Co-financed by the Connecting Europe Facility of the European Union

increasinG tRust with eId for Developing buSiness

D4.3 APIs Development and Deployment

Document Identification			
Status	Final	**Due Date**	28/04/2021
Version	1.0	**Submission Date**	12/05/2021

Related Activity	Act 4	**Document Reference**	D4.3
Related Deliverable(s)			
Lead Participant	ADACOM	**Lead Author**	Stamatis Zamanis (ADACOM)
Contributors	INFOCERT UAegean	**Reviewers**	Nikolaos Triantafyllou (UAegan) Romualdo Carbone (INFOCERT)

Keywords
eIDAS, AML, KYC, KYB, GRIDS, Aggregate, BAA

Document Information

List of Contributors	
Name	Partner
Nikolaos Triantafyllou	UAegean
Romualdo Carbone	INFOCERT
Stamoulis Zamanis	ADACOM
Serafeim Makris	ADACOM
Konstantinos Nousias	ADACOM
Emmannouil Giakoumakis	ADACOM

Document History			
Version	Date	Change editors	Changes
0.1	08/04/2021	Stamoulis Zamanis (ADACOM)	Preparation of the document structure First draft.
0.2	14/04/2021	Konstantinos Nousias (ADACOM)	Minor changes
0.3	26/04/2021	Stamoulis Zamanis (ADACOM)	Low level design update
0.4	27/04/2021	Emmannouil Giakoumakis (ADACOM)	Layout update
0.5	28/04/2021	Stamoulis Zamanis (ADACOM) Serafeim Makris (ADACOM)	Update based on contributors review
0.6	29/04/2021	Emmannouil Giakoumakis (ADACOM)	Layout update, Submition
0.7	30/04/2021	Juan Alonso, Juan Carlos Perez (Atos)	Quality Assement
0.8	07/05/2021	Emmannouil Giakoumakis (ADACOM)	Final update
1.0	12/05/2021	Juan Alonso, Juan Carlos Perez (Atos)	Review of the final version before submission

Quality Control		
Role	Who (Partner short name)	Approval Date
Deliverable leader	Stamatis Zamanis (ADACOM)	07/04/2021
Peer reviewers	1st reviewer - Konstantinos Noussias (ADACOM) 2st reviewer – Serafeim Makris (ADACOM) 3rd reviewer - Emmannouil Giakoumakis (ADACOM)	28/04/2021 29/04/2021 29/04/2021
Quality Manager	Juan Alonso (Atos)	29/04/2021

Table of Contents

List of Figures

List of Acronyms

Abbreviation / acronym	Description
AML	Anti Money Laundering
API	Application Programming Interface
CDD	Customer due diligence
DC	Data Consumer
DP	Data Provider
DS	Data Subject
EC	European Commission
eIDAS	eIDAS Regulation (EU) No. 910/2014 on electronic identification and trust services for electronic transactions in the internal market
FIs	Financial Institutions
GDPR	General Data Protection Regulation
JWKS	JSON Web Key Set
KYB	Know Your Business
KYC	Know Your Customer
LEI	Legal Entity Identifier
LOU	Local Unit Operator
OIDC	OpenID Connect
OIDC IDA	IDA OpenID Connect for IDentity Assurance
SDK	Software Development Kit

Executive Summary

GRIDS identifies the importance of bundled services for worldwide commerce, business and financial sectors that require KYC – Know Your Customer guidelines and identifies the service receiver through an internationally accepted eIDAS eID system.

As a result, part of the project focuses on existing services, whose addressable market is really large and will have a significant growth, especially due to the Covid pandemic period.

Deliverable "D4.3 – APIs Development and Deployment" is related to Activity 4 of the GRIDS project (increasinG tRust with eId for Developing buSiness).

The goal of this deliverable is to provide an SDK that will allow end-users to connect to GRIDS service offerings through standard, publicly available and GDPR compliant APIs. This SDK will allow the seamless and SSO-based integration of end-users with KYC services, greatly enhancing the efficiency and accuracy of the due diligence process.

1 Introduction

1.1 Purpose of the document

This document refers to the Activity 4 – End User APIs for Business Users of GRIDS Services, and more specifically, to task 4.2 dedicated to record and design 'End-User APIs Development and Deployment'.

Within this document a tool for developing, known as a Software Development Kit (SDK), is analysed and described. This tool will help Data Consumers to consume GRIDS APIs as described by Activity 4.2 in grand agreement.

1.2 Relation to other project work

All deliverables related to this project are depended on:

- Activity 2,
  - Task 2.2: Architecture and APIs Design and action plan
- Activity 3,
  - Task 3.1: BAA Development

1.3 Structure of the document

This document is structured in 4 major chapters

Chapter 2 presents an overview of a Software Development kit

Chapter 3 presents a high-level overview of the SDK development for the GRIDS project

Chapter 4 presents the conclusions of the deliverable.

2 GRIDs Software development kit

The purpose of GRIDS Software development kit is to offer Data Consumer (DC) a quick and straightforward way to implement and consume GRIDS API services.

2.1 What is API

An application programming interface (API) is a set of protocols and tools for building application software. It is a set of clearly defined methods of communication between various software components. Most companies, especially tech companies, have built APIs for their customers or for internal use.

2.2 What is SDK

An SDK, or Software Development Kit, is a set of tools, guidelines, and programs used to develop applications for a specific platform. Suggested by the name, an SDK is a kit for developing software. SDKs can include APIs (or multiple APIs), IDE's, Documentation, Libraries, Code Samples, and other utilities. SDKs boast a set of robust features and functionalities which reduce the complexity of developing programs and applications.

2.3 Details

GRIS SDK uses JAVA as its main language and a JAR package file is generated and shared with the Data Consumers.

The main purpose of the SDK is providing a set of interfaces, models and services(wrappers) following OpenID Connect, Identity Assurance and eKYC specifications that the DC can use in order to connect and consume GRIDS API endpoints.

2.4 GRIDS API endpoints implemented by SDK

The DCC supports configuration end point as described on OpenId specifications[1]. This will make available the OIDC end points and metadata needed for Data Consumer clients communicating with the BAA.

The BAA acts as an OIDC Provider (OP) and will follow the OIDC IDA 1.0 specification to advertise the KYC claims and trust framework it supports over distributed claim sources in the GRIDS network.

Below is the BAA Components diagram:

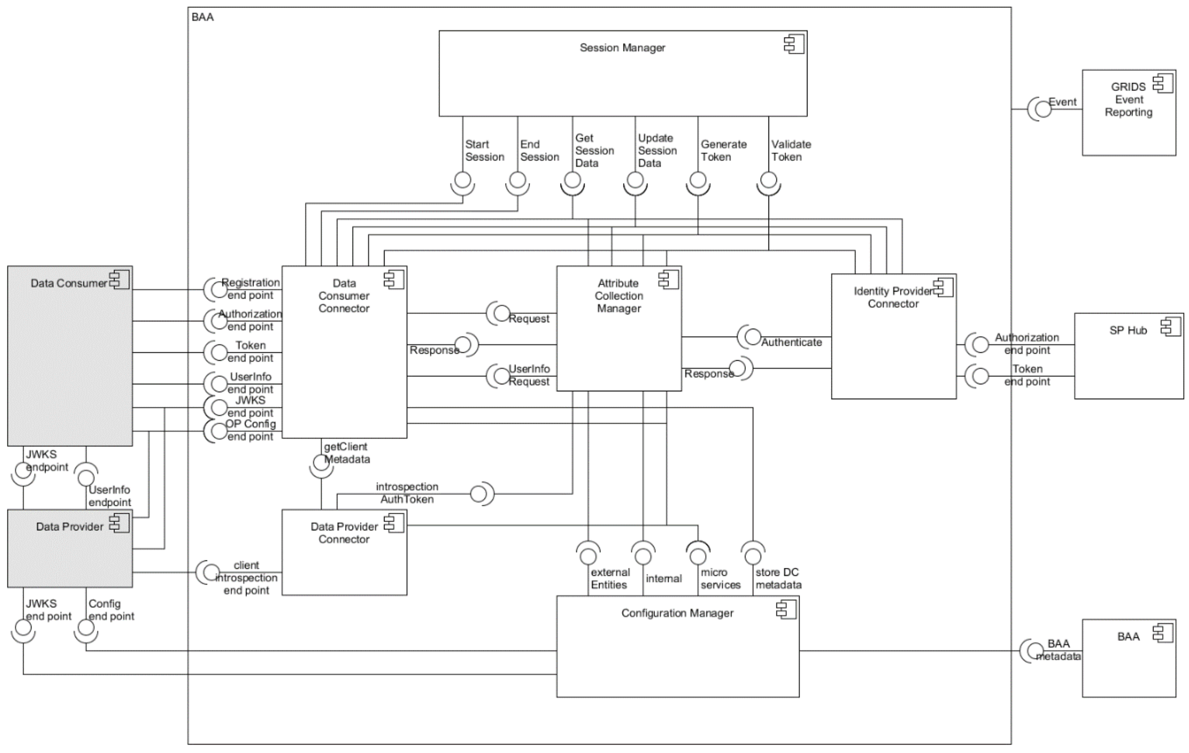

Figure 1: BAA Component diagram

Following the BAA Component diagram, GRIDs SDK (grey area in Figure 1) should wrap all the end points that can be used by the Data Consumer:

- **Data Consumer Connector**
  - **Regististration end point**. The Client Registration Endpoint is an OAuth 2.0 Protected Resource through which a new Client registration can be requested. The OpenID Provider MAY require an Initial Access Token that is provisioned out-of-band (in a manner that is out of scope for this specification) to restrict registration requests to only authorized Clients or developers
  - **Authorization end point.** The Authorisation Endpoint supports an OAuth 2.0 Authorization Request that requests that the End-User be authenticated by the Authorization Server

[1] https://openid.net/specs/openid-connect-discovery-1_0.html#ProviderConfig

- **Token end point**. To obtain an Access Token, an ID Token, and optionally a Refresh Token, the RP (Client) sends a Token Request to the Token Endpoint to obtain a Token Response, when using the Authorization Code Flow.
    - **UserInfo end point.** The UserInfo Endpoint is an OAuth 2.0 Protected Resource that returns Claims about the authenticated End-User. To obtain the requested Claims about the End-User, the Client makes a request to the UserInfo Endpoint using an Access Token obtained through OpenID Connect Authentication
    - **JWKS end point**. The signing key(s) the RP uses to validate signatures from the OP.
    - **OP Config end point**. The configuration of any GRIDS Endpoint: BAA or Data Provider can be retrieved via OpenID Configuration published by the respective Issuer. OpenID Providers supporting Discovery MUST make a JSON document available at the path formed by concatenating the string /well-known/openid-configuration to the Issuer
- **Data Provider**
    - **UserInfo end point**. The UserInfo Endpoint is an OAuth 2.0 Protected Resource that returns Claims about the authenticated End-User. To obtain the requested Claims about the End-User, the Client makes a request to the UserInfo Endpoint using an Access Token obtained through OpenID Connect Authentication

3 Low level design

Low level design for GRIDS SDK is a technical content deliverable, containing instructions about how to effectively use and integrate with GRIDS. It's a concise reference manual containing all the information required to work with the SDK, with details about the functions, classes, return types, arguments and more, supported by tutorials and examples.

3.1 GRIDSClientManager

Client applications must be registered with OpenID Connect Dynamic Client Registration, which extends OAuth 2.0 Dynamic Client Registration Protocol and OAuth 2.0 Dynamic Client Registration Management Protocol before they can send authorisation requests to it. GRIDSClientManager wraps this logic and helps Data Consumer to dynamically register itself with Data Consumer Connector.

3.1.1 Constructors

GRIDS Client Manager (URI clients Endpoint) - Initializes a new instance of the GRIDSClientManager

3.1.1.1 Parameters

Parameter	Type	Description	Required
clientsEndpoint	URI	OpenID provider client registration endpoint	true

3.1.2 Properties

Property	Type	Description
clientsEndpoint	URI	OpenID provider client registration endpoint

3.1.3 Methods

Method	Type	Description
registerClient (clientMetadata, masterToken)	OIDCClientInformation	Register's client

3.2 GRIDSIssuer

GRIDSIssuer encapsulates a discovered OpenID Connect Provider and its metadata. Provides the methods for getting an authorization URL, consuming callbacks, triggering token endpoint and getting userInfo from BAA and DP.

3.2.1 Constructors

GRIDSIssuer (URI, String, String, URI) - Initializes a new instance of the GRIDSIssuer

3.2.1.1 Parameters

Parameter	Type	Description	Required
endpointURI	URI	OpenID provider metadata URL	true
clientId	String	The client identifier issued to the client during the registration process	true
clientSecret	String	The client secret	true
callbackURI	URI	Client's redirection endpoint previously established with the authorization server during the client registration process	true

3.2.2 Properties

Property	Type	Description
endpointURI	URI	OpenID provider metadata URL
clientId	String	The client identifier issued to the client during the registration process
clientSecret	String	The client secret
callbackURI	URI	Client's redirection endpoint previously established with the authorization server during the client registration process

3.2.3 Methods

Method	Type	Description
getOPMetadata()	OIDCProviderMetadata	Requests OPMetadata from DCC
getAuthorizationUrl(OIDCClaimsRequest claims)	String	Returns the complete URI representation for DC to redirect in order to start the authorization
requestToken(String callbackResponse)	OIDCTokens	Requests Token from DCC. Code etc will be parsed from url. Url ex. https://dc.com/callback?state=12345G8&code=aaaabbbbbbcccccddd
getUserInfo(token)	UserInfo	Requests userinfo from DCC
getDPUserInfo(userInfoURI,token)	UserInfo	Requests userinfo from DP

A description and reference of each type used can be found on chapter 3.4.

3.3 GRIDS SDK flow

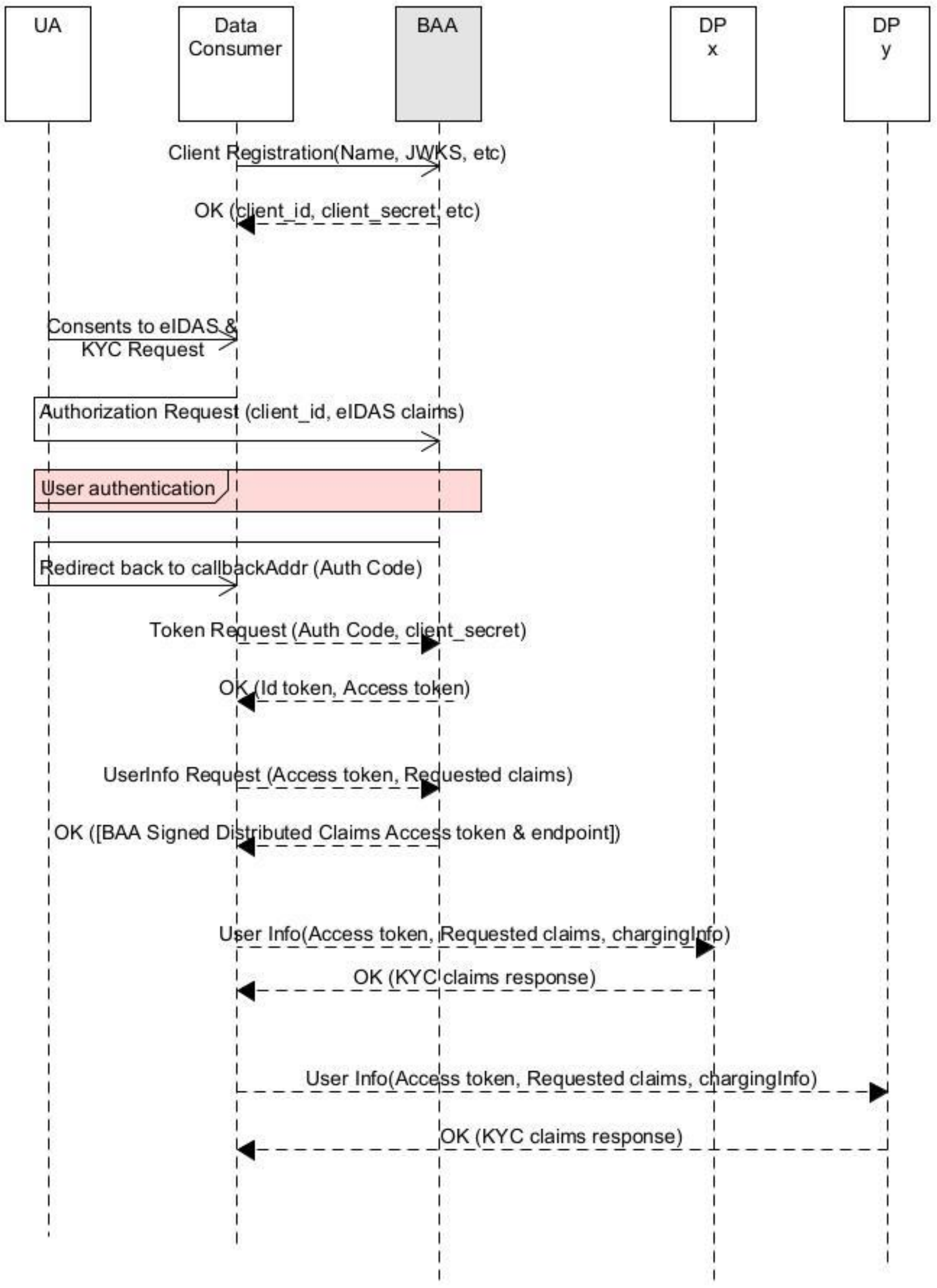
UA
Data Consumer
BAA
DP x
DP y
Client Registration(Name, JWKS, etc)
OK (client_id, client_secret, etc)
Consents to eIDAS & KYC Request
Authorization Request (client_id, eIDAS claims)
User authentication
Redirect back to callbackAddr (Auth Code)
Token Request (Auth Code, client_secret)
OK (Id token, Access token)
UserInfo Request (Access token, Requested claims)
OK ([BAA Signed Distributed Claims Access token & endpoint])
User Info(Access token, Requested claims, chargingInfo)
OK (KYC claims response)
User Info(Access token, Requested claims, chargingInfo)
OK (KYC claims response)

Figure 2: SDK flow diagram

The flow in the above sequence diagram is described in detail below:

1) Data Consumer uses **GRIDSClientManager** to register a new client with DCC and receives a client id and a client_secret. That can be achievent by calling **registerClient** and passing all required parameters, example: name, DC JWKS enpoint, callback url, etc.
   A Client Registration example can be found below:

```
URI clientURL = new URI("https://dcc.grids.eu/clients ");
GRIDSClientManager gridsClientManager = new GRIDSClientManager(clientURL);

ClientMetadata clientMetadata = new ClientMetadata();
clientMetadata.setRedirectionURI(URI.create("https://example.com/cb"));
clientMetadata.setJWKSetURI(URI.create("https://example.com/jwks"));
clientMetadata.setName("My Client App");

OIDCClientInformation clientInfo =
gridsClientManager.registerClient(clientMetadata, "master access token");

String clientID = clientInfo.getID().getValue();
```

Figure 3: Client registration example

2) User Consents and wants to login or register with GRIDS.
3) Data Consumer Initializes a GRIDSIssuer using the GRIDS enpointUrl, client_id, client_secret and callbackUrl. After initialization DC can request BAA configuration using getOPMetadata method in order to retrieve all required metadata in order to continue.
   A GRIDS Issuer and OP Metadata example can be found below:

```
URI wellKnownURI = new URI("https://dcc.grids.eu/.well-known/openid-configuration
");
URI callbackURI = new URI("https://dc.example.com/callback");

GRIDSIssuer gridsIssuer = new GRIDSIssuer(wellKnownURI, "client_id",
"client_secret", callbackURI);
```

Figure 4: GRIDS Issuer example

```
OIDCProviderMetadata opMetadata = gridsIssuer. getOPMetadata();

//Read metadata example
URI issuer = metadata.getIssuer(); // ex "https://DCC.example.com"
URI authorizationURI=op.getAuthorizationEndpointURI()//
https://BAA.example.com/connect/authorize
URI tokenURI = op.getTokenEndpointURI() // https://DCC.example.com/connect/token
List<String> st =  metadata.getSubjectTypes() // ["public", "pairwise"]
URI jwks = metadata.getJWKSetURI() // https://BAA.example.com/jwks.json
Boolean svc = metadata.supportsVerifiedClaims() // true
List<String> itf = metadata.getIdentityTrustFrameworks() //
["eidas_ial_substantial", "eidas_ial_high"]
List<String> idt = metadata.getIdentityDocumentTypes() // ["idcard", "passport"]
List<String> ivm = metadata.getIdentityVerificationMethods() // ["pipp", "sripp",
"eid"]
List<String> vcs = metadata.getVerifiedClaims() // ["given_name", "family_name",
```

Figure 5: OP Metadata example

4) DC will call **getAuthorizationUrl** passing all the request claims as a parameter. Doing so, a url will be returned. Using this url DC will redirect Users browser in order for the user to complete register or login.

Document name:	D4.3 APIs Development and Deployment				**Page:**	14 of 18
Reference:	D4.3		**Version:**	1.0	**Status:**	Final

A Get Authorization Url example can be found below:

```
JSONObject verification = new JSONObject();
verification.put("trust_framework", null);

OIDCClaimsRequest claimsRequest = new OIDCClaimsRequest()
    .withUserInfoVerifiedClaimsRequest(
    new VerifiedClaimsSetRequest()
    .withVerificationJSONObject(verification)
    .add("given_name")
    .add("family_name")
    .add("address")

URI request = gridsIssuer. getAuthorizationUrl (claimsRequest);

//redirect users browser
```

Figure 6: Get Authorization Url example

5) After user succefully logins or registers, DCs callbackUrl will be triggered. Passing that into **GRIDSIssuer requestToken** will request from DCC and return an Access token.

A Request Token example can be found below:

```
// The request url from OIDC Provider to Data Consumer
// https://dc.com/callback?state=12345G8&code=aaaabbbbbbcccccccddd
OIDCTokens tokens = gridsIssuer. requestToken (requestUrl);

JWT idToken = tokens.getIDToken();
AccessToken accessToken = tokens .getAccessToken();
RefreshToken refreshToken = tokens.getRefreshToken();
```

Figure 7: Request Token example

6) Passing that Access Token into **getUserInfo** will request from DCC and return users info including a set of distributed claims.

A Get User Info example can be found below:

```
UserInfo userInfo = gridsIssuer.getUserInfo(token);
String subject = userInfo.getSubject().getValue();

Set<DistributedClaims> set = userInfo.getDistributedClaims();
for (DistributedClaims c : set) {
    String claimName = c.getNames();
    URI dpEndpoing = c.getSourceEndpoint();
    String token = c.getAccessToken().getValue();
}
```

Figure 8: Get User Info example

7) Using those claims, DC can user claim url and claim access token to call **getDPUserInfo**. Doing so GRIDSIssuer will make an API call to the specified DP in order to receive the verified claims.

A Get DP User Info example can be found below:

```
List<String> claimNamesToGet = Arrays.asList("given_name", "family_name",
"birthdate");
Set<DistributedClaims> set = userInfo.getDistributedClaims();
for (String claimName : claimNamesToGet) {
    for (DistributedClaims c : set) {
        Set<String> claimNames = c.getNames();

        if (claimNames.contains(claimName)) {
            URI dpEndpoing = c.getSourceEndpoint();
            String token = c.getAccessToken().getValue();
            UserInfo userInfo = gridsIssuer.getDPUserInfo(token);

            VerifiedClaimsSet verifiedClaims = userInfo.getVerifiedClaims().get(0);
            PersonClaims verifiedPersonClaims = verifiedClaims.getClaimsSet();
            String givenName = verifiedPersonClaims.getGivenName(); // Max
            String familynName = verifiedPersonClaims.getFamilyName(); // Meier
            String birthDay = verifiedPersonClaims.getBirthdate(); // 1956-01-28
        }
    }
}
```

Figure 9: Get DP User Info example

3.4 External reference

External references are thrird packages that are being used for the develepoment of GRIDS SDK. All the following references are present in order to have a complete picture in the capabilities of the SDK. The following reference are opensource and subject to Apache Licence 2[2].

3.4.1 ClientInformation

ClientInformation[3] encapsulates the registration and metadata details of an OAuth 2.0 client:

- The client identifier.
- The client metadata.
- The optional client secret for a confidential client.
- The optional registration URI and access token if dynamic client registration is permitted.

3.4.2 OIDCProviderMetadata

OIDCProviderMetadata[4] OpenID Provider (OP) metadata.

[2] https://www.apache.org/licenses/LICENSE-2.0

[3] https://www.javadoc.io/doc/com.nimbusds/oauth2-oidc-sdk/latest/com/nimbusds/oauth2/sdk/client/ClientInformation.html

[4] https://www.javadoc.io/doc/com.nimbusds/oauth2-oidc-sdk/latest/com/nimbusds/openid/connect/sdk/op/OIDCProviderMetadata.html

3.4.3 OIDCTokens

OIDCTokens[5] includes the ID token, access token and optional the refresh token.

3.4.4 OIDCClaimsRequest

OIDCClaimsRequest[6] specifies individual OpenID claims to return from the UserInfo endpoint and / or in the ID Token.

3.4.5 UserInfo

UserInfo [7]claims set, serialisable to a JSON object. Supports normal, aggregated and distributed claims.

[5]https://www.javadoc.io/doc/com.nimbusds/oauth2-oidc-sdk/latest/com/nimbusds/openid/connect/sdk/token/OIDCTokens.html
[6]https://www.javadoc.io/static/com.nimbusds/oauth2-oidc-sdk/9.4/com/nimbusds/openid/connect/sdk/OIDCClaimsRequest.html
[7]https://www.javadoc.io/doc/com.nimbusds/oauth2-oidc-sdk/latest/com/nimbusds/openid/connect/sdk/claims/UserInfo.html

4 Conclusions

The purpose of this document to provide the main outcomes of the connection between end-users and GRIDS service offerings through standard, publicly available and GDPR compliant APIs. The aforementioned deployment of the SDK was successfully connected to the third-party DCs to DP also on eIDAS node. This SDK Generated to brige all third party DCs with a generic and solid workflow on Grids And eIDAS node. At this time the SDK is based on JAVA framework is common for more of the enterprise companies. Based on the analysis conveyed, this research aims to communicate and generate all authorization and authentication OpenId connect requests from DC to DP. Modeling json response, verified claims and use valid access token between transactions to get extra metadata user Information.