User Management

NextGEOSS User Guide

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 730329.

Table of Contents

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 730329.

Introduction

The NextGEOSS User Management (UM) Service provides users with Single-Sign-On (SSO) authentication for accessing GEOSS data and services in a federated environment. It is based on OpenID Connect (OIDC) for authentication and UMA for the single-point authorization management. It allows the definition of integration points to provide a federated management of user account and user access to online services under a NextGEOSS Single-Sign-On user experience. Those federated systems could be based on different protocols: OpenID Connect, SAML2 and OAuth2 integrated through proxies.

In summary, the NextGEOSS UM Service currently demonstrates the following functionalities:

- Allow registration of users into a community (targeting the GEO/GEOSS community) and managing related user 'identity' information (user name, family name, email, telephone number, gender, ...);
- Allow registration directly via an existing social network login (from a user account on Google, Twitter, Facebook, ...) by importing that user 'identity' information;
- Allow authentication and authorization mechanisms towards acknowledged third-party services (targeting GEO/GEOSS services), based on user credentials defined at the level of the NextGEOSS UM Service;
- Allow registration of GEO/GEOSS services or applications (i.e. data harvesting, discovery, access, processing) that shall be subject to the definition of authentication and authorization mechanisms within the NextGEOSS UM Service.
- Provide SSO capability that enables a registered user to log in once, and access multiple GEO/GEOSS applications where the user signed-up already, without being required to authenticate for each application separately.

Furthermore, it will also address in a next stage the following capabilities:

- Allow integration of other GEO-related user management systems (e.g. identity providers such as ESA-https://eo-sso-idp.eo.esa.int or NASA-https://urs.earthdata.nasa.gov), to be handled as

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 730329.

- identity providers similarly to the handling of social network providers presented above, in order to provide to existing EO data users a federation of GEO/GEOSS resources. These systems could be based on different protocols: OpenID Connect, SAML2, Oauth2, ...
- Allow User-Managed Access (UMA) authorisation. The UMA framework is expected to be available on the NextGEOSS UM Service after its official v1.0 release. It will allow clients to dynamically perform authorization based on claims, scopes and policies. It will also enable dynamic registering of resources, indicating the set of claims/scopes associated with them.

This version of the user guide is focusing on the processes to register and integrate NextGEOSS client services or applications (i.e. data harvesting, discovery, access and processing) under the NextGEOSS SSO, and leverages the authentication based on OIDC protocol, and the authorization mechanisms based on OIDC scopes or UMA protocol.

The following section aims at defining the interface method between the NextGEOSS UM Service and client applications, for their developers to register a new client on the service.

1. Static Registration

In order to allow a client application to delegate its sign-in function under the NextGEOSS SSO, it is necessary to provide the following parameters:

- Application Type: An application could be either NATIVE or WEB.
- Policy and ToS URI: These resources contain the application policies regarding the usage of user personal information.
- Redirect Login / Logout URI: Only the first is mandatory. Indicates the URL or App Link where the sign-in service will redirect users after login.
- Required OAuth2 Scopes: These scopes indicate which kind of information and access the Client Application is able to grant to users (OpenID scope is mandatory and geoss_user is default for this system).
- Logo URI (Optional): This resource should contain a Logo for the Client Application.
- Client URI (Optional): This should point to a general-purpose web URI for the Client Application.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 730329.

After the application has been approved and configured, the following parameters, necessary to connect to the SSO service, will be provided:

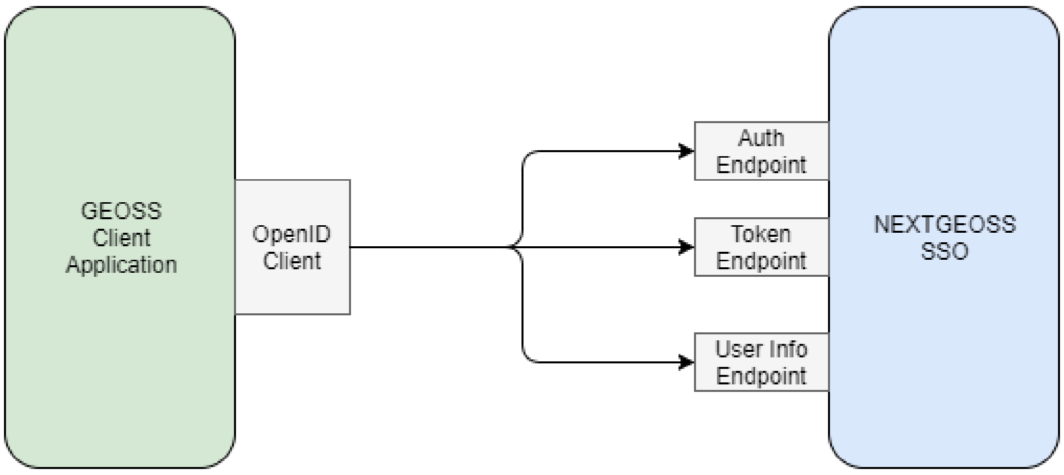
- Client ID: Unique identification sequence for your client.
- Client Secret: Necessary to perform Authentication on the Token Endpoint.

Additionally, these parameters are necessary in order to match the configuration of the application in our service:

- Encryption Algorithm: Used to sign ID Tokens, defaults to RS256
- Authentication method: Used to authenticate access to the Token Endpoint. Defaults to client_secret_basic. This implies that it is necessary to provide Client ID and secret to each Token Endpoint request.
- HTTPS Configuration: Currently, the NextGEOSS UM Service provides a self-signed intermediate certificate, which has to be configured as trusted on mobile apps in order to implement any authorization flow. Please go to Appendix I for further information.

2. Authentication

The NextGEOSS UM Service allows many different kinds of applications to use its authentication services by means of the OpenID Connect 1.0 protocol, and requests made to the available endpoints.



The OpenID Connect 1.0 protocol

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 730329.

3. Authentication Flows

In order to interact with this service, it is necessary to implement one of the three possible authorization flows defined by OpenID Connect 1.0 standard: implicit flow, authorization code flow and hybrid flow.

- Implicit flow. This method is recommended for browser-based apps. Its main steps are:

1. A request is made to the **Authorization Endpoint**. The NextGEOSS SSO will redirect the user to the NextGEOSS UM Service sign-in page.
2. The end-user will then authenticate with a set of required credentials.
3. The NextGEOSS SSO will answer back with a **redirection URI and an Access Token**.
4. The client application can now use this Access Token to request information about the end-user via the **User Info Endpoint**.

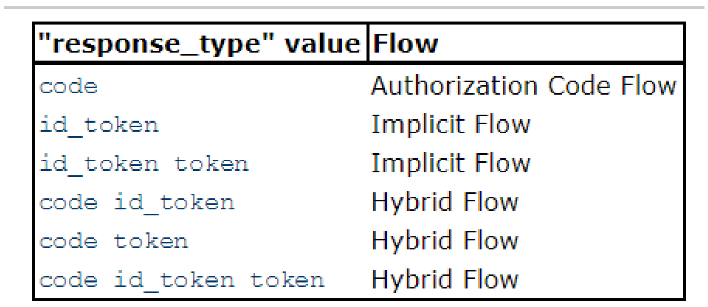
- Authorization Code flow. This method is recommended for apps running on a web server or native mobile applications. Its main steps are:

1. A request is made to the **Authorization Endpoint**. The NextGEOSS SSO service will redirect the user to NextGEOSS UM Service sign-in page.
2. The end-user will then authenticate with a set of credentials required.
3. The NextGEOSS SSO will answer back with an **authorization code**.
4. The Client can now use the received code to request an Access Token through the **Token Endpoint.**
5. Once the client application has acquired an Access Token, it will be possible to request information about the end-user via the **User Info Endpoint**.

- Hybrid Code flow. This method merges characteristic from both of the previously mentioned methods. Currently, the usage of this method is not recommended with the NextGEOSS UM Service.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 730329.

To implement any of the flows, it is necessary to specify the response type. OpenID Connect specification indicates the combination of response types necessary to implement each flow:



"response_type" value	Flow
`code`	Authorization Code Flow
`id_token`	Implicit Flow
`id_token token`	Implicit Flow
`code id_token`	Hybrid Flow
`code token`	Hybrid Flow
`code id_token token`	Hybrid Flow

OpenID Connect "response_type" Values

4. *Client Authentication*

When accessing the Token Endpoint, clients using the implicit flow are not required to be authenticated, but if the client uses the Authorization Code Flow, it must provide its credentials. These credentials will be provided by the NextGEOSS SSO: client_id and client_secret.

5. *Request Endpoints*

All endpoints require a set of mandatory parameters in order to generate a valid response. Their URLs can be obtained by means of a Discovery URI that answers back with a set of endpoints and their URLs. All these URLs must be accessed by means of a GET request, and require a set of mandatory parameters:

- Discovery URI: https://um.nextgeoss.eu/.well-known/openid-configuration
- Authorization Endpoint (GET): https://um.nextgeoss.eu/oxauth/restv1/authorize
  - Parameters:
    - scope: The request should include an array of scopes, with one of them being "openid".
    - response_type: Its value should be desired combination according to the OpenID Connect response type table.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 730329.

- **client_id:** Provided by NextGEOSS.
- **redirect_uri:** It should point to the client application and match the URI given on the acceptance request.
- **state (optional):** Opaque value used to maintain state between the request and the callback. Typically, Cross-Site Request Forgery (CSRF, XSRF) mitigation is done by cryptographically binding the value of this parameter with a browser cookie.
- **nonce (optional / required for Implicit Flow):** String value used to associate a Client session with an ID Token, and to mitigate replay attacks

- Example:

GET
oxauth/restv1/authorize?scope=openid&client_id=@!5C7F.E36B.5DE3.15EE!0001!6B53.87B4!0008!A121.D32B.8BCD.4E14&redirect_uri=app://test&response_type=code

If the application is trying to authenticate without user input, user credentials must be provided through the Authorization header. The code will be encoded as a parameter in the Location response header.

- Token Endpoint (POST): https://um.nextgeoss.eu/oxauth/restv1/token
  - Parameters:
    - **grant_type**: implicit / authorization_code (depending on the authorization flow).
    - **Code***: Used only with grant_type=authorization_code
    - **redirect_uri**: It should point to the client application and match the URI given on the acceptance request.
    - **scope**: The request should include an array of scopes, with one of them being “openid”.
    - **client_id*:** Provided by NextGEOSS, only necessary with grant_type=authorization_code.
    - **client_secret*:** Provided by NextGEOSS, only necessary with grant_type=authorization_code

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 730329.

        - **username (optional):** Use only when no user interaction is required
        - **password (optional):** Use only when no user interaction is required

- User-Info Endpoint (GET): https://um.nextgeoss.eu/oxauth/restv1/userinfo
    - Parameters:
        - **access_token**: Acquired via Token or Authorization endpoints.
    - Example:
        - GET oxauth/restv1/userinfo?access_token=<TOKEN>
- Token Introspection Endpoint (GET): https://um.nextgeoss.eu/oxauth/restv1/introspection
    - Parameters:
        - **access_token**: Acquired via Token or Authorization endpoints.
    - Example:
        - GET oxauth/restv1/introspection?access_token=<TOKEN>

6. Authorization based on scopes

It is possible to define a list of attributes indicating which users can or cannot access a service/resource. These attributes can be associated to an OpenID scope (default is geoss_user) that allows to discriminate the access to the resources/operations between users. So far, the following basic boolean attributes are defined for each end-user:

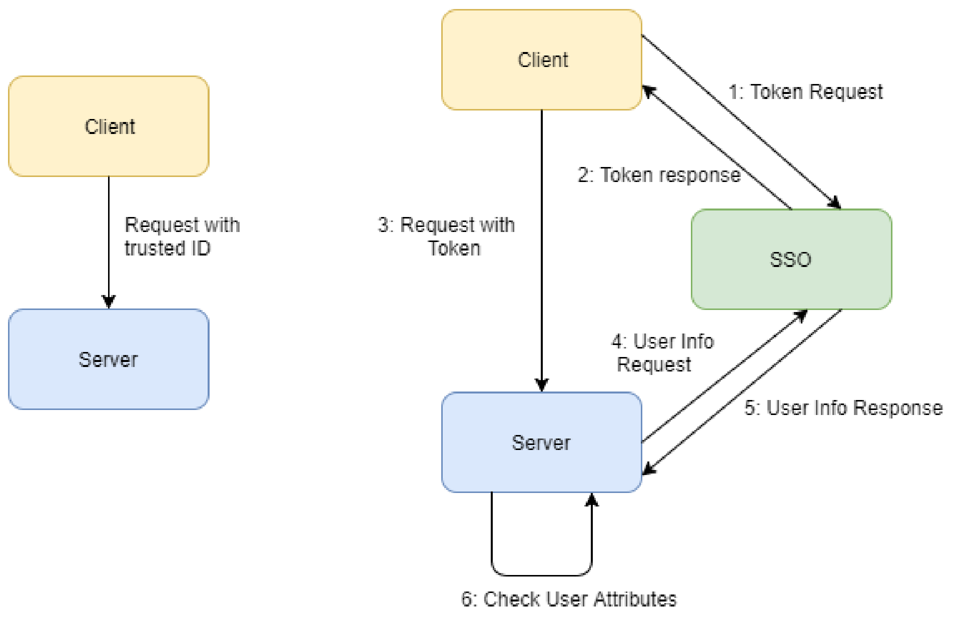
- Harvesting (harvestingUser): Allows retrieval of metadata from a registered Data Provider API. By default, the attribute is set to false. The administrator will be able to modify the attribute value.
- Discovery (discoveryUser): Allows querying of records from a registered catalog API. By default, the attribute is set to true.
- Catalogue (catalogueUser): Allows updating/modifying records from a registered catalog service . By default, the attribute is set to false. The administrator will be able to modify the attribute value.
- Access (accessUser): Allows direct access to data on a registered data server. By default, the attribute is set to false. The administrator will be able to modify the attribute value.
-

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 730329.

- Processing (processingUser): Allows the processing of datasets on a registered processing service. By default, the attribute is set to false. The administrator will be able to modify the attribute value.

- Analytics (analyticsUser): Allows access to NextGEOSS analytics (i.e: User Management KPIs). By default, the attribute is set to false. The administrator will be able to modify the attribute value.

All this information can be used to provide claims about users interacting with external clients and using the authentication flows previously mentioned. On the other hand, internal clients should be implemented using a set of specific default users, taking into account the following figure and sections.



Access control via tokens, allocated from User Attributes and OpenID scopes

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 730329.

7. Server-Side Implementation

Any component within NextGEOSS should define which user attributes are required to interact with its services. After that, any received request must contain an access token (obtained using the Authorization/Token Endpoint).
This access token can be included in an internal request to the NextGEOSS UM Service's **User Info Endpoint**, which returns the set of default attributes that can be checked internally by the service in order to determine if access should be granted.

8. Client-Side Implementation

Any client trying to connect to a service must request an access token in order to provide claims about the user. This can be achieved with no user interaction by calling the Token Endpoint directly, using a special grant type named "password".
POST /oxauth/restv1/token
Body (Form)

- scope=openid geoss_user profile
- client_secret=<secret>
- redirect_uri=app://test
- client_id=<client_id>
- grant_type=password
- username=<user>
- password=<password>

9. Libraries and documentation

In order to develop a Client Application that satisfies the needs specified by OpenID Connect 1.0 and NextGEOSS SSO, specifications can be found here:

- OAuth2.0: https://oauth.net/2/

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 730329.

- OpenID Connect: http://openid.net/specs/openid-connect-core-1_0.html

All the information provided in the previous sections combined with the specifications of the standards involved in this service can be more easily implemented through available libraries dedicated to each platform and environment:

- Single Page Applications (SPA): applications based on JavaScript clients. https://github.com/IdentityModel/oidc-client-js

- Native Apps for iOS and macOS: https://github.com/openid/AppAuth-iOS

- Native Apps for Android: https://github.com/openid/AppAuth-Android

- Reverse Proxy Solutions: https://github.com/pingidentity/mod_auth_openidc

- Oauth2.0 plugin for JIRA: https://marketplace.atlassian.com/plugins/com.miniorange.oauth.jira-oauth/server/overview

- Web applications:
  - Django rest framework for OAuth2.0: https://github.com/PhilipGarnero/django-rest-framework-social-oauth2
  - Python social Auth: http://python-social-auth.readthedocs.io/en/latest/
  - Django OAuth Toolkit: https://django-oauth-toolkit.readthedocs.io/en/latest/

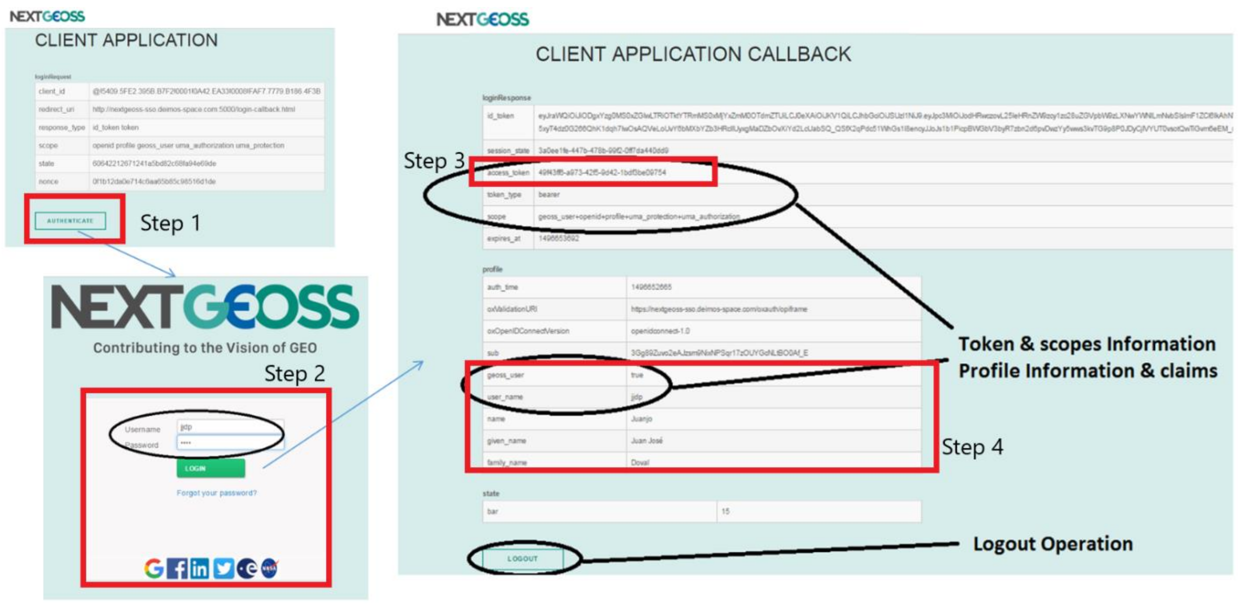
These libraries contain documentation and samples that implement a basic Client App that can be used on NextGEOSS. For specific examples, please contact the development team.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 730329.

10. *Integration samples*

Sample 1: Single Page Application - JavaScript client

This example implements Implict Flow which does not require client authentication. Steps 1 through 4 of this Flow are marked on the figure.



loginRequest

client_id	@!5409.5FE2.395B.B7F2!0001!0A42.EA33!0008!FAF7.7779.B186.4F3B
redirect_uri	http://nextgeoss-sso.deimos-space.com:5000/login-callback.html
response_type	id_token token
scope	openid profile geoss_user uma_authorization uma_protection
state	60642212671241a5bd82c68fa94e69de
nonce	0f1b12da0e714c0aa65b65c98516d1de

Username: jidp
Password: ••••

loginResponse

id_token	[illegible]
session_state	3a0ee1fe-447b-478b-99f2-0ff7da440dd9
access_token	49f43ff6-a973-42f5-9d42-1bdf3be09754
token_type	bearer
scope	geoss_user+openid+profile+uma_protection+uma_authorization
expires_at	1496653692

profile

auth_time	1496652665
oxValidationURI	https://nextgeoss-sso.deimos-space.com/oxauth/opiframe
oxOpenIDConnectVersion	openidconnect-1.0
sub	3Gg89Zuvo2eAJzsm9NixNPSqr17zOUYGdNLtBO0Af_E
geoss_user	true
user_name	jidp
name	Juanjo
given_name	Juan José
family_name	Doval

state

bar	15

Interaction sequence for Single Page Application - JavaScript client

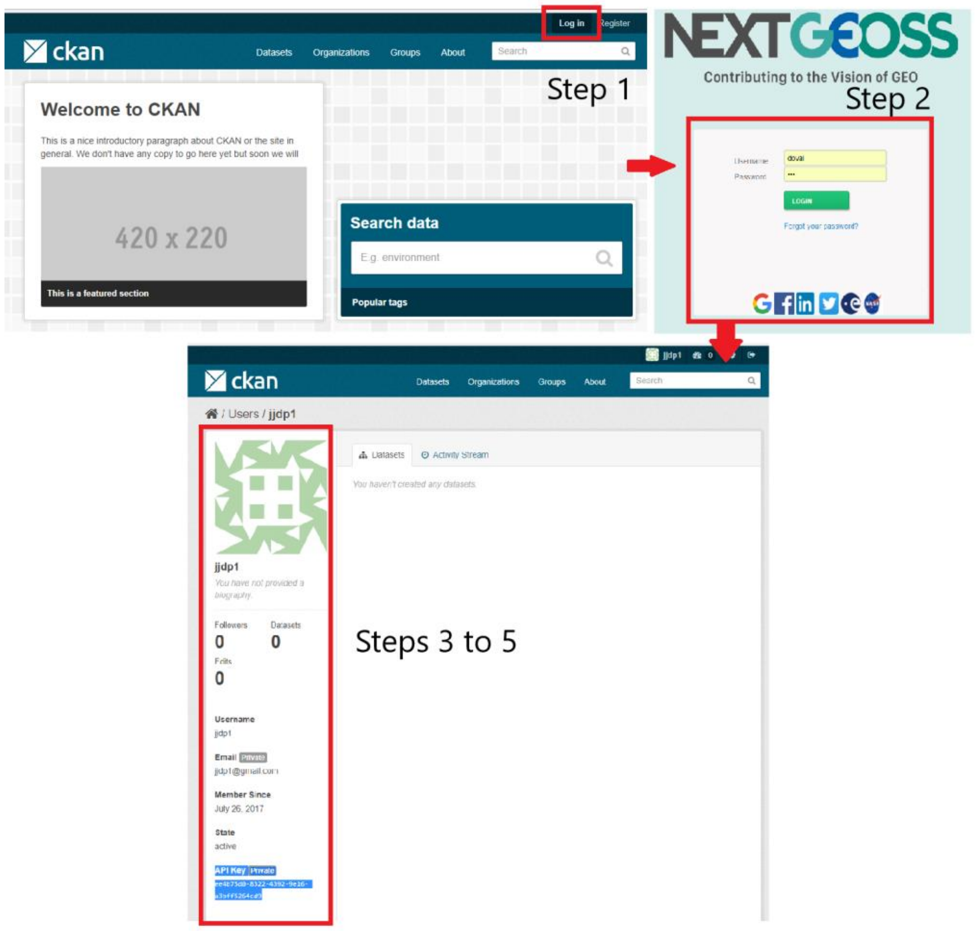
- The authenticate button performs a call to the Authorization Endpoint, which redirects the user to NextGEOSS Log-In page (Step 1).
- Once the user has entered a username and a password successfully, a redirect to a callback URI is triggered containing the access token (Steps 2 and 3).
- This token can be used to obtain user info, mainly for authentication purposes, but also access authorization based on scopes (Step 4).

Sample 2: Web-based Application - CKAN client

This method implements an Authorization Code Flow with basic client authentication. All the steps related to this Flow are mostly performed on the back-end of the application (requests to authentication, token and user info endpoints). Additionally, the authentication requests could include different scopes if there is a need to restrict access to some resources.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 730329.

Steps 1 to 5 from the Authorization Code Flow:



This is a nice introductory paragraph about CKAN or the site in general. We don't have any copy to go here yet but soon we will

You haven't created any datasets.

You have not provided a biography.

jjdp1

jjdp1@gmail.com

July 26, 2017

active

Interaction sequence for Web-based Application - CKAN client

- The authenticate button performs a call to the Authorization Endpoint, which redirects the user to NextGEOSS Log-In page (Step 1)
- Once the user has entered a username and a password successfully, a redirect to a callback URI is triggered containing the authorization code (Step 2).
- This code must be used to perform a call to the Token Endpoint (Steps 3 and 4). After obtaining it, a call to the User Info Endpoint will allow the application to acquire scopes and information (Step 5) about the user that can be used to generate a profile in the application (in this case, assigning an API key).

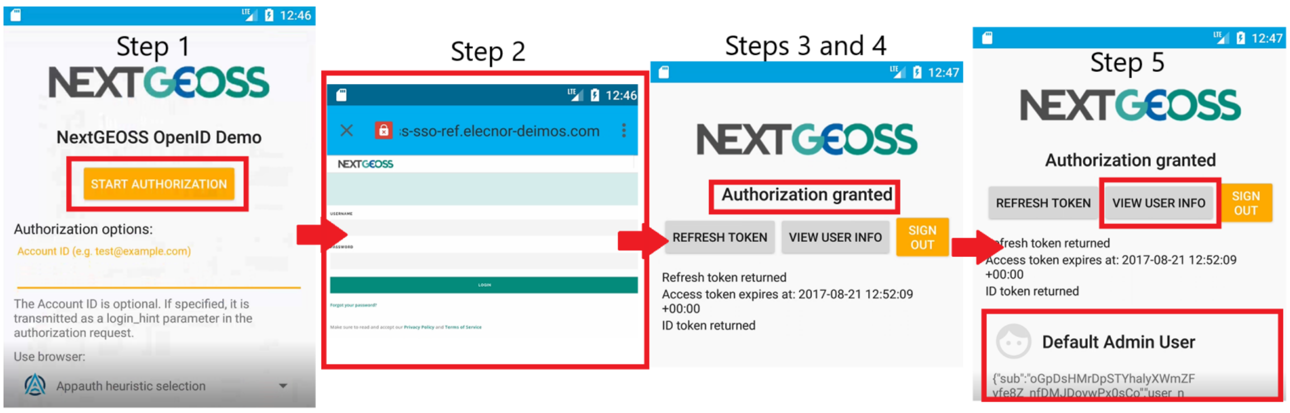
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 730329.

Sample 3: Native Mobile Application - AppAuth Android

Similar to the previous example, this method implements an Authorization Code Flow with basic client authentication. All the steps related to this Flow are mostly performed on the back-end of the application (requests to authentication, token and user info endpoints). Additionally, the authentication requests could include different scopes if there is a need to restrict access to some resources.

Note: for the configuration of Android SSL connections, when configuring an OpenID Client on a native mobile application, it might be necessary to use an intermediate certificate. **NextGEOSS will provide this intermediate certificate to developers.** After that, this certificate should be included on the application and added as an SSL security exception.

Steps 1 to 5 from the Authorization Code Flow:



Refresh token returned
Access token expires at: 2017-08-21 12:52:09 +00:00
ID token returned

{"sub":"oGpDsHMrDpSTYhalyXWmZF yfe8Z_nfDMJDoywPx0sCo","user_n

Interaction sequence for Native Mobile Application - AppAuth Android

- The authenticate button performs a call to the Authorization Endpoint, which redirects the user to NextGEOSS Log-In page (Step 1).
- Once the user has entered a username and a password successfully, a redirect to a callback URI is triggered containing the authorization code (Step 2).
- This code must be used to perform a call to the Token Endpoint (Step 3 and 4). After obtaining it, a call to the User Info Endpoint will allow the application to acquire scopes and information (Step 5) about the user that can be used to generate a profile in the application (in this case, assigning an API key).

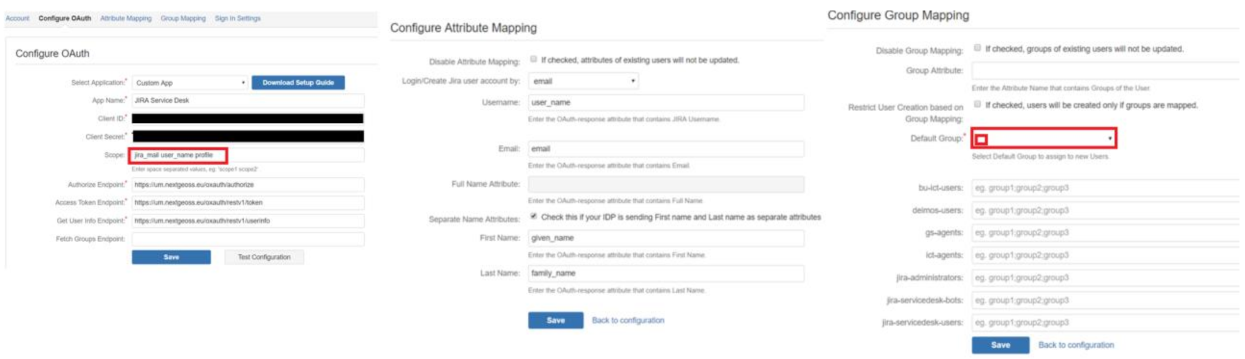
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 730329.

Sample 4: OAuth2 plugins - JIRA Service Desk

This method will implement the Authorization Code Flow using an OAuth2 plugin available on Atlassian Marktplace available here. Since there are no usable OpenID Connect plugins available, this method only uses OAuth2, meaning that session management won't be enabled and identity tokens won't be encrypted or signed by default. The configuration steps are:

- Request a Client ID and Client Secret

- Fill in the parameters as shown on the following figures:



Configure OAuth

Select Application:* Custom App
App Name:* JIRA Service Desk
Client ID:* [redacted]
Client Secret:* [redacted]
Scope: jira_mail user_name profile
Enter space separated values, eg: 'scope1 scope2'
Authorize Endpoint:* https://um.nextgeoss.eu/oxauth/authorize
Access Token Endpoint:* https://um.nextgeoss.eu/oxauth/restv1/token
Get User Info Endpoint:* https://um.nextgeoss.eu/oxauth/restv1/userinfo
Fetch Groups Endpoint:

Configure Attribute Mapping

Disable Attribute Mapping: ☐ If checked, attributes of existing users will not be updated.
Login/Create Jira user account by: email
Username: user_name
Enter the OAuth-response attribute that contains JIRA Username.
Email: email
Enter the OAuth-response attribute that contains Email.
Full Name Attribute:
Enter the OAuth-response attribute that contains Full Name.
Separate Name Attributes: ✓ Check this if your IDP is sending First name and Last name as separate attributes
First Name: given_name
Enter the OAuth-response attribute that contains First Name.
Last Name: family_name
Enter the OAuth-response attribute that contains Last Name.

Configure Group Mapping

Disable Group Mapping: ☐ If checked, groups of existing users will not be updated.
Group Attribute:
Enter the Attribute Name that contains Groups of the User.
Restrict User Creation based on Group Mapping: ☐ If checked, users will be created only if groups are mapped.
Default Group:*
Select Default Group to assign to new Users.
bu-ict-users: eg. group1;group2;group3
delmos-users: eg. group1;group2;group3
gs-agents: eg. group1;group2;group3
ict-agents: eg. group1;group2;group3
jira-administrators: eg. group1;group2;group3
jira-servicedesk-bots: eg. group1;group2;group3
jira-servicedesk-users: eg. group1;group2;group3

Configuration for OAuth2 plugins - JIRA Service Desk

- After this step, JIRA admins will be given a callback URL that should be passed on to NextGEOSS UM in order to complete the last configuration step.

- Additional configuration parameters:
    - <u>IMPORTANT:</u> The option for "External Management of Users" should be enabled to avoid discrepancies between JIRA records and NextGEOSS UM records

    - A default group should be defined using JIRA procedures. All users coming from NextGEOSS UM that log-in for the first time will be included in this group. If the administrator wants to give higher permissions to one of these users, the process has to be manual using JIRA procedures.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 730329.

After this steps, a new button for OAuth sign-in will be made available by this plugin. Interacting with this button will redirect to NextGEOSS UM service as shown in previous examples of this specific kind of authentication flow.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 730329.