NEANIAS AAI Documentation

NEANIAS AAI Team

Jul 26, 2021

Contents:

CHAPTER 1

Neanias AAI

1.1 Fit in NEANIAS Ecosystem

The NEANIAS AAI service offers a horizontal solution for all NEANIAS services to cover the common requirements of authenticated access, whether these come in the form of direct user requests or as cross service invocations. Furthermore, provisions to support some degree of the authorization of these requests is offered. The solution is based on widely accepted protocols and standards to ensure wide applicability of the approach.

The NEANIAS AAI Service is backed by the Open Source Keycloak (keycloak.org) application. The application consist of a set of web API accessible endpoints to complete the needed authentication protocol interactions, as well as a set of web graphical user interfaces to support login, profile managements and administrative operations.

1.2 Protocol

1.2.1 OIDC

We currently support Open ID Connect as the main protocol for granting access to protected resources.

1.2.2 SAML

SAML is not currently activated but can be added as an option if needed.

1.3 Realms

NEANIAS AAI currently support three realms, one for each deployment environment in a service's life-cycle.

1.3.1 Development

Applications that are in the development life-cycle should request access to the Dev Realm of NEANIAS AAI. You will need to contact the NEANIAS AAI administration team in order to register your application's redirect urls and you will be provided with a new client id and client secret

1.3.2 Staging

Applications in pre-production should request access to the Staging Realm of NEANIAS AAI. You will need to contact the NEANIAS AAI administration team in order to register your application's redirect urls and you will be provided with a new client id and client secret

1.3.3 Production

Production ready applications should request access to the Production Realm of NEANIAS AAI. You will need to contact the NEANIAS AAI administration team in order to register your application's redirect url and you will be provided with a new client id and client secret

1.4 Identity Providers

NEANIAS AAI will suport the following Identity Providers:

1.4.1 Local

Support for NEANIAS AAI local users with username password credentials.

1.4.2 NEANIAS SSO

Integration with NEANIAS SSO allows authentication to all the partner members that have access to other NEANIAS services.

1.4.3 EOSC

NEANIAS AAI is currently in the process of integrating EOSC AAI to the list of supported federated Identity Providers. This way we will be able to integrate users from all the external identity services that EOSC AAI supports.

1.4.4 Automatic assignment roles / groups

We are looking for ways to simplify and automate the process of assigning users with roles and groups. Currently NEANIAS AAI administrators will need to assign roles to users. The goal is that roles will be automatically assigned to users based on mappings between the roles/groups and properties of the client application such as the client id or the origin of the request.

1.5 Client Application

1.5.1 Process to create a client application

The process of creating new browser facing client applications in NEANIAS AAI is as follows

- You will need to specify to the AAI team in which realm your application will operate
- You will need to inform the AAI team the NEANIAS services that your app will need access to.
- You will need to provide to the AAI team a valid redirect url in your app in order to acquire an application code
- You will receive through a secure channel the client id and client secret for your application

1.5.2 IntraService invocations

The process of creating client applications for intra service communication in NEANIAS AAI is as follows

- You will need to specify to the AAI team in which realm your application will operate
- You will need to inform the AAI team the NEANIAS services that your app will need access to.
- You will receive through a secure channel the client id and client secret for your application

1.5.3 Roles

To enable role based autorization, two types of roles scopes and respective assignments are available:

- **Realm / Global scope:** A global set of roles will be defined to facilitate cross client / service user role authorities. The usage and definition of these roles will be gradually build based on service consenus and use case generalization
- **Client / Service Scope:** Each client can define the roles they requires users that utilize their services to be annotated with. Service Providers need to communicate this infroamtion to the AAI adminsitration teem for these roles to be set up

1.5.4 Groups

- **Realm / Global scope:** A global set of groups will be defined to facilitate cross client / service user groups. The usage and definition of these groups will be gradually build based on service consenus and use case generalization

1.6 User Management

Through registration users will be created but will have no roles. Your services should have a base role without which no access or very limited access will be granted. Role and group assignment will be managed from NEANIAS AAI. We are in the process of automating the assignment of basic roles but for the time being assignment will have to be done manually

1.7 Auditing

We are looking for ways to integrate NEANIAS AAI auditing in the centralised auditing services provided by NEANIAS. Login requests audits, refresh token requests audits etc will need to be available in a centralised component along with other auditable actions.

1.8 Authorization

1.8.1 Static Roles / Groups

Authorization can be achieved using the static roles / groups defined globally or in the scope of a client. User roles and groups will be available to each application through the access token. Each application can use this information to authorize the user to the protected resources. For more fine-grained resource authorization we can define more client specific roles as discussed in section *Roles* and *User Management*

1.8.2 Resource based

We are working towards Resource based authorization integration for the services in the NEANIAS Ecosystem

1.9 Development

For development reasons we provide a docker image that you can use in you local instalation to integrate with keycloak.

The image extends the original keycloak image and allows you to specify realms, clients and users for development reasons.

In order to import realms, users and clients you will need to mount a volume with the appropriate *-realm.json files to /opt/jboss/realms

1.9.1 Realm configuration

In order to import realms, users and clients to keycloak you will need to provide json files with name {realmName}-realm.json and the following format:

```
{
    //Realm Name
    "realm": "example",
    //Whether the realm is enabled or disabled by default
    "enabled": true,
    "sslRequired": "external",
    //Whether the realm allows registration
    "registrationAllowed": true,
    "privateKey": "",
    "publicKey": "",
    //What kind of credentials the realm supports
    "requiredCredentials": [ "password" ],
    //Bootstrap Users
    "users" : [
        {
            "username" : "test-user",
```

(continues on next page)

(continued from previous page)

```
            //Whether the user is enabled or disabled by default
            "enabled": true,
            "email" : "test-user@example",
            "firstName": "Test",
            "lastName": "User",
            //User credentials. Type of credential and value
            "credentials" : [
                {
                    "type" : "password",
                    "value" : "password",
                    //The credential is temporary. User will be prompted to set a␣
↪pass when he tries to sign in
                    "temporary": true
                }
            ],
            //Realm scoped roles
            "realmRoles": [ "user" ],
            //Client scoped roles
            "clientRoles": {
                "account": ["view-profile", "manage-account"]
            }
        }
    ],
    //Role configuration
    "roles" : {
        "realm" : [
            {
                "name": "user",
                "description": "User privileges"
            },
            {
                "name": "admin",
                "description": "Administrator privileges"
            }
        ]
    },
    "scopeMappings": [
        {
            "client": "test-client",
            "roles": ["user"]
        }
    ],
    //Client configuration
    "clients": [
        {
            "clientId": "test-client",
            "enabled": true,
            "publicClient": true,
            "baseUrl": "http://localhost:4200/",
            "redirectUris": [
                "http://localhost:4200/*"
            ],
            "webOrigins": []
        }
    ],
    "clientScopeMappings": {
        "account": [
```

(continues on next page)

(continued from previous page)

```
            {
                "client": "test-client",
                "roles": ["view-profile"]
            }
        ]
    }
}
```

1.9.2 Execution

In order to start a local instance of AAI Service image open a terminal and execute

```
docker run -e USER=<USERNAME> -e PASSWORD=<PASSWORD> -e IMPORT=true -p 8080:8080 -v /
↪path/to/*-realm.json/:/opt/jboss/realms gitlab.neanias.eu:5050/aai-service/server
```

This will start an instance of AAI Service at http://localhost:8080

The **USERNAME** and **PASSWORD** parameters will be used to create the initial credentials for the administrator of the AAI Service Installation

1.9.3 Misc

By default, restarting the AAI Service container will delete data and configuration from previous runs.

In order to make your changes persistent you can add a mounted volume from your filesystem to /opt/jboss/keycloak/standalone/data inside the container

For example run:

```
docker run -e USER=<USERNAME> -e PASSWORD=<PASSWORD> -e IMPORT=true -p 8080:8080 -v /
↪path/to/*-realm.json/:/opt/jboss/realms -v ./aai-service-db:/opt/jboss/keycloak/
↪standalone/data gitlab.neanias.eu:5050/aai-service/server
```

CHAPTER 2

Indices and tables

- search