin.realms.by_name('realm-name').clients.by_id('#client id').roles.by_
↪name('role-name')
```

The following methods are available:

3.3.5 Users

Manage users in a REALM

```
users = realm.admin.realms.by_name('realm-name').users
```

The following methods are available:

`Users.create`(*username*, ***kwargs*)

Create a user in Keycloak

http://www.keycloak.org/docs-api/3.4/rest-api/index.html#_users_resource

Parameters

- **username** (*str*) –
- **credentials** (*object*) – (optional)
- **first_name** (*str*) – (optional)
- **last_name** (*str*) – (optional)
- **email** (*str*) – (optional)
- **enabled** (*boolean*) – (optional)

3.4 UMA (User-Managed Access)

The UMA client can be retrieved from the realm object.

http://www.keycloak.org/docs/latest/authorization_services/index.html#_service_overview

```
from keycloak.realm import KeycloakRealm


realm = KeycloakRealm(server_url='https://example.com', realm_name='my_realm')

uma_client = realm.uma()
```

3.4.1 Async

```
from keycloak.aio.realm import KeycloakRealm


async def main(loop=None):
    realm_params = dict(
        server_url='https://example.com',
        realm_name='my_realm',
        loop=loop
    )
    async with KeycloakRealm(**realm_params) as realm:
        uma_client = realm.uma()
        # do something


if __name__ == '__main__':
    import asyncio

    loop = asyncio.get_event_loop()
    loop.run_until_complete(main(loop))
```

3.4.2 Resource Set management

KeycloakUMA.**resource_set_create**(*token*, *name*, ***kwargs*)

Create a resource set.

https://docs.kantarainitiative.org/uma/rec-oauth-resource-reg-v1_0_1.html#rfc.section.2.2.1

Parameters

- **token** (*str*) – client access token
- **id** (*str*) – Identifier of the resource set
- **name** (*str*) –
- **uri** (*str*) – (optional)
- **type** (*str*) – (optional)
- **scopes** (*list*) – (optional)
- **icon_url** (*str*) – (optional)
- **DisplayName** (*str*) – (optional)
- **ownerManagedAccess** (*boolean*) – (optional)
- **owner** (*str*) – (optional)

Return type str

KeycloakUMA.**resource_set_update**(*token*, *id*, *name*, ***kwargs*)

Update a resource set.

https://docs.kantarainitiative.org/uma/rec-oauth-resource-reg-v1_0_1.html#update-resource-set

Parameters

- **token** (*str*) – client access token
- **id** (*str*) – Identifier of the resource set
- **name** (*str*) –
- **uri** (*str*) – (optional)
- **type** (*str*) – (optional)
- **scopes** (*list*) – (optional)
- **icon_url** (*str*) – (optional)

Return type str

KeycloakUMA.**resource_set_read**(*token*, *id*)

Read a resource set.

https://docs.kantarainitiative.org/uma/rec-oauth-resource-reg-v1_0_1.html#read-resource-set

Parameters

- **token** (*str*) – client access token
- **id** (*str*) – Identifier of the resource set

Return type dict

KeycloakUMA.**resource_set_delete**(*token*, *id*)

Delete a resource set.

https://docs.kantarainitiative.org/uma/rec-oauth-resource-reg-v1_0_1.html#delete-resource-set

Parameters

- **token** (*str*) – client access token
- **id** (*str*) – Identifier of the resource set

KeycloakUMA.**resource_set_list**(*token*, ***kwargs*)

List a resource set.

https://docs.kantarainitiative.org/uma/rec-oauth-resource-reg-v1_0_1.html#list-resource-sets

Parameters

- **token** (*str*) – client access token
- **name** (*str*) – (optional)
- **uri** (*str*) – (optional)
- **owner** (*str*) – (optional)
- **type** (*str*) – (optional)
- **scope** (*str*) – (optional)

Return type list

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Index