Guide to EPOS AAAI for Service Provider

Version 0.2, 11.12.2019

This guide provides information about how to integrate portals and EPOS web services with EPOS AAAI.

Introduction to EPOS AAAI

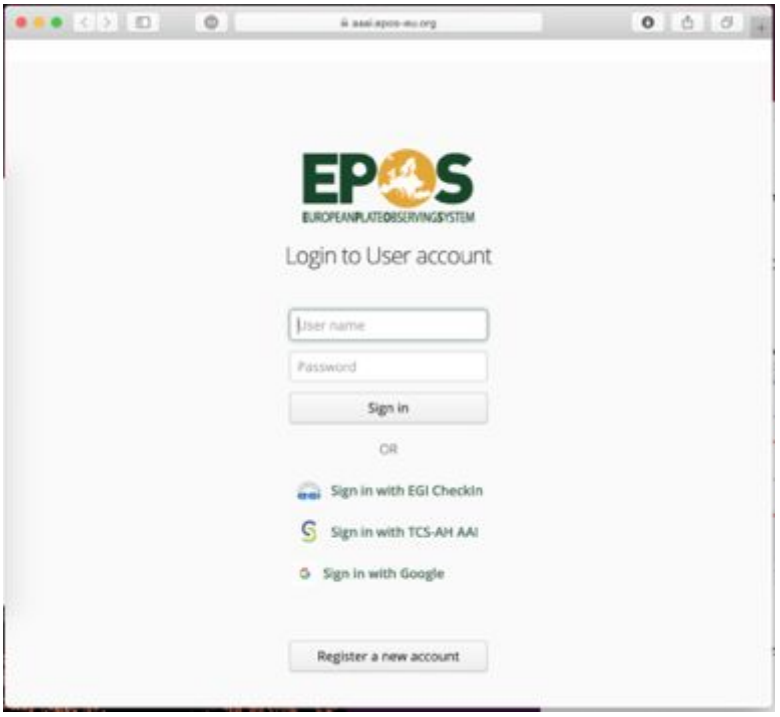
Functions

EPOS AAAI is an integrated service for:

- Authentication, understood as central EPOS service for keeping user credentials and ensure a user is recognized in EPOS community
- Authorization, understood as a secure access to authentication attributes agreed in the EPOS community, that would be useful in the specific component to enable access to specific resources and functions (details are not yet defined)
- Accounting, understood as a collector of usage data (details are not yet defined)

Address and Technical bases

EPOS AAAI service is at: **aaai.epos-eu.org**



The service is based on UNITY-IDM software and provides wide functionalities documented at http://www.unity-idm.eu/documentation/unity-2.8.0/. The software provides wide configuration and supports a wide range of Authentication and Authorization standards.

Main contacts:

- Responsibility for the EPOS AAAI and instance is at ACC Cyfronet AGH.
- Responsible person: Tomasz Szepieniec <t.szepieniec@cyfronet.pl>
- Technical support: Wojciech Ziajka <w.ziajka@cyfronet.pl>

Features

EPOS AAAI provides the following features:

- Registration for local accounts
- Registration/login through external Authentication Providers: Google Account, EGI CheckIn Account, TCS-AH account (others will be added soon)
- Required profile information and complementary registration for external providers
- Registration for OAuth2 Service Providers (valid for Portal)
- Authentication based on OAuth2 tokens
- Federated authentication based on OpenID Connect scheme

Guide for Integrating with Portals

This section is dedicated for service providers that want to use EPOS AAAI as Authentication Provider. It means that a Service will allow EPOS Users to log in to the Service using EPOS AAAI account.

External Authentication Explained

Authentication will be performed based on OpenID Connect (https://openid.net/connect/) which is a specific scenario (protocol) build on top of broader OAuth2 2.0 protocol.

Assuming that a user is going to use ICS services and needs to login into the system. Authentication process has the following steps:

1. The Portal redirect a user to a dedicated EPOS AAAI *webentry*, which enable EPOS AAAI to recognize the Portal that initialized the connection.
2. The User is authenticating in EPOS AAAI webpage.
3. EPOS AAAI redirects the User back the Portal (using registered *return URL*) passing connection *token* (according to OpenidConnect)
4. The Portal might use *token* to get *user info*, *which is a set of* attributes of the User. The token is used as an authentication mechanism that represents the user. It might be used to in delegation scenarios, too.

Developers of the Portal may use one of the many client libraries, that are available in most the languages. Please refer the following page to look for solutions available for your Portal technology: https://openid.net/developers/certified/.

More details can be found at https://openid.net/specs/openid-connect-basic-1_0.html. Reading this please have in mind that EPOS AAAI plays a role of OpenID Provider (OP) and a Portal is named Relying Party (RP).

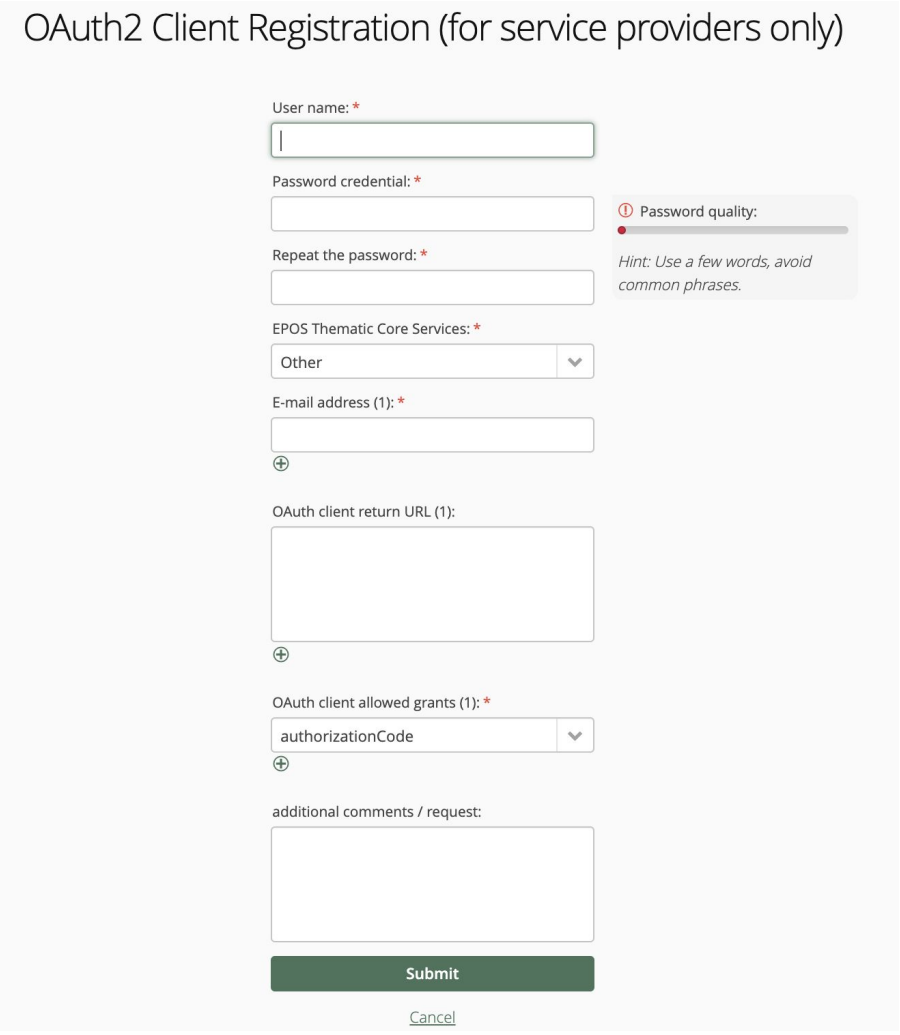
Details related to the exact implementation of this process in UNITY are documented in http://www.unity-idm.eu/documentation/unity-2.8.0/manual.html#endp-oauth-as

Steps to perform

Registration of the Provider

The first step to integrate portal is to fill the dedicated registration form available here:

https://aaai.epos-eu.org/unitygw/pub?form=OAuth2%20Client%20Registration#!registration



Comments for the field in the form:

- **User name** & **password** is for authenticating the portal in EPOS AAAI
- **EPOS TCS** is for better grouping of providers

- **E-mail address** will be used as a contact for providers, you can provide more than one e-mail
- **OAuth client return URL** is a list of URLs to which the user can be redirected after successful authentication. EPOS AAAI is configured to support wildcards in return URLs. Please follow the following (from UNITY documentation):
    - The first allowed URI should rather be a plain URI as will be used as a default return URI if client has not provided any.
    - Ant-style wildcards use single star to match arbitrary characters in single path segment and two stars to match strings across path segments.
- **OAuth client allowed grants** is a list of methods that will be used. It depends on your library. If not sure add "all supported".

In a few days after your registration you will be contacted by AAAI operator to confirm the account and relation with EPOS. After verification the account will be activated.

Configuration of OAuth Client

Depending on OAuth Client library that you choose the configuration may look different, but the most needed specific URL and parameters are provided under Provider configuration: https://aaai.epos-eu.org/oauth2/.well-known/openid-configuration

Important URLs are the following:

- Authorisation: https://aaai.epos-eu.org/oauth2-as/oauth2-authz
- Token: https://aaai.epos-eu.org/oauth2/token
- User Info: https://aaai.epos-eu.org/oauth2/userinfo

In the process of testing, in case of something is not working correctly, you can contact EPOS AAAI technical support. Usually checking logs on both client and server side makes it easier to fix the issues.

Guide for Integration with Web Services

This section is dedicated to providers that aim to integrate their web service (later referred as WS) with EPOS AAAI.

Please note: **Authentication described here is a standard for TCS web services connected to ICS. Namely, ICS will always add an authentication header with the user token. WS may perform authentication and/or accounting action based on this or ignore it.**

Delegation explained

The term delegation (more precisely authorization delegation) means that the service is acting on behalf of a subject (a user) on client service. Therefore, the client service perform authentication not based on the service that sends the request, but based on the user that is authorized (or not) to access specific resources or performs specific operations.

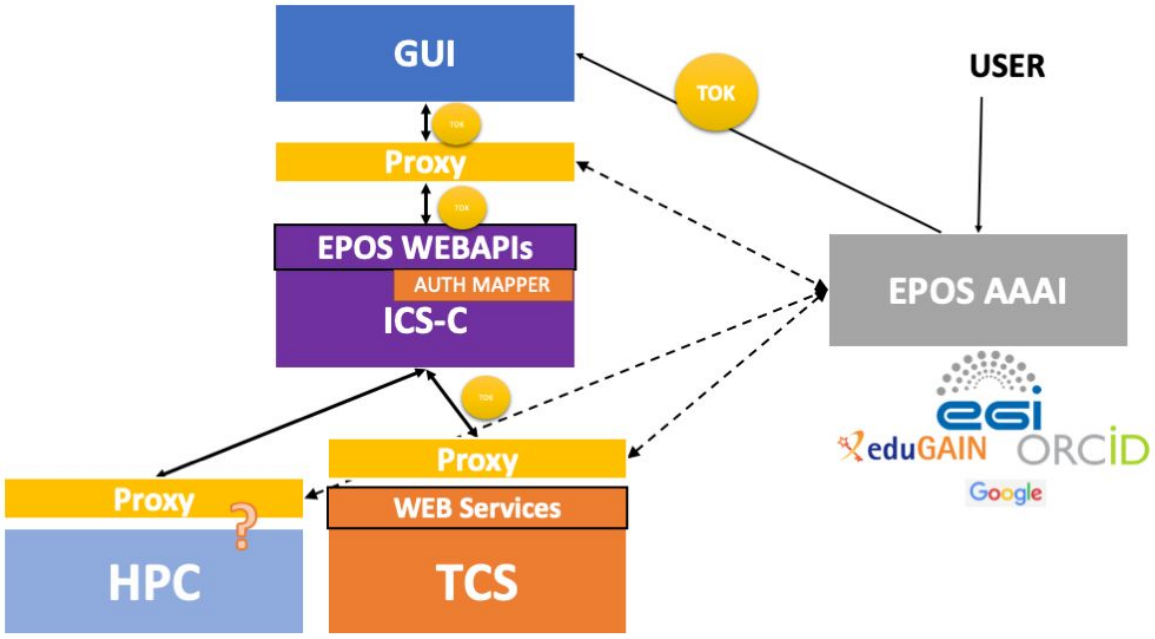
In EPOS, due to distributed architecture and many access points of using services, it is important that each WS have the possibility to execute authentication and authorization based on the user profile data. For example, the user may call a service through ICS, ICS-D or using a script. In each case, the access should be granted the same way.

In this scheme full control of authorization is in providers hands. EPOS AAAI is however responsible to provide agreed and reliable attributes of the user.

Another aspect is that all EPOS TCS web services are open for access from the whole Internet (it is not easy to restrict usage based on origin IP). Integrating authentication based on EPOS AAAI give a possibility to limit access and avoid potential waste for resources due to uncontrolled usage or DoS attacks (Denial of Service).

Scheme of delegation in EPOS

In the diagram, we show the simplified architecture of EPOS services and how delegation is performed.



The idea is that a *token* obtained from EPOS AAAI in ICS GUI is transferred to other services as the authentication method. Each of them has a possibility to check the validity of the token in EPOS AAAI and obtain information about user profile (list of user attributes).

This scheme is useful also from the perspective of accounting. Making EPOS AAAI the central place for checking tokens, makes it possible for global collecting information about the identified usage.

Integrating token-based authentication and authorization in WS

From the perspective of a single service (WS) the idea is simple to grasp and easy to implement as it is based on standard HTTPS mechanisms.

Method of Authentication

Each WS call in EPOS should have standard authentication HTTP header with a token. ICS GUI and WebAPI should pass a valid token to all other services they use, including TCS services.

HTTP standard says that the WS may ignore any header, so service providers are free to process this header based on the procedure described below or do nothing. Therefore introducing this header to WS should not spoil any existing integration.

The authentication header looks like this:

```
Authorization: Bearer AbCdEf123456
```

where “AbCdEf123456” is a token that represents a user.

The best way to validate such token is to use it to obtain user into from EPOS AAAI service. EPOS AAAI is configured to give each user access to their profile (attributes) on URL: https://aaai.epos-eu.org/oauth2/userinfo. This request must be authorized, and one of the method to authorize this request is to use the token!

So, to validate the token and obtain user attribute you need only to send GET request to the URL above with the same Authentication header.

Using curl for this would be the following:

```
export TOKEN=token_payload
curl -v --header "Authorization: Bearer $TOKEN" https://aaai.epos-eu.org/oauth2/userinfo
```

Response to such request might be the following:

- In case the token is valid:
  - response: HTTP 200
  - Payload with JSON with:
    - User persistence ID under “sub”
    - Profile attributes under “scope”

    Example for the author of this guide:

```
{
   "sub":"4554df41-bb28-4fa8-9682-566f176813ad",
   "lastName":"Szepieniec",
   "firstname":"Tomasz",
   "eduPersonUniqueId":"xxxxxxxxxxxxxxxxxxx@aaai.epos-ue.org",
   "email":"t.szepieniec@cyfronet.pl"
}
```

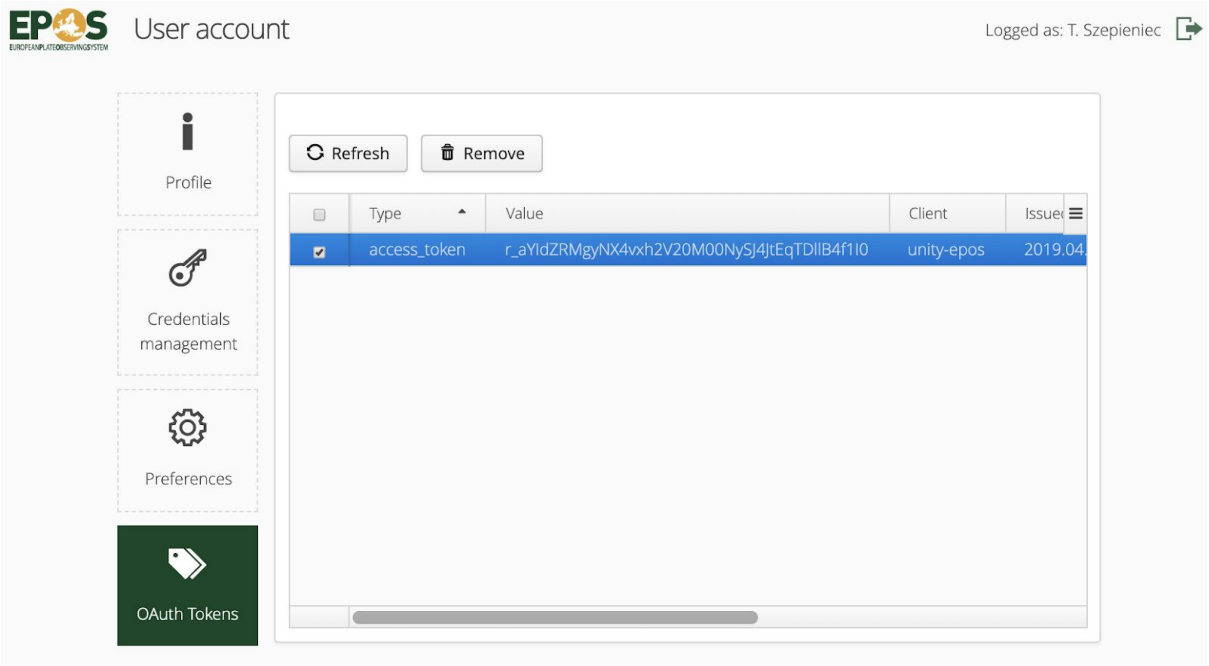

- In case the token is invalid:
    - response : HTTP 403
- In case the token is missing:
    - response : HTTP 401

Obtaining a Token for Testing

To obtain a valid token for testing you need to:

- Log in to one of EPOS Portals
    - ICS GUI
    - TCS AH (https://tcs.ah-epos.eu/) choosing EPOS AAAI as the method of authentication
- Open page: https://aaai.epos-eu.org/home/home and choose “OAuth Tokens” tab

The screen should look as below:



You can copy “Value” of the token in the table. The token is valid for 1 hour in the current configuration.

Legacy services case

One of the requirements for EPOS AAAI was that authentication system must be feasible to implement for legacy service (services that cannot be reimplemented or altered for various reasons). So, AAAI team tried to find a way to make possible using authentication also in such case.

The idea was to develop, so-called *authentication proxy* that can be implemented in various ways. Such proxy would perform validation of the token and act according to WS policy. In case the validation is successful, the proxy can translate authentication to the one required by the legacy system (for example translate EPOS token to a general EPOS account).

A feasibility study was done using a lua script (https://www.lua.org/) added to the NGINX web server (https://www.nginx.com/). Installation requires two steps:

Step 1: NGINX with LUA Support

The simplest way to have lua script support in NGINX is to use *openresty* (https://openresty.org/en/). After sources are downloaded and extracted from the archive, we can compile openresty.

```
./configure --prefix=/tmp/openresty --with-http_ssl_module

make

make install
```

Step 2: Configuration of Proxy

Now, we add lua script to NGINX configuration. We will use:

- NGINX *proxypass* to invoke authorization endpoint
- lua to parse authorization response and set required headers

```
http {
    ...
    upstream api {
        server 127.0.0.1:8083;
    }

    server {
        listen       8081;
        server_name  localhost;
```

```
        location / {
            access_by_lua_block {
                local res = ngx.location.capture("/auth")

                if res.status == 200 then
                    local cjson = require("cjson")
                    local value = cjson.decode(res.body)
                    ngx.req.set_header("X-Auth-UserId", value["sub"])
                    ngx.req.set_header("X-Auth-Scope", value["scope"])
                else
                    ngx.exit(ngx.HTTP_UNAUTHORIZED)
                end
            }
            proxy_pass http://api;
        }

        location = /auth {
            internal;
            proxy_pass aaai.epos-eu.org/oauth2/userinfo;
            proxy_pass_request_body off;
            proxy_set_header Content-Length "";
        }
    }
}
```

The above configuration provides the following behavior (that can be modified according to needs) of the legacy WS located at `http://api` :

- In case the token is invalid or missing response of the service is HTTP_UNAUTHORIZED (HTTP 401)
- In case the token is valid proxy sets two additional headers (an example of adaptation to legacy system) `X-Auth-UserId` and `X-Auth-Scope` and call the target WS.

Note, that `/auth` section is using NGINX proxy (forwarding), so it just keeps all original headers, delete body (if any) and such request forward to EPOS AAAI.