



Common Infrastructure for National Cohorts in Europe, Canada, and Africa - CINECA -

Deliverable D2.2 - A report on how the GA4GH researcher ID specifications have been implemented in the ELIXIR AAI

Work Package:	WP2 - Interoperable Authentication and Authorisation Infrastructure
Lead Beneficiary:	European Molecular Biology Laboratory
WP Leader(s):	Slavek Licehammer (MU), Mikael Linden (CSC)
Contributing Partner(s):	MU, CSC, SickKids, ClinicaGeno, EMBL-EBI, UCT
Contractual Delivery Date:	December 31, 2021
Actual Delivery Date:	December 16th 2021
Authors of this Deliverable:	Mikael Linden (CSC), Martin Kuba (MU), Jorge Izquierdo Ciges (EBI), Mamana Mbiyavanga (UCT)
Reviewed by:	Tommi Nyrönen (CSC), Dominik F. Bucik (MU)
Approved by:	Thomas Keane (EMBL-EBI)
Dissemination Level:	Public
Type of Deliverable:	Report
Grant agreement:	No. 825775 Horizon 2020 (H2020-SC1-BHC-2018-2020)
Type of action:	RIA
Start Date:	1 Jan 2019
Duration:	48 months

Table of contents:

1. Executive Summary	3
2. Project objectives	3
3. Detailed report on the deliverable	3
3.1 Background	3
3.1.1. Passport and AAI specifications	3
3.1.3. Passport visas	4
3.1.4. Passport in AAI protocol flow	7
3.2 Description of Work	11
3.2.1. Overview of ELIXIR AAI	11
3.2.2. Overview of Passports in ELIXIR AAI	13
3.2.3. Visas for controlled access in ELIXIR AAI	14
3.2.4. Visas for registered access in ELIXIR AAI	15
3.2.5. Implementation of Passport broker in ELIXIR AAI	17
3.2.6. ELIXIR AAI Passport clearinghouses	18
3.3 Next steps	19
3.3.1. Passport specification update	19
3.3.2. Self-Sovereign identity (SSI)	20
3.3.2.1. Limitations of the current approach	21
3.3.2.2. Self-Sovereign Identity (SSI) approach	22
3.3.2.3. Adopting SSI to deliver GA4GH Passports	23
3.3.2.4. Proof of concept done within the project	24
4. References	25
5. Abbreviations	26
6. Delivery and schedule	26
7. Adjustments made	27
Appendix 1: EGA's Permissions API specification	28
Appendix 2: Instructions for the SSI test service	39



1. Executive Summary

GA4GH Passport (a.k.a. GA4GH Researcher ID) is the GA4GH standard for expressing an authenticated researcher's roles and data access permissions (a.k.a. passport visas). Together with the GA4GH Authentication and Authorisation Infrastructure (AAI) specification, it describes how passport visas are issued and delivered from their authority (such as a Data Access Committee) to the environment where the data access takes place (such as an analysis platform or cloud). The work package has contributed to the Passport and AAI standards which were approved by the GA4GH in October 2019.

This deliverable provides an overview of the GA4GH Passport and AAI standards (version 1.0) in general. It then describes in detail how ELIXIR AAI has implemented the specification as a Passport broker service. Some directions of the next version of the GA4GH Passport and AAI standards, which are still in development in the GA4GH DURi workstream, are displayed. Finally, a hands-on experiment presents how passports could alternatively leverage self-sovereign identity, an emerging identity and access management paradigm in the industry.

2. Project objectives

The objectives of WP2 are:

- 1) To ensure Authentication and Authorisation Infrastructure (AAI) is interoperable between European, Canadian, and African cohorts.
- 2) To provide AAI services to enable federated cohort data discovery (WP1), analysis (WP4), and clinical applications (WP5).
- 3) To upgrade ELIXIR and Canada AAI to support GA4GH protocols to ensure CINECA interoperability with other global cohort infrastructures.
- 4) To provide assistance and expertise to enable organisations in Europe, Canada, and Africa to access AAI services.

With this deliverable, the project has contributed to objectives 2) and 3).

3. Detailed report on the deliverable

3.1 Background

The Global Alliance for Genomics and Health (GA4GH) is a policy-framing and technical standards-setting organisation, seeking to enable responsible genomic data sharing within a human rights framework. The work in GA4GH is organised into eight work streams covering different aspects of genomic data sharing. Researcher identity and data access authorisation aspects belong to the Data Use & Researcher Identities (DURI) work stream. In October 2019, the DURI work stream delivered the Passport version 1.0 specification as the basis for expressing researcher IDs in GA4GH.

This section introduces the Passport specification [GA4G19] and the related AAI [GA4G19b] specification. For more information, refer to a recent paper on passports in Cell Genomics [VOIS21] and a GA4GH webinar on passports [GA4G20].



3.1.1. Passport and AAI specifications

A GA4GH Passport is a collection of visas describing their holder's role (e.g. researcher), affiliation (with their employing organisation, commonly referred to as their home organisation), and data access permissions. Passports align with the GA4GH's 3-tier access model, where data and services are either **public**, **restricted** for bona fide researchers and clinical care professionals only or implement a **controlled** access policy, requiring a researcher to present a data access request (DAR) that is reviewed and approved by a Data Access Committee (DAC) representing the original data collector [VOIS21]. The visas in a passport are issued and digitally signed by their authoritative sources, such as DACs (or the data archive systems representing them) for controlled access visas and home organisations for researcher status visas. Passport brokers (such as ELIXIR AAI) authenticate the researcher when they log in, pull their visas from the various issuers, assemble and present them as a passport to the downstream relying services, dubbed as Passport clearinghouses.

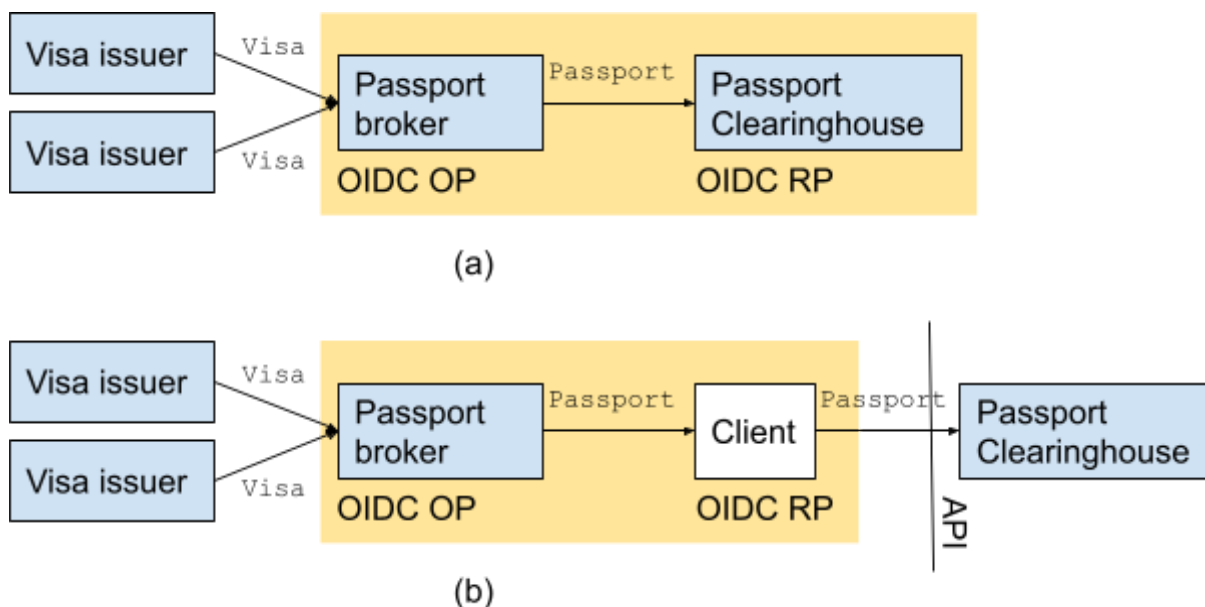


Figure 1. Passports are either (a) used directly by the relying party or (b) presented to a downstream server in an API call. The scope of the Passport specification is highlighted by blue rectangles. The scope of the AAI profile is marked with a yellow rectangle.

While the Passport standard [GA4GH19] defines the presentation (syntax) of the visas and a passport as their container, the interaction between a Passport broker and a Passport clearinghouse is defined in the GA4GH Authentication and Authorisation Infrastructure (AAI) standard [GA4G19b] developed by the Data Security work stream. The AAI relies on an OpenID Connect (OIDC) protocol [SAKI14] profile where the Passport broker implements an OpenID Connect provider (OP) responsible for authenticating a user and delivering their visas (Figure 1). The Passport clearinghouse implements an OpenID Connect Relying Party (RP), who receives the passport from the broker and enforces access locally (Figure 1(a)), or uses the passport to authorise an API call. In the latter case, the server

implementing the API is the Passport clearinghouse that enforces access based on the visas (Figure 1(b)).

3.1.3. Passport visas

Table 1 introduces the five visa types defined by the GA4GH Passport ver 1.0 standard. The visa types are further described below.

Table 1. Passport visas defined in GA4GH Passport ver 1.0 standard

Passport Visa type	Visa type and value description
ControlledAccessGrants	A dataset or other object for which controlled access has been granted to this Passport Visa Identity. - e.g. https://ega-archive.org/datasets/EGAD00001006673
ResearcherStatus	The person has been acknowledged to be a researcher of a particular type or standard. - e.g. https://doi.org/10.1038/s41431-018-0219-y (bona fide researcher for registered access)
AcceptedTermsAndPolicies	The Passport Visa Identity or the "source" organization has acknowledged the specific terms, policies, and conditions - e.g. https://doi.org/10.1038/s41431-018-0219-y (registered access attestations)
AffiliationAndRole	The Passport Visa Identity's role within the identity's affiliated institution - e.g. faculty@cam.ac.uk - standard (eduPersonAffiliation) - e.g. nih.researcher@med.stanford.edu - proprietary (dot ".")
LinkedIdentities	The identity as indicated by the {"sub", "iss"} pair (aka "Passport Visa Identity") of the Passport Visa is the same as the identity or identities listed in the "value" field. - e.g. " mikael@elixir-europe.org equals to mlinden@csc.fi "

The visa type serving controlled access is ControlledAccessGrants visa. Its value identifies the data set to which it permits access. For instance, a value of <https://ega-archive.org/datasets/EGAD00001006673> permits the visa subject to access CINECA synthetic cohort EUROPE UK1 (identified by the Dataset ID EGAD00001006673) in the European Genome-phenome Archive EGA (see Table 3 below for an example visa).

The visas relevant for registered access are ResearcherStatus and AcceptedTermsAndPolicies visas. The ResearcherStatus indicates that the visa subject has been acknowledged as a researcher according to a particular definition. The exact definition (such as [DYKE18]) is then referenced by the visa value that is supposed to be a URL resolving to the definition. It is further assumed that, in order to qualify for registered access, the researcher (or their home organisation) needs to commit to certain terms, policies and conditions, such as "I will refrain from trying to re-identify individuals from



the samples I will get access to". Those terms, commonly known as the attestations for registered access, are identified by the value of the AcceptedTermsAndPolicies visa.

The AffiliationAndRole visa describes its subject's affiliation at their home organisation, such as a faculty member status. That value can be further mapped to a ResearcherStatus visa. Finally, the LinkedIdentities visa indicates that the visa subject is confirmed to be using different identifiers in different services (such as archives of controlled access data). Being able to confirm that the different identifiers represent the same person is necessary for demonstrating the integrity of a passport. Omission of the LinkedIdentities visa would leave room for a doubt that, due to a malfunction or a security flaw, visas in a passport might belong to several distinct persons.

Table 2 describes the fields in a visa besides *Type* and *Value*, which were already described in Table 1. The *source* identifies the body that asserted the visa and *by* describes its type; for instance, a ControlledAccessGrants visa is normally asserted by a DAC and a person's affiliation with an organisation by an authorised representative of the organisation (often dubbed as a signing official, or SO). Some visas, such as a ResearcherStatus visa, may be asserted also by a peer researcher [DYKE18]. Attestations for registered access are typically made by the researcher themselves [DYKE18].

Table 2. Visa object fields. An asterisk (*) indicates a mandatory field.

Visa field	Description
type *	See Table 1.
value *	URI Value of the claim <ul style="list-style-type: none"> - e.g. https://doi.org/10.1038/s41431-018-0219-y for the registered access attestation, as per S.O.M. Dyke et al - e.g. https://ega-archive.org/datasets/EGAD00001006673 for a controlled access grant
source *	A URL that provides at minimum the organization that made the assertion <ul style="list-style-type: none"> - e.g. https://ega-archive.org/dacs/EGAC00001000908 for an EGA DAC - e.g. https://grid.ac/institutes/grid.225360.0 for EMBL-EBI
by	The level or type of authority within the "source" organization of the assertion <ul style="list-style-type: none"> - vocabulary: self, peer, system, so, dac
asserted *	Seconds since unix epoch that represents when the Passport Visa Assertion Source made the claim <ul style="list-style-type: none"> - e.g. when did the DAC approve the data access application - not the same as iat (issued at) which indicates when the JWT was minted
conditions	Indicates that the Passport Visa is only valid if the clauses of the conditions match to capture external dependencies <ul style="list-style-type: none"> - e.g. "person still has affiliation faculty@helsinki.fi"



Asserted is a timestamp that indicates when the visa was asserted by the source. For a *ControlledAccessGrants* visa, it is supposed to be the moment the DAC approved the DAR. For *ResearcherStatus*, it indicates when that status was last confirmed, etc. Finally, the *conditions* field enables the visa source to attach extra conditions that the visa subject needs to satisfy. For instance, the visa is valid only as long as the person is affiliated with the home organisation that signed the data access agreement pertaining to the controlled access datasets.

Table 3 exposes a GA4GH Passport visa as a Json web token (JWT, [RFC7519]) as observed “on-the-wire”, with the base64 encoding removed and the digital signature and related key information stripped. The visa fields introduced in Table 2 can be observed in the *ga4gh_visa_v1* claim and are followed by standard JWT claims [RFC7519]. In brief, *sub* identifies the visa subject (i.e. the researcher), *iss* identifies the visa issuer (i.e. the visa signer), *iat* is the timestamp when the visa was issued and *exp* indicates when the visa will expire. *jti* uniquely identifies the JWT itself.

Table 3. Example visa (decoded, digital signature and related key information stripped)

```
{
  "ga4gh_visa_v1": {
    "type": "ControlledAccessGrants",
    "asserted": 1623936445,
    "value": "https://ega-archive.org/datasets/EGAD00001006673",
    "source": "https://ega-archive.org/dacs/EGAC00001000908",
    "by": "dac"
  },
  "sub": "EGAW00000019020",
  "iss": "https://ega.ebi.ac.uk:8053/ega-openid-connect-server/",
  "exp": 1628600552,
  "iat": 1628596952,
  "jti": "ec840446-24fc-4388-8efa-5d331a6863b8"
}
```

The example itself is a *ControlledAccessGrants* visa, issued and signed by EGA and describes the *sub*'s permission to CINECA Synthetic cohort EUROPE UK1, as granted by the competent DAC. Decoding the timestamps (expressed as seconds since January 1970) shows that the permission was granted on June 17, 2021 at 1:27:25 PM and the visa issued on August 10, 2021 at 12:02:32 PM. The visa will expire in 3600 seconds, after which a new one needs to be retrieved from EGA.

For a Passport clearinghouse implementer it is important to observe that the analysis workflows may span well beyond the expiration of the visa. The implementation needs to decide its approach for handling such a situation; for instance, if it will request a new visa from its issuer (and terminate the analysis if that request fails) or if the visa is examined only in the beginning and the analysis can finish despite the expired visa.

3.1.4. Passport in AAI protocol flow

GA4GH AAI OIDC Profile [GA4G19b] relies on OpenID Connect protocol [SAKI14] for the reliable delivery of a passport from the Passport broker to the Relying party and eventually the Passport



clearinghouse (Figure 1). How the broker further fetches the visas from the issuers is out of scope for the AAI specification and left to the deployment to decide. This section shortly introduces the AAI OIDC profile, using ELIXIR AAI in the examples.

The most common OIDC protocol flow is the authorisation code flow described in Figure 2. The arrows in the figure are exemplified in the text boxes below.

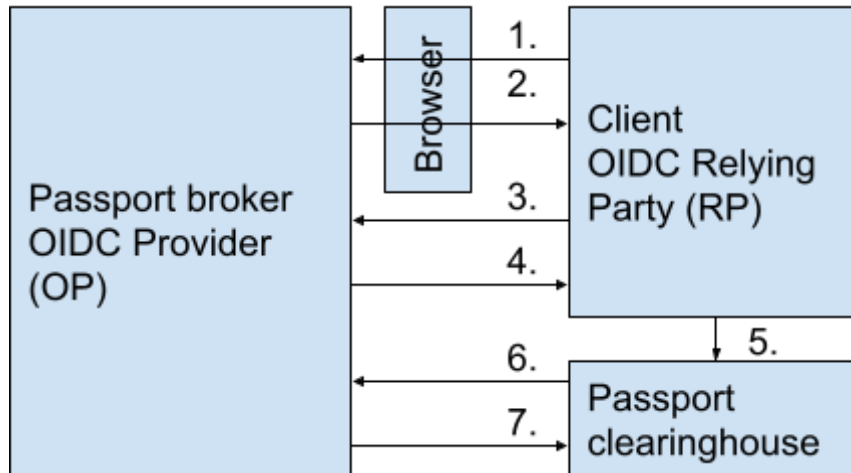


Figure 2. Interaction of a Passport broker and a Passport clearinghouse using the authorisation code flow of OpenID Connect (OIDC) protocol.

It is assumed that the OIDC Relying Party (RP) has been registered to the OIDC Provider (OP) beforehand, and the OP has assigned the RP an identifier (called client-ID) and a symmetric key (called client-secret) for authentication. The flow starts when a user lands on the RP with their web browser and clicks a login button there.

Message 1. Redirect user to OP's /authorize endpoint

```

GET /oidc/authorize
  ?response_type=code
  &scope=openid ga4gh_passport_v1
  &client_id=<your-client-id>
  &state=<state>
  &redirect_uri=https://your.host.org/callback HTTP/1.1
Host: https://login.elixir-czech.org
  
```

After initiating the login procedure triggered by the login button click event, the RP redirects (message 1) the user's web browser to OP's Authorization endpoint (*/authorize*), and in the query string,

- requests the use of authorisation code flow (parameter *response_type*),
- requests a GA4GH Passport over OIDC protocol (parameter *scope* with the value *openid ga4gh_passport_v1*)
- indicates its client-ID (parameter *client_id*)
- provides a parameter that the OP will return to help the RP to keep state (parameter *state*)

- provides the URL to which the OP returns the browser after the user has authenticated (parameter *redirect_uri*). A set of permitted values that can be passed via this parameter (URLs) must be registered to the OP beforehand.

Message 2. OP redirects user back to your registered callback address

```
HTTP/1.1 302 Found
Location: https://your.host.org/callback
        ?code=<authorisation-code>
        &state=<state>
```

In the following step, the OP authenticates the user (potentially using a third-party service, such as the researcher's home organisation's identity provider). After authenticating the user the OP informs them and asks their permission to release their data to the RP, including their Passport. In OIDC terminology, this is called authorization. After the user's approval, the OP redirects (message 2) the user's browser back to the URL indicated via the *redirect_uri* parameter in message 1, together with the *state* parameter with the same value as presented in message 1. The OP includes an *authorisation-code* in the query string parameter.

Message 3. RP sends authorisation-code to OP's /token REST API

```
POST https://login.elixir-czech.org/oidc/token HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Authorization: Basic <your-client-id><client-secret>

grant_type=authorization_code
        &code=<authorisation-code>
        &redirect_uri=https://your.host.org/callback
```

After the user landing on the URL resolved from the *redirect_uri* parameter, the RP collects the *authorisation code* from message 2 and includes it in a POST request made to OP's REST Token endpoint (*/token*). It also uses the *redirect_uri* parameter from messages 1-2 and specifies the *authorisation code grant type* using parameter *grant_type=authorization_code*.

In exchange for the *authorisation code*, the OP returns (message 4) an *access token* to the RP, together with additional descriptions on it, such as its lifetime, type and the information that it is an *access token of Passport type*. The OP also returns another token called an *ID token* that provides additional information about the authenticated user, such as their identifier (*sub*, see Table 3) and the method the user was authenticated by (such as using a password or performing a multi-factor authentication).

Message 4. OP sends user's Access token to RP in response

```
HTTP/1.1 200 OK
Content-Type: application/json
```



```
{
"access_token": "<access-token>",
"expires_in": 3600,
"token_type": "Bearer",
"scope": "openid ga4gh_passport_v1",
"id_token": "<id-token>"
}
```

The OIDC specification leaves the syntax of the access token undefined, implying that it is assumed to be an opaque string. The main extension the GA4GH AAI OIDC profile makes is that the syntax of an access token is specified to be a JWT.

GA4GH Passport-scoped access token (decoded, signature and related key information stripped)

```
{
"iss": "https://login.elixir-czech.org/oidc/",
"sub": "370fa949dd5cac7906dde14164545c64a908ec92@elixir-europe.org",
"aud": "7ceef937-6740-4774-aec3-40fb75b23e30",
"scope": "ga4gh_passport_v1 openid",
"iat": 1628670378,
"exp": 1628706378,
"jti": "881da85d-d38b-43c0-adde-e6a8ae239dec"
}
```

The GA4GH AAI OIDC profile's JWT access token contains

- the broker that has issued the access token (field *iss*)
- the authenticated user's identifier (field *sub*)
- the relying party's client-id (field *aud*)
- information that the access token is a passport (field *scope*)
- timestamps when the access token was issued (field *iat*) and when it expires (field *exp*)
- JWT's unique identifier (field *jti*)

An example of a Passport-scoped access token issued by the ELIXIR AAI is provided in the text box *GA4GH Passport-scoped access token*.

(Optional) Message 5. Attach the Access token to a GA4GH API call

```
GET /query HTTP/1.1
Host: beacon.example.com
Authorization: Bearer <access-token>
```

If the RP wants to use the Passport-scoped access token to authorise the call of an API (such as a GA4GH API), it must attach the access token to the API call (message 5). In the classic approach, that is done by placing the access token to the Authorization header of the REST API request. If the access token is not used to authorise an API call but consumed by the RP itself (Figure 1(a)), this step can be omitted.



3.2 Description of Work

ELIXIR AAI's Passport implementation was one of the two independent implementations required for the specification approval in GA4GH in October 2019. This section describes how the GA4GH Passport and AAI specifications are implemented in ELIXIR AAI.

3.2.1. Overview of ELIXIR AAI

ELIXIR AAI [LIND18] is an ELIXIR Compute Platform service for authenticating researchers and helping the relying services to decide their permissions. It was developed in the ELIXIR EXCELERATE project and launched in November 2016. ELIXIR AAI is operated by the Czech and Finnish ELIXIR nodes. By the end of 2020, ELIXIR AAI had 102 relying services and 5863 active end users.

The operations of ELIXIR AAI receives infrastructure service funding from ELIXIR, but further development and new features are funded via external grants. ELIXIR AAI's support of GA4GH Passports was initially developed in the CINECA project in 2019.

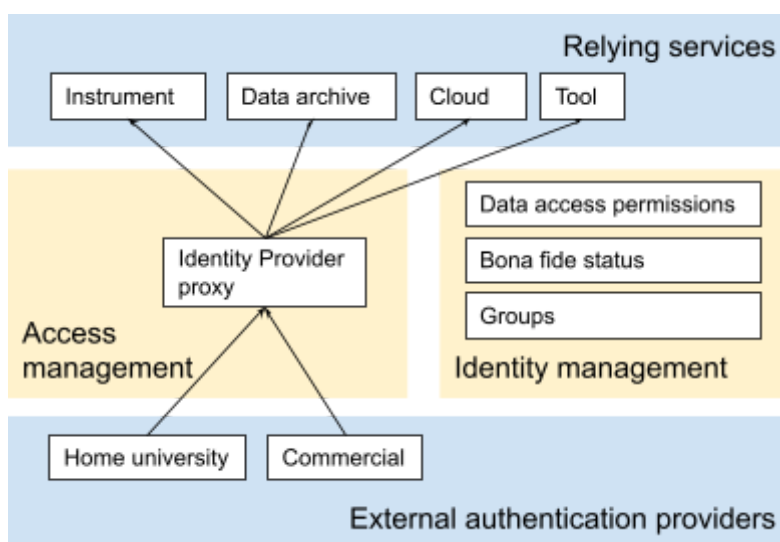


Figure 3. High level overview of ELIXIR AAI (in yellow) and its key integrations.

Figure 3 presents a high-level overview of the ELIXIR AAI. ELIXIR AAI does not deliver primary authentication credentials (such as passwords) to the end users but expects them to use an external authentication provider for authentication. Researchers are encouraged to use their home organization identity provider (via the eduGAIN interoperation service¹) which can deliver up-to-date information about the user's role and affiliation (c.f. *AffiliationAndRole* visa, section 3.1.3). If the home organization authentication is not possible, a commercial (such as Google or LinkedIn) or community services (such as ORCID) are supported.

ELIXIR AAI assigns a unique, non-reusable identifier to the user (which then becomes the value of the *sub* claim, see section 3.1.3) and decorates the users with extra claims it manages (the identity management component in Figure 3). After authentication, the Identity Provider proxy (the access

¹ www.edugain.org



management component in Figure 3) collects the user's claims and presents them to the relying services. Typical relying services (see Figure 3) are those that generate data (instruments such as microscopes or sequencers), store data (data archives and repositories) and perform computation or processing on data (clouds and compute clusters). ELIXIR AAI also supports various collaborative tools, such as wikis and mailing list servers.

For cross-Atlantic interoperability in the CINECA project, ELIXIR AAI has been integrated with the Canadian Distributed Genomics (CanDIG) AAI system. Reflected to Figure 3, the CanDIG AAI would appear

- as another External Authentication provider in the lower part of the figure to let Canadian researchers log into the ELIXIR AAI and its relying services; and
- as another Relying service in the upper part of the figure to let ELIXIR researchers log into the CanDIG AAI and its relying services.

For more information on ELIXIR /CanDIG AAI interoperability see deliverable D2.1 [PROC20].

3.2.2. Overview of Passports in ELIXIR AAI

Figure 4 extends Figure 3 by highlighting the system components that contribute to the implementation of GA4GH Passports in ELIXIR AAI. The Identity Provider proxy of ELIXIR AAI implements the Passport broker functionality which is invoked when a relying party of ELIXIR AAI requests the *ga4gh_passport_v1* scope. Such a request activates a module in ELIXIR AAI's OIDC Provider that assembles a JWT access token for the relying party and delivers the visas from its *Userinfo (/userinfo)* endpoint as described in section 3.1.4.

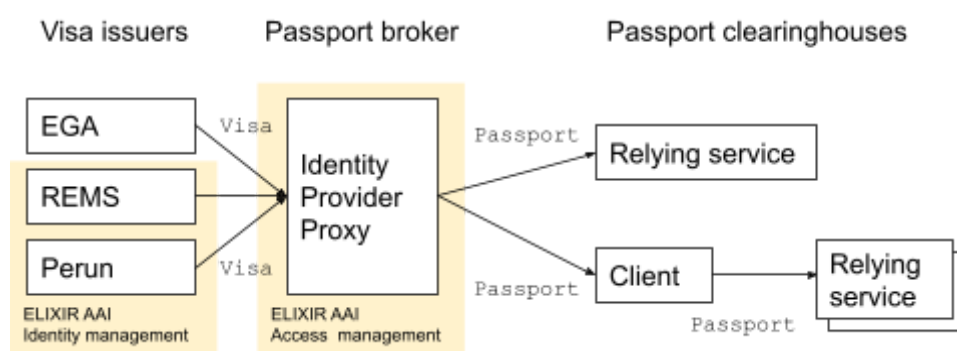


Figure 4. Overview of Passports in ELIXIR AAI using ELIXIR AAI system component names (in yellow background).

Table 3 introduces the visas currently supported in ELIXIR AAI and their issuer and source. An ELIXIR user can at any time log into a dedicated service² that presents their GA4GH Passport as a human-readable web page.

Table 3. Visa types currently supported by ELIXIR AAI, their issuers and sources. See Table 1 for more information on the visa types.

Visa type	Access tier	Visa issuer	Visa assertion source
-----------	-------------	-------------	-----------------------

² <https://echo.aai.elixir-czech.org/>

ControlledAccessGrants	Controlled access	EGA REMS	DAC
ResearcherStatus	Registered access	ELIXIR AAI	Home organisation Perun
AcceptedTermsAndPolicies	Registered access	ELIXIR AAI	Perun
AffiliationAndRole		ELIXIR AAI	Home organisation
LinkedIdentities		ELIXIR AAI EGA REMS	Visa issuer

The key system components in identity management tasks performed in the ELIXIR AAI are Perun³, used for the user and identity management-related activities, and REMS⁴, used for managing users' data access permissions. The ControlledAccessGrants visas were, at the time of writing, issued by the REMS, except for visas that describe permissions to datasets deposited in EGA. These are issued by EGA as described in depth in section 3.2.3.

ELIXIR AAI assumes an approach proposed by Dyke et al. [DYKE18] for registered access. The visas are issued by the ELIXIR AAI Identity Provider proxy, based on the user information it receives from the Perun identity management system. In particular, it is based on a person's group memberships registered in the system (see section 3.2.4 for detail). If a person has a "faculty" status in an AffiliationAndRole visa, it is reflected in the ResearcherStatus visa.

AffiliationAndRole visa carries the role value retrieved from the user's home organisation when they logged in using their home organisation authentication for the last time. It is believed that the home organisation has the most up-to-date and accurate information on their affiliated users' institutional roles, using the defined vocabulary of *faculty*, *student* and *member*. LinkedIdentities visa can be issued by any assertion source who assigns a (new) identifier to the user.

3.2.3. Visas for controlled access in ELIXIR AAI

Since the beginning, ELIXIR AAI has worked closely with EGA to serve the needs of the ELIXIR human data communities. EGA, the European Genome-phenome Archive, has been seen as a natural integration point for ELIXIR AAI, enabling it to deliver a researcher's data access permissions to the computing environments for access control enforcement purposes.

In August 2021, EGA had 7342 datasets⁵ deposited and managed by 1429 DAC⁶s. After approving a researcher's DAR, the DAC grants them a permission to access the dataset. EGA then stores this permission, together with the researcher's EGA ID. In the CINECA project, the EGA has implemented a

³ <https://github.com/CESNET/perun>

⁴ <https://github.com/CSCfi/remis/>

⁵ <https://ega-archive.org/datasets>

⁶ <https://ega-archive.org/dacs>



Permission API (Appendix 1) that ELIXIR AAI uses to query an authenticated ELIXIR ID holder's data access permissions in EGA. EGA responds to the queries by issuing a ControlledAccessGrant visa (see Table 3 for an example) and delivering it to ELIXIR AAI that relays it to the relying party as a GA4GH Passport, together with their other visas (see message 7 in section 3.1.4).

EGA identifies users with an EGA ID, while ELIXIR AAI uses an ELIXIR identifier, instead. To be able to deliver appropriate ControlledAccessGrant visas to ELIXIR AAI, the end user needs to link the two identifiers. This is done by logging into a dedicated web page managed by EGA using their EGA ID, and then subsequently log in using their ELIXIR ID. This enables EGA to associate the EGA ID holder's validated ELIXIR ID with their record in EGA and issue LinkedIdentities visas together with the ControlledAccessGrants visas, as described in section 3.1.3.

To facilitate visa freshness, ELIXIR AAI fetches fresh visas from EGA's Permission API any time a client presents a valid access token to ELIXIR broker's /userinfo endpoint. To counter denial of service attacks, the visas are cached in ELIXIR AAI for a short period of time.

Traditionally, a researcher's interaction with a DAC is predominantly manual and involves sending an application with related attachments in e-mail or postal mail. To streamline and formalise the application process the Finnish ELIXIR node has implemented the REMS [LIND13]. It is an electronic application workflow tool, which can be used by a researcher for applying for access to an identified resource (in particular, a dataset) and by a DAC to review and perform decisions on the application. Based on an approved DAR, REMS issues a ControlledAccessGrants visa that ELIXIR AAI can attach to an authenticated researcher's passport for delivery to a downstream relying party (Figure 5(a)).

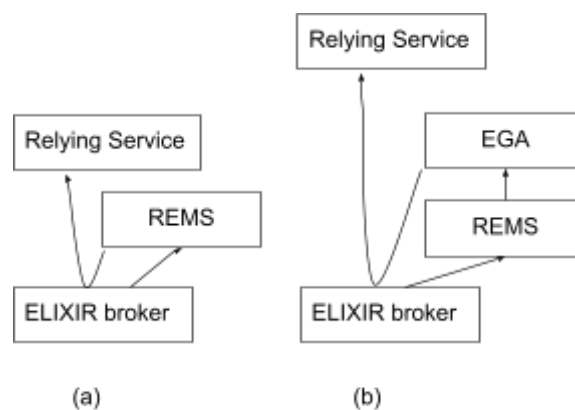


Figure 5. Different approaches to integrate the REMS tool to ELIXIR Passport broker.

In the CINECA project, REMS has been integrated to manage the data access application process for EGA datasets. By policy, EGA is the authoritative source for all data access rights pertaining to datasets deposited at EGA. Due to that, for the EGA integration, REMS does not issue nor deliver the ControlledAccessGrants directly to the ELIXIR broker (Figure 5(a)). Instead, it writes the permissions to EGA (Figure 5(b)), which exposes them to the ELIXIR broker, as described above. This approach enables a DAC to monitor their current granted permissions in a single place.

3.2.4. Visas for registered access in ELIXIR AAI

ELIXIR AAI implements the approach described by Dyke et al. [DYKE18] for registered access. The flow of receiving the visas for registered access is illustrated in Figure 6. Firstly, the researcher needs to demonstrate they are a bona fide researcher, after which ELIXIR AAI issues them a ResearcherStatus visa (see Table 1). To be granted access to data and services in the registered access tier, a researcher needs to make the necessary attestations, yielding them an additional related AccepterTermsAndPolicies visa. When released to a downstream Passport clearinghouse, these two visas together justify access to services in the registered access tier.

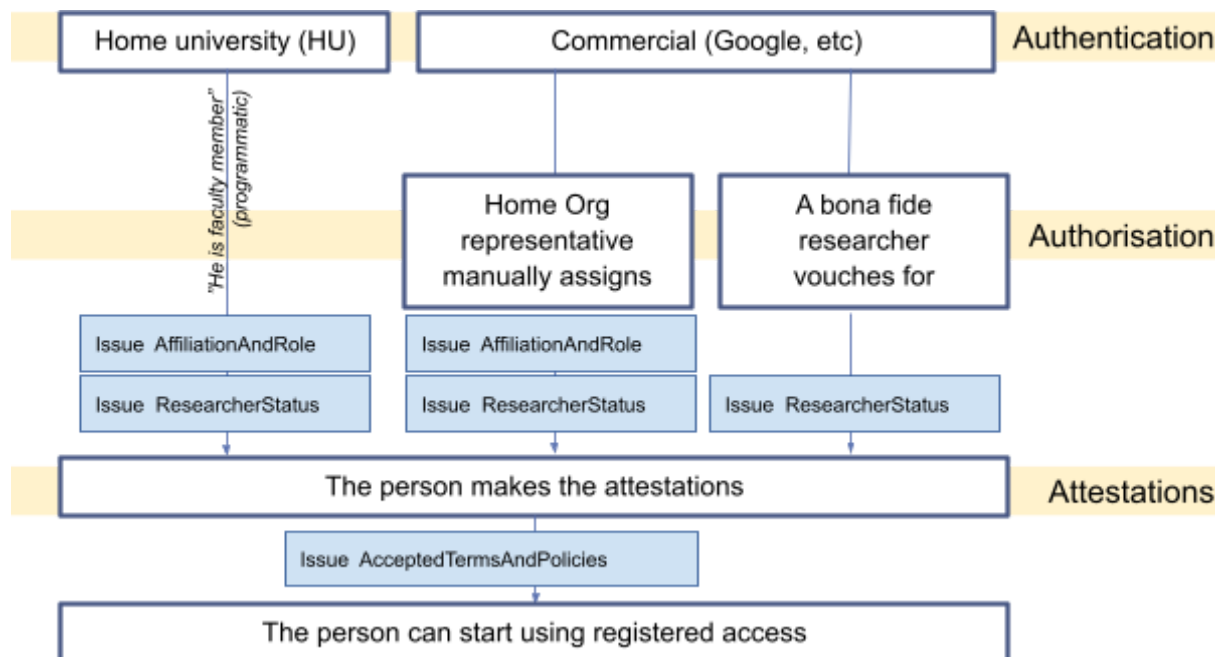


Figure 6. “A Triple-A approach” implementation in ELIXIR AAI. A user needs to use one of the three parallel channels to demonstrate their status as a bona fide researcher and make the attestations for registered access.

A researcher can use three alternative approaches to demonstrate their status as a bona fide researcher. The preferred approach is that they log in to ELIXIR AAI using their home university authentication provider (the left-hand side path in Figure 3) and the home university uses a standard schema⁷ to signal they are a faculty member of the university. This approach is fully automated and requires no manual steps by anyone other than the researcher, who needs to authenticate at their home university. The faculty information is delivered programmatically from the home university identity management system where it is assumed to be fresh and accurate. The role is also populated into the AffiliationAndRole visa, which will expire in 12 months if the user does not authenticate using their home university identity provider before the expiration date..

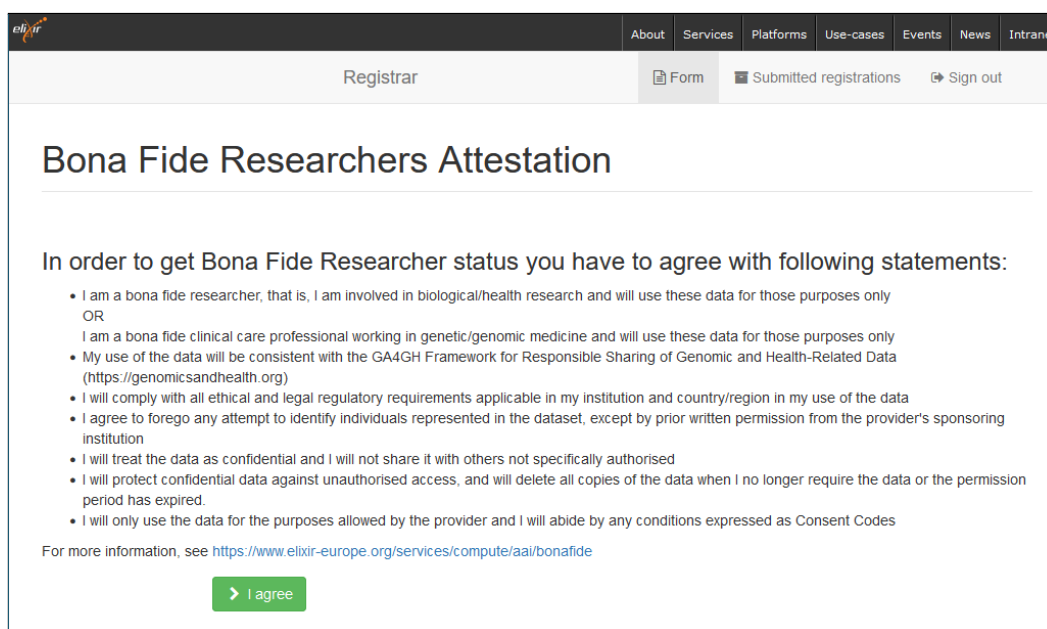
Not all universities support logging into the ELIXIR AAI and programmatically delivering the researcher status from the home university databases. To serve those users, the ELIXIR AAI also

⁷ eduPersonScopedAffiliation attribute of eduPerson schema, <https://refeds.org/eduperson>

supports commercial and community logins (such as Google and ORCID), which in general cannot deliver authentic information on a user's role in a university. For managing the ResearcherStatus of such users, a dedicated representative (sometimes called a signing official, SO) of the university (or other types of organisation, if applicable) can become elevated to a designated role in the ELIXIR AAI. This role permits them to manually assign a researcher affiliation and status to an arbitrary ELIXIR ID holder, using a graphical user interface of ELIXIR AAI (the middle path in Figure 6). Technically, in the ELIXIR AAI's Perun Identity management system, such a person is made a manager of a group called "faculty at <university>". All users, who become members of such a group by invitation from the designated person, are granted the related ResearcherStatus and AffiliationAndRole visas. The visas are valid for 12 months. After this period, the same procedure is necessary to renew them.

To allow maximum flexibility while maintaining sufficient control and audit trail, Dyke et al [DYKE18] propose a third, peer-to-peer approach for asserting a bona fide status, which is also implemented in ELIXIR AAI (the right-hand side path in Figure 6). An ELIXIR ID holder can receive a bona fide status if a peer bona fide researcher vouches for it. The vouching person themselves must have received their bona fide status from their home university (left or middle path in Figure 6).

In the ELIXIR AAI, the vouching is implemented using the REMS system designed for controlled access (see the previous section). For these purposes, a specific resource named "bona fide status" has been created. An ELIXIR ID holder can apply for this resource, and identify a person they think could vouch for their status as a bona fide researcher. The specified person then receives and reviews the application for bona fide status. If they feel confident to approve it and satisfy the requirements for the vouching person, ELIXIR AAI grants the requesting person a ResearcherStatus visa. AffiliationAndRole visa is not issued in the peer-based flow.



The screenshot shows the ELIXIR AAI Registrar interface. At the top, there is a navigation menu with links for About, Services, Platforms, Use-cases, Events, News, and Intranet. Below the menu, the word "Registrar" is displayed, along with buttons for "Form", "Submitted registrations", and "Sign out". The main heading is "Bona Fide Researchers Attestation". Below this, a text block states: "In order to get Bona Fide Researcher status you have to agree with following statements:". This is followed by a list of seven bullet points:

- I am a bona fide researcher, that is, I am involved in biological/health research and will use these data for those purposes only
- OR
- I am a bona fide clinical care professional working in genetic/genomic medicine and will use these data for those purposes only
- My use of the data will be consistent with the GA4GH Framework for Responsible Sharing of Genomic and Health-Related Data (<https://genomicsandhealth.org>)
- I will comply with all ethical and legal regulatory requirements applicable in my institution and country/region in my use of the data
- I agree to forego any attempt to identify individuals represented in the dataset, except by prior written permission from the provider's sponsoring institution
- I will treat the data as confidential and I will not share it with others not specifically authorised
- I will protect confidential data against unauthorised access, and will delete all copies of the data when I no longer require the data or the permission period has expired.
- I will only use the data for the purposes allowed by the provider and I will abide by any conditions expressed as Consent Codes

Below the list, there is a link: "For more information, see <https://www.elixir-europe.org/services/compute/aaibonafide>". At the bottom, there is a green button with a right-pointing arrow and the text "I agree".

Figure 7. The attestations for registered access [DYKE18].

After the ELIXIR ID holder has received the ResearcherStatus visa, they can proceed to making the attestations for registered access. Technically, in the ELIXIR AAI Identity management system, the

person is invited to apply for membership in a dedicated group whose membership application form presents them the attestations (Figure 7). After submitting the form the ELIXIR ID holder automatically joins the group. All group members are issued an AcceptedTermsAndPolicies visa.

3.2.5. Implementation of Passport broker in ELIXIR AAI

The implementation of the OpenID Connect protocol in the ELIXIR AAI is based on a certified open-source reference implementation of OpenID Connect in the Java programming language named MITREid Connect⁸. It is customized to fetch data from the Perun Identity Management system, so it is called Perun MITREid⁹. It extends the original software with the ability to add new scopes and claims, and the possibility to modify the content of issued access tokens. The implementation of the ELIXIR Passport broker is located in the Java class GA4GHClaimSource.java¹⁰ which produces the contents of the *ga4gh_passport_v1* claim defined by the Passport specification (see section 3.1.4).

ELIXIR Passport broker reads ControlledAccessGrants visas for a given user from a configured set of visa issuers, includes the visas in the claim without modification, and adds a LinkedIdentities visa for each issuer that provided at least one visa in order to link the ELIXIR ID of the user with the user ID from the visa issuer.

The other types of visas (namely AffiliationAndRole, AcceptedTermsAndPolicies, and ResearcherStatus) are produced directly by the Java class from the user data read from the Perun Identity management system following the rules described in the previous sections, i.e. based on group membership and user attributes. The produced visas are assembled and digitally signed using the same key as access tokens and id tokens issued by the OpenID Provider, and included into the *ga4gh_passport_v1* claim.

3.2.6. ELIXIR AAI Passport clearinghouses

A majority of relying services of ELIXIR AAI does not relate to human genomics or other sensitive data and do not support GA4GH Passports. In August 2021, 6 production relying services in ELIXIR AAI were subscribing to the *ga4gh_passport_v1* scope (see section 3.1.4) that best indicates they consume passports (either directly or in a downstream API call).

⁸ <https://github.com/mitreid-connect/OpenID-Connect-Java-Spring-Server>

⁹ <https://github.com/CESNET/perun-mitreid>

¹⁰

<https://github.com/CESNET/perun-mitreid/blob/master/oidc-idp/src/main/java/cz/muni/ics/oidc/server/elixir/GA4GHClaimSource.java>



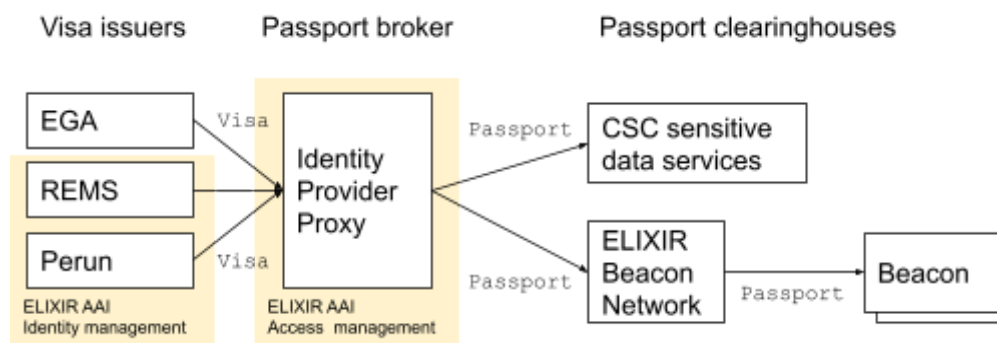


Figure 8. Two example relying services of ELIXIR AAI

Figure 8 exemplifies two ELIXIR AAI's relying services consuming GA4GH Passports. CSC Sensitive data services¹¹ is the sensitive data environment of CSC - the IT Center of Science where researchers can store and publish their sensitive datasets and perform computational tasks on them. If a user logs in using the ELIXIR AAI, it delivers their ControlledAccessGrants visas from EGA, enabling them to access local copies of their permitted EGA datasets in the sensitive data environment.

The other example in Figure 4 represents an ELIXIR Beacon Network¹², a service of ELIXIR that consumes claims for registered access. In the ELIXIR Beacon Network, a bona fide researcher can present extended queries on, e.g., allele frequencies which may reveal some sensitive information on the sample donors. The Beacon Network relays the queries to multiple connected downstream beacons using the GA4GH Beacon API. For the queries' authorisation in the registered access tier, the Beacon network presents the Passport-scoped access token received from the ELIXIR AAI to the downstream API. The beacon then fetches and validates the users' visas from the ELIXIR AAI.

3.3 Next steps

This section sheds some light on the (potential) future steps of the GA4GH Passport and AAI standards and speculates the new approaches the GA4GH could take. In the beginning, this section makes a short overview of the new directions in the preparations of the next Passport standard. Then it introduces a practical experiment on integrating passports to self-sovereign identity, an emerging paradigm in identity management.

3.3.1. Passport specification update

After their approval in GA4GH in October 2019, new requirements on the GA4GH Passports and AAI standards have been identified in the GA4GH. The DURi and security workstreams of GA4GH has started the process towards amending the Passport and AAI specifications. While the exact results of the work have not been finished by the due date of this deliverable, some elements of the amendment can already be identified, based on the recent GA4GH meetings and discussions [GA4G21].

¹¹ <https://research.csc.fi/sensitive-data>

¹² <https://beacon-network.elixir-europe.org/>

In the standard version 1.0, as described in section 3.1.4, a client presents an access token to an API that the server implementing it uses to query the passport visas from the Passport broker's Userinfo (/userinfo) endpoint (Figure 9(a)). This approach has been considered insufficient in the GA4GH community, as there are fears that the environment where the analysis takes place ('server' in Figure 9(a)) does not necessarily have Internet access for security reasons. Therefore it is proposed that the access token should be replaced by a **self-contained passport** that itself contains all the visas that are needed for the analysis. The client would use the access token to retrieve the self-contained passport from a dedicated broker endpoint (Figure 9(b)).

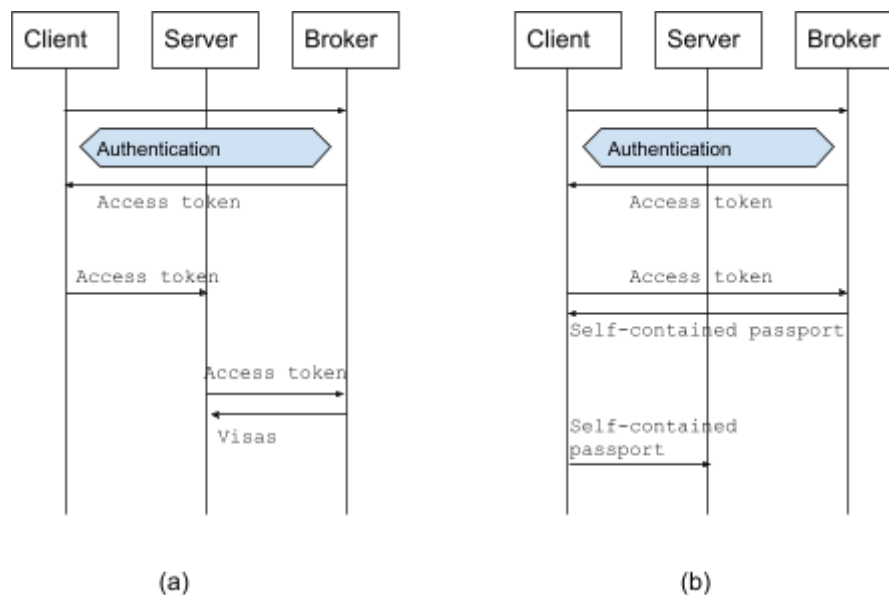


Figure 9. In current ver 1.0 Passport specification (a) a client presents an access token to the server, which uses it to fetch user's visas from the broker. In the proposed new approach (b) the client exchanges the passport for a self-contained passport that contains the visas directly and presents it to the server.

Based on the calculations made, it has been estimated that the size of a typical GA4GH visa (see section 3.1.4) is around one kilobyte. While new API specifications and implementations also support the passing of larger API call parameters, the older ones are invoked using the GET method of HTTP where the parameters are placed into the HTTP header (typically: Authorization header, see section 3.1.4). The size of the HTTP headers is set by the server implementing the API. A conservative estimate is that there exist servers that limit the size of the request parameters to 4 kilobytes. Therefore, a **condensed visa syntax** is proposed to replace the syntax defined in Passport ver 1.0 (see section 3.1.3). Making such adjustments would maximise the number of visas that a server is able to accept in a self-contained passport.

The Self-contained passport with the condensed visas is supposed to be an extension of the current Passport and AAI specifications. A broker would signal if they support it and the client could choose which of the two approaches it decides to use.

3.3.2. Self-Sovereign identity (SSI)

As described in section 3.1.4, the GA4GH Passports ver 1.0 relies on the OpenID Connect protocol (OIDC) and a dedicated Passport broker that authenticates the user and pulls their visas from the various Visa issuers to deliver them to a Passport clearinghouse.

This section studies an alternative approach where the OIDC and the Passport broker are replaced by a self-sovereign identity (SSI) approach. In the SSI, the Visa issuers deliver the visas directly to the users who relay them to the Passport clearinghouses. A proof of concept prepared in the project is presented in this section. The section starts by describing limitations identified in the current approach.

3.3.2.1. Limitations of the current approach

As introduced in section 3.1.1, the Passport broker is an important component to which the Passport clearinghouse redirects the user for authentication. After authenticating the user, the Passport broker consults various visa issuers to pull the user's visas and assemble them to a passport that is delivered to the downstream Passport clearinghouse. While this architecture relies on the well-established OIDC protocol, it has certain downsides:

Fragmented user identifiers. The various Visa issuers may identify the user with different user identifiers. For instance, the ELIXIR AAI knows the user by their ELIXIR Identifier, the EGA by the EGA ID, and the National Institute of Health (NIH) by their eRA Commons account. When the Passport broker authenticates the user, the broker and the Visa issuers need to match the different identifiers the user has in various issuers to assemble the passport correctly. This task can be done, for instance, through identity linking (see section 3.2.3 for the EGA and ELIXIR ID linking), resulting in a LinkedIdentities visa, as described in section 3.1.3. However, regular users may consider identity linking to be complex and not intuitive. From the information security perspective, the linking itself is a fragile procedure. A flawed linking may result in a security breach.

Coupling authentication and visa delivery. As described in section 3.1.4, the Passport broker first authenticates the user and subsequently delivers their visas to the downstream Passport clearinghouse. These two steps cannot be decoupled. If the Passport clearinghouse supports several authentication methods (Passport brokers), the user needs to select the particular one that is able to deliver the visas the Passport clearinghouse needs. Such a situation can be unintuitive for the user.



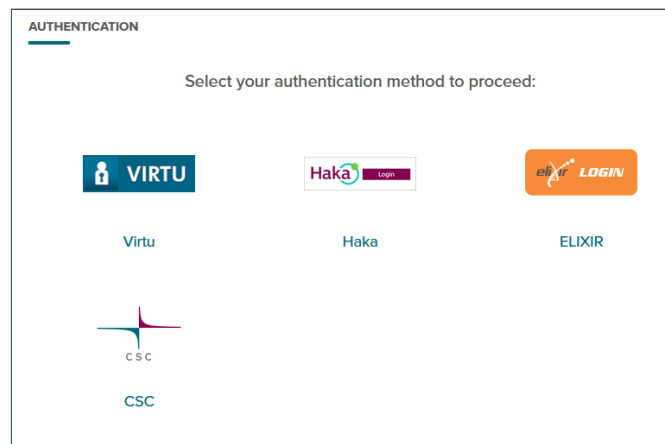


Figure 10. Authentication to CSC - the Finnish IT Center of Science's sensitive data services.

Figure 10 presents the authentication dialogue of CSC - the Finnish IT Center of Science's sensitive data services. There are four alternative ways to authenticate. First, two options are making use of national federated identity management solutions (Virtu for researchers in government agencies and Haka for researchers in universities) that the researchers can use for logging in with their home organisation credentials. Button "CSC" refers to logging in with a CSC username and ELIXIR Login triggers authentication against the ELIXIR Passport broker, as described in section 3.2.2. Out of these four alternatives, only the ELIXIR Passport broker can deliver visas to CSC's sensitive data services. It may be difficult for a user to understand that to project their controlled access files to the computing environment, they must use the ELIXIR Login.

Visas from multiple brokers. It can be foreseen that several Passport brokers will emerge, serving different stakeholders globally. Unless the Passport brokers find a way to interact with each other, a Passport Clearinghouse may need to interact with several brokers to collect all the visas for authorising the data analysis tasks.

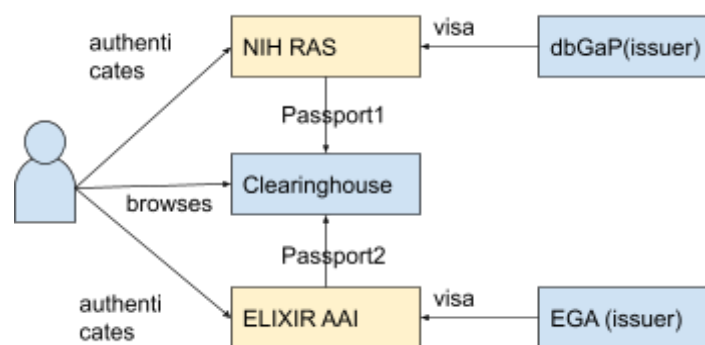


Figure 11. A Passport clearinghouse that needs visas from multiple Passport brokers.

Figure 11 presents a hypothetical example where a clearinghouse needs ControlledAccessGrants visas describing data access permissions to dbGaP and EGA datasets. Visas for dbGaP are delivered by the NIH RAS, a Passport broker managed by the NIH. However, visas for EGA datasets are received from the ELIXIR AAI. In the current approach, to pull the visas from the two issuers, the clearinghouse needs to redirect the user to NIH RAS for authentication. Then subsequently, it needs to redirect the

user to authenticate at the ELIXIR AAI. It may be difficult for a user to understand why a single authentication is not sufficient.

Single point of failure. The Passport broker is an integral component in any interaction between the Visa issuers and Passport clearinghouses, making it a single point of failure. In some communities, it may also be difficult to agree on who can be trusted to operate a broker and what kind of information security and privacy implications it has. On the other hand, even small communities setting up a broker of their own might lead to fragmentation and related issues, such as the one described in Figure 11.

3.3.2.2. Self-Sovereign Identity (SSI) approach

Self-sovereign identity (SSI) is an implementation of a user-centric approach to identity management which emphasises the role of the user as an actor who mediates their attributes (dubbed as verifiable credentials, VC), such as the GA4GH visas, from their issuer to their consumer (dubbed as a verifier). While the flow in OIDC (see section 3.1.4) requires the user to authorise the release of their claims (visas) in the OP (Passport broker), the release of the claims takes place directly between the OP and the RP (Passport clearinghouse). In SSI, the issuer and verifier do not communicate directly.



Figure 12. Actors in SSI and related GA4GH AAI actors.

The SSI basic architecture is described in Figure 12. In SSI architecture, there are no Identity Providers necessary for authenticating users and delivering their attributes. Instead, the issuers sign and deliver the Verifiable Credentials (VC) directly to the user, who can download and store them in their wallet. The wallet is a system component exclusively under the control of the user. It is proposed that the wallet should be implemented as, or controlled by, an application in the user's smartphone. The user then uses their wallet to expose the VC (or a proof that they have it) to the verifier. This can be done, for instance, by using the wallet to log in to a website.

Table 4. Key OpenID Connect, GA4GH Passport/AAI and Self-Sovereign Identity concept mapped.

OpenID Connect, OIDC	GA4GH Passport/AAI	Self-sovereign identity, SSI
claim	visa	verifiable credential, VC
-	visa issuer	issuer
OIDC Provider, OP	Passport broker	-
Relying Party, RP	Passport clearinghouse	verifier

Table 4 maps some OpenID Connect, GA4GH Passport/AAI and SSI concepts. The wider architecture of SSI, including connections, agents, agencies, distributed ledgers, etc., is out of the scope of this deliverable. Also the necessary technical and governance issues are omitted here.

3.3.2.3. Adopting SSI to deliver GA4GH Passports

Figure 13 describes how SSI could be leveraged for managing the delivery of a researcher's GA4GH visas. In step (1), DAC 1 reviews the researcher's DAR and permits them to access dataset 1. DAC 1 offers to the researcher to write the related ControlledAccessGrants visa to their wallet. The researcher approves the smartphone-based retrieval of the visa. This process is repeated in step (2) for Dataset 2. As a result, the researcher has two ControlledAccessGrants in their wallet. In step 3, the researcher logs in to the computing environment using their wallet. The computing environment requests the researcher to release their ControlledAccessGrants visas from the wallet to the computing environment. The researcher uses their smartphone to approve the release.

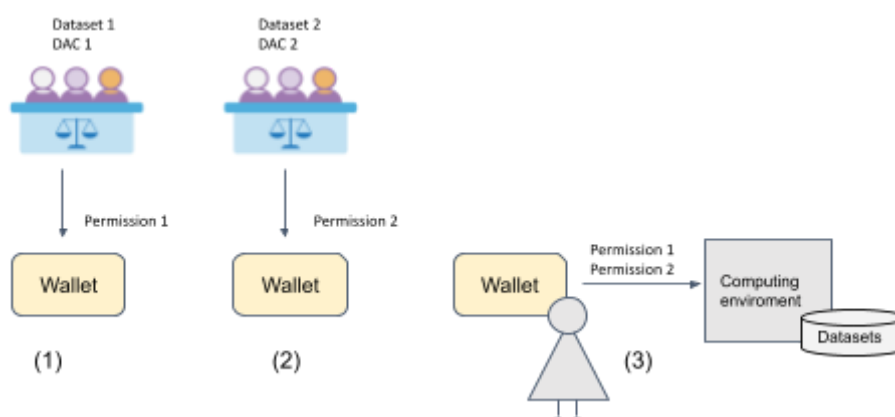


Figure 13. The data access committees write the data access permissions to a researcher's wallet (steps 1 and 2). The researcher releases the permissions to the computing environment (step 3).

The SSI approach could help to overcome the limitations described in section 3.3.2.1:

- **Fragmented user identifiers.** In SSI, the identifier of the user is their DID (Distributed identifier). The DACs would associate the permissions with that identifier, which would later be presented to the computing environment together with the permissions. SSI also supports cryptographic protocols (zero-knowledge proof) enabling the researcher to prove they are the subject of the VC a DAC has issued to them.
- **Coupling authentication and visa delivery.** SSI can be used just for visa delivery, while other approaches could be used for user authentication. However, in the long run, it would be desirable to use SSI also for user authentication.
- **Visas from multiple brokers.** SSI can deliver visas issued by multiple visa issuers (Figure 13) as long as they support writing the visas into the researcher's wallet.
- **Single point of failure.** No Passport broker would be a single point of failure.

Realistically, a mixed model would likely be used, where some visa issuers would support the SSI-based approach, and some would not. If the SSI model can demonstrate clear benefits over the current model, a transition from the current model to SSI could take place. To support wider adoption, the GA4GH Passport specification would have to be adapted to SSI as they currently have significant overlapping functionality (e.g. for digital signatures of the visa/VC).

3.3.2.4. Proof of concept done within the project

This section describes a practical experiment done in the project. The proof of concept relied on Evernym's commercial Verity Service Development Kit (SDK)¹³, Evernym's Connect.me wallet app¹⁴, the REMS¹⁵ DAC tool developed by ELIXIR-Finland and a test Passport clearinghouse which resembled the sensitive data services access management portal of CSC -- the Finnish IT Center for Science (Figure 10).

The flow from a user perspective was the following:

1. **Write user's ELIXIR Identifier to their wallet as a VC.** The user logged in to a Relying Party of ELIXIR AAI, which received their ELIXIR Identifier from the ELIXIR AAI and wrote it to the user's wallet as a VC. In a genuine SSI-based environment, distributed identifiers (DID) would probably make ELIXIR Identifiers unnecessary. However, in a mixed environment, a Passport clearinghouse must also be able to receive users' ELIXIR ID from their wallet.
2. **Write user's ControlledAccessGrant visa to the wallet.** The user had applied for and received the data access permissions to CINECA Synthetic Cohort EUROPE UK1 using REMS. For the experiment, a tool that used the ELIXIR Identifier to read a user's ControlledAccessGrants visa from REMS and wrote it as a VC to their wallet has been implemented. The syntax of the payload followed the GA4GH Passport standard for visas.
3. **Use the wallet to log in to a Passport clearinghouse.** The user used their wallet to log in to a test Passport clearinghouse. The clearinghouse requested the user to release their ELIXIR identifier and GA4GH Passport visas from the wallet. The user confirmed the release and the test clearinghouse displayed the visas received on a web page.

A screencast video of the process above is available on Youtube¹⁶. For the time being, a test service is also available for any ELIXIR ID holder to try. The instructions are in Appendix 2.

The experiment was not intended to be used directly in production but, instead, its purpose was to demonstrate the concept that relies on the SSI. The eventual decision to adopt the SSI approach as a Passport standard needed eventually to be done by the GA4GH. While the experiment encourages to study the approach further, it also exposes some concerns:

- **Limited wallet user interface.** In SSI, the user typically uses their smartphone as the primary user interface to their wallet. The smartphone's user interface is probably sufficient for consumer use cases, like releasing a person's name, e-mail address and credit card number to a verifier. However, a researcher may have dozens of GA4GH visas in their wallet, and they need to manage their release using a relatively small phone user interface.
- **Limitations of wallet software.** There will likely be several wallet implementations for the end-users to choose from. The wallet implementation the users choose will be outside the control of the GA4GH community unless the GA4GH wants to implement a wallet of its own and promote its use. Otherwise, the GA4GH can hardly influence the capabilities the wallet user interface supports and how well they fit the GA4GH Passport use cases. In contrast, the

¹³ <https://www.evernym.com/verity/>

¹⁴ <https://www.evernym.com/connectme/>

¹⁵ <https://github.com/CSCfi/remis/>

¹⁶ https://www.youtube.com/watch?v=K_VRxtKrzHo



user-facing interface of a Passport broker is a web page under the control of the organisation that operates the broker and can better influence the user experience.

4. References

- DYKE18 Dyke, S., Linden, M., Lappalainen, I., De Argila, J., Carey, K., Lloyd, D., Spalding, D., Cabili, M., Kerry, G., Foreman, J., Cutts, T., Shabani, M., Rodriguez, L., Haeussler, M., Walsh, B., Jiang, X., Wang, S., Perrett, D., Boughtwood, T., Matern, A., Brookes, A., Cupak, M., Fiume, M., Pandya, R., Tulchinsky, I., Scollen, S., Törnroos, J., Das, S., Evans, A., Malin, B., Beck, S., Brenner, S., Nyrönen, T., Blomberg, N., Firth, H., Hurles, M., Philippakis, A., Rättsch, G., Brudno, M., Boycott, K., Rehm, H., Baudis, M., Sherry, S., Kato, K., Knoppers, B., Baker, D., Flicek, P. Registered access: authorizing data access. *European Journal of Human Genetics* 26, pages 1721–1731 (2018). DOI: <https://doi.org/10.1038/s41431-018-0219-y>
- GA4G20 GA4GH Passport Webinar Series. Global Alliance for Genomics and Health. 28 May 2020. <https://www.ga4gh.org/news/ga4gh-passports-webinar-series/>
- GA4G19 GA4GH Passport. Version: 1.0.2. https://github.com/ga4gh-duri/ga4gh-duri.github.io/blob/master/researcher_ids/ga4gh_passport_v1.md
- GA4G19b GA4GH AAI OpenID Connect Profile. Version 1.0.4. <https://github.com/ga4gh/data-security/blob/master/AAI/AAIConnectProfile.md>
- GA4G21 Global Alliance for Genomics and Health. Passport Technical Progress. Passport Sub Group Meeting Update. November, 2021. <https://docs.google.com/presentation/d/16PkH0jHR3-Jv4VzTNJMd-aegWmpc8hD50xo6jeGm9c/> Referenced 17 November 2021
- LIND13 Linden, M., Nyrönen, T., Lappalainen, I. Resource Entitlement Management System. Selected papers of TNC13 conference. https://github.com/CSCfi/remss/blob/master/docs/presentations/tnc2013_misc_Tnc2013PaperRemsTnc-fullpaper.pdf
- LIND18 Linden, M., Procházka, M., Lappalainen, I., Bucik, D., Vyskocil, P., Kuba, M., Silén, S., Belmann, P., Sczyrba, A., Newhouse, S., Matyska, L., Nyrönen, T. Common ELIXIR Service for Researcher Authentication and Authorisation. August, 2018. DOI: <https://doi.org/10.12688/f1000research.15161.1>
- PROC20 Prochazka, M., Linden, M. CanDIG and ELIXIR AAI interoperability demonstration. CINECA Project deliverable D2.1. June, 2020. DOI: <https://doi.org/10.5281/zenodo.3938916>
- RFC7519 Jones, M., Bradley, J., Sakimura, N. JSON Web Token (JWT). Request for Comments: 7519. Internet engineering task force. May, 2015.



SAKI14	Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., Mortimore, C. Connect core 1.0. November, 2014. https://openid.net/specs/openid-connect-core-1_0.html	OpenID
VOIS21	Voisin, C., Linden, M., Dyke, S., Bowers, S., Alper, P., Barkley, M., Bernick, D., Chao, J., Courtot, M., Jeanson, F., Konopko, M., Kuba, M., Lawson, J., Leinonen, J., Li, S., Ota Wang, V., Philippakis, A., Reinold, K., Rushton, G., Spalding, D., Törnroos, J., Tulchinsky, I., Auvil, J., Nyrönen, T. GA4GH Passport standard for digital identity and access permissions. Cell Genomics, volume 2, issue 1, 2021. DOI: https://doi.org/10.1016/j.xgen.2021.100030	

5. Abbreviations

AAI	Authentication and Authorisation Infrastructure
API	Application Programming Interface
DAC	Data Access Committee
DAR	Data Access Request
DID	Decentralised Identifier
DURI	Data Use and Researcher Identifiers workstream
EGA	European Genome-Phenome Archive
GA4GH	Global Alliance for Genomics and Health
HTTP	Hypertext Transfer protocol
ID	Identity
JWT	Json Web Token
NIH	National Institute of Health
OIDC	OpenID Connect
OP	OIDC Provider
REMS	Resource Entitlement Management System
REST	Representational State Transfer
RP	Relying Party
SO	Signing Official
SSI	Self-Sovereign Identity
URL	Uniform Resource Locator
VC	Verifiable Credential

6. Delivery and schedule

Work has been done on time.

7. Adjustments made



Appendix 1: EGA's Permissions API specification

The following specification is aimed to provide approved users with the necessary functionality to manage permissions of datasets on behalf or as part of a Data Access Committee. The EGA Permissions API follows the specifications of GA4GH Researcher Identities (Passports and Visas)¹⁷ for communicating permissions to support interoperability.

1. Authentication

All the operations made by any user need to be authenticated. The user can opt to use either an OpenID Connect (OIDC) bearer token of an accepted AAI server or a generated API key for machine-machine communication.

1.1. Bearer Token

This is an OIDC standard bearer token. At the moment EGA only supports both the internal EGA AAI server and ELIXIR AAI. For information about getting an ELIXIR Token, please refer to their Development Guide.

The EGA AAI is available at URL: <https://ega.ebi.ac.uk:8053/ega-openid-connect-server/>

Use OpenID Connect 'password' grant type with EGA {username} and {password}. Access tokens are valid for 1 hour.

```
curl -d
"grant_type=password&client_id=0b54fa73-8124-4c59-93a3-aed7f96d85ae&client_
secret=B_aEAR_HaL2fgjhXUdtYvMLNyw&username={}&password={}&scope=openid" -H
"Content-Type: application/x-www-form-urlencoded" -k
https://ega.ebi.ac.uk:8053/ega-openid-connect-server/token
```

If a refresh token is required also specify the scope "offline_access"

This produces a JSON response containing these elements (the "access_token" is necessary to access the API; the "refresh_token" is optional if the client needs to run longer than 1 hour):

```
{
  "access_token": "e[...]Rg",
  "token_type": "Bearer",
  "refresh_token": "e[..]Q.",
  "expires_in": 3599,
  "scope": "openid offline_access",
  "id_token": "e[..]I"
}
```

Every request intended to use Bearer Token authentication must have a valid token in the Header according to the following format:

¹⁷ https://github.com/ga4gh-duri/ga4gh-duri.github.io/tree/master/researcher_ids



Location: Header
 Name: Authorization
 Format: Bearer [Token]

Please note that the token must be prepended with the 'Bearer' string followed by an empty space.

1.2. API Key

It is also possible to authenticate using a generated API_KEY. Using this authentication method is only intended for programmatic access to the API by automated systems (machine-machine). Users should use Bearer token authentication whenever possible. Check the API Key Endpoint section for more details.

API Key Tokens can be invalidated when they are no longer needed. Every request intended to use API Key authentication must have a valid token in the header according to the following format:

Location: Header
 Name: Authorization
 Format: api-key [Token]

Please note that the token must be prepended with the 'api-key' string followed by an empty space.

2. Environment

The following are the root publicly available uris that map to both the OIDC server and the permissions API service as well as the available swagger documentation.

Environment	EGA OpenID Server*	Permissions API*	Swagger
Production	https://ega.ebi.ac.uk:8443/ega-openid-connect-server	https://ega.ebi.ac.uk:8443/ega-permissions https://ega.ebi.ac.uk:8443/ega-permissions/1.0.0	https://ega.ebi.ac.uk:8443/ega-permissions/swagger-ui/index.html

*Root application URLs are not browsable. For browsable documentation, use the Swagger link.

3. Operations

Operation	Format*	Method	Path
Read permissions	plain/jwt	GET	/permissions
Create or update permissions	plain/jwt	POST	/permissions
Delete permissions	-	DELETE	/permissions
List users with access to a Dataset	-	GET	/datasets/{datasetId}/users



Current user permissions	plain/jwt	GET	/me/permissions
Generate API Key	-	GET	/api_key/generate
List API Keys for current user	-	GET	/api_key
Invalidate a specific API Key	-	DELETE	/api_key/{key}

**Some operations allow a format parameter to be specified in order to accept/return Signed JWT objects instead of plain JSON objects. Visit the Usage section for more information.*

3.1. Schema

The following objects are used among the request/response in the Permissions API:

3.1.1. Visa

The Visa object is a representation of the Passport Claim part in the payload of a GA4GH WS-Encoded JSON Web Token (JWT). It contains the following attributes:

Attribute	Type / Format	Description
sub (<i>required</i>)	string	User for whom visa has been issued
iss	string	Issuer
iat	integer	Timestamp of when the Passport Visa was minted
jti	string	jti (JWT ID) claim
exp	integer (int64)	Generally, it is seconds since unix epoch of when the Passport Visa Assertion Source requires such a claim to be no longer valid
ga4gh_visa_v1	PassportVisaObject	Passport Visa details, see PassportVisaObject
format	String	Format (PLAIN)

3.1.2. PassportVisaObject

As described in the [GA4GH specification](#), the PassportVisa Object is a JWT claim value for the ga4gh_visa_v1 JWT claim name.

Attribute	Type / Format	Description
type (<i>required</i>)	string	Visa Type (EGA currently supports ControlledAccessGrants)
asserted	integer (int64)	Timestamp when the action was added or last updated
value	string	Object to which permissions will apply
source	string	The authority who sanctioned this action (EGA DAC)
by	string	Indicates that the person, service, or broker that applied the action on behalf of the 'source'



3.2. Format parameter

Some endpoints accept a **format** parameter in the query that can be either PLAIN or JWT (default JWT). This allows clients to specify the format for their requests. Note that format will be applied to input and output objects, meaning that a POST request would need to send JWT or PLAIN if the parameter is specified.

A discriminator attribute **format** is also added to the returned objects so the client is aware of the format being used.

For example:

A request to **/permissions?format=PLAIN** will return PLAIN JSON object with a response similar to:

```
[
  {
    "attr1": "val",
    "attr2": "val",
    "format": "PLAIN"
  }
]
```

Note the format attribute specifying the format used to elaborate the response. If the request doesn't specify a format parameter, the default response will be JWT

3.3 Header and query parameters (Account ID)

Some endpoints allow clients to provide parameters either in the query or header of the request. Passing ids in the URI is not always desirable, as they can be logged by web servers. By passing these parameters on the header we can avoid them being logged outside of the intended ensuring a higher degree of privacy. If both query and header parameters are set, the value in the header takes priority.

For example:

Path	/permissions	/permissions
Query	account-id=XYZ	
Header		x-account-id=XYZ
Full URL	/permissions?account-id=XYZ	/permissions
Result	OK: Returns permissions for account XYZ	OK: Returns permissions for account XYZ

3.4 Read permissions



Returns a list of plain or JWT encoded JSON objects containing the permissions currently assigned to a given user (`accountId`).

HTTP Method: GET

Endpoint: `/permissions`

Parameter	Type	Location	Description
x-account-id	string	header	User account ID*
account-id	string	query	User account ID*
format	string	query	Format (PLAIN, JWT). (see section 5.3)

*User Account ID can be passed either in the query or header of the request. See Header and query parameter for more information.

Response codes:

HTTP Code	Description
200	Successful operation
404	User account invalid or not found
40X	Other client-related errors (Unauthorized, Bad Request, etc)
50X	Server-related errors

Request example:

GET `/permissions?account-id=EGAW00000015388`

Response example (PLAIN format):

```
[
  {
    "sub": "EGAW00000015388",
    "iss": "https://ega.ebi.ac.uk:8053/ega-openid-connect-server/",
    "exp": 1592824514,
    "iat": 1592820914,
    "jti": "f030c620-993b-49af-a830-4b9af4f379f8",
    "ga4gh_visa_v1": {
      "type": "ControlledAccessGrants",
      "asserted": 1568814383,
      "value": "EGAD00001002069",
      "source": "EGAC00001000514",
      "by": "dac"
    },
    "format": "PLAIN"
  }
]
```

3.5 Create or update permissions



Receives a set of plain or JWT encoded objects and stores them in the system. The permissions represent the access a particular user has to one or more datasets. If the object submitted is already present in the database, it will be updated. A response code and message is generated for each item in the request list. Requests will only be processed if they are made by users who are approved to manage access to the specified dataset.

HTTP Method: POST

Endpoint: /permissions

Parameters:

Parameter	Type	Location	Description
account-id	string	query	User account ID*
x-account-id	string	header	User account ID*
value	Array of PassportVisaObject	Request body	List of permissions to be created/added
format	string	query	Format (PLAIN, JWT). (see section 5.3)

*User Account ID can be passed either in the query or header of the request. See Header and query parameter for more information.

Response codes:

HTTP Code	Description
207	Multi-status with inner status for individual items
400	Invalid ID supplied
404	User account not found
40X	Other client-related errors (Unauthorized, Unprocessable Entity, etc)
50X	Server-related errors

Request examples (PLAIN format):

POST /permissions?account-id=EGAW00000015388&format=PLAIN

```
[
  {
    "type": "ControlledAccessGrants",
    "asserted": 1568814383,
    "value": "EGAD00001002069",
    "source": "EGAC00001000514",
    "by": "dac"
    "format": "PLAIN"
  }
]
```



Response examples (PLAIN format):

```
[
  {
    "ga4gh_visa_v1": {
      "type": "ControlledAccessGrants",
      "asserted": 1568814383,
      "value": "EGAD00001002069",
      "source": "EGAC00001000514",
      "by": "dac"
    },
    "status": 201,
    "message": "Success",
    "format": "PLAIN"
  }
]
```

Note: In case multiple objects are sent in the request body, each one would have an independent inner status. This means there can be one 201 and one 500 for a request containing two items.

3.6. Delete permissions

Receives an account ID and dataset ID and revokes the permissions of a particular user to a particular dataset.

HTTP Method: DELETE

Endpoint: /permissions

Parameters:

Parameter	Type	Location	Description
account-id	string	query	User account ID*
x-account-id	string	header	User account ID*
value	string	path	Dataset ID

*User Account ID can be passed either in the query or header of the request. See Header and query parameter for more information.

Response codes:

HTTP Code	Description
200	Successful operation
204	No content (no record has been deleted)
400	Invalid ID supplied
40X	Other client-related errors (Unauthorized, etc)
50X	Server-related errors

Request example:

DELETE /permissions?account-id=EGAW00000015388&value=EGAD0000123

3.7 List users with access to dataset

This endpoint will return a list of user account IDs that have access to a particular dataset along with the asserted time. Only users that are controllers for the dataset being queried would be able to see the list of users with granted permissions to it. The service will return empty responses instead of error messages for invalid inputs.

HTTP Method: GET

Endpoint: /datasets/{datasetId}/users

Parameters:

Parameter	Type	Location	Description
datasetId	string	path	Dataset ID

Response codes:

HTTP Code	Description
200	successful operation
40X	Other client-related errors (Unauthorized, etc)
50X	Server-related errors

Request example:

GET /datasets/EGAD0000123/users

Response example:

```
[
  {
    "accountId": "EGAW00000015388",
    "asserted": 12344
  }
]
```

3.8 Current user permissions

Returns a list of plain or JWT encoded JSON objects containing the permissions assigned to the current user.

HTTP Method: GET

Endpoint: /me/permissions

Parameters:

Parameter	Type	Location	Description
-----------	------	----------	-------------



format	string	query	Format (PLAIN, JWT). (see section 5.3)
--------	--------	-------	--

Response codes and other and sample responses are similar to the ones in Read Permissions.

3.9. Generate API Key

Allow authenticated users to generate an API_KEY token that can be used in requests. Return a message with the id, the expiration date and the api token.

HTTP Method: GET

Endpoint: /api_key/generate

Parameters:

Parameter	Type	Location	Description
id	string	query	ID of the API_KEY
expiration_date	string	query	Expiration date in YYYY-MM-DD format
reason	string	query	Reason

Response codes:

HTTP Code	Description
200	successful operation
40X	Other client-related errors (Unauthorized, etc)
50X	Server-related errors

Request example:

GET /api_key/generate?id=SampleID&expiration_date=2021-12-27&reason=MyReason

Response example:

```
{
  "id": "SampleID",
  "expiration_date": 1592824514,
  "token": "generated.token="
}
```

3.10 List API Keys for current user

Returns a list of all the API_KEY tokens the current user has generated. Return the ids of the keys, reasons and their expiration_date.

HTTP Method: GET

Endpoint: /api_key

Parameters: No parameters



Response codes:

HTTP Code	Description
200	successful operation
40X	Other client-related errors (Unauthorized, etc)
50X	Server-related errors

Request example:

GET /api_key

Response example:

```
[
  {
    "id": "Key1",
    "expiration_date": 1592824514,
    "reason": "Test 1"
  },
  {
    "id": "Key2",
    "expiration_date": 1692824514,
    "reason": "Test 2"
  }
]
```

3.11 Invalidate a specific API Key

Remove an API_TOKEN whenever it is no longer needed.

HTTP Method: DELETE

Endpoint: /api_key/{key}

Parameters:

Parameter	Type	Location	Description
key	string	path	ID of the API_KEY to be removed

Response codes:

HTTP Code	Description
200	successful operation
40X	Other client-related errors (Unauthorized, etc)
50X	Server-related errors

Request example:

DELETE /api_key/SampleID

Appendix 2: Instructions for the SSI test service

SSI Demo - give a try

Instructions for using the CINECA SSI demo



Disclaimer: this is a technical demonstrator which does not cover all possible flows and may also be occasionally down for upgrades.

See [a recorded screencast video](#) of the demo.

Prerequisites

1. Register an ELIXIR ID here: <https://elixir-europe.org/register> (if you don't have one)
2. Register to ELIXIR AAI test environment here: https://perun.elixir-czech.cz/fed/registrar/?vo=elixir_test
 - a. SSI Demo is not a production service. You need to opt in as a test user
3. Install Evernym's Connect.me app in your smartphone from the appstore ([IOS](#), [Android](#))

1. Write ELIXIR ID to the wallet as a verifiable credential

What you are supposed to do	What you are supposed to see
Browse to https://kyc-ssi-dev-coop.rahtiapp.fi/ Click "Continue"	<p>Verifiable Credential Issuer BfP3k36PH97rPBJ2FcoVz2 for ELIXIR ID</p> <p><small>CSC Sensitive Data Services DEMO for passing your ELIXIR ID as a Verifiable Credential to your Connect.Me wallet application.</small></p> <p>Prerequisites</p> <ol style="list-style-type: none"> 1. Install Connect.Me application to your mobile phone either from Apple or Google store. 2. Initialise Connect.Me application to use Sovrin Staging as ledger. 3. You need to have ELIXIR ID and Test VO enabled. <ol style="list-style-type: none"> 1. Register ELIXIR ID 2. Enable Test VO for your ELIXIR ID <p><small>Continue</small></p>
Click ELIXIR Login to log in with your ELIXIR ID	<p>AUTHENTICATION</p> <p>Select your authentication method to proceed:</p> <p></p> <p>ELIXIR</p>
Log in using your your ELIXIR ID	<p></p>



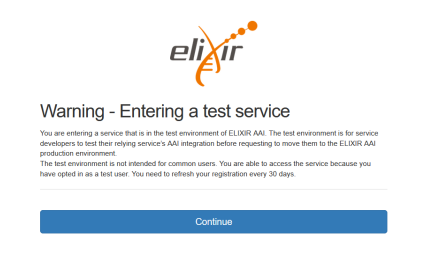
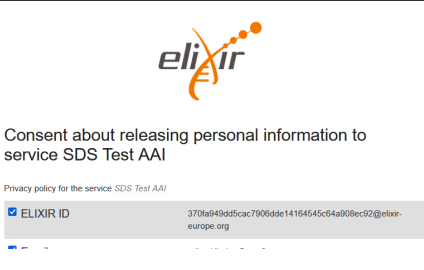
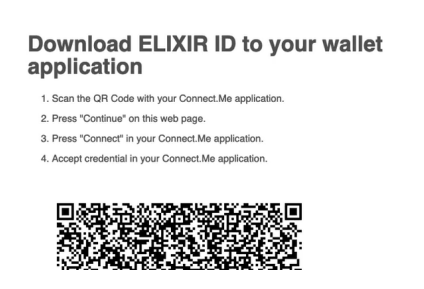
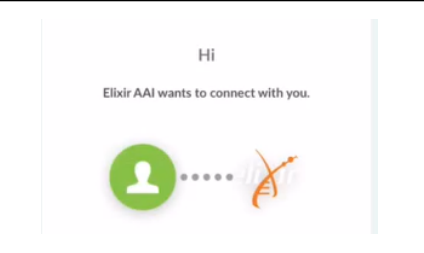
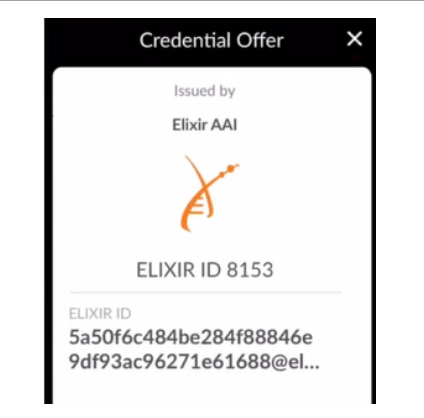
Choose how to log in

Use your previous selection

CSC - IT Center for Science Ltd.

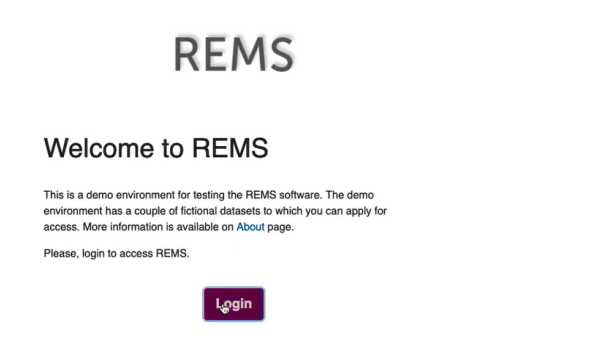
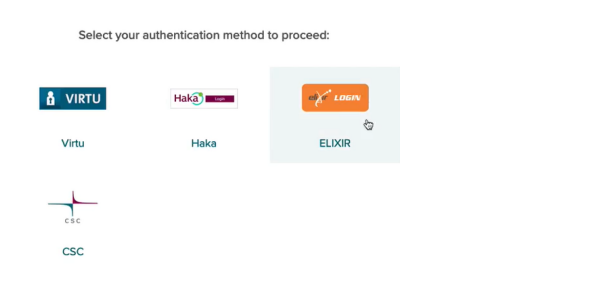
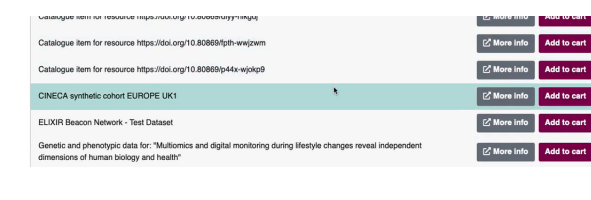
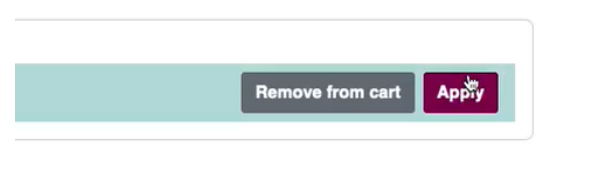
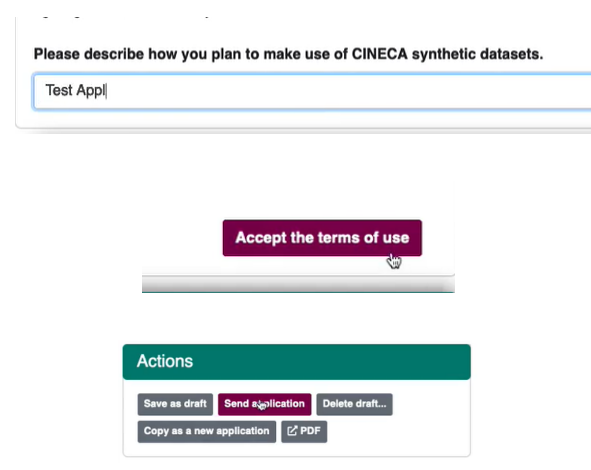
or

Sign in using another institute or account

<p>Click “Continue” to enter a test service</p>	 <p>The screenshot shows the ELIXIR logo at the top. Below it, the text reads "Warning - Entering a test service". A smaller text block explains that the user is entering a test environment and that the test environment is not intended for common users. At the bottom, there is a blue "Continue" button.</p>
<p>Confirm the release of your personal data</p>	 <p>The screenshot shows the ELIXIR logo at the top. Below it, the text reads "Consent about releasing personal information to service SDS Test AAI". There is a link for the "Privacy policy for the service SDS Test AAI". A checkbox labeled "ELIXIR ID" is checked, and a small text block shows a long alphanumeric string and an email address.</p>
<p>Open Connect.me and scan the QR code from the browser. Click “Continue” in your browser.</p>	 <p>The screenshot shows the heading "Download ELIXIR ID to your wallet application". Below the heading is a numbered list of four steps: 1. Scan the QR Code with your Connect.Me application. 2. Press "Continue" on this web page. 3. Press "Connect" in your Connect.Me application. 4. Accept credential in your Connect.Me application. A QR code is displayed below the list.</p>
<p>Accept ELIXIR AAI connection in Connect.me app by clicking “Connect”</p>	 <p>The screenshot shows a "Hi" greeting followed by "Elixir AAI wants to connect with you." Below this text are two circular icons: a green one with a white person silhouette and an orange one with a DNA helix, connected by a dotted line.</p>
<p>Accept ELIXIR AAI’s credential offer in Connect.me app by clicking “Accept credential”.</p> <p>This stores your ELIXIR ID as a verifiable credential in your wallet.</p>	 <p>The screenshot shows a "Credential Offer" window. It displays "Issued by Elixir AAI" with the ELIXIR logo. Below that, it shows "ELIXIR ID 8153". At the bottom, it lists the "ELIXIR ID" as a long alphanumeric string and an email address.</p>


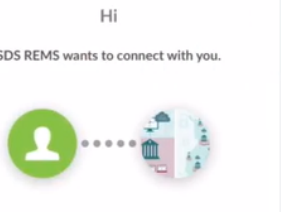
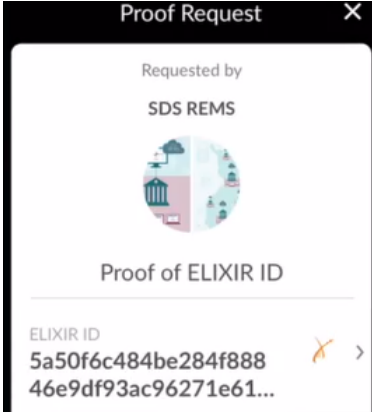


2. Apply for access rights to a dataset in REMS

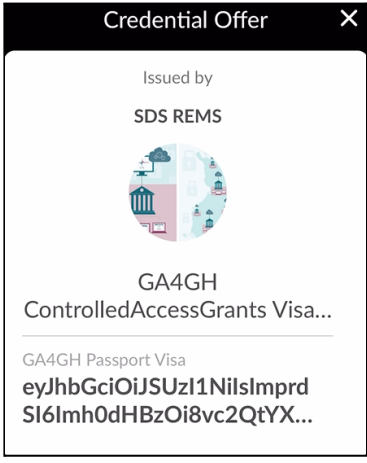

<p>Browse to https://sd-apply.csc.fi/</p> <p>Click “Login”</p>	 <p>The image shows the REMS welcome page. At the top, it says 'REMS' in large letters. Below that, it says 'Welcome to REMS'. A paragraph of text explains that this is a demo environment for testing the REMS software, with a few fictional datasets available for access. It includes a link to an 'About' page and a prompt to 'Please, login to access REMS.' At the bottom, there is a purple 'Login' button.</p>
<p>Select “ELIXIR login”.</p> <p>Log in using ELIXIR AAI as above.</p>	 <p>The image shows a screen titled 'Select your authentication method to proceed:'. There are three main options: 'VIRTU' with a blue icon, 'Haka' with a red icon, and 'ELIXIR' with an orange icon and a 'LOGIN' button. Below these, there are logos for 'CSC'.</p>
<p>Add “CINECA synthetic cohort EUROPE UK1” from the catalogue to cart.</p>	 <p>The image shows a list of datasets in a catalogue. Each entry has a 'More info' button and an 'Add to cart' button. The entry 'CINECA synthetic cohort EUROPE UK1' is highlighted in green.</p>
<p>Click Apply</p>	 <p>The image shows a single item in a cart. It has a 'Remove from cart' button and a purple 'Apply' button.</p>
<p>Provide short reasoning.</p> <p>Accept the terms of use.</p> <p>Then Send the application. You can observe your application is approved automatically. A data access permission is issued for you.</p>	 <p>The image shows a form for providing reasoning, with a text input field containing 'Test Appl'. Below the form is a purple 'Accept the terms of use' button. At the bottom, there is an 'Actions' section with buttons for 'Save as draft', 'Send application', 'Delete draft...', 'Copy as a new application', and 'PDF'.</p>

3. Download the data access permissions to your wallet



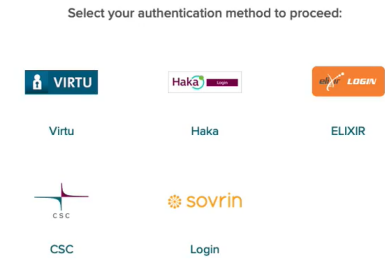

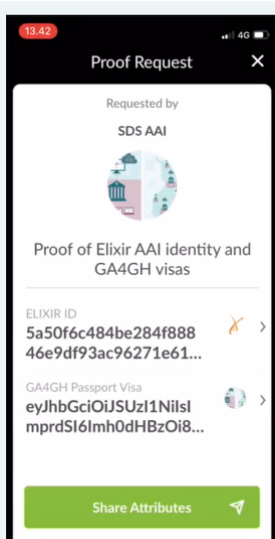
<p>Browse to https://rems-ssi-dev-coop.rahtiapp.fi/</p> <p>Click "Continue"</p>	<p>Download GA4GH Visa from REMS in your wallet</p> <p>CSC Sensitive Data Services DEMO that fetches your GA4GH Passport visa from REMS and passes it as a Verifiable Credential to your Connect.Me wallet application</p> <p>Prerequisites</p> <ol style="list-style-type: none"> 1. Connect.Me wallet application containing your ELIXIR ID verifiable credential issued by B1F9k36PH97/PBj2FcoVz2.
<p>Open Connect.me and scan the QR code from the browser.</p> <p>Click "Continue" in your browser.</p>	<p>Download GA4GH Visa in your wallet</p> <ol style="list-style-type: none"> 1. Scan the QR Code with your Connect.Me application. 2. Press "Continue" on this web page. 3. Press "Connect" in your Connect.Me application. 4. Press "Share attributes" in Connect.Me application to share your ELIXIR ID. 5. Accept credential(s) in your Connect.Me application. 
<p>Accept SDS REMS connection in Connect.me app by clicking "Connect"</p>	 <p>Hi</p> <p>SDS REMS wants to connect with you.</p>
<p>Click "Share attributes" in your Connect.me app to share your ELIXIR ID with SDS REMS</p>	 <p>Proof Request</p> <p>Requested by SDS REMS</p> <p>Proof of ELIXIR ID</p> <p>ELIXIR ID 5a50f6c484be284f888 46e9df93ac96271e61...</p>



<p>Accept SDS REMS's credential offer in Connect.me app by clicking "Accept credential".</p> <p>This stores your data access permissions as a verifiable credential in your wallet.</p>	 <p>Credential Offer</p> <p>Issued by SDS REMS</p>  <p>GA4GH ControlledAccessGrants Visa...</p> <p>GA4GH Passport Visa eyJhbGciOiJSUzI1NiIsImprdiSI6Imh0dHBzOi8vc2QtYX...</p>
---	---



4. Use wallet to log in and present visas

<p>Browse to https://service-ssi-dev-coop.rahtiapp.fi</p> <p>Select “Sovrin” as the authentication method.</p>	
<p>Open Connect.me and scan the QR code from the browser.</p> <p>Click “Continue” in your browser.</p>	<p>CSC Sensitive Data Services platform Sovrin Demo Authentication</p> <ol style="list-style-type: none"> 1. Scan the QR Code with your Connect.Me application. 2. Press Continue on this web page. 3. Press Connect/Deny in your Connect.Me application. 4. Respond as requested in your Connect.Me application. 
<p>Click “Share attributes” in your Connect.me app to share your ELIXIR ID and data access permissions with SDS AAI.</p> <p>You can also select a particular attribute value to share.</p>	
<p>The demo service echos your ELIXIR ID and data access permissions in your browser.</p>	<p>Sensitive Service</p> <p>CSC Sensitive Data Services DEMO representing service requiring GA4GH Visa from user.</p> <p>Wellcome 5a50f6c484be284f88846e9df93ac96271e61688@elixir-europe.org.</p> <pre>Authorization as ga4gh visa.{"iss":"https://sd-apply.csc.fi", "sub":"5a50f6c484be284f88846e9df93ac96271e61688@elixir-europe.org", "iat":1634726436, "exp":1666262436, "ga4gh_visa_v1":{"type":"ControlledAccessGrants", "value":"EGAD00001006673", "source":"https://sd-apply.csc.fi/", "by":"dac", "asserted":1634720343}}</pre>

