MyDataShare OpenID Connect Integration Specification

Background

The purpose of this document is to describe the technical details of integrating a Relying Party application with the MyDataShare using OpenID Connect protocol.

OpenID Connect Identity Provider Metadata

OpenID Connect compliant access management Identity Provider metadata can be fetched from following URL: https://gluu.alpha.mydatashare.com/.well-known/openid-configuration

Integrating applications can use the metadata to get information about the OpenID Connect interfaces and configuration parameters supported by MyDataShare (e.g. token endpoint, authorization endpoint) and about the claims that are supported.

Pre-requisites

Using MyDataShare OpenID Connect endpoints requires the Relying Parties to be registered. Relying Party registration must be asked separately from MyDataShare and it will be done manually through the administration console.

Information Needed from Relying Parties

In order to manually register an application, following information is needed:

- URLs where browser can be redirected after successful authentication (HTTPS URLs or mobile application custom scheme URLs)
- URLs where browser can be redirected after successful logout (HTTPS URLs or mobile application custom scheme URLs)
- Name of the application
- Contact email address(es)
- Integration type: Authorization Code, Client Credentials Grant, Refresh Token Grant. Implicit grant type is deprecated.
- Scopes that need to be available (`openid` by default). Available scopes are listed below in section "Supported Scope Values"
- ID token signing algorithm (`RS256` by default)
- Token endpoint authentication method (`client_secret_basic` by default). Available authentication methods are listed in OpenID Connect Core specification.)
- Subject Identifier Type (`pairwise` or `public`, `pairwise` by default)

Information Supplied to Relying Parties

After registration the integrating application will get following information:

- Client ID: Unique identifier of the application
- Client Secret: Secret generated for the application

Supported Scope Values

Following table describes the scopes that are supported by MyDataShare:

Scope	Description
`openid`	Basic scope defined in the OpenID Connect Core specification.
`profile`	OpenID Connect Core compliant scope to fetch basic profile attributes of the authenticated user.
`admin`	Provides administrator access to MOP APIs.
`organization`	Provides organization-level access to MOP APIs.
`wallet`	Provides access to MOP Wallet APIs.
`extended_introspection`	Determines whether access token introspection returns additional claims. Only usable for MOP.

Supported ACR Values

Following table contains the list of supported authentication methods that are currently available:

`acr_values`	`preselectedExternalProvider`	Authentication Assurance Level	Description
`passport_social`	`ewogICAicHJvdmlkZXIiIDogIm9wZW5pZGNvbm5lY3QiCn0`	2	Strong user authentication with Signicat remote ID broker.
`passport_social`	`ewogICAicHJvdmlkZXIiIDogInNpc3VpZCIKfQ`	0, 1 or 2 (readable from `loa` claim)	Mobile authentication with SisuID.

Integrating relying party can request specific authentication method with combination of `acr_values` and `preselectedExternalProvider` request parameters when calling the MyDataShare authorization endpoint.

Supported UI Locales

Following table contains the list of supported UI locales:

Key	Value
en	English

Supported Claims

Common Claims

ID token payload is a key-value type of JSON data structure. Individual key-value pairs are called claims in the JWT terminology. ID Token contains a set of common claims. These are defined in the ID Token specification. Below table lists these claims:

Claim	Scope	Mandatory	Description	Example Value
`iss`	`openid`	Yes	Identifier of the creator of the ID token. This is populated with the URL of access management.	https://gluu.beta.mydatashare.com
`aud`	`openid`	Yes	Relying Party for whom the ID token is generated. Client ID of the Relying Party is populated here.	`@!596F.C4DF.5333.39D2!0001!4440.D05E!0008!04EA.2671`
`exp`	`openid`	Yes	Session expiration timestamp.	`1464078639`
`iat`	`openid`	Yes	ID Token creation timestamp.	`1464075039`
`auth_time`	`openid`	No	Authentication timestamp.	`1464075039`
`nonce`	`openid`	No	Value that the Relying Party sent to the Identity Provider when calling the authorization endpoint.	`2310aa8f-6333-47bc-bfa2-54c847517396`
`acr`	`openid`	No	Authentication method that was used to authenticate the end-user.	`passport_social`
`at_hash`	`openid`	No	Hash of the access token associated to the ID token. Generation of this value is described in the OpenID Connect specification.	`sXG8ntmpPQj8KMdOvb_lmU38DpH0uDxx8txv8X8KHMo`
`sub`	`openid`	Yes	Persistent unique identifier of the authenticated user. If subject identifier type for the client is `public`, the value is the same for all clients. If the subject identifier type is `pairwise`, the value will be client specific.	`@!2027.861B.4505.5885!0001!200B.B5FE!0000!08AC.7BE7`

MyDataShare Specific Claims

Following table lists the claims that are supported in the OpenID Connect compliant UserInfo endpoint and are also populated to the ID token when the associated scope is in place and client is configured to receive the claims in ID Token:

Claim	Scope	Mandatory	Description	Example Value
`name`	`profile`	No	Full name.	Test Tester
`given_name`	`profile`	No	First names.	Test
`family_name`	`profile`	No	Last name.	Tester
`govID`	`profile`	No	Government Unique Identifier.	010101-0101
`inum`	`profile`	Yes	Gluu persistent unique identifier	`@!2027.861B.4505.5885!0001!200B.B5FE!0000!08AC.7BE7`
`updated_at`	`profile`	Yes	Last update timestamp	`1590665332693`

For users who have authenticated with `sisuid` and low authentication level, the claims are populated in following way: * `name` contains the mobile phone number * `given_name`, `family_name`, and `govID` are not populated

For users who have authenticated with `sisuid` and strong authentication level, or with `signicat`, the claims are populated in following way: * `name` contains the full name of the user * `given_name` contains the first names of the user * `family_name` contains the last name of the user * `govID` contains the government unique identifier of the user

Using Authorization Code Grant Flow

Web applications that can keep the client_secret secure (i.e. have a backend) can use the OpenID Connect compliant Authorization Code Grant Flow. Additionally client applications (native mobile applications or Javascript based single-page applications) that cannot keep the client secret secure, can use the Authorization Code Grant Flow with PKCE extension.

Authorization code grant flow is initiated with a call that the Relying Party does to the Identity Provider authorization endpoint:

```
https://<domain>/oxauth/restv1/authorize?
response_type=code&client_id=@!2027.861B.4505.5885!0001!200B.B5FE!0008!84D4.82F3&redirect_uri=https://sec-
221.nixu.fi/redirect&scope=openid profile&state=93118d46-e50c-4682-956d-51370c7970f2&nonce=f2f4a9cd-cdc0-4a84-ac33-
d9810a961f0b&acr_values=passport_social&preselectedExternalProvider=ewogICAicHJvdmlkZXIiIDogInNpc3VpZCIKfQ
```

Following table describes the parameters that need to be provided in the call:

Parameter	Mandatory	Description	Example Value
`response_type`	Yes	Authorization flow to be used. Must be populated with value `code`.	`code`
`client_id`	Yes	Unique identifier of the registered Relying Party.	`@!2027.861B.4505.5885!0001!200B.B5FE!0008!84D4.82F`
`redirect_uri`	Yes	URL where the browser is redirected after successful authentication and authorization. This must be one of the `redirect_uri`	`https://sec-221.nixu.fi/redirect`

		values registered for the Relying Party.	
`scope`	Yes	Space separated list of scopes that will be associated to the created access token. List of supported scopes is provided above.	`openid profile`
`state`	Yes	Random value used in CSRF protection. Relying Party must store the value to session or to cookie when initiating the call to authorization endpoint. The Relying Party must also validate that the value of the `state` parameter has not changed when the Identity Provider redirects back to Relying Party. The specification says that this parameter is not mandatory, but using this parameter is recommended strongly.	`93118d46-e50c-4682-956d-51370c7970f2`
		Random value that will be populated in the generated ID token. Used for protection	

`nonce`	Yes	against replay attacks. The specification says that the parameter is not mandatory, but using this parameter is recommended strongly.	`f2f4a9cd-cdc0-4a84-ac33-d9810a961f0b`
`ui_locales`	No	UI locale to be used. Default value is `en`.	`en`
`acr_values`	Yes	Authentication method to be used. Supported values are provided above.	`passport_social`
`preselectedExternalProvider`	Yes	External Identity Provider to be used. Supported values are provided above.	`ewogICAicHJvdmlkZXIiIDogInNpc3VpZCIKfQ`

After sending the request to authorization endpoint, access management checks for the presence of an existing single sign-on session. If no valid session exists, the user must authenticate.

After successful authentication and authorization, user's browser is redirected back to redirect_uri set by the Relying Party:

```
https://sec-221.nixu.fi/redirect?state=93118d46-e50c-4682-956d-51370c7970f2&code=831daff2-9353-438e-9aaf-
afe97518b2e0
```

Following table describes the parameters that are populated in the `redirect_uri`:

Parameter	Mandatory	Description	Example Value
`state`	Yes	If the Relying Party populated the `state` parameter when calling the authorization endpoint, the same value is provided here when redirecting back to the Relying Party.	`93118d46-e50c-4682-956d-51370c7970f2`
`code`	Yes	Authorization Code that the Relying Party can exchange into a valid access token by calling the token endpoint.	`831daff2-9353-438e-9aaf-afe97518b2e0`

After receiving the authorization, the Relying Party will exchange the authorization into a valid access token that it can use to make calls to resource servers. Below is an example request to resolve the access token through the token endpoint:

```
POST /oxauth/restv1/token
Host: <domain>
Accept: application/json
Authorization: Basic ********
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code&redirect_uri=https://sec-221.nixu.fi/redirect&code=831daff2-9353-438e-9aaf-
afe97518b2e0
```

Following table describes the parameters that need to beprovided when calling the token endpoint:

Parameter	Mandatory	Description	Example Value
`grant_type`	Yes	Identifier of the authentication method. This must be populated with value `authorization_code`.	`authorization_code`
`redirect_uri`	Yes	`redirect_uri` where browser was redirected.	`https://sec-221.nixu.fi/redirect`
`code`	Yes	Authorization code that the Relying Party received.	`831daff2-9353-438e-9aaf-afe97518b2e0`

In order to call the token endpoint the Relying Party must authenticate itself. The default authentication method is used for this purpose is HTTP-Basic. The username is the `client_id` of the Relying Party and the password is the `client_secret`. Other authentication methods listed in the OpenID Connect specification may also be used as defined in the OpenID Connect Core specification.

Access management validates the call to the token endpoint in following way: * Authorization code is mandatory and it must be present in access management side * Redirect URI is mandatory and the value must match the URL which was requested when calling the authorization endpoint * Grant Type is mandatory and it must have value `authorization_code` * Client ID used in application authentication must be the same that was used when calling the authorization endpoint

If request validation fails, access management will respond with HTTP status code 400 or 401 as defined in the OAuth 2.0 specification. HTTP response body contains additional information about the error in JSON format.

If the request to token endpoint is successful, access management will respond with HTTP status code 200 and following kind of response:

```
HTTP/1.1 200 OK
Date: Wed, 25 May 2016 10:12:38 GMT
Content-Length: 1145
Content-Type: application/json

{"access_token":"7f648110-505d-4960-868a-
3dfdf0599cad","token_type":"bearer","expires_in":14399,"refresh_token":"a63a3f0e-7222-49d8-bdd3-
146ba0cb12f6","scope":"openid profile","id_token":"..."}
```

Following table contains the parameters returned by token endpoint:

Parameter	Mandatory	Description	Example Value
`access_token`	Yes	Access token that can be used to make calls to resource servers.	`7f648110-505d-4960-868a-3dfdf0599cad`
`token_type`	Yes	Access token type. This will always have value `bearer` which means that access tokens following the guidelines from RFC 6750.	`bearer`
`expires_in`	Yes	Access token lifetime in seconds.	`14399`
`refresh_token`	No	Refresh token, which is a one-time usable token that can be exchanged into a valid access token. Refresh token usage is described in separate chapter.	`a63a3f0e-7222-49d8-bdd3-146ba0cb12f6`
`scope`	No	Scope associated to the `access_token`.	`openid profile`
`id_token`	No	ID Token which is a JSON Web Token containing information about the authentication session and about the authenticated identity.	See OpenID Connect Core specification.

Using Implicit Grant Flow

Usage of implicit grant flow has been deprecated in favor of authorization code grant flow with PKCE extension. This is the recommended integration method for client applications that are not capable of keeping their `client_secret` secure, such as Javascript MVC web applications and native mobile applications.

Using Client Credentials Grant Flow

Backend applications that require application authentication, but do not require end-user authentication, may use the OAuth 2.0 compliant client credentials grant flow.

Client credentials grant flow usage needs to be separately requested when registering the Relying Party.

When using client credentials grant flow, the Relying Party makes following request to the token endpoint:

```
POST /oxauth/restv1/token
Host: <domain>
Accept: application/json
Authorization: Basic ********
Content-Type: application/x-www-form-urlencoded

grant_type=client_credentials&scope=openid profile
```

Following table contains the parameters that need to be provided in the call:

Parameter	Mandatory	Description	Example Value
`grant_type`	Yes	Authorization flow to be used. Must be populated with value `client_credentials`.	`client_credentials`
`scope`	No	Space separated list of scopes that will be associated to the created access token. List of supported scopes is provided above.	`openid profile`

In order to call the token endpoint the Relying Party must authenticate itself. The default authentication method is used for this purpose is HTTP-Basic. The username is the `client_id` of the Relying Party and the password is the `client_secret`. Other authentication methods listed in the OpenID Connect specification may also be used as defined in the OpenID Connect Core specification.

Access management validates the call to the token endpoint in following way: * Grant Type is mandatory and it must have value `client_credentials` * Relying Party authentication information must be available in the `Authorization` HTTP header and must match the values configured for the Relying Party

If request validation fails, access management will respond with HTTP status code 400 or 401 as defined in the OAuth 2.0 specification.

HTTP response body contains additional information about the error in JSON format.

If the request to token endpoint is successful, access management will respond with HTTP status code 200 and following kind of response:

```
HTTP/1.1 200 OK
Date: Thu, 26 May 2016 06:17:33 GMT
Content-Length: 600
Content-Type: application/json

{"access_token":"7f648110-505d-4960-868a-3dfdf0599cad","token_type":"bearer","expires_in":299,"scope":"openid
profile"}
```

Following table contains the parameters returned by token endpoint:

Parameter	Mandatory	Description	Example Value
`access_token`	Yes	Access token that can be used to make calls to resource servers.	`7f648110-505d-4960-868a-3dfdf0599cad`
`token_type`	Yes	Access token type. This will always have value `bearer` which means that access tokens following the guidelines from RFC 6750.	`bearer`
`expires_in`	Yes	Access token lifetime in seconds.	`299`
`scope`	No	Scope associated to the `access_token`.	`openid profile`

The response will contain no `refresh_token` or `id_token`. If the Relying Party needs a new access token, it must perform the call to token endpoint again.

Refresh Token Usage

Applications using authorization code grant will receive a refresh token as a response to the call to the token endpoint. This can be used to offer persistent access to API interfaces on behalf of the user, because when the access token expires, the refresh token can be used to generate a new access token. Refresh token usage is described in detail in OAuth 2.0 specification.

Relying Parties can use refresh tokens by making a call to the token endpoint in following way:

```
POST /oxauth/restv1/token
Host: <domain>
Accept: application/json
Authorization: Basic ********
Content-Type: application/x-www-form-urlencoded

grant_type=refresh_token&scope=openid profile&refresh_token=c033515f-eac6-4a4f-a6cd-d158896b216a
```

Following table contains the parameters that need to be provided in the call:

Parameter	Mandatory	Description	Example Value
`grant_type`	Yes	Authorization flow to be used. Must be populated with value `refresh_token`.	`refresh_token`
`scope`	No	Space separated list of scopes that will be associated to the created access token. List of supported scopes is provided above.	`openid profile`
`refresh_token`	Yes	Refresh token.	`c033515f-eac6-4a4f-a6cd-d158896b216a`

In order to call the token endpoint the Relying Party must authenticate itself. The default authentication method is used for this purpose is HTTP-Basic. The username is the `client_id` of the Relying Party and the password is the `client_secret`. Other authentication

methods listed in the OpenID Connect specification may also be used as defined in the OpenID Connect Core specification.

Access management validates the call to the token endpoint in following way:

- Refresh token is mandatory and it must be present in access management side
- Grant Type is mandatory and it must have value `refresh_token`
- Relying Party authentication information must be available in the `Authorization` HTTP header and must match the values configured for the Relying Party
- Client ID used in application authentication must be the same as the one that is tied to the refresh token

If request validation fails, access management will respond with HTTP status code 400 or 401 as defined in the OAuth 2.0 specification. HTTP response body contains additional information about the error in JSON format.

If the request to token endpoint is successful, access management will respond with HTTP status code 200 and following kind of response:

```
HTTP/1.1 200 OK
Date: Wed, 25 May 2016 10:12:38 GMT
Content-Length: 1145
Content-Type: application/json

{"access_token":"7f648110-505d-4960-868a-
3dfdf0599cad","token_type":"bearer","expires_in":299,"refresh_token":"fb6446a3-d944-434b-9a51-
18144303ff29","id_token":"...","scope":"openid profile"}
```

Following table contains the parameters returned by token endpoint:

Parameter	Mandatory	Description	Example Value
`access_token`	Yes	Access token that can be used to make calls to resource servers.	`7f648110-505d-4960-868a-3dfdf0599cad`
`token_type`	Yes	Access token type. This will always have value `bearer` which means that access tokens following the guidelines from RFC 6750.	`bearer`
`expires_in`	Yes	Access token lifetime in seconds.	`14399`
`refresh_token`	No	Refresh token, which is a one-time usable token that can be exchanged into a valid access token. Refresh token usage is described in separate chapter.	`fb6446a3-d944-434b-9a51-18144303ff29`
`scope`	No	Scope associated to the `access_token`.	`openid profile`
`id_token`	No	ID Token which is a JSON Web Token containing information about the authentication session and about the authenticated identity.	See OpenID Connect Core specification.

ID Token Validation

The ID Token returned by access management is a JSON Web Token which is defined in the OpenID Connect Core Specification and in RFC 7519.

When the Relying Party receives an ID Token from the Identity Provider, it must validate the ID Token based on following rules:

- `iss` claim must match the URL of the identity provider
- `aud` claim must contain the Relying Party `client_id`
- ID Token signature must be verified as defined in RFC 7515. Public keys to verify the signature can be requested from JWKS endpoint
- `exp` claim must be checked so that the ID Token has not expired
- `iat` claim must be check so that not too long has passed since the ID Token creation
- `auth_time` claim may be checked to determine how long it has passed since the user has last authenticated

- `acr` claim may be checked to verify that the authentication was done with a trusted authentication method.

Requesting User Information through UserInfo Endpoint

After successful authentication and authorization and receiving a valid access_token, the Relying Party can request the information of the authenticated user through the OpenID Connect compliant UserInfo endpoint.

The claims supported by the Userinfo endpoint are listed above in "Supported Claims".

Below is an example request to the UserInfo endpoint:

```
GET /oxauth/restv1/userinfo
Host: <domain>
Accept: application/json
Authorization: Bearer ...
```

Below is an example response from the UserInfo endpoint:

```
{"sub":"0035-eda95a7a5eafa07c849dc65fc0ea93c6-
fbb81765","inum":"@!2027.861B.4505.5885!0001!200B.B5FE!0000!08AC.7BE7","name":"Test
Tester","family_name":"Tester","given_name":"Test"}
```

Logout

Relying parties can logout an end-user by using the OpenID Connect compliant RP initiated logout endpoint.

Following is an example URL to perform the logout:

```
https://<domain>/oxauth/restv1/end_session?post_logout_redirect_uri=https://sec-221.nixu.fi/test&id_token_hint=...
```

Following table contains the parameters that need to be provided in the call:

Parameter	Mandatory	Description	Example Value
`post_logout_redirect_uri`	Yes	URL where the browser is redirected after successful logout. URL must be one of the valus configured for the Relying Party.	`https://sec-221.nixu.fi/test`
`id_token_hint`	Yes	ID Token assigned to the user being logged out.	Described above.

Access Token Validation in Resource Server

When receiving an access token sent by the Relying Party, the resource servers can validate the access token by calling the RFC 7662 compliant introspection endpoint.

`Authorization` header must contain a valid access token. Clients requiring additional claims to be incorporated in the introspection response must provide an access token with `extended_introspection` scope.

Below is an example request to the token introspection endpoint:

```
POST /oxauth/restv1/introspection
Host: <domain>
Accept: application/json
Authorization: Bearer eda34...
token=eda34........
```

Below is an example response from the introspection endpoint without `extended_introspection` scope:

```
{"active":true,"scopes":
["openid"],"client_id":"@!4475.51D1.A110.13CE!0001!A2B0.0497!0008!C46C.0D3D.60B9.A4B7","username":"Test
User","token_type":"bearer","exp":1591686571,"iat":1591600171,"sub":"@!4475.51D1.A110.13CE!0001!A2B0.0497!0000!2600.
9F11.7B4E.966B","aud":"@!4475.51D1.A110.13CE!0001!A2B0.0497!0008!C46C.0D3D.60B9.A4B7","iss":"https:\/\/gluu.alpha.my
datashare.com","jti":null,"acr_values":null,"scope":["openid"]}
```


Below is an example response from the introspection endpoint with `extended_introspection` scope:


```
{"active":true,"scopes":
["openid"],"client_id":"@!4475.51D1.A110.13CE!0001!A2B0.0497!0008!C46C.0D3D.60B9.A4B7","username":"Test
User","token_type":"bearer","exp":1591686388,"iat":1591599988,"sub":"@!4475.51D1.A110.13CE!0001!A2B0.0497!0000!2600.
9F11.7B4E.966B","aud":"@!4475.51D1.A110.13CE!0001!A2B0.0497!0008!C46C.0D3D.60B9.A4B7","iss":"https:\/\/gluu.alpha.my
datashare.com","jti":null,"acr_values":null,"scope":
["openid"],"given_name":"Test","family_name":"User","uid":"test.user@nixu.com"}
```


Below table describes the common parameters that are returned by the token introspection endpoint:

Parameter	Mandatory	Description	Example Value
`active`	Yes	Indicates whether the access token is active or not.	`true`
`scope`	No	JSON array containing scopes associated to the access token.	`["openid"]`
`client_id`	Yes	Client for which the access token was issued.	`@!4475.51D1.A110.13CE!0001!A2B0.0497!0008!C46C.0D3D.60B9.A4B7`
`username`	Yes	Display name of the authenticated user.	`Test User`
`token_type`	Yes	Token type of the access token. This is always populated with value `bearer`.	`bearer`
`exp`	Yes	Expiration timestamp of the access token.	`1591686388`
`iat`	Yes	Access token creation timestamp.	`1591599988`
`sub`	Yes	Persistent unique identifier of the authenticated user. If subject identifier type for the client is `public`, the value is the same for all clients. If the subject identifier type is `pairwise`, the value will be client specific.	`@!4475.51D1.A110.13CE!0001!A2B0.0497!0000!2600.9F11.7B4E.966B`
`aud`	Yes	Client for which the access token was issued.	`@!4475.51D1.A110.13CE!0001!A2B0.0497!0008!C46C.0D3D.60B9.A4B7`
`iss`	Yes	Issuer of the access token.	`https:\/\/gluu.alpha.mydatashare.com`
`acr_values`	No	Authentication method that was used when authenticating.	`passport_social`

Following additional claims are populated when using `extended_instrospection` scope:

Parameter	Mandatory	Description	Example Value
`given_name`	No	User's first names.	Test
`family_name`	No	User's last name.	Tester
`id_type`	No	Identifier type, either `sisuid` or `ssn` if populated.	`sisuid`
`id_source`	No	Identity source. Possible values are `sisuid` and `signicat`	`sisuid`
`uid`	Yes	User's username.	`test.user@nixu.com`
`govID`	No	User's government issued identifier.	`010101-0101`
`c`	No	Country of government issued identifier ISO 3166-alpha-2 format.	`FI`
`loa`	No	Identity assurance level of the identity. Either `0` or `2`.	`2`
`pairwise_identifiers`	No	JSON Array containing pairwise identifiers of the user.	`["pairwise1", "pairwise2" ]`

When user has authenticated with SisuID and low-level authentication, the additional claims are populated in following way: * `given_name` and `family_name` are not available * `id_type` is `sisuid` * `id_source` is `sisuid` * `uid` contains the unique SisuID persistent identifier * `govID` and `c` are not available * `loa` is populated with `0` * `pairwise_identifiers` contains the `pairwise` values for the user * `username` contains the mobile phone number of the user * `acr_values` is populated with `passport_social`

When user has authenticated with SisuID and strong authentication, the additional claims are populated in following way: * `given_name` is populated with the first names of the user * `family_name` is populated with the last name of the user * `id_type` is `sisuid` * `id_source` is `sisuid` * `uid` contains the unique SisuID persistent identifier * `govID` contains the government issued identifier * `c` contains the country corresponding to government issued identifier in ISO 3166-alpha-2 format. * `loa` is populated with `2` * `pairwise_identifiers` contains the `pairwise` values for the user * `username` contains the displayname (first names and last name) of the user. * `acr_values` is populated with `passport_social`

When user has authenticated with Signicat strong authentication, the additional claims are populated in following way: * `given_name` is populated with the first names of the user * `family_name` is populated with the last name of the user * `id_type` is `ssn` * `id_source` is `signicat` * `uid` contains random unique user name created for the user * `govID` contains the government issued identifier * `c` contains the country corresponding to government issued identifier in ISO 3166-alpha-2 format. * `loa` is populated with `2` * `pairwise_identifiers` contains the `pairwise` values for the user * `username` contains the displayname (first names and last name) of the user. * `acr_values` is populated with `passport_social`

Links

OpenID Connect Core 1.0: http://openid.net/specs/openid-connect-core-1_0.html

OpenID Connect Discovery 1.0: http://openid.net/specs/openid-connect-discovery-1_0.html

RFC 6749: The OAuth 2.0 Authorization Framework: https://tools.ietf.org/html/rfc6749

RFC 6750: The OAuth 2.0 Authorization Framework: Bearer Token Usage: https://tools.ietf.org/html/rfc6750

RFC 7519: JSON Web Token (JWT): https://tools.ietf.org/html/rfc7519

RFC 7515: JSON Web Signature (JWS): https://tools.ietf.org/html/rfc7515

RFC 7517: JSON Web Key (JWK): https://tools.ietf.org/html/rfc7517

RFC 7636: Proof Key for Code Exchange: https://tools.ietf.org/html/rfc7636

RFC 7662: OAuth 2.0 Token Introspection: https://tools.ietf.org/html/rfc7662