PHR authorization

My Kanta Pages PHR

24.9.2019 Kela

Change history

Version	Change	Author	Date
0.82	First published draft version	Kela Kanta services	22.5.2017
0.83	Updated request parameters and removed openid scope	Kela Kanta services	27.2.2018
1.0	The first final version	Kela Kanta services	23.10.2018
1.1	Updated section 2, Client types	Kela Kanta services	21.12.2018
1.2	Updated state-parameter, added maintenance only -scopes	Kela Kanta services	5.4.2019
1.3	Added section 9, Error messages	Kela Kanta services	23.5.2019
1.4	Updated using of state-parameter to token endpoint.	Kela Kanta services	24.9.2019

Contents

1 Introduction

This document describes authorization profiles and flows for My Kanta Personal Health Record (Finnish Kanta PHR). The document is intended for implementers of applications that communicate with Finnish Kanta PHR.

All Kanta PHR related material for application implementers is published at https://www.kanta.fi/jarjestelmakehittajat/omakannan-omatietovaranto

The endpoint addresses in this document are exemplary and actual addresses will be published by Kela.

Should you have any comments on this document, please provide feedback via kanta@kanta.fi.

2 Client types

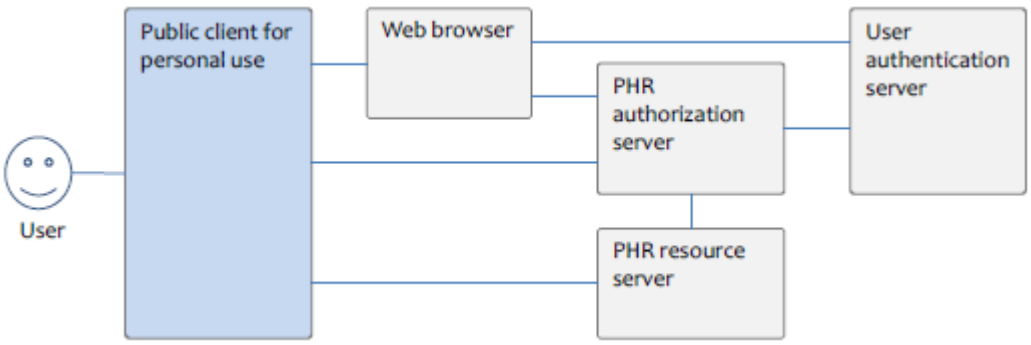
There are three types of PHR clients. Authorization protocols differ depending on the client type being used.

Note that native applications are not supported at the moment.

2.1 Personal use

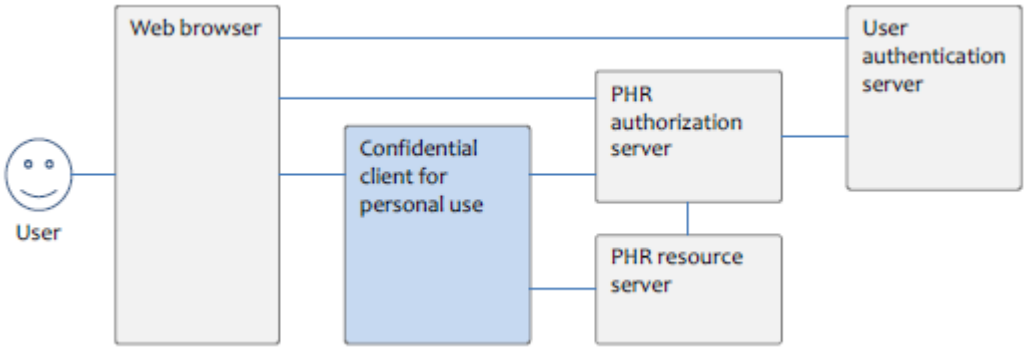
Public client for personal use. Native applications designed for mobile operating systems (e.g. Android, iOS, Windows Phone) that can store client credentials in a secure environment provided by the mobile device. Client credentials are instance-specific, which means that they are generated by the client separately for every installation of the application on every device.

Fig. 1: Data and control flows for public clients for personal use. Note that the vertical or horizontal order of the connectors is not necessarily the same as the order of phases in the authorization process.



Confidential client for personal use. Web applications that have server side business logic and that are capable of protecting application-specific client secrets. Such applications can also be uniquely identified through client authentication with a client certificate, using mutual TLS. Examples of confidential clients are web-based customer portals.

Fig. 2: Data and control flows for confidential clients for personal use. Note that the vertical or horizontal order of the connectors is not necessarily the same as the order of phases in the authorization process.



When the application requests a token and authorization, it should know which user they concern. Pseudonym is returned with the token and they should be matched by the application. Refresh token is user specific and must be stored safely.

The refresh token for server-based applications expires after one year if integration has not been used, in which case users must renew the application's permissions. Applications may not extend the validity of the refresh token without the user's permission.

2.2 Professional use

Client for professional use. All clients that are targeted for use by health and social care professionals are considered confidential clients. This does not mean that professionals cannot use native mobile apps or desktop clients to access PHR data. However, if that is the case, all communication with PHR must be done via their organization's authorization server acting as a backend for the apps, i.e. such implementations will not communicate directly with PHR. The organization's authorization server is also responsible for the authentication of users and for ensuring that they have necessary professional rights for the access to patient or customer data.

Fig. 4: Data and control flows for professional clients. Note that the vertical or horizontal order of the connectors is not necessarily the same as the order of phases in the authorization process.

3 Client registration

Before a client can use PHR, the client must be tested and certified. Kela provides testing requirements and coordinates testing. Certification is performed by Kela or an assigned third party operator according to the guidelines published by the National Institute for Health and Welfare (THL). A tested and certified client can be registered to PHR. Registration information is maintained by Kela and is used in the user-initiated authorization process.

At the sandbox environment developers can register clients with self-service registration.

3.1 Registering client for personal use

The minimum information needed for the registration of a public client for personal use is:

- Application name. The name must be the same as the one used for marketing purposes and for registering the client in the application stores of mobile OS providers. The name will be displayed to the user by the authorization server during the authorization process.
- Application id. A unique identifier, assigned to the client by the Kela in registration process. The identifier will be used by the client instance registration endpoint to identify the client software in order to register its instances.
- Application version. The version identifier of the registered application software
- Application description. A short description of purpose of the application.
- Redirect URI. The URI to which an authorization code is sent after successful authorization of the client by its user.
- Contact information. Contact details of the organization responsible for marketing and maintaining the client.
- Application logo. The logo must be the same as the one used for marketing purposes and for registering the client in the application stores of mobile OS providers. The logo will be displayed to the user by the authorization server during the authorization process.
- Scopes. The scopes define the rights that the client requires for its correct functioning. The scopes are set when the client is registered and cannot be extended or narrowed by the client or its user during the authorization process. The scopes are described in Section 7.
- Authentication method. What kind of user authentication service or method client uses for user authentication.

For confidential clients Backend certificate OID is needed. The certificate used by the client's backend for communication with the authorization server and the resource server. The certificate must be issued by the Population Register Centre of Finland (VRK). Certificates are not used in the sandbox environment.

Upon registration, Kela will provide the client developer a client secret if the client can’t use a client certificate (native mobile apps). The client secret is used in the client instance registration flow as described in Section 4. A reasonable protection of the client secret must be implemented (e.g. obfuscation) so that its extraction from the software distribution package or its installation is not trivial.

After the registration of the software, software id and initial access token are returned for the software developers to store securely and for using in the dynamic registration of client instances.

3.2 Registering a client for professional use

The minimum information needed for the registration of a client for professional use is:

- The organization oid of the social and healthcare service provider
- Contact information. Contact details of the organization responsible for marketing and maintaining the client.

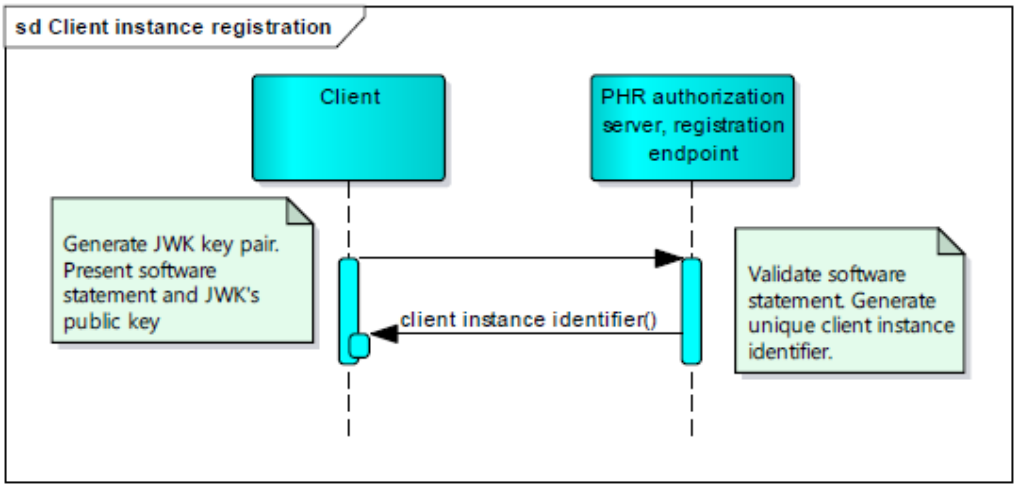
- Backend certificate OID. The certificate used by the client's backend for communication with the authorization server and the resource server. The certificate must be issued by the Population Register Centre of Finland (VRK). Certificates aren't used in the sandbox environment.

- Scopes. The scopes define the rights that the client requires for its correct functioning. The scopes are set when the client is registered and cannot be extended or narrowed by the client or its user during the authorization process. The scopes are described in Section 7.

4 Client instance registration for public clients for personal use

Prior to the use of the authorization endpoint, instances (each installation) of public clients for personal use must be registered according to OAuth 2.0 Dynamic Client Registration Protocol [RFC7591]. An initial access token is provided to the client developer by Kela after successful certification of the client software and after its registration with the PHR service. The initial access token is used as part of the software statement for registering instances of the registered client software.

Fig. 5 Client instance registration



A client instance starts the registration flow starts by generating a JSON web key (JWK) set [RFC7517] on the device. The set is composed of two EC keys, private and public. The private key MUST be stored in the secure memory of the device. Keys MUST NOT be shared among instances of client software.

The following key types and other parameters are supported in the description of the public key that will be required in the next step of the registration flow:

Name	Meaning	Cardinality	Value
`kty`	Key type Elliptic Curve	required	`"EC"`
`use`	Public key use	required	`"sig"`
`kid`	Key ID	required	`Random and unique uuid`
`crv`	Cryptographic	required	`"P-256"`

	curve used with the key.		
x	x coordinate for the Elliptic Curve point	required	Base64url encoding of the octet string representation of the coordinate
y	y coordinate for the Elliptic Curve point	required	Base64url encoding of the octet string representation of the coordinate
alg	Algorithm used	required	"ES256"

Next, the client instance generates a software statement that includes the following fields and values:

Name	Meaning	Cardinality	Value
client_name	Application name	required	Application name that MUST match the one provided in the client registration.
software_id	Application ID	required	Application ID that MUST match the one provided in the client registration.
device_id	Id for the device software is installed on	required	Unique identifier for the installation instance of the software, such that the users can separate between multiple installations on devices
jwks	JSON web key set	required	JSON web key set containing the public key generated by the client instance.
initial_access_token	Initial access token	required	The initial access token that MUST match the one provided by Kela to the software vendor upon client registration.
scope	Scopes	required	The scopes needed by the client instance. All of the scopes the instance need access to must be listed here and all of them must be registered for the software.
grant_types	Token flows used	required	Possible options are "authorization code" for the auth code flow and "refresh_token" if the application can store the refresh token securely and use it to refresh the tokens.

The following is an example of the software statement:


```
{
  "client_name": "The Example Software client",
  "software_id": "ExampleId123",
  "device_id" : "Example Phone 1",
  "jwks": {
    "keys": [
      {
        "kty": "EC",
        "use": "sig",
        "kid": "6a8747e1-cd1f-4a12-b417-c151ebdae68c",
        "crv": "P-256",
        "x": "4Q_39cHeN6_s7du40b2FkiTGfEFKiOj3x7YydXDMf9A",
        "y": "He4HSzRyMfap08gaL0YbQLpHKG7cNJD3JvQ64TMrpGY"
      }
    ]
  },
  "initial_access_token":
"JncmFudF9ds0eXBlcyI6WyJhdXRob3JsdpemF0aW9uX2NvZGUiLsdCJyZWZyZgfXNoX3Rvad2V
uIldfg0sImFwcGdxpY2F0aW9uX",
  "scope": "offline_access patient/Observation.read
patient/Observation.write",
  "grant_types": [
    "authorization_code",
    "refresh_token"
   ],
  "token_endpoint_auth_method": "private_key_jwt",
  "request_object_signing_alg": "ES256",
  "token_endpoint_auth_signing_alg": "ES256",
  "redirect_uris": [
    "http://localhost/application/example"
  ],
  "application_type": "native",
  "jti":
"WhTTEdYQ1VyRTBNdWI1bFFaeTBfdzh0cTJxQ0JOY3hFeFAwMlhQeUs5SWZ4a2JuSFZ1VlBtRXl
KbjdYdHpYcVpWQ21neTBrSl9RVHhrTm9kQVE3M1dwdUpRZm84eWY4cVZmc1ZTWEp6MEpxdnplSW
dlUlBtTVFNTFhOVWg2cFphcVR4Z2QwUjZMOHlxS0pSeTcxV1c1dUp5dGYzRjlZMGJXWHlYbXhic
kF4ZjN1Um1kb0N4NGhfbzd4OG1"
}
```

The software statement is asymmetrically signed by the client instance using its private key according to specification JSON Web Signature (JWS) [RFC7517]. Then it is submitted to the registration endpoint of the authorization server. The following is an example of the registration request, signed with the private key using EC DSA with SHA-256 algorithm:

```
POST /register HTTP/1.1
Content-Type: application/json
Accept: application/json
Host: phrauth.kanta.fi

{
  "software_statement":
"eyJhbGciOiJFUzI1NiJ9.eyJzb2Z0d2FyZV9pZCI6IkV4YW1wbGVJZDEyMyIsImluaXRpYWxfY
WNjZXNzX3Rva2VuIjoiSm5jbUZ1ZEY5ZHMwZVhCbGN5STZXeUpoZFhSb2IzSnNkcGVtRjBhVzl1
WDJOdlpHVWlMc2RDSnlaV1p5WmdmWE5vWDNSdmFkMlZ1SWxkZmcwc0ltRndjR2R4cFkyRjBhVzl
1WCIsInRva2VuX2VuZHBvaW50X2F1dGhfc2lnbmluZ19hbGciOiJFUzI1NiIsImp3a3MiOnsia2
V5cyI6W3sia3R5IjoiRUMiLCJ1c2UiOiJzaWciLCJjcnYiOiJQLTI1NiIsImtpZCI6IjZhODc0N
2UxLWNkMWYtNGExMi1iNDE3LWMxNTFlYmRhZTY4YyIsIngiOiJCSXVQVUJRS0VQdmE1cV9XNEs2
NklGdXc0cWViVDZTYllDME5yY0FWZVpBIiwieSI6ImYwVkhkU1g5TmRjTVBLVEpQVEVRYVloNUF
VLWNpTUk3UlgwVTQ3R25LVE0ifV19LCJncmFudF90eXBlcyI6WyJhdXRob3JpemF0aW9uX2NvZG
UiLCJyZWZyZXNoX3Rva2VuIl0sImFwcGxpY2F0aW9uX3R5cGUiOiJuYXRpdmUiLCJzY29wZSI6I
m9mZmxpbmVfYWNjZXNzIHBhdGllbnRcL09ic2VydmF0aW9uLnJlYWQgcGF0aWVudFwvT2JzZXJ2
YXRpb24ud3JpdGUiLCJyZXF1ZXN0X29iamVjdF9zaWduaW5nX2FsZyI6IkVTMjU2IiwicmVkaXJ
lY3RfdXJpcyI6WyJodHRwOlwvXC9sb2NhbGhvc3RcL2FwcGxpY2F0aW9uXC9leGFtcGxlIl0sIm
NsaWVudF9uYW1lIjoiVGhlIEV4YW1wbGUgU29mdHdhcmUgY2xpZW50IiwidG9rZW5fZW5kcG9pb
nRfYXV0aF9tZXRob2QiOiJwcml2YXRlX2tleV9qd3QiLCJqdGkiOiJXaFRURWRZUTFWeVJUQk5k
V0kxYkZGYWVUQmZkemgwY1RKeFEwSk9ZM2hGZUZBd01saFFlVXM1U1daNGEySnVTRloxVmxCdFJ
YbEtiamRZZEhwWWNWcFdRMjFuZVRCclNsOVJWSGhyVG05a1FWRTNNMWR3ZFVwUlptODRlV1k0Y1
ZabWMxWlRXRXA2TUVweGRucGxTV2RsVWxCdFRWRk5URmhPVldnMmNGcGhjVlI0WjJRd1VqWk1PS
Gx4UzBwU2VUY3hWMWMxZFVwNWRHWXpSamxaTUdKWFdIbFliWGhpY2tGNFpqTjFVbTFrYjBONE5H
aGZiemQ0T0cxIn0._SfWoOo_blX7hLYD6E1cTbM4z9hx8O_o4TTyl1_UsFGSVY9V0hTFfiYlZfQ
a2mJA-R9I3KIiA49iF7XI1kX7EQ”
}
```

Upon successful validation of the registration request the authorization server will assign a `client_id` to the client instance. The following is an example of the registration response:

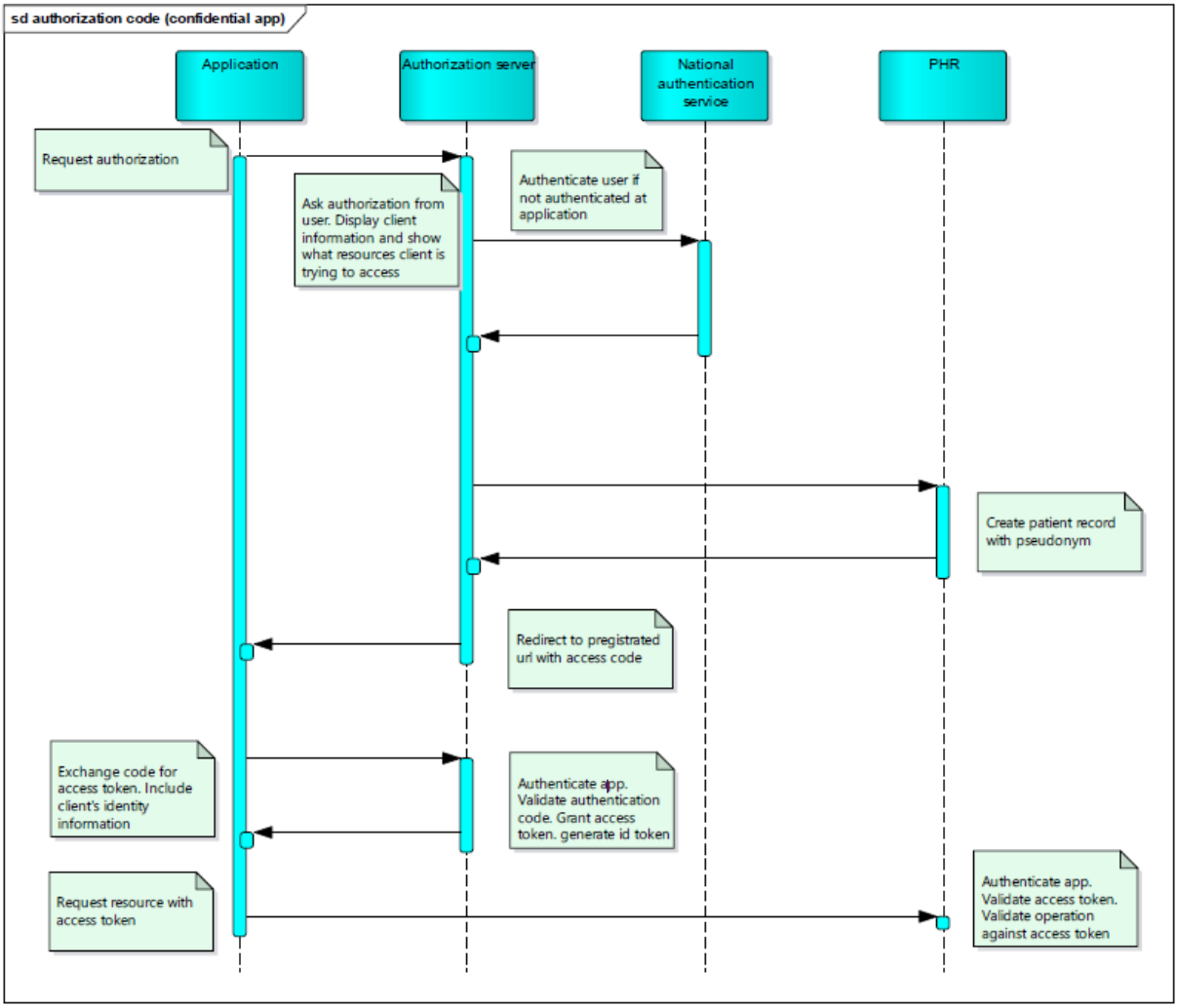
```
      HTTP/1.1 201 Created
      Content-Type: application/json
      Cache-Control: no-store
      Pragma: no-cache

      {
  "client_id": "b3886ff9-374b-4bd4-bbb3-0a6bb8d8108d",
  "client_id_issued_at": 1495115628,
  "registration_access_token":
"eyJraWQiOiJyc2ExIiwiYWxnIjoiUlMyNTYifQ.eyJhdWQiOiJiMzg4NmZmOS0zNzRiLTRiZDQ
tYmJiMy0wYTZiYjhkODEwOGQiLCJpc3MiOiJodHRwczpcL1wvcGhyLWF1dGguaWcua2FudGEuZm
lcL3Boci1hdXRoc2VydmVyLXNhbmRib3hcLyIsImlhdCI6MTQ5NTExNTYyOCwianRpIjoiYmM3O
DFlOTQtNWYyZi00YTE3LWI4M2YtMjQ4YzMxYjIzNDE4In0.F58ZAHwLiB_KnoiUYrCaT8M5SwX1
p5XUyIaYLjwcBVSGx7JS49NmbksSzpt4QA4Nn3WUAdbLA7e34MMhwlzyMu2OD7EpEgPqsDOQwcP
dfQaW w_96HjBS4Sfmp-RqzdICo-hxn_ORuf2cm-k_hftAqxujyqGvk3N-sFdKyqiU74",
  "registration_client_uri": "https://phrauth.kanta.fi/phr-
authserver/register/b3886ff9-374b-4bd4-bbb3-0a6bb8d8108d",
  "redirect_uris": ["http://localhost/application/example"],
  "client_name": "The Example Software client",
  "token_endpoint_auth_method": "private_key_jwt",
  "scope": "patient/Observation.read offline_access
patient/Observation.write",
  "grant_types": [
    "refresh_token",
    "authorization_code"
  ],
  "response_types": ["code"],
  "jwks": {"keys": [ {
    "kty": "EC",
    "crv": "P-256",
    "kid": "6a8747e1-cd1f-4a12-b417-c151ebdae68c",
    "x": "BIuPUBQKEPva5q_W4K66IFuw4qebT6SbYC0NrcAVeZA",
    "y": "f0VHdSX9NdcMPKTJPTEQaYh5AU-ciMI7RX0U47GnKTM"
  }]},
  "application_type": "native",
  "request_object_signing_alg": "ES256",
  "token_endpoint_auth_signing_alg": "ES256",
  "software_statement":
"eyJhbGciOiJFUzI1NiJ9.eyJpbml0aWFsX2FjY2Vzc190b2tlbiI6InRpbWkiLCJ0b2tlbl9lb
mRwb2ludF9hdXRoX3NpZ25pbmdfYWxnIjoiRVMyNTYiLCJqd2tzIjp7ImtleXMiOlt7Imt0eSI6
IkVDIiwiZCI6IjZjWEprYVE0VENxU2lQSUdPeVpGNG9CRmRfQTNCYXpJRzhsblU1UlVrOVUiLCJ
jcnYiOiJQLTI1NiIsImtpZCI6IjZhODc0N2UxLWNkMWYtNGExMi1iNDE3LWMxNTFlYmRhZTY4Yy
IsIngiOiJCSXVQVUJRS0VQdmE1cV9XNEs2NklGdXc0cWViVDZTYllDME5yY0FWZVpBIiwieSI6I
mYwVkhkU1g5TmRjTVBLVEpQVEVRYVloNUFVLWNpTUk3UlgwVTQ3R25LVE0ifV19LCJncmFudF90
eXBlcyI6WyJhdXRob3JpemF0aW9uX2NvZGUiLCJyZWZyZXNoX3Rva2VuIl0sImFwcGxpY2F0aW9
uX3R5cGUiOiJuYXRpdmUiLCJpc3MiOiJodHRwOlwvXC9hcnRlbWVzaWEubG9jYWwiLCJyZWRpcm
VjdF91cmlzIjpbImh0dHA6XC9cL2tlbGEuZmkiXSwidG9rZW5fZW5kcG9pbnRfYXV0aF9tZXRob
2QiOiJwcml2YXRlX2tleV9qd3QiLCJzb2Z0d2FyZV9pZCI6IlRpbWluIHNvZnR3YXJlIiwic2Nv
cGUiOiJvcGVuaWQgb2ZmbGluZV9hY2Nlc3MgcGF0aWVudFwvT2JzZXJ2YXRpb24ucmVhZCBwYXR
pZW50XC9PYnNlcnZhdGlvbi53cml0ZSIsInJlcXVlc3Rfb2JqZWN0X3NpZ25pbmdfYWxnIjoiRV
MyNTYiLCJjbGllbnRfbmFtZSI6IlRpbWluIHBhcmFzIGFwcHNpIiwianRpIjoiZjU2YjgyMWQtY
jY0Ni00YmIyLWJkOTUtNGQwZTM5ZDFkOWY5In0.fB_Do19i0klTdn5vIsl_HsPAVjevPFEGLrg1
k_i0sMt1daL-XW3D9F76OrzSZ1W8Rqx-u7OwPe5820idH-1a-w”
}
```

The authorization server will associate the issued `client_id` with the public key of the client instance. The client instance will be authenticated at the authorization endpoint and token endpoint using JWT tokens as per [RFC7523]. The tokens MUST be signed by the client instance using its private key and the algorithm named in the dynamic registration.

5 Authorization code flow

Authorization code flow is used by public clients for personal use and by confidential clients for personal use. The flow is shown in Fig. 3. The client retrieves a short-lived authorization code from the authorization endpoint of the authorization server, in order to trade it later for a set of tokens at the token endpoint. The authorization endpoint is called when the client needs authorization from the user to access resources. This may be the first time the client is used or if the client has not been granted a scope that it needs to access a resource. Authorization and token endpoints are described in [RFC6749]. Authorization and token urls may differ from each other.

Fig. 6: Authorization code flow



5.1 Authorization endpoint

The authorization endpoint is used by the client in order to retrieve a short-lived authorization code which will be traded later for a set of tokens at the token endpoint. Before issuing the code, the authorization server will authenticate the user, request the user to confirm the access to their PHR account, and offer the user to act on someone's behalf using the rights delegation service.

Due to reasons described in [DIONA], public clients for personal use running on mobile platforms MUST use the system browser of the said platform or another external user-agent for calls to the authorization endpoint. Web-views (embedded user-agents) MUST NOT be used. If the platform supports in-app browser tabs, their use is RECOMMENDED for usability reasons.

The user-agent MUST be requested to open the authorization endpoint's URL with the following parameters:

Name	Cardinality	Value
`response_type`	required	"code"
`client_id`	required	The identifier of the client instance. For confidential clients, the id is provided in the client registration process, and for public clients the id is generated through dynamic registration of instances at the registration endpoint.
`redirect_uri`	required	Must match the URI registered for the client software at the registration time. Guidelines for redirect URI naming are provided in [DIONA].
`scope`	optional	The scopes that the client requests to be granted, separated using the + character. Scopes are defined in Section 7. If the value is skipped or empty, the authorization server will assume that the client is requesting all scopes registered for it.
`state`	required	The client MUST generate an unpredictable state parameter with at least 128 bits of entropy for each user session. The authorization server will include the state value when redirecting the user-agent to the redirect URI. The client MUST validate the state value for any request sent to its redirect URI.

All parameters MUST be "application/x-www-form-urlencoded" formatted as defined in the Appendix B of [RFC6749]. The following is an example of the request sent to the authorization endpoint:

```
Location:
https://phrauth.kanta.fi/authorize?response_type=code&client_id=8d415 da7-
bec9-44a3-8979-105ea5bf8ee4&redirect_uri=fi.sw-vendor.app%3A%2Fafter-auth&
scope=patient%2FObservation.read+patient%2FObservation.write&state=adfh56ki
wshti2k4
```

After verifying the parameters of the call, the authorization server will redirect the user-agent to Suomi.fi e-Identification [SUOMI.FI], the Finnish national citizen authentication service. If the application uses the same authentication service that will show the citizen notification about authenticating to Kanta PHR. After successful authentication of the user, the authorization server shows the scopes that are requested by the client. The user is asked to confirm the scopes.

If the request fails, the client identifier is not valid or the access request is denied, the authorization server must not redirect the user-agent to the invalid redirection URI, but inform the resource owner or the client instead, depending on the case (see RFC6749 4.1.2.1. for details).

The authorization server will assign each user a pseudonym to be used with the PHR service. Pseudonyms are random UUID identifiers that are directly associated with the Finnish national identification numbers of the same persons. Clients will never receive original national identification numbers from the PHR service. User’s pseudonym remains the same, for example when the application is authorized again.

After generating any required pseudonyms, the authorization server will redirect the user-agent to the redirect URI (client's redirection endpoint) provided in the authorization request. The following parameters are supplied with the redirect call:

Name	Cardinality	Value
`code`	required	The short-lived authorization code generated by the authorization server.
`state`	required	The value of the state parameter exactly as supplied by the client in the authorization request. The client MUST validate the state value for any request sent to its redirect URL and check whether it matches a submitted authorization request.

The following is an example of the redirect call:

```
Location: fi.sw-vendor.app:/after-
auth?code=ahui560zxs12n3dq&state=adfh56kiwshti2k
```

The authorization code is valid for 5 (five) minutes.

5.2 Token endpoint

After receiving an authorization code through the redirect call performed by the authorization server in the previous step, the client accesses the token endpoint in order to receive an access token and a refresh token. The client presents the authorization code along with its own credentials to the authorization server's token endpoint to obtain the said set of tokens. When an access token is expired, the client can request a new access token by presenting a valid refresh token.

5.2.1 Client authentication at the token endpoint

Confidential clients are authenticated with mutual TLS using client certificates.

Public clients for personal use are authenticated at the token endpoint using a JWT Bearer Token as per Section 2.2 of [RFC7523], following profiles [ARGONAUT] and [CORECONNECT]. The authentication JWT is self-issued by the client instance and signed using its private key generated during the client instance registration process (Section 4). The signature format follows JSON Web Signature (JWS) [RFC7517]. The authentication JWT SHALL contain the following claims.

Name	Meaning	Cardinality	Value
`iss`	Issuer	required	client_id of the client instance
`sub`	Subject	required	client_id of the client instance
`aud`	Audience	required	The URL of the authorization server's token endpoint (the same URL to which this authentication JWT will be posted)
`exp`	Expiration time	required	The time on or after which the authentication JWT MUST NOT be accepted for processing. The time MUST be expressed in seconds since the "Epoch" (1970-01-01T00:00:00Z UTC). This time MUST be no more than 5 (five) minutes in the future.
`jti`	JWT ID	required	A unique identifier (nonce) of this authentication JWT. MUST have at least 128 bits of entropy and MUST NOT be reused in

			another token. The authorization server SHALL check for reuse of jti values and SHALL reject all tokens issued with duplicate jti values.
kid	Key id	required	Key id of the key pair used to digitally sign this token. MUST match the value supplied with the software statement used during the client instance registration process.
iat	Issued at	required	The time on which the authentication JWT was generated.

Example:

```
{
"iss": "8d415da7-bec9-44a3-8979-105ea5bf8ee4",
"sub": "8d415da7-bec9-44a3-8979-105ea5bf8ee4",
"aud": "https://phrauth.kanta.fi/token",
"jti": "a9sk105fpwqn2n20",
"iat": 1505996055,
"exp": 1505996355,
"kid": "8bdd589e-ba42-4d6e-aea6-0d3ba43f5ed7"
}
```

5.2.2 Trading the authorization code for a set of tokens

The client trades the code for an access token, a refresh token and an ID token via a POST call to the PHR authorization server's token endpoint URL.

The following parameters are supplied with the call:

Name	Cardinality	Value
client_id	required	The id of the client
grant_type	required	"authorization_code"
code	required	The short-lived authorization code received from the authorization server.
redirect_uri	required	MUST match the URI used in the authorization request.
client_assertion_type	conditional	Required for public clients, with fixed value "urn:ietf:params:oauth:client-assertion-type:jwt-bearer". Omit for confidential clients.
client_assertion	conditional	Required for public clients. The value is a signed authentication JWT as described in Section 5.2.1. Omit for confidential clients.
state	optional	An opaque value used by the client to maintain state between the request and callback. If state parameter is provided with the request, PHR authorization server will return the exact value to the client.

All parameters MUST be "application/x-www-form-urlencoded" formatted as defined in the Appendix B of [RFC6749]. The following is an example of the token request (with line wraps within values for display purposes only):

```
POST https://phrauth-token.kanta.fi/phr-authserver/token HTTP/1.1 Content-
Type: application/x-www-form-urlencoded
grant_type=authorization_code
&code=4lKCd5
&state=adfh56kiwshti2k4
&redirect_uri=https%3A%2F%2Flocalhost
&client_id=4393ab31-7753-472b-af74-dcb8b7b64c93
&client_assertion_type=urn%3Aietf%3Aparams%3Aoauth%3Aclient-assertion-
type%3Ajwt-bearer
&client_assertion=eyJhbGciOiJFUzI1NiJ9.eyJzdWIiOiI0MzkzYWIzMS03NzUzLTQ3MmIt
YWY3NC1kY2I4YjdiNjRjOTMiLCJhdWQiOiJodHRwczpcL1wvcGhyLWF1dGguaWcua2FudGEuZml
cL3Boci1hdXRoc2VydmVyLXNhbmRib3hcLyIsImtpZCI6IjZhODc0N2UxLWNkMWYtNGExMi1iND
E3LWMxNTFlYmRhZTY4YyIsImlzcyI6IjQzOTNhYjMxLTc3NTMtNDcyYi1hZjc0LWRjYjhiN2I2N
GM5MyIsImV4cCI6MTQ5NTExNjY3MywiaWF0IjoxNDk1MTE2MzczLCJqdGkiOiJlMmYwYTkxNS02
ZjhiLTQ4NzYtYmU5Ny03NjYwNzU2ZWU1YmYifQ.9UL58yw4mxrQUGwbCdZjqDwsYWEkNY__EWFt
U0vuJTrewNxnDT36fajlx6aXlXZQm7zJjk7497XTVSNdR9fUog
```

The PHR authorization server will return a JSON structure that includes an access token or a message indicating that the authorization request has been denied.

The JSON structure includes the following parameters:

Name	Cardinality	Value
access_token	required	The access token issued by the authorization server
token_type	required	Fixed value: Bearer
expires_in	required	Lifetime in seconds of the access token, after which the token SHALL NOT be accepted by the resource server
scope	required	Scope of access authorized.
state	required	The exact value received from the client in the token request.
refresh_token	required	Token that can be used to obtain a new access token
sub	required	Application user's PHR pseudonym
id_token	optional	Authenticated patient pseudonym identity, added if openid scope exist
principal	optional	Child's PHR pseudonym (NYI)

```
{
"access_token":"eyJhbGciOiJSU...8h0eQ",
"token_type":"Bearer",
"expires_in":3599,
"scope":"patient/Observation.read+patient/Observation.write+openid",
"sub": "44a12254-b28d-42f9-8bec-4a468473ef9f",
"refresh_token":"eyJhbGciOiJSUzI1NiI...ZGFicmE=",
"id_token":"eyJhGciOiJSUzI1NiI...ESL0eIX7eg1_DA"
```

```
"state":"adfh56kiwshti2k4"
}
```

The access token is used in all calls to the resource server to obtain resources. The token parameter shall be sent as the Bearer http-header to the FHIR resource server as defined in section 2.1 in RFC6750.

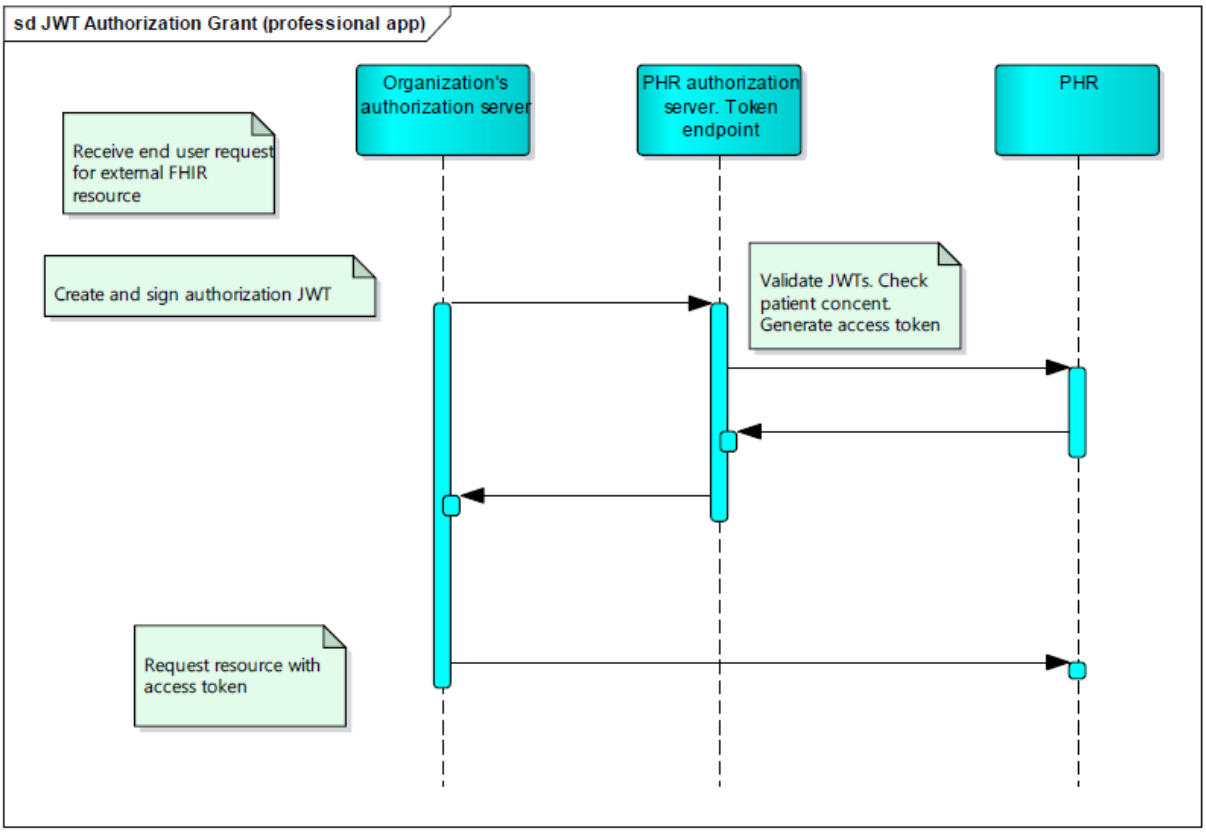
In addition to the access token native app for personal use need to use similar client assertion JWT token to authenticate with the resource server. It should be sent in the header *PhrJWTAuthentication*.

6 Client for professional use

This profile is used when the application user is healthcare professional. When this profile is used the user isn't authenticated with Suomi.fi authentication service, the responsibility for authenticating the application user is in the client or in the EHR if the client is integrated to EHR system.

Also in this profile the application user doesn't give consent to the client to use PHR information because the information that is handled isn't user's own information. Access to the information is granted by the resource owner (citizen) with separate consent.

This profile is somewhat similar to Argonaut project's draft "Cross-Organization Data Access Profile". JWT used for authorization grant is similar, but the organization's authorization server is authenticated with mutual TLS instead of authentication JWT which is used in the Argonaut's model.



6.1 Token request

The authorization request is a JWT, as defined in RFC7519 and contains the details the PHR authorization server will need to know in order to mediate the request for access to a FHIR resource. The HTTP parameters for transporting the authorization JWT from the organization's authorization

server to the PHR authorization server's token endpoint is as defined in The OAuth Assertion Framework RFC7521, with the following specific parameter values and encodings.

The value of the "grant_type" MUST BE "urn:fi:kela:kanta:phr:oauth:grant-type:finnish-phr-jwt".

The value of the "assertion" parameter MUST contain a single authorization JWT.

Example

POST https://phrauth.kanta.fi/phr-authserver/token HTTP/1.1

```
grant_type=urn%3Afi%3Akela%3Akanta%3Aphr%3Aoauth%3Agrant-type%3Afinnish-
phr-
jwt&redirect_uri=https%3A%2F%2Flocalhost&client_id=ProfessionalClientExampl
e&assertion=eyJhbGciOiJub25lIiwidHlwIjoiSldUIn0.eyJpc3MiOiJQcm9mZXNzaW9uYWx
DbGllbnRFeGFtcGxlIiwic3ViIjoidGVyaGktMTIzIiwiYXVkIjoiaHR0cHM6Ly9waHJhdXRoLm
thbnRhLmZpL3Boci1hdXRoc2VydmVyLyIsInJlcXVlc3RlZF9yZWNvcmQiOiIyNDA0OTctOTA3M
CIsInJlcXVlc3RlZF9zY29wZXMiOiJwYXRpZW50L09ic2VydmF0aW9uLnJlYWQrcGF0aWVudC9P
YnNlcnZhdGlvbi53cml0ZSIsInJlcXVlc3RpbmdfcHJhY3RpdGlvbmVyIjp7InJlc291cmNlVHl
wZSI6IlByYWN0aXRpb25lciIsImlkZW50aWZpZXIiOlt7InN5c3RlbSI6IjEuMi4yNDYuNTM3Lj
I2IiwidmFsdWUiOiIwMDAwMTIzMTIzMTIzIn1dLCJuYW1lIjp7InRleHQiOiJUZXN0UHJhY3Rpd
GlvbmVyIn0sInByYWN0aXRpb25lclJvbGUiOlt7Im1hbmFnaW5nT3JnYW5pemF0aW9uIjp7InJl
c291cmNlVHlwZSI6Ik9yZ2FuaXphdGlvbiIsImlkZW50aWZpZXIiOlt7InN5c3RlbSI6InVybjp
vaWQ6MS4yLjI0Ni41MzcuNi4yMDIuMjAwOCIsInZhbHVlIjoiMS4yLjI0Ni41MzcuMTIzNTY3OD
kwIn1dLCJuYW1lIjoiSGVhbHRoY2FyZVRlc3RPcmdhbml6YXRpb24ifX1dfSwicmVhc29uX2Zvc
l9yZXF1ZXN0IjoiMS4yLjI0Ni4xMjM1Njc4OTAiLCJleHAiOjE0OTUxMTcyOTcsImtpZCI6Ijc4
OTAiLCJqdGkiOiJzb21lLXVuaXF1ZS1ub25jZS1hYmMiLCJpYXQiOjE0OTUxMTcyNDd9.
```

PHR authorization checks care relationship according to encounter oid and that the user has given consent.

The authorization JWT contains claims relating to the resource being requested (e.g., FHIR patient resource, data scope, requesting practitioner, reason) and claims necessary to help ensure the security of the exchange (expiration time, issuer, subject, a token identifier; see RFC7523 for details).

Following table presents parameters used in authorization JWT:

Claim	Priority	Argonaut	PHR
iss	required	Requesting EHR's issuer URI.	client_id
sub	required	EHR-A's id for the user on whose behalf this request is being made. Matches requesting_practitioner.id	Practitioner's identification, SSN or Terhikki-/Suosikki or valid identification number of VRK identification card
acr	not used	Level of assurance of the requesting user's identity (e.g. NIST level 0-4, as defined in NIST SP 800-63-2)	Not used
aud	required	EHR-B authorization server's token_URL (the URL to which this authorization JWT will be posted)	PHR-Authorization server URL
requested_record	required	The FHIR patient resource being requested	Patient's social security number

requested_scopes	optional	Patient data being requested	Scopes requested
requesting_ organization	required		The organization oid of the social and healthcare service provider making request
requesting_ practitioner	required	FHIR practitioner resource making the request	Practitioner's identification, SSN or Terhikki-/Suosikki or valid identification number of VRK identification card
reason_for_request	required	Purpose for which access is being requested	Encounter OID that is used to check the care relationship with the patient
application_ version	required		Version identifier of the application
application_ name	required		Name of the application
special_reason	conditional		If the encounter is created by the same professional that is making the request to PHR the reason for doing so
exp	required	Expiration time integer after which this authorization JWT MUST be considered invalid; expressed in seconds since the "Epoch" (1970-01-01T00:00:00Z UTC). This time MUST be no more than five minutes in the future.	Expiration time of the token, this time MUST be no more than five minutes in the future.
kid	required	Key id of the encryption key used to digitally sign this token	OID of the SSL certificate used to authenticate the client that sent the JWT
jti	required	A nonce string value that uniquely identifies this authorization JWT. MUST have at least 128 bits of entropy and MUST NOT be reused in another token. EHR-B MUST check for reuse of jti values and MUST reject all tokens issued with duplicate jti values.	Unique nonce

`iat`	required	The UTC time the JWT was created by EHR-A.	Time of JWT creation, this time MUST be no more than five minutes in the past.

Example of the JWT

```
{
  "iss": "ProfessionalClientExample",
  "sub": "terhi-123",
  "aud": "https://phrauth.kanta.fi/phr-authserver/",
  "requested_record": "240497-9070",
  "requested_scopes": "patient/Observation.read+patient/Observation.write",
  "requesting_practitioner": "0000123123123",
  "requesting_organization": "1.2.246.123123123",
  "application_version": "1.0",
  "application_name": "Phr Professional Client for Company",
  "reason_for_request": "1.2.246.123567890",
  "exp": 1495117297,
  "kid": "7890",
  "jti": "some-unique-nonce-abc",
  "iat": 1495117247
}
```

6.2 Access token response

The PHR authorization server returns either a JSON structure that includes an access token, as defined in RFC6749 and RFC6750, or a message indicating that the authorization request has been denied.

The JSON structure includes the following parameters:

Field	Explanation
`access_token`	The access token issued by the PHR authorization server
`sub`	Registration number (Terhikki/Suosikki/SSN)
`token_type`	Fixed value: Bearer
`expires_in`	Lifetime in seconds of the access token, after which the token isn't accepted by the resource server. Lifetime of the token in PHR is one hour.
`scope`	Scope of access authorized
`patient`	Patient's PHR pseudonym

7 Supported scopes in Finnish Kanta PHR

Scopes supported by the Finnish Kanta PHR can be divided into the scopes that grant access to specific FHIR resources stored on the resource server and to scopes that allow applications to obtain other information and keep the authorization active.

7.1 User scopes for data access

Scopes than can be granted to access resources on the resource server are defined similarly to SMART on FHIR scopes.

To read a resource you need to have the patient/Resource.read –scope. For writing, updating and deleting the resource patient/Resource.write scope is needed. A scope is needed only for the main resource type, contained resources that are inline in the resource to be read or written follow the scope of the resource that they are part of. Referenced resources are subject of the scope of their respective type.

All requested scopes that can be authorized by user to the application, need to be registered for the application at the registration time. Only registered scopes are allowed to request authorization for. All scopes that are included in the access token need to be authorized by the user.

Scope	Contents
patient/Observation.read	Reading patient observations, like heart rate
patient/Observation.write	Creating, updating and deleting observation
patient/MedicationStatement.read	Reading the medication statement-resource
patient/MedicationStatement.write	Creating, updating and deleting medication statement
patient/MedicationAdministration.read	Reading the medication administration-resource
patient/MedicationAdministration.write	Creating, updating and deleting medication administration
patient/QuestionnaireResponse.read	Reading the questionnaire response-resource
patient/QuestionnaireResponse.write	Creating, updating and deleting questionnaire response
patient/CarePlan.read	Reading the care plan-resource
patient/CarePlan.write	Creating, updating and deleting care plan
patient/Consent.write	Creating and updating consent
patient/Consent.read	Reading the consent resource

7.2 Non user-specific scopes

Scope	Contents
CapabilityStatement.read	Reading the capability statement
StructureDefinition.read	Reading different structure definitions
StructureDefinition.write	Creating, updating and deleting structure definitions
ValueSet.read	Reading different value sets
ValueSet.write	Creating, updating and deleting different value sets
CodeSystem.read	Reading different code systems
CodeSystem.write	Creating, updating and deleting different code systems

Questionnaire.read	Reading the questionnaire-resource
Questionnaire.write	Creating, updating and deleting questionnaire

These scopes are reserved for internal PHR use only.

7.3 Technical scopes

There are in addition the user scopes that provide access to protected resources on the server some more technical scopes. These are the “offline_access” scope and “openid” scope.

The “offline_access” is defined in OpenId Core specification and enables the client to request new access token after expiration using the refresh token granted at the authorization time. The “openid” is defined in OpenId core specifications and enables the client to identify the user.

8 Security considerations

All transactions MUST be protected in transit by TLS as described in BCP195 [BCP195].

All clients MUST conform to applicable recommendations found in the Security Considerations sections of [RFC6749] and those found in the OAuth 2.0 Threat Model and Security Considerations document [RFC6819].

All clients MUST conform to applicable recommendations in the OWASP Mobile security project’s Secure Mobile Development Guidelines [OWASP].

9 Error messages

The following error codes and messages are used in Finnish Kanta PHR.

Code	Definition
1000	Unable to write uselog
1001	Internal error
1002	Database error
1003	Database connection error
1004	Error validating token
1005	Unidentified error code
1006	Unable to read token
1007	Unable to read error codes
1008	Unable to create authorization token
1009	Authorization server has no PHR information set
5000	Invalid or missing header

5001	Missing JWT in request
5002	Client's registered certificate oid and actual certificate oid do not match
5003	No valid token found
5004	No scopes found from token
5005	No access rights
5006	No profile found
5007	No value set found
5008	Missing profile
5009	No Questionnaire found
5010	No CodeSystem found
5011	No Resource found
5012	Resource with this Id already in use
5014	Data is invalid
5015	Multiple documents found
5016	Too many resources found with the criteria
5017	Too many profiles found from the resources
5018	Resource with this url already in the system
5019	Unknown client
5020	Token is not sufficient for this operation
5021	Token is invalid
5022	Operation is not permitted for professional applications
5023	Resource format is not supported
5024	Only identifier is supported
5025	Invalid method
5026	Reference type is missing
5027	Reference type is not allowed
5028	Missing required parameter
5029	Unauthorized grant type
5030	Person is too young
5031	Page not found or it’s expired

5032	Invalid token request

10 References

[RFC6749] Hardt, D. The OAuth 2.0 Authorization Framework, RFC 6749, DOI 10.17487/RFC6749, October 2012.

[RGC6819] T. Lodderstedt, M. McGloin, P. Hunt.OAuth 2.0 Threat Model and Security Considerations, RFC6819. January 2013.

[RFC7517] M. Jones. JSON Web Key (JWK), RFC 7517. May 2015.

[RFC7591] OAuth 2.0 Dynamic Client Registration Protocol

[OWASP] OWASP Mobile Security Project, Mobile Application Coding Guidelines https://www.owasp.org/index.php/OWASP_Mobile_Security_Project#tab=Secure_Mobile_Development

[DIONA] https://tools.ietf.org/html/draft-ietf-oauth-native-apps-03

[RFC7523] M. Jones, B. Campbell, C. Mortimore. JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants. RFC 7523. May 2015.

[ARGONAUT] Argonaut Project. Cross-Organization Data Access Profile. Working draft of a OAuth 2.0 profile to support the EHR-to-EHR use case. December 2015. https://github.com/smart-on-fhir/smart-on-fhir.github.io/wiki/cross-organizational-auth

[CONNECTCORE] N. Sakimura, J. Bradley, M. Jones, B. de Medeiros, C. Mortimore. OpenID Connect Core 1.0 incorporating errata set 1. November 2014. http://openid.net/specs/openid-connect-core-1_0.html

[BCP195] Y. Sheffer, R. Holz, P. Saint-Andre. Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Tarnsport Layer Security (DTLS). May 2015.

[SUOMI.FI] https://www.suomi.fi/page/about-eidentification

Mutual X.509 Transport Layer Security (TLS) Authentication for OAuth Clients https://tools.ietf.org/html/draft-campbell-oauth-tls-client-auth-00