D. Pinkas
DP Security Consulting

Internet-Draft
Intended status: Standards Track
Expires: 20 February 2022
19 August 2021

Grant Negotiation and Authorization Protocol
draft-pinkas-gnap-core-protocol-00

Abstract

This protocol enables an Authorization Server (AS) to issue access tokens to permit an end-user using a client software to perform operations on a protected resource hosted by a Resource Server (RS). These access tokens allow to support capabilities and/or user attributes.

The protocol includes means of specifying how the end-user can potentially be involved in an interactive fashion during the process. The client and/or the RS will use these interaction mechanisms to involve the end-user, as necessary, to take decisions.

The protocol uses HTTPS for all communications between the client and the AS, as well as between the client and the RS.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This protocol allows end-users to interact with a piece of software, the client instance, to request access tokens to Authorization Servers (ASs) to perform operations either on protected resources hosted by a Resource Server (RS) or on the RS itself to create a user account.

The protocol also allows end-users to collect from an AS information about themselves.

When an end-user is willing to perform an operation on a protected resource hosted by a RS or on the RS itself to create a user account, the end-user operating the client interacts with the RS to assert his consent, authenticates to the AS and then requests an access token from the AS.

The protocol allows to support Attribute Based Access Control (ABAC), as well as Capability Based Access Control (CBAC), where a capability is an authorization to perform an operation on a resource.

This specification also discusses discovery mechanisms for the client instance to discover the access token formats supported by a RS or an AS and whether attributes (for ABAC) and/or capabilities (for CBAC) are needed in order to perform a given operation on a resource or a registration on a RS.

The focus of this protocol is to provide interoperability between the different parties acting in each role, but is not to specify implementation details of each. However the structure of access tokens is detailed, but the syntax of the access tokens is left open. The security of the protocol relies on the presence and on the verification by the RS of some of the fields that must be present in an access token.

The protocol takes into consideration the ease of use for the end-user: an end-user can use any number of clients without the burden to manage them (as long as he can trust the client instance). The protocol also take into consideration some privacy laws and recent regulations (e.g. the EU General Data Protection Regulation) [GDPR] in order to allow the RSs to comply with the legislation.

Direct communications between an AS and a RS are not necessary for the execution of this protocol. The means for an AS and a RS to interoperate directly are not discussed in this document.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.2. Vocabulary

attribute : characteristics related to an end-user

right : ability given to an end-user to perform a given operation on a protected resource under the control of a RS

Note: a "right" is denoted as a "capability" in the security literature.

access token : data artifact representing a set of rights and/or attributes

grant (verb) : to permit an instance of client software to receive some attributes at a specific time and valid for a specific duration and/or to exercise some set of delegated rights to access a protected resource

grant (noun) : the act of granting

privilege : right or attribute associated with an end-user

protected resource : API (Application Programming Interface) served by an RS and that can be accessed by a client, if and only if a valid access token is provided.

1.3. Abbreviations

ABAC	Attribute Based Access Control
CBAC	Capability Based Access Control

1.4. Roles

The parties in GNAP perform actions under different roles. Roles are defined by the actions taken and the expectations leveraged on the role by the overall protocol.

Resource Server (RS) : server that provides operations on resources, where operations on resources protected by GNAP require a valid access token issued by an Authorization Server (AS)

Authorization Server (AS) : server that grants delegated privileges to a particular instance of client software in the form of access tokens

Client : application operated by an end-user that consumes resources from one or several RSs, possibly requiring access tokens from one or several ASs

End-user : natural person that operates a client instance

Example: a client can be a mobile application, a web application, etc.

Resource Owner (RO) entity that may grant or deny operations on resources it has authority upon

Note: the act of granting or denying an operation may be manual (i.e. through an interaction with a physical person) or automatic (i.e. through predefined rules).

The following Figure 1 illustrates the relationships between the different roles

```
                                     +---------------+
                                     |               |
                                     | Authorization |
                         +---------->|    Server     |
                         |           |               |
                         |           +---------------+
                         |                   ~
                         |                   ~ When CBAC is supported
                         |                   ~
                    +----------+     +---------------+
   +----------+     |          |     |               |
   | End-user |+ + +|  Client  |     |   Resource    |
   +----------+     | Instance |     |     Owner     |
                    |          |     |               |
                    +----------+     +---------------+
                         |                   ~
                         |                   ~ When ABAC is supported
                         |                   ~
                         |           +---------------+
                         |           |               |
                         +---------> |   Resource    |
                                     |    Server     |
                                     |               |
                                     +---------------+

   Legend

   + + + indicates an interaction between a human and computer
   ----- indicates an interaction between two pieces of software
   ~ ~ ~ indicates a possible interaction between two roles
```

Figure 1: Relationships between the different roles

When a resource hosted by the RS is access control protected using attributes (ABAC), then the RO interfaces with the RS.

When a resource hosted by the RS is access control protected using capabilities (CBAC), then the RO interfaces with the AS.

A RS may host protected resources where some of them are access control protected using attributes and while some others are access control protected using capabilities. A same resource can even be protected using attributes and capabilities.

1.5. Trust relationships

All the exchanges between a Client and a RS SHALL be protected using HTTPS, i.e. HTTP [RFC7231] over TLS [RFC8446]. The verification of the identity of the RS shall be done according to RFC 6125 [RFC6125].

All the exchanges between a Client and an AS SHALL be protected using HTTPS, i.e. HTTP [RFC7231] over TLS [RFC8446]. This includes the authentication exchange between an end-user and an AS. The verification of the identity of the AS shall be done according to RFC 6125 [RFC6125].

In order to trust an AS public key certificate or a RS public key certificate, a client SHALL install and use a trust anchor that allows to verify each AS or RS public key certificate. The overall certification scheme SHALL allow each client to test the revocation status of each AS or RS public key certificate using CRLs [RFC5280] or OCSP [RFC6960].

In order to trust an AS public key certificate, a RS SHALL install and use a trust anchor that allows to verify each AS public key certificate. The overall certification scheme SHALL allow each RS to test the revocation status of each AS public key certificate.

In order to allow for the revocation of the public key certificate of an AS or a RS, the public key that allows to verify that public certificate SHALL itself be certified using a public key certificate issued by an upper CA.

The end-user is trusting his client to manage the interactions with the AS and with the RS, whether these interactions are performed using APIs or using a User Interface (UI).

The end-user is trusting the AS to manage his attributes and, upon request, to disclose them to his client.

In order to allow the checking of the integrity and the authenticity of the content of an access token, each AS SHALL sign its access tokens using a private key certified by a CA (Certification Authority).

For the delivery of either rights or attributes in an access token, a RS may trust one or more ASs.

The remaining trust conditions are different whether rights or attributes are supported by the RS to allow performing operations on a given resource under the control of an RS.

Rights

An access control scheme where rights are supported by the RS to perform operations on a resource is usually known in the security literature as a system using capabilities. When rights are supported by a RS to allow performing operations on a resource under the control access token.

In such a case, an of the RS, the RS may trust one or more ASs for the delivery of rights in arrangement needs to be established between each AS/RS pair: the RS MUST designate to the AS a RO that will be responsible to deliver rights which will be placed into access tokens.

For each resource protected using capabilities and for each authenticated user, the responsible RO is able to indicate to the AS which operations may be permitted on the resource by an authenticated end-user.

A RS may decide to accept from a given RO only a reduced set of operations on some resources.

By default and in order to allow interoperability tests, all the resources placed under the control of one RS SHOULD be managed by a single RO. Using private arrangements between the RS and the AS, finer or coarser granularities may override the default behavior.

Attributes

An access control scheme where attributes are supported by the RS to perform operations on a resource is usually known in the security literature under the acronym of ABAC (Attribute Based Access Control). When attributes are supported by a RS to allow performing operations on a resource under the control of a RS, the RS may trust one or more ASs for the delivery of attribute types and/or attribute values in access tokens.

For each resource protected using attributes, the responsible RO is able to indicate to the RS which operations may be permitted on the resource, when a proper set of attribute types and attribute values are present in an access token.

In such a case, the RS may globally indicate which attribute types and/or attribute values in access token will be accepted from a given AS.

Note: It should be observed that when attributes are being used the

AS does not need to perform any kind of pre-arrangement with the RSs.

1.6. Prior arrangements before the protocols can be used

The following arrangements are supported using out-of-bands means that are outside the scope of the protocol.

Every end-user MUST have an account opened with at least one AS. When the account is settled between the end-user and the AS, a user identifier and an authentication method SHALL be agreed. The end-user MUST receive information that allows him to perform a first authentication exchange with success.

When capabilities are supported by a RS, that RS SHALL designate a RO that will interface with at least one AS.

1.7. Short term and long term user accounts

Two types of accesses may be proposed by a RS to allow an end-user to use:

- short-term user accounts, or
- long-term user accounts.

In the first case, the user may be willing to remain anonymous towards the RS by either using a capability or disclosing a set of attributes that will be insufficient to identify him unambiguously. Once the session will be closed, the RS will not maintain information about the session, except in his audit trail.

In the second case, the user may be willing to retrieve some data, to deposit some data or to modify some data that will be saved by the RS. A typical example, is an access to a bank account.

1.8. Structure of an access token

This section describes the structure of an access token by enumerating both required and optional fields.

For each field, it prescribes both its semantics and the processing that SHALL be done on it, but it does not prescribe its syntax. In the future, additional documents may define detailed access token formats including their encoding.

An access token is composed of two parts: a signed part and an unsigned part which is optional.

Before accepting an access token, a RS SHALL check the mandatory fields of the access token.

1.8.1. Signed part of an access token

This part contains:

- a required field (called "valid") that indicates the validity period of the access token using the UTC time. The start of the validity period is the time at which the access token was issued.

- a required field (called "as_pkc") that allows to identify the issuer of the access token, i.e. the AS. Either the PKC (Public Key Certificate) [RFC5280] of the AS SHALL be included or, if the corresponding certificate is placed in the unsigned part of the access token, a hash value of that certificate SHALL be included.

- an optional field (called "hash_algo") that indicates the identifier of the hash algorithm used to compute the digital signature of the access token.

- one of the two following optional fields, i.e. "rs_url" or "hidden_url", SHALL be present:

  - an optional field (called "rs_url") that allows a RS to make sure that the access token is indeed intended for itself. That field SHALL contain at least one URL of a RS.

  - an optional field (called "hidden_url") that allows a RS to make sure that the access token is intended for itself. That field SHALL contain at least one value that MUST be combined with a field ("reveal_url") from the unsigned part of the access token to recover the real value of the target URL of a RS.

- a required field (called "buid") which is a "binding user identifier" that allows a RS to verify that the access token is associated with the right (short-term or long-term) user account. This field allows to detect the inappropriate use of an access token by a malicious user, including in the case of a collusion between users. This field is composed of a type chosen by the client and of a value chosen by the AS. Five different types may be requested by the client :

The first four types are used in the context of long-term user accounts managed by the RS, while the last type is used in the context of short-term user accounts managed by the RS.

The four types used in the context of long-term user accounts managed by a RS are:

(1) a unique user identifier used to identify a user for each User/ RS pair, or

Note: this option cannot be implemented in the context of a "software-only" solution. It requires the use, by the end-user, of a secure element with specific security properties. [This option is not detailed any further at the moment].

(2) a unique user identifier used to identify a user for each AS / RS pair, or

(3) a locally unique user identifier used to identify a user whatever RS is being involved, or

(4) a globally unique user identifier.

The last type used in the context of short-term user accounts managed by a RS is:

(5) a short-term user unique identifier.

Note: The four first types are used when a long-term user account is being used on a RS. The last and fifth type is used when a short-term user account is being used on a RS.

- at least one of the two following optional fields, i.e. "attrs" or "rights", SHALL be present (both may be present):

  - an optional field (called "attrs") that contains one or more attributes that are associated with the end-user. This field is an array where each element of the array is composed of the identifier of an attribute type and of the associated attribute value.

  - an optional field (called "rights") that contains one or more rights (i.e. capabilities) that have been granted to the user by a RO hosted by the AS and which is associated with the RS protecting the resource. This field is an array where each element of the array is composed of one or more methods and of the URL of a resource.

- an optional field (called "at_uid") that contains a unique identifier for the access token. The identifier value MUST be assigned by the AS in a manner that ensures that there is a negligible probability that the same value will be accidentally assigned twice. This field may be useful for audit purposes.

1.8.2. Unsigned part of an access token

This part contains:

- the signature value (called "sign") of the access token.

- an optional field (called "path") that contains a certification path [RFC5280], starting with the PKC of the AS.

- an optional field (called "reveal_url") that allows to retrieve the true value(s) of the URL(s) of RS(s) that has (have) been hidden to the AS.

  This field is an array where each element of the array is composed of a random value called "rnd_value" and a method identifier. If the field "hidden_url" is present in the signed part of the access token, then the field "reveal_url" MUST be present. This field MUST be used by the RS to combine the "hidden_url" value with the appropriate "rnd_value" to verify that the end result matches with its own URL value.

  A one way hash function (OWHF) SHALL be used by the client to compute the "hidden_url" to be placed into the access token, by first choosing a large random number for the "reveal_url" value and combining it with the base URL of the RS using the following formula: "hidden_url" = OWHF ("reveal_url", RS_URL). When the client receives the access token from the AS and before communicating it to the RS, the "reveal_url" value SHALL be inserted by the client into the unsigned part of the access token.

1.9. Mandatory checks to be done by a RS on an access token

When a resource is protected using GNAP, several checks need to be done.

The ordering of the following checks is not mandated as long as all the checks are performed.

However, the following list has been established taking into consideration that an invalid access token should be rejected as soon as possible which means that the fastest checks should be done before.

When receiving an access token, the RS SHALL check that:

- it is well-formed,

- it contains an "attrs" field if the RS supports attributes and it contains a "rights" field if the server supports capabilities. If a RS supports both, one of these two fields SHALL be present.

- the access token is currently within its validity period by using the "valid" field and the UTC time. The start of the validity period cannot be sooner than the current time expressed using the UTC time. A time skew of a dozen of seconds SHOULD be allowed. The validity period of the access token SHOULD NOT exceed 25 hours.

- that the access token is indeed targeted to itself by using either the "rs_url" field or the "hidden_url" field, and in the later case also the "reveal_url" field.

When the RS receives an access token that contains both an "hidden_url" and a "reveal_url" field, it SHALL verify that the access token is targeted to itself, by using its RS_URL and then computing the value: OWHF (reveal_url, RS_URL) and finally verify that it matches with the value contained in the "hidden_url" field.

- that the access token is apparently coming from one of the ASs trusted by the RS using the DN (Distinguished Name) of the AS that is present in the AS's PKC.

  In order to perform that check, the AS's PKC SHALL be retrieved either directly from the "as_pkc" field or using the hash value of that PKC if present in the "as_pkc" field to retrieve it from the "path" field. The validity period of that PKC [RFC5280] SHALL be verified.

- that the signature of the access token is valid. In order to perform this verification, the RS SHALL first construct a certification path between an appropriate trust anchor and the RS's PKC [RFC5280]. If present, the field "path" SHOULD be used to construct that certification path. Once the certification path has been verified as being valid (taking in consideration the revocation status of each PKC from the certification path) the certified key SHALL be extracted from the PKC and used to verify the field "sign" of the access token [RFC5280]. Depending upon the public key algorithm being used, the field "hash_algo", if present, SHALL also be used.

  Note: additional checks depend whether the resource is protected using attributes or capabilities.

2. An overview of the protocols

Different protocols can be used for different purposes:

(1) to discover the main features supported either by a RS or an AS;

(2) to allow an end-user to query which of his attributes are known by an AS;

(3) to create a long-term user account on a RS (that will last between different sessions) using an access token;

(4) to create a short-term user account on a RS (that will last during the duration of a single session) using an access token;

(5) to perform an operation on a resource hosted by a RS.

2.1. RS and AS Discovery APIs

2.1.1. RS Discovery API

A GNAP RS can publish its features on a well-known discovery document using the URL ".well-known/gnap-rs" appended to the base URL of the RS.

The discovery response is a JSON document [RFC8259] consisting of a single JSON object with the following fields:

trusted_AS (array) : REQUIRED: A list of the ASs trusted by the RS.

The array contains an enumeration of ASs and for each AS, the following information SHALL be present :

- the base URL of the AS, and
- one AS PKC.

It is RECOMMENDED to include an image within the AS certificate according to RFC 6170 (Internet X.509 Public Key Infrastructure Certificate Image). The purpose of the certificate image is to aid human interpretation of a certificate by providing meaningful visual information to a user interface (UI).

In addition it is RECOMMENDED to publish the next AS PKC, when it has already been issued by the responsible CA.

Note: the ordering of these ASs is important. See section 5.1.

user_interaction_endpoint (string): REQUIRED. The location of the RS's user interaction endpoint, used by the client to conduct a dialogue between the end-user and the RS.

The goal of this dialogue is to present one or more options to the end-user, so that, after being informed of the consequences of these options, he may provide its consent to the RS.

The location MUST be a URL [RFC3986] with a scheme component that MUST be https, a host component, and optionally, port, path query components and no fragment components.

RS_token_formats_&_syntaxes_supported (array of strings) OPTIONAL: A list of token formats and syntaxes supported by the RS.

long_term_user_account (Boolean) OPTIONAL: indicates whether long-term user accounts are supported by the RS.

short_term_user_account (Boolean) OPTIONAL: indicates whether short-term user accounts are supported by the RS.

Note : either the long_term_user_account flag or the short_term_user_account flag MUST be present and set to TRUE.

user_registration_endpoint (string): OPTIONAL. The RS's user registration endpoint, used by the client to create either a short-term user account or a long-term user account.

rs_cert_path (array): OPTIONAL. A set of CA certificates that may help a client to built a certification path between a RS certificate and a trust anchor.

2.1.2. AS Discovery API

A GNAP AS can publish its features on a well-known discovery document using the URL ".well-known/gnap-as" appended to the base URL of the AS.

The discovery response is a JSON document [RFC8259] consisting of a single JSON object with the following fields:

attributes_supported (Boolean) OPTIONAL: indicates whether attributes requested by the end-user may be included into an access token.

capabilities_supported (Boolean) OPTIONAL: indicates whether capabilities requested by the end-user may be included into an access token.

Note : either the attributes_supported Boolean or the capabilities_supported Boolean MUST be present and set to TRUE.

AS_token_formats_&_syntaxes_supported (array of strings) OPTIONAL: A list of token formats and syntaxes supported by the AS.

grant_request_endpoint (string): REQUIRED. The location of the AS's grant request endpoint, used by the client to request access tokens. The location MUST be a URL [RFC3986] with a scheme component that MUST be https, a host component, and optionally, port, path query components and no fragment components.

as_cert_path (array): OPTIONAL. A set of CA certificates that may help a client to built a certification path between an AS certificate and a trust anchor.

2.2 Queries from an end-user to an AS

Before making this query, the client SHALL establish an HTTPS connection with the AS. The client SHOULD be able to communicate to the AS the preferred language(s) of the end-user. The end-user SHALL successfully authenticate with the AS.

Any kind of authentication method can be used, e.g. using asymmetric cryptography, symmetric cryptography or even using end-user identifiers associated with (long) passwords, since the connection is both integrity and confidentiality protected using HTTPS.

Since the AS knows some attributes that belong to the end-user, they will be returned in the response to this query. Each attribute has a type and a value. These attributes are considered as personal data and, as such, the end-user SHALL be able to have access to them. In some cases, it may be allowed to correct some of them or to propose corrections to them.

Note: In this case, no access token will be returned.

2.3. Creation a long-term user account on a RS

Two different cases needs to be considered, whether the RS already "knows" the end-user or not. The client SHOULD first make sure that the RS has set the long_term_user_account flag, otherwise no long-term user account can be created.

2.3.1. The RS already "knows" the end-user

The RS may already "know" the end-user because it already holds some information about him and the end-user is already identified by the RS under a user account number.

The user would like now to like to get on on-line access to perform some operations on some information associated with his user account.

The client performs a RS Discovery operation to know which ASs are trusted by the RS. If the user has an account on one of these ASs, the end-user (or the client) may select one of these ASs.

The RS asks the client to provide some user attributes types already known by the AS that has been selected by the client. The dialogue with the end-user SHALL be handled using the user_interaction_endpoint.

The RS allows the user to know the reason(s) why such attribute types are being requested (User Notice). The user may agree or deny to provide them (User choice and User Consent) and, if he agrees, the client will request to the AS an access token that should contain these attribute types.

For example, the attribute types may be: first name, family name, birth date and birth location.

In order to detect a possible replay or use of a delivered access token by an AS for a given RS by another client, the client indicates that the access token should be protected under, e.g. :

(1) a unique user identifier used to identify a user for each User/ RS pair;

(2) a unique user identifier issued by the AS to identify the end-user for each AS / RS pair (e.g. a different pseudonym for each AS / RS pair), or

(3) a locally unique user identifier used by the AS to identify the user, whatever server is involved (e.g. a single pseudonym used for all the servers), or

(4) a globally unique user identifier (e.g. a personal email address, a social security number including the issuing country, a passport number including the issuing country, a driving license including the issuing state or country).

In order to detect the replay of the access token by a RS towards another RS, the client indicates the identifier of the intended RS. The fields "rs_url" or "hidden_url" of the access token may be used to allow such a detection.

The returned access token will include some attributes. These attributes are placed in the "attrs" field of the access token.

The returned access token will then include in the "binding user identifier" field ("buid") of the access token, the type of the binding unique user identifier requested and a value assigned by the AS.

If the attribute types and values contained in the access token match with the already known attribute types and values, the operation will be granted.

It should be observed that, when using the choice (2), it is not possible to hide to the AS the URL of the RS, since this value is needed to generate a different pseudonym for each AS / RS pair.

It may be observed that, when using the choice (3), it is possible to hide to the AS the URL of the RS, since this value is not needed to generate a single pseudonym used for all the servers. However, in this case, all the RSs receiving access tokens from the same AS are able to link their user accounts.

It may be observed that, when using the choice (4), it is possible to hide to the AS the URL of the RS, since this value is not needed to generate a globally unique user identifier. However, in this case, not only all the RSs receiving access tokens from the same AS will be able to link their user accounts, but also other servers using the same globally unique user identifier.

The choice (4) should be avoided, unless all the RSs issuing attributes or rights are managed by the same organization that is managing the AS and that organization is purposely willing to link all the user accounts of its RSs. At the moment, the "best" choices in the Internet environment are be restricted between the choices (2) and (3) and will be a compromise between two privacy properties, where each of these two choices has an advantage and a drawback :

- the choice (2) does not allow to hide to the AS the URL of the RS, but prevents the RSs receiving access tokens from the same AS to link their user accounts, while

- the choice (3) allows to hide to the AS the URL of the RS, but allows the RSs receiving access tokens from the same AS to link their user accounts.

That choice will be left to the end-user (or to the client).

> Note: the choice (1) would allow both to hide to the AS the URL of the RS and to prevent the RSs to perform a linkage between their user accounts.

Later on, if the client requests another access token, the client or the end-user should remember its original choice, e.g. (2) or (3), for the same RS, otherwise the access token will be rejected by the RS.

It is recommended that each client locally manages for each end-user a "privacy preference profile" to associate a choice value with the base URL of each RS.

In order to detect the replay of the access token by a RS towards another RS, the client indicates the identifier of the intended RS using either the "rs_url" field for the choice (2) or the "hidden_url" field for the choice (3) of the access token.

The returned access token will then include some attributes in the field "attrs" of the access token and the binding user identifier issued by the AS in the field "buid" of the access token.

Before presenting this access token to the RS, the end-user SHALL establish an HTTPS connection with the RS. Then after, the access token SHALL be send to the user_registration_endpoint of the RS. Once the RS has verified that the access token is valid, it memorizes the "binding user identifier" field ("buid") of the access token. All subsequent access tokens issued by that AS SHALL contain the same value, otherwise they will be rejected.

It is recommended that each client locally manages for each end-user a "privacy preference profile" to associate a choice value with the base URL of each RS.

2.3.2. The RS does not already "know" the user

The RS does not yet "know" the user: he has no user account for the end-user on that RS. The end-user wants to create a user account.

The client performs a RS Discovery operation to know which ASs are trusted by the RS. If the user has an account on one of these ASs, the end-user (or the client) may select one of these ASs.

In order to create a RS user account, a RS will likely ask for both claimed attributes and certified attributes (i.e. attributes contained in an access token delivered by an AS trusted by the RS). Claimed attributes are simply attributes declared by the user.

The user_interaction_endpoint of the RS will be used to conduct a dialogue between the RS and the end-user in order to request both claimed attributes and certified attributes. During that dialogue, the end-user has the ability to know the reason(s) an/or the rational for the collection of each attribute type and which kind of treatment will be made of it.

Once the end-user has agreed to request some attributes types to the selected AS, the client requests to that AS an access token that contains some attribute types known by that AS.

As in the previous case, in order to detect a possible replay or use of a delivered access token by an AS for a given RS by another client, the client indicates that the access token should be protected under, e.g. :

(1) a unique user identifier used to identify a user for each User/ RS pair;

Note: this option is only possible when the end-user is using a specific secure element.

(2) a unique user identifier issued by the AS to identify the user for each AS / RS pair (e.g. a different pseudonym for each AS / RS pair), or

(3) a locally unique user identifier used by the AS to identify the user, whatever server is involved (e.g. a single pseudonym used for all the servers), or

(4) a globally unique user identifier (e.g. a personal email address, a social security number including the issuing country, a passport number including the issuing country, a driving license including the issuing state or country).

The choice (4) should be avoided. At the moment, the "best" choices will be restricted between the choices (2) and (3) and will be a compromise between two privacy properties, since each of these two choices has an advantage and a drawback :

- the choice (2) does not allow to hide to the AS the URL of the RS, but prevents the RSs receiving access tokens from the same AS to link their user accounts, while

- the choice (3) allows to hide to the AS the URL of the RS, but allows the RSs receiving access tokens from the same AS to link their user accounts.

If the attribute types and values contained in the access token match with the expected attribute types and if the requested claimed attributes are also received, some verifications will be done by the RS. Such verifications introduce some delay.

This is why the final response about the creation of the user account on the RS is asynchronous.

Later on, if the client requests another access token, the client or the end-user should remember its original choice, e.g. (2) or (3), for the same RS, otherwise the access token will be rejected by the RS.

It is recommended that each client locally manages for each end-user a "privacy preference profile" to associate a choice value with the base URL of each RS.

In order to detect the replay of the access token by a RS towards another RS, the client indicates the identifier of the intended RS using either the "rs_url" field for the choice (2) or the "hidden_url" field for the choice (3) of the access token.

The returned access token will then include some attributes in the field "attrs" of the access token and the binding user identifier issued by the AS.

Before presenting this access token to the RS, the end-user SHALL establish an HTTPS connection with the RS. Then after, the access token SHALL be send to the user_registration_endpoint of the RS. Once the RS has verified that the access token is valid, it memorizes the "binding user identifier" field ("buid") of the access token. All subsequent access tokens issued by that AS SHALL contain the same value, otherwise they will be rejected.

2.4. Creation a short-term user account on a RS

The client performs a RS Discovery operation to make sure that the RS has set the short_term_user_account flag, otherwise no short-term user account can be created.

The client performs a RS Discovery operation to know which ASs are trusted by the RS. If the user has an account on one of these ASs, the end-user (or the client) may select one of these ASs.

In order to detect a possible replay or use of a delivered access token by an AS for a given RS by another client, the client indicates that the access token SHALL be protected using a short-term user unique identifier.

The client (or the end-user) has then two options: either to hide or to reveal the URL of the RS. If it wants to hide the URL of the RS, it can only use one session. For this purpose, it SHALL use the "hidden_url" field of the access token. In order to connect with another RS, it SHALL explicitly release that session. If it agrees to reveal the URL of the RS, it SHALL use the "rs_url" field of the access token. It can then use multiple short-term sessions with different RSs in parallel. The two choices are exclusive.

The returned access token will then include in the "binding user identifier" field ("buid") of the access token, the type of the binding user identifier requested and a value assigned by the AS. This value assigned by the AS SHOULD be a large pseudo-random number.

In order to detect the replay of the access token by a RS towards another RS, the client indicates the identifier of the target RS. The fields "rs_url" or "hidden_url" of the access token may be used to allow such detection.

Note: none of the two optional fields "attrs" and "rights" needs to be present in this access token.

Before presenting this access token to the RS, the end-user SHALL establish an HTTPS connection with the RS. Then after, the access token SHALL be send to the user_registration_endpoint of the RS. Once the RS has verified that the access token is valid, it memorizes the "binding user identifier" field ("buid") of the access token.

All subsequent access tokens issued by that AS SHALL contain the same value, otherwise they will be rejected.

2.5. Operation on a resource hosted by a RS

2.5.1. Operation on a resource without an access token

The client does not necessarily know in advance, whether the resource is or is not protected. If the resource is unprotected and if the API is well formed, then the access will be granted.

When a client instance calls an RS without an access token, or with an invalid access token, if the resource is protected and if the API is well formed, then different HTTP errors types may be returned, in particular:

401 "Unauthorized" which indicates an authentication error. It always includes a WWW-Authenticate header that describes how to authenticate.

In this particular case, the RS SHALL respond with a header field that contains one challenge for the "GNAPv1" scheme and two additional parameters "attrs" and "rights" to indicate whether the RS supports attributes and/or rights for that resource.

After each parameter (i.e. "attrs" and "rights") follows a pointer to the AS(s) trusted by the RS as enumerated in the "trusted_AS" field from the RS Discovery API.

For example: WWW-Authenticate: GNAPv1 attrs= 1,3,4 rights= 2,3

In this example, both attributes and rights are supported. AS 1 and AS 4 support attributes only, AS 3 supports both attributes and capabilities (i.e. rights) while AS 2 supports capabilities (i.e. rights) only.

If either attributes or rights are not supported, the following values SHALL be used respectively: "attrs= 0" or "rights= 0".

The client SHOULD then use the "RS Discovery API" to find out which are the ASs trusted by the RS.

403 "Forbidden" which indicates that the server understood the request but that the client instance does not have permission to access this resource, even if it has been authenticated. A server that wishes to make public why the request has been forbidden can describe that reason in the response payload (if any).

When the request is recognized by the server but sent "without an access token or with an invalid access token", the HTTP status 403 Forbidden SHOULD be used. If the server wants to make known why a request is forbidden, it can provide the reason in the payload.

> Note 1: 403 "Forbidden" is dedicated to authorization errors, whereas 401 "Unauthorized" is dedicated to authentication errors.
>
> Note 2: A server that wishes to hide the existence of a forbidden target resource MAY instead respond with a status code of 404 (Not Found).

404 "Not Found" which indicates either that the server did not find a current representation for the target resource or that the server is not willing to disclose that one exists.

405 "Method not allowed" which indicates that the method received in

the request-line is known by the server but not supported by the target resource. In that case, the server MUST generate an Allow header field containing a list of the target resource's currently supported methods.

Note: If the method is incorrect, 405 "Method not allowed" SHALL have precedence over 403 "Forbidden".

Note: These error codes are defined in RFC 7231 [RFC7231]and RFC 7235 [RFC7235] as follows:

- 400 Bad Request	Section 6.5.1 of RFC 7231
- 401 Unauthorized	Section 3.1 of RFC 7235
- 403 Forbidden	Section 6.5.3 of RFC 7231
- 404 Not Found	Section 6.5.4 of RFC 7231
- 405 Method Not Allowed . . .	Section 6.5.5 of RFC 7231
- 406 Not Acceptable	Section 6.5.6 of RFC 7231

2.5.2. Operation on a resource using an access token

The client instance determines which operation on a resource is needed and which RS to approach for access.

Unless the client already knows from a previous experience what kind of additional data needs to be presented, the client has the possibility to query the RS to know which kind of protection is being used by the RS for that resource.

It requests on operation on the intended resource and voluntarily omits to send any access token. It will then get an 401 "Unauthorized" error that indicates that GNAPv1 is supported and whether attributes and/or rights should be presented in an access token.

It will also know which ASs, if any, are trusted by the RS to deliver attributes in an access token and which ASs, if any, are trusted by the RS to deliver capabilities in an access token.

The client instance determines that the RS supports GNAP and the process may continue.

2.5.2.1. Dialogue between the end-user and his client

The client may be configured by the end-user with the URLs of the ASs where he has an AS user account. It may then propose to select one or more ASs showing at the same time, if capabilities or attributes may be requested on these ASs to be included into an access token.

If some choices remain, then the client SHOULD ask the end-user to make these choices. The client then knows which AS has been chosen and whether attributes (ABAC) or capabilities (CBAC) will later on be requested to that AS.

The client SHALL determine whether the end-user is willing to make an access to the RS using a short-term user account or a long-term user account. Usually, it should know it from the context of the operation. However, if it doesn't know, it SHALL ask the end-user to make that choice.

If the client used by the end-user has recently opened a short-term user account, it SHOULD be usable if it has not been closed, otherwise a new short-term user account SHOULD be created.

If the access to the RS should be done using a long-term user account, the same choice as originally made by the end-user for the binding user identifier when opening his user account on the RS should be done. If the client locally manages for each end-user a "privacy preference profile" that associates a choice value with the base URL of each RS, it should re-use the same choice. Otherwise, the question should be raised again to the end-user.

2.5.2.2. Dialogue between the end-user and the RS

When using ABAC, a dialogue needs to be established between the end-user and the RS. Such dialogue needs to be supported using a port able to support a User Interface (UI). The address of this port SHALL be published by the RS and made available using the RS Discovery API. It SHALL be a URL hosted by the RS.

When initiating the dialogue on the UI port of the RS, the client communicates to the RS which AS has been selected by the end-user.

The RS indicates to the end-user which attribute types and optionally attribute values should be present in an access token issued by that AS.

Before making a call to that AS, the end-user may wish to be informed of the treatments or usages that will be done by the RS with each requested attribute type, including a possible disclosure to other third parties. Such information is usually present in a User Notice. A simple click or a selection of an attribute type should be sufficient to disclose the terms of this User Notice.

The UI SHALL clearly ask the end-user whether it accepts to request these attribute types and SHALL require an action or a gesture from the end-user to indicate its acceptance.

At this point of time, the choice made by the end-user SHOULD be memorized by the RS and also by the client.

The memorization done by the RS is not for technical reasons, but to comply with some laws or regulations.

As an example, the General Data Protection Regulation [GDPR] from the EU indicates in Article 7(1) :

> "Where processing is based on consent, the controller SHALL be able to demonstrate that the data subject has consented to processing of his or her personal data. "

The memorization done by the client is for technical reasons: the client SHALL use the choice made by the end-user to request an access token to the AS that SHOULD include the attribute types selected by the end-user.

A markup language such as XML needs to be used in order to delineate the meaning of each field and its value, when present.

Note: when the RS supports CBAC for a protected resource, no dialogue is needed between the end-user and the RS.

2.5.2.3. The access token request

The client knows whether the access token will be used for a short-term RS user account or a long-term RS user account.

If a long-term user account is being used, the client SHOULD already know the privacy preference of the user since they have been chosen when creating the long-term user account (see section 3). If the user is using a new client (i.e. device), then the new client SHOULD inquire it again.

The client already knows whether the resource is protected using attributes and/or rights (see section 5.1). If it is protected using attributes, it already knows which types of attributes should be requested to the AS, and optionally which attribute values.

The request MUST be sent as a JSON object in the body of the HTTP POST request with Content-Type "application/json".

Each field placed in the body of the HTTP POST request is described below:

at_format : REQUIRED. The format of the access token.

at_crypto : REQUIRED. The asymmetric crypto algorithm and the one way hash function to be used for computing the digital signature of the access token.

val_period (string) OPTIONAL. It is the desired validity period of the access token expressed in hours and minutes. The AS may override this value.

buid_type (integer) : REQUIRED. The binding user identifier type to be used with the RS. The allowed values are 1 to 5.

target_rs (string) : either a "rs_url" value or a "hidden_url" value that SHALL be placed into the access token.

Note : if the buid_type is set to 2, then a "hidden_url" value SHALL NOT be accepted.

operations (array of string) : REQUIRED when capabilities are requested. It is a list of capabilities, where each capability consists of :

- one or more methods and
- the URL of the protected resource.

attrs_types (array of string) : REQUIRED when attributes are requested. It is a list of attribute types. These attributes may be static or computed from a static attribute. As an example, an age categorization attribute would be a computed attribute composed of one or two values, like "over 12" and "under 18" (see min-age and max-age).

The naming and the definitions used in Table 1 (Registered Member Definitions) from the "OpenID Connect Core 1.0 specification" [OIDC_1.0] are re-used in this document.

See : https://openid.net/specs/openid-connect-core-1_0.html#Claims

As a consequence, the following attributes types are defined:

name, given_name, family_name, middle_name, nickname, preferred_username, profile, picture, website, email, email_verified, gender, birthdate, zoneinfo, locale, phone_number, phone_number_verified, address.

In addition, subsets of the previous fields may be returned. For example, for an address, the country and region only may be requested. This leads to recognize the following attribute types: street_address, locality, region, postal_code, country.

In addition, the following attribute types may also be useful: shipping_address, payment_info, eye_color, max_age, min_age.

In order to support group memberships, the two following attribute types are also defined: hierarchical_group and functional_group.

Since a user may belong to more than one functional group, the value(s) that should be placed into the access token should be indicated in the request, otherwise all the functional groups will be returned.

The end-user is able to know which values have been affected to this attribute type by sending a query to the AS (see section 2.2). As a consequence, the client is able to specify which attribute value(s) should be returned for the functional_group attribute type.

2.5.2.4. The access token response

The response MUST be sent as a JSON object in the body of the HTTP POST request with Content-Type "application/json". It is an access token. Its semantics and structure are described in section 1.7.

2.6. Flow of operations for one access

Once either a short-term user account or a long-term user account has been created on a RS, the flow of operations is illustrated hereafter:

```
                                        +---------------+
                         +-------------->|               |
                         |(5)           | Authorization |
                         |              |    Server     |
                         |      +-------|               |
                         |      |(6)    +---------------+
                         |      |                ~
                         |      |                ~
                       +----------+     +---------------+
                       |          |     |               |
      +----------+ (3) |  Client  |     |    Resource   |
      | End-user |+ + +| Instance |     |     Owner     |
      +----------+     |          |     |               |
           ^           +----------+     +---------------+
           |            ^   ^   |               ~
           |            |   |   |               ~
           |            |   |   |(7)    +---------------+
           |            |   |   +------>|               |
           |            |   |(2)        |               |
           |            |   +---------->|    Resource   |
           |            |(1)            |     Server    |
           |(4 Opt.)    +-------------->|               |
           +---------------------------->|               |
                  User Interface        +---------------+
```

Figure 2: Flow of operations for one access

(1) RS Discovery
(2) Operation on a resource hosted by a RS without an access token
(3) Dialogue between the end-user and the client

```
    (4)  Optional dialogue between the end-user and RS
         when the RS requests attributes
    (5)  Access Token request to AS
    (6)  Access Token response from AS
    (7)  Access Token presentation to RS
```

Note: The creation of RS user accounts is not illustrated
on Figure 2.

3. IANA Considerations

This specification will require the registration of the various attributes types defined in section 5.2.3:

name, given_name, family_name, middle_name, nickname,
preferred_username, profile, picture, website, email,
email_verified, gender, birthdate, zoneinfo, locale,
phone_number, phone_number_verified, address,

street_address, locality, region, postal_code, country,

shipping_address, payment_info, eye_color, max_age, min_age.

4. Security Considerations

Since HTTPS is used between the client and the RS and between the client and the AS, the client can be confident that the information it receives is indeed coming from the intended RS and from the intended AS.

Since the authentication exchange between the end-user and the AS is protected by HTTPS, the AS can be confident that the end-user is using the unknown connected client. Any kind of authentication exchange can be used, including the simple "id and password" scheme, since the password is both integrity and confidentiality protected during its transfer.

The appropriate version (or versions) of TLS will vary over time, based on the widespread deployment and known security vulnerabilities. Refer to [BCP195] for up to date recommendations on transport layer security.

The transmission of an access token obtained by one end-user to another end-user cannot be prevented, but can be detected by the RS.

In order to detect the transmission of the access token by one client towards another client, the AS includes in every access token a binding user identifier ("buid").

Every access token is bound to a RS user account (either a short-term user account or a long-term user account). This is done by including a binding user identifier ("buid") in every access token. A binding user identifier is composed of a type and of a value. The client can choose the type of the binding user identifier but not its value which is only assigned by the AS.

All access tokens that are presented to a RS in the context of a given RS user account must contain the same binding user identifier.

If an end-user obtains an access token from a collaborative end-user, he cannot use it on his own user account since it will not contain the same binding user identifier ("buid").

Let us use an example to illustrate the topic.

Some goods or activities with some preferred rates are only disclosed by a town to its residents and if there are over 21. In such a case, two attribute types and values will be requested by the RS, e.g. :

```
    Town of residence: Nashville - Tennessee - 840
    Age categorization: over 21
```

Let us assume that Alice has the two following attributes:

```
    Town of residence: Nashville - Tennessee - 840
    Age categorization: over 16
```

while Bob has the two following attributes:

```
    Town of residence: San Francisco - California - 840
    Age categorization: over 21
```

If Alice asks for an access token that only contains:

```
    Town of residence: Nashville - Tennessee - 840
```

and Bob asks for an access token that only contains:

```
    Age categorization: over 21
```

if they agree to collaborate, they will not be able to combine their attributes to perform an operation on the server managed by the town of Nashville, since they will not contain the same binding user identifier ("buid").

It is proposed to use to the wording "binded token" to designate an access token that contains a binding user identifier.

Note: It should be noted that access tokens do not need to be "protected" by a private key known by the client. Such a protection would be illusory, since a collaborative client could perform all the cryptographic computations that another collaborative client would need to claim to be the "owner" of the access token. This can be done even these private keys are protected using an hardware security module (HSM).

5. Privacy Considerations

ISO/IEC 29100 (Privacy framework) [ISO29100] lists eleven privacy principles that are valid in a system with two entities which correspond in this document to the relationships between one end-user and one RS.

Note: ISO/IEC 29100 is available from ISO for free.

Among the privacy principles from ISO/IEC 29100 enumerated on page 14 in Table 3, the following privacy principles are particularly important:

- Individual participation
- Purpose legitimacy and specification,
- Consent and choice,
- Collection limitation,
- Data minimization,
- Use, retention and disclosure limitation,
- Openness, transparency and notice.

Since this document considers two access control schemes: Attribute Based Access Control (ABAC) and Capability Access Control (CBAC), the privacy considerations for both of them are first addressed (section 5.1) followed by the privacy considerations for each of them (sections 5.2 and 5.3).

However, the current document considers a more complex system with at least three entities: the end-user, the RS and the AS, where each of them may exists more than once.

In this environment, some additional privacy properties also need to be considered, in particular those that apply between RSs (section 5.4) and privacy properties that apply between the end-user and the AS (section 5.5).

5.1. Privacy Considerations for both ABAC and CABC

Since the AS knows some attributes from the end-user, applying the "individual participation" principle from ISO/IEC 29100, means giving to the end-users the ability to access and review their attributes, provided their identity is first authenticated with an appropriate level of assurance and using a language which is both clear and appropriately adapted to the circumstances.

This principle is reached using queries from an end-user to an AS (see section 2.2) since such a query is performed once the client has established an HTTPS connection with the AS and the end-user has successfully authenticated with the AS.

For both ABAC and CABC a user account SHALL be created on the RS. This account may be either short-term or long-term. For a long-term user account, at the moment, the end-user needs to choose between hiding to the AS the URL of the RS or allowing RSs to link their user accounts.

5.1.1. Privacy Considerations for ABAC

In this access control scheme, end-user attributes need to be presented to the RS. The end-user SHALL be able to :

- select the AS that will be contacted to deliver attributes,
- know which attribute types and attribute possibly values are requested by the RS,
- know the reasons and/or the rational for providing these attribute types and possibly values,
- consent or disagree with the provision of these attributes from an AS that he has selected.

The previous functionalities are supported by a local dialogue between the end-user and the client and by another dialogue between the end-user and the RS.

This allows adhering to the purpose of legitimacy principle: communicating the purpose(s) to the end-user before the time the information is collected or used for the first time for a new purpose.

It is possible for the client to hide to the AS the URL of the RS. However, at this time, until a secure element is being used, the price to pay for that feature is to allow RSs to link their user accounts. So the end-user will have to make a choice between these two features.

Note: The core principles of ABAC permit to the client to hide to the AS the method that will be performed on a resource as well as the URL of that resource.

5.1.2. Privacy Considerations for CBAC

In this access control scheme, capabilities need to be presented to the RS. The end-user SHALL be able to :

- select the AS that will be contacted to deliver capabilities,
- consent or disagree with the provision of these capabilities from an AS that he has selected.

The above functionalities are supported by using a local dialogue between the end-user and the client.

The core principles of CBAC do not permit the end-user to hide to the AS the URL of the RS, nor to hide to the AS the URL of the resource, nor to hide to the AS the method that will be performed on the resource.

5.2. Privacy Considerations between RSs

Two RSs should not be able to link their user accounts, by using the content of the access tokens they receive from the same AS or from different ASs. This property may be referred as: Unlinkeability between RS user accounts. For short-term user accounts, it is always achieved. However, for long-term user accounts, it may only be achieved, for the moment, if the client/end-user accepts to disclose to the AS the URL of the RS.

5.3. Privacy Considerations between the end-user and the AS

5.3.1. Transparency

When a client receives an access token from an AS, it should be able to verify that the access token contains the requested privileges, i.e. no more or not less, and, if not, it should be able to prevent the transmission of the access token to the RS.

This property can be achieved as long as "Token introspection", as currently described in OAuth, is not being used. It should be noticed that Token introspection may allow an AS to deliver to the client more privileges than the ones inserted into an access token.

These two properties are directly related to the "Transparency" of the system since they allow the end-users to be confident in the system they are using.

For these reasons, access tokens are not considered to be opaque to the clients but may be considered to be opaque for the end-users.

5.3.2. Hiding to the AS the URL of the RS and its use

The AS should not be able to know when an issued access token will be indeed consumed by a RS. This property can be achieved by using the "hidden_url" field and as long as "Token introspection" is not being used.

6. References

6.1. Normative References

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List(CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008. <https://www.rfc-editor.org/info/rfc5280>.

[RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March 2011, <https://www.rfc-editor.org/info/rfc6125>.

[RFC6960] Santesson, S., Myers, M., Ankney, R., Malpani, A., Galperin, S., and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", RFC 6960, DOI 10.17487/RFC6960, June 2013, <https://www.rfc-editor.org/info/rfc6960>.

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <https://www.rfc-editor.org/info/rfc7231>.

[RFC7235] R. Fielding, J. Reschke, Hypertext Transfer Protocol (HTTP/1.1): Authentication, June 2014, <https://www.rfc-editor.org/info/rfc7235>

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/info/rfc8259>

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

6.2. Informative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[BCP195] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, May 2015. <https://www.rfc-editor.org/info/bcp195>

[OIDC_1.0] OpenID Connect Core 1.0 specification <ttps://openid.net/specs/openid-connect-core-1_0.html>

[ISO29100] Information technology - Security techniques - Privacy framework. 2011. ISO/IEC 29100 is available for free at: https://standards.iso.org/ittf/PubliclyAvailableStandards/c045123_ISO_IEC_29100_2011.zip

[GDPR] Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). <https://eur-lex.europa.eu/eli/reg/2016/679/oj>

Authors' Address

Denis Pinkas
DP Security Consulting

Email: denis.ietf@free.fr