Security handling

Version 1.2

Version History

Version	Date	Comment
1.0	01.01.2021	Initial Version
1.1	20.06.2022	Refinement of Login Request recommendation
1.2	20.02.2024	Refresh Endpoint and OpenId Connect added

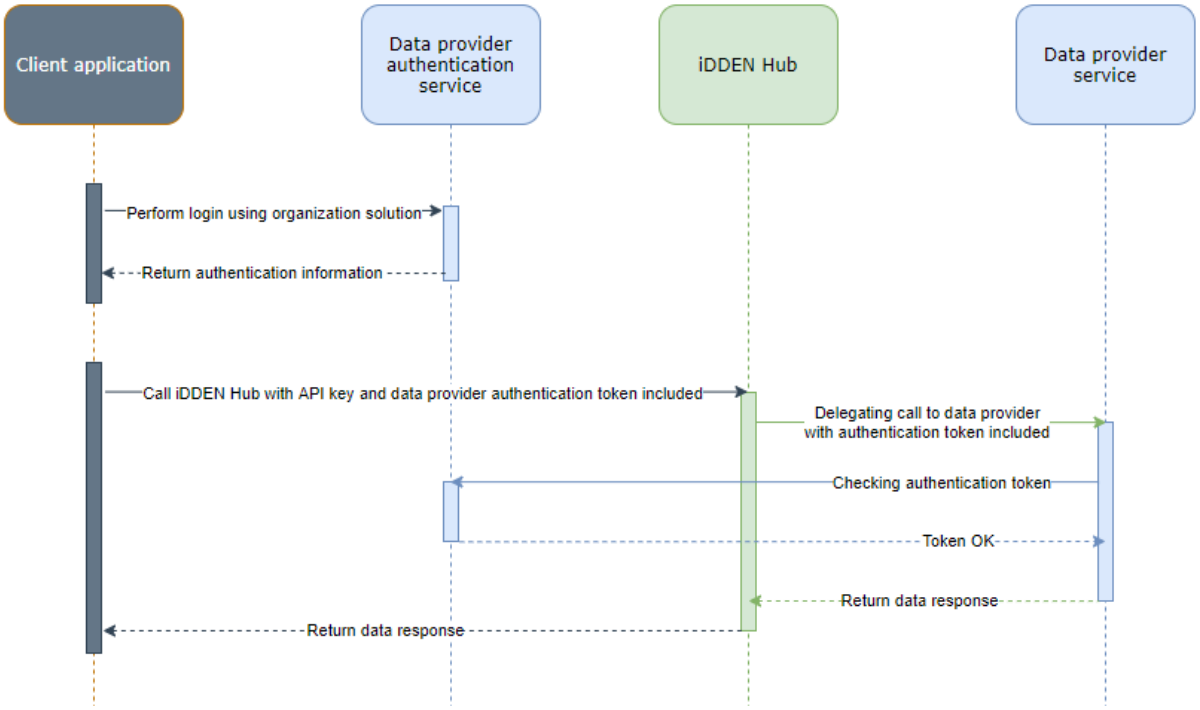
Introduction

The purpose of this document is to give an overview of how security is handled in the iDDEN interface.

The iDDEN –interface security solution follows these general principles:

- iDDEN does not perform any end user authentication/authorisation logic by itself but assumes that the authentication will be implemented by the backend system of each organisation.
- iDDEN assumes that the organisation authentication produces a single string-type authentication token which is included within the HTTP header in the iDDEN requests. This token can be of any form since iDDEN only passes it through and does not parse or process it in any way.

This basic principle is illustrated in the picture below.



(Picture 1. iDDEN authentication principle)

From the iDDEN interface point of view, this principle is very clear. However, from the FMS point of view the big question here is how to get a valid authentication token for use in the iDDEN data exchange calls. There are significant differences between different security solutions and the chances to get a unified security solution are very limited.

Basics for the client to obtain a security token

Standardised authentication web service

This is a recommendation to standardise the iDDEN user authentication so that each supported data delivering partner implements its own simple authentication REST action using a common standard. Doing so it will only be implemented once per Service and can be used multiple time by any client for different services:

- POST method type
- JSON content type so that the authentication information in the following format is accepted by the function:

```
{
    “userName”: “[specific farm identifier or iDDEN-ID]”,
    “password”: “[user password]”
}
```

- Upon successful authentication, the function returns an authentication token in the following JSON format:

```
{
    “authToken”: “[token value]”,
    “refreshToken”: “[token for refreshing]”,      // optional depending
    “expireTime”: [seconds until expiration]       // on availability
}
```

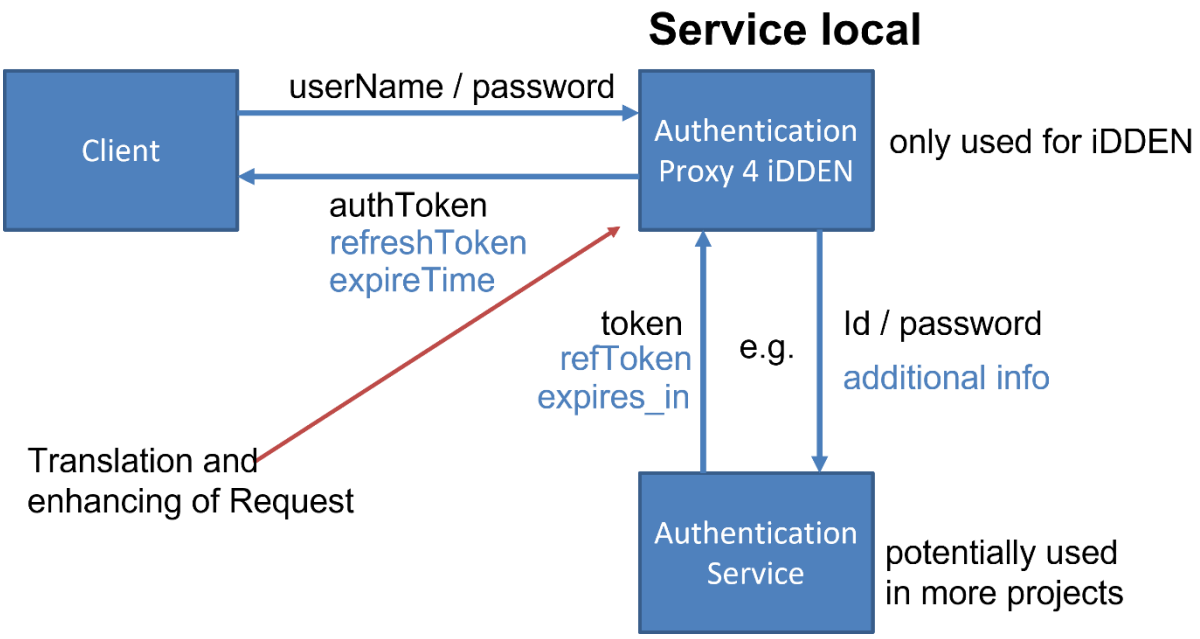
All authentication services, which deliver a refreshToken should offer a second service endpoint to refresh the authToken by sending the refreshToken to this Endpoint:

- POST method Type
- JSON content type so that the token will be accepted:

```
{
    “refreshToken”: “[token for refreshing]”
}
```

- The response will be the same as in the original login request (see above)

The easiest implementation for this is a proxy service, that forwards the authentication request to the national or companywide real authentication service. Here anyone has the chance to translate the properties and enhance the request as shown (optional properties in blue) below. The advantage of this is, that every authentication service looks the same from the client side.



The call to the refresh service should be handled in the same way. The recommended way would be to have the endpoint for refresh parallel to the original login service.

- https://{service-address}/{version}/{optional path}/login
- https://{service-address}/{version}/{optional path}/refresh

Using OpenID Connect for authentication

The second way to obtain a token for making calls through the iDDEN Hub is to use the web standard OpenId Connect. This is built around the OAuth2 standard and can be used in many different flows.

The main reasons to do so, is to avoid the storing of the user credentials. Therefore we recommend the following flow to be used in iDDEN environment:

- Authorization Code Flow with Proof Key for Code Exchange (PKCE)
- Authorization Code Flow

The first is preferred, as it is recommended by the standard for native and single page clients.

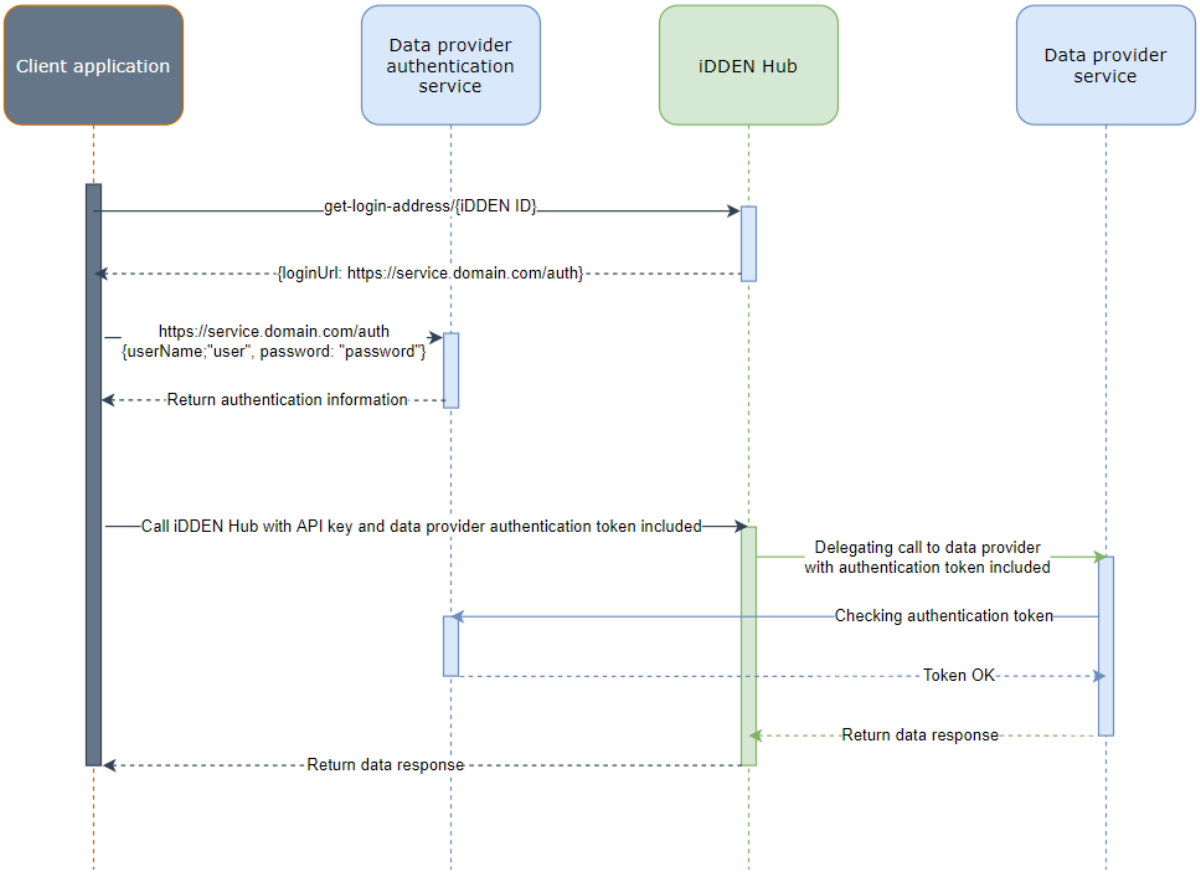
For reference check:

- https://auth0.com/docs/get-started/authentication-and-authorization-flow/authorization-code-flow-with-pkce
- https://openid.net/specs/openid-connect-basic-1_0.html#CodeFlow

Lookup service for authentication services

The iDDEN interface can also provide a single REST function named /get-login-address/{iDDEN ID} that returns the address of the authentication function of that country so that national service addresses can be maintained in the iDDEN.

The solution principle is illustrated in the sequence diagram below. Note that the authentication calls are made directly against the national authentication web service.



(Picture 2. iDDEN-standardized authentication web service)